Appendix C
R Code
Jon Baldock and Kimberly Garland-Campbell

Scripts followed by "jb" indicate that Jon Baldock is the author; and scripts followed by "kagc" indicate that Kimberly Garland-Campbell is the author.

Many chapters in this book provided examples using SAS code and output. For readers who prefer R, this Appendix translates much of the SAS code to R and provides the corresponding output. The Appendix identifies each chapter and author(s) to help readers easily find the R code they seek. The chapter authors did not participate in the translation of this code and have no responsibility for the contents of this Appendix.

In preparing the scripts in this appendix, we have focused on getting similar numbers or graphs to those from SAS and not so much on the format. In several of the early examples, we have shown how to work on the format, but in general, getting formats and order of results to agree has been secondary. For easier comparison with SAS output, we have highlighted the key results in yellow and made comments highlighted in green. There are two R versions that we have used in working with these scripts: version 3.4.1 "Single Candle" and version 3.4.2 "Short Summer" (R Core Team, 2017). Teetor (2013, Pages 62-65) and Chang (2017, Running_a_script) show how to run scripts. R Core Team (2017) and Chang (2017) provide general on-line help writing and running R programs. Teetor (2013), Faraway (2015), and Chang (2017) are excellent hardcopy resources on statistical analyses and graphics in R. Fox and Weisburg (2014) and Clark (2018) are outstanding on-line resources for the statistical analyses described in this appendix.

Table of Contents

Chapter 2. Analysis of Variance and Hypothesis Testing by Marla S. McIntosh.

Table 6. Experimental design.

Appropriately, the first SAS code to translate is designing an experiment; specifically, the hypothetical Statbean study. It consists of a 2-by-5 factorial treatment plan in a Randomized Complete Block Design (RCBD) at three locations. There are three blocks (1 replication per block) at each site. There are a number of R packages that could be used to do the randomizations for this study. I have used the agricolae package because it automatically includes plot numbers for RCBDs and can arrange them in either left to right or serpentine numbering. (WARNING: the agricolae package contains several outdated functions for analyzing data.) In the following script, I have simply copied and pasted the steps to design an RCBD at one site for the other two locations. There are ways to automate this in R, but it would require writing a user defined function. That is not a big problem, but for small numbers it is easier to copy and paste. Table 1 shows the equivalent treatment layouts to those of Table 6 in Chapter 2.

Script 2.1 jb

```
#activate additional packages needed
library(agricolae)

#statistical analyses
#set number of treatments (2 x 5 factorial) and blocks
trt <- 1:10 ; rep <-3
#create design for Central location & variable for loc.
loc_c <- design.rcbd(trt, r=rep, serie=2, seed=123)
loc_c$book$Loc<-"Central"
#print layout
print(loc_c$sketch)
#do designs for East and West locations & print layouts
loc_e <- design.rcbd(trt, r=rep, serie=2, seed=456)
loc_e$book$Loc<-"East"
print(loc_e$sketch)
loc_w <- design.rcbd(trt, r=rep, serie=2, seed=789)
loc_w$book$Loc<-"West"
print(loc_w$sketch)
#combine the 3 design books into one data frame
all3 <- rbind(loc_c$book, loc_e$book, loc_w$book)
#print columnwise
print(all3)
#save the combined design in a .csv file
write.csv(all3, "table6-design.csv", row.names=FALSE)
```

Output 2.1 jb

```
> source("c1-table6.r", echo=T)

> #activate additional packages needed
```

```
> library(agricolae)

> #statistical analyses
> #set number of treatments (2 x 5 factorial) and blocks
> trt <- 1:10 ; rep <- 3
> #create design for Central location & variable for loc.
> loc_c <- design.rcbd(trt, r=rep, serie=2, seed=123)
> loc_c$book$Loc<-"Central"
#print layout

> print(loc_c$sketch)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "6"  "9"  "8"  "10" "7"  "1"  "4"  "5"  "2"  "3"
[2,] "8"  "2"  "9"  "3"  "4"  "6"  "1"  "5"  "10" "7"
[3,] "1"  "9"  "5"  "7"  "10" "2"  "3"  "8"  "4"  "6"

> #do designs for East & West locations & print layouts
> loc_e <- design.rcbd(trt, r=rep, serie=2, seed=456)
> loc_e$book$Loc<-"East"

> print(loc_e$sketch)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "10" "7"  "8"  "4"  "6"  "1"  "5"  "2"  "9"  "3"
[2,] "9"  "2"  "4"  "10" "5"  "1"  "6"  "8"  "3"  "7"
[3,] "6"  "3"  "9"  "2"  "10" "5"  "8"  "7"  "4"  "1"

> loc_w <- design.rcbd(trt, r=rep, serie=2, seed=789)
> loc_w$book$Loc<-"West"

> print(loc_w$sketch)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "7"  "4"  "2"  "1"  "3"  "6"  "5"  "10" "9"  "8"
[2,] "6"  "3"  "8"  "4"  "10" "2"  "1"  "5"  "9"  "7"
[3,] "8"  "4"  "10" "6"  "7"  "2"  "9"  "5"  "1"  "3"
```

Note the seeds were set so the output could be replicated exactly. Reset them for each design run or remove them and let the computer generate them. The numbers on the left of the layout are the block (aka rep) numbers and the numbers at the top are the plot numbers. The Save To File option in R allows one to capture these layouts in a .txt file and combine them as in Table 1.

```
> #combine the 3 design books into one data frame
> all3 <- rbind(loc_c$book, loc_e$book, loc_w$book)

> #print columnwise
> print(all3)

   plots block trt     Loc
1    101     1   6 Central
2    102     1   9 Central
3    103     1   8 Central
4    104     1  10 Central
5    105     1   7 Central
6    106     1   1 Central
7    107     1   4 Central
```

```
8     108      1    5 Central
9     109      1    2 Central
10    110      1    3 Central
11    201      2    8 Central
12    202      2    2 Central
13    203      2    9 Central
14    204      2    3 Central
15    205      2    4 Central
16    206      2    6 Central
17    207      2    1 Central
18    208      2    5 Central
19    209      2   10 Central
20    210      2    7 Central
21    301      3    1 Central
22    302      3    9 Central
23    303      3    5 Central
24    304      3    7 Central
25    305      3   10 Central
26    306      3    2 Central
27    307      3    3 Central
28    308      3    8 Central
29    309      3    4 Central
30    310      3    6 Central
31    101      1   10      East
32    102      1    7      East
33    103      1    8      East
34    104      1    4      East
35    105      1    6      East
36    106      1    1      East
37    107      1    5      East
38    108      1    2      East
39    109      1    9      East
40    110      1    3      East
41    201      2    9      East
42    202      2    2      East
43    203      2    4      East
44    204      2   10      East
45    205      2    5      East
46    206      2    1      East
47    207      2    6      East
48    208      2    8      East
49    209      2    3      East
50    210      2    7      East
51    301      3    6      East
52    302      3    3      East
53    303      3    9      East
54    304      3    2      East
55    305      3   10      East
56    306      3    5      East
57    307      3    8      East
58    308      3    7      East
59    309      3    4      East
60    310      3    1      East
61    101      1    7      West
```

```
62   102      1   4    West
63   103      1   2    West
64   104      1   1    West
65   105      1   3    West
66   106      1   6    West
67   107      1   5    West
68   108      1  10    West
69   109      1   9    West
70   110      1   8    West
71   201      2   6    West
72   202      2   3    West
73   203      2   8    West
74   204      2   4    West
75   205      2  10    West
76   206      2   2    West
77   207      2   1    West
78   208      2   5    West
79   209      2   9    West
80   210      2   7    West
81   301      3   8    West
82   302      3   4    West
83   303      3  10    West
84   304      3   6    West
85   305      3   7    West
86   306      3   2    West
87   307      3   9    West
88   308      3   5    West
89   309      3   1    West
90   310      3   3    West
```

The .csv file saved to the working directory is exactly like the above print out except there are commas instead of blank spaces. It is easily read by most spreadsheet programs, which makes it simple to add variables and data.

```
> #save the combined design in a .csv file
> write.csv(all3, "table6-design.csv", row.names=FALSE)
=======================================================
```

Table 7 (Chapter 2). Descriptive statistics.

Continuing with the hypothetical Statbean Study, the author used JMP, not SAS, to calculate and create a table of descriptive statistics and to graph a set of box and whisker plots (Table 7 and Fig. 5 in Chapter 2). R can compute the same descriptive statistics and can create an HTML or LaTex file containing those statistics and the formatting necessary to produce a table similar to Table 7 in Chapter 2. R can also create box and whisker plots comparable to Fig. 5 in Chapter 2. The following R script implements those capabilities. The data from the supplemental materials were stored in a file named Statbean-data.csv. This exercise revealed that there is a difference in the default methods SAS and R use to round numbers and determine quantiles. The output for this script includes the print out from the head(), tail(), and summary() functions that verify the data has been read in correctly. Because these results are lengthy and unnecessary in the rest of this Appendix (I'll have already checked them), they will be omitted from all remaining outputs.

Script 2.2 jb

```
#options and packages needed
library(xtable)             #xtable()
library(tidyverse)          #tibble(), group_by(), recode()...

# read in the variable names and data
mdat <- read.csv("Statbean-data.csv")
#attach as modern data frame; i.e. tibble
#print/check first and last six rows of data
head(mdat)
tail(mdat)
#print/check 6 descriptive stats per variable
summary(mdat)

#statistical analysis
#calculate descriptive statistics
sumdat <- mdat %>%
   group_by(Loc, Mulch, Ca_Trt) %>%
    summarize(N=length(pH), Mean=mean(pH),
      StdDev=sd(pH), Min=min(pH), Max=max(pH))
#find numeric columns & round them to tenths
nv <- sapply(sumdat, is.numeric)
sumdat[nv] <- lapply(sumdat[nv], round, digits=1)
#make N a charater variable to keep it as one digit
sumdat$N <- as.character(sumdat$N)
#create table in html format and save it
xtbl <- xtable(sumdat, caption="Descriptive statistics for pH",
digits=1)
print.xtable(xtbl, type="html", file="c2-s2tbl.html")

#create boxplot of pH vs. Ca_Trt by location & save it
ggplot(mdat, aes(x=Ca_Trt, y=pH)) +
   geom_boxplot(aes(color=Loc)) +
   ggtitle("pH vs. Ca_Trt") +
   theme_bw() +
   theme(plot.title=element_text(face="bold",hjust=0.5))
ggsave("c2-fig5x.png", dpi=600)
```

Output 2.2 jb

```
> source("c2-s2.r", echo=T)

> #options and packages needed
> library(xtable)       #xtable()
> library(tidyverse)    #tibble(), group_by(), recode()...

> # read in the variable names and data
> mdat <- read.csv("Statbean-data.csv")
> #attach as modern data frame; i.e. tibble
> mdat <- as.tibble(mdat)
> #print/check first and last six rows of data
```

```
> head(mdat)
# A tibble: 6 x 9
`    Loc    Blk Trt_No  Mulch  Ca_Trt     pH      Ca     Mg PTotal
   <fctr> <fctr>  <int> <fctr>  <fctr> <dbl>   <dbl>  <dbl>  <dbl>
1 Central      I      1     no control  5.58  765.60 145.04   5.00
2 Central      I      2     no   g0730  5.91 1072.40  66.76   5.93
3 Central      I      3     no   g1460  5.55  884.12  43.67  13.33
4 Central      I      4     no   l0730  6.07  976.68 135.75   7.86
5 Central      I      5     no   l1460  6.07 1030.77 136.48  12.86
6 Central      I      6    yes control  6.05  893.02 163.08   2.86

> tail(mdat)
# A tibble: 6 x 9
`    Loc    Blk Trt_No  Mulch  Ca_Trt     pH      Ca     Mg PTotal
  <fctr> <fctr>  <int> <fctr>  <fctr> <dbl>   <dbl>  <dbl>  <dbl>
1   East    III      5     no   l1460  4.12   83.37  11.23   1.60
2   East    III      6    yes control  3.95    9.77  10.18   0.00
3   East    III      7    yes   g0730  3.91  173.93  11.10   0.00
4   East    III      8    yes   g1460  3.75  402.16  17.56   1.60
5   East    III      9    yes   l0730  4.21  123.31  14.88   0.69
6   East    III     10    yes   l1460  4.14  210.88  16.99   1.48

> #print/check 6 descriptive stats per variable
> summary(mdat)
`      Loc      Blk         Trt_No    Mulch         Ca_Trt          pH
Central:30  I  :30  Min.   : 1.0  no :45   control:18 Min.   :3.750
East   :30  II :30  1st Qu.: 3.0  yes:45   g0730  :18 1st Qu.:4.210
West   :30  III:30  Median : 5.5           g1460  :18 Median :5.730
`                  Mean   : 5.5           l0730  :18 Mean   :5.557
`                  3rd Qu.: 8.0           l1460  :18 3rd Qu.:6.565
`                  Max.   :10.0                       Max.   :7.310

`       Ca                Mg              PTotal
 Min.   :   9.77   Min.   :  8.78   Min.   : 0.00
 1st Qu.: 297.03   1st Qu.: 19.86   1st Qu.: 3.39
 Median : 911.96   Median : 63.34   Median :14.81
 Mean   :1100.80   Mean   : 74.61   Mean   :17.36
 3rd Qu.:1790.74   3rd Qu.:119.17   3rd Qu.:27.35
 Max.   :3146.43   Max.   :171.64   Max.   :61.43
```

Based on the head(), tail(), and summary() functions, the input step appears to have proceeded correctly. All the columns read in to the end and the data classes are as expected. There are 30 observations at each location, the treatment codes and numbers are correct, and the measurement data are reasonable based on the six statistic summaries.

```
> #statistical analysis
> #calculate descriptive statistics
> sumdat <- mdat %>%
+    group_by(Loc, Mulch, Ca_Trt) %>%
+     summarize(N=length .... [TRUNCATED]
> #find numeric columns & round them to tenths
> nv <- sapply(sumdat, is.numeric)
> sumdat[nv] <- lapply(sumdat[nv], round, digits=1)

> #recode Ca_Trt levels to match Table 7 of Chapter 2
> #make N a charater variable to keep it as one digit
> sumdat$N <- as.character(sumdat$N)
> #create table in html format and save it
```

```
> xtbl <- xtable(sumdat, caption="Descriptive statistics for pH",
digits=1)

> print.xtable(xtbl, type="html", file="c2-s2tbl.html")
```

The results are shown in Table 2, which is exactly as it came into MSWord as an .html file, except that it was centered and the caption moved from the bottom to the top. Additional format changes, such as converting it from a vertical to a horizontal layout, can be done in MSWord. The statistics match those from JMP, except there are 5 values that differ by 0.1. Four of those are because JMP (and I believe SAS) use the "always round a trailing 5 up" rule, but R uses the "round a trailing 5 to an even number" rule. The other discrepancy must be due to using binary numbers to represent decimal values as well as the rounding difference.

```
> #create boxplot of pH vs. Ca_Trt by location & save it
> ggplot(mdat, aes(x=Ca_Trt, y=pH)) +
+    geom_boxplot(aes(color=Loc)) +
+    ggtitle("pH  ..." ... [TRUNCATED]
> ggsave("c2-fig5x.png", dpi=600)
Saving 5.76 x 5.75 in image
```

The boxplot is displayed as Figure 2.1.

==========

SAS code in Tables 9 and 13 (Chapter. 2). Mixed model ANOVA and related analyses.

Tables 10-13, and 18 plus Figures 6 and 7 of Chapter 2 summarize the preliminary ANOVAs and related analyses for the Statbean Study. The SAS code for these analyses is given in Tables 9 and 13. An R script to duplicate most of the SAS results in those tables follows. The data from the supplemental materials were stored in a file named Statbean-data.csv. Note it is possible to set up an automatic way in R to work through the analysis for each site (similar to the BY statement in SAS); however, it requires writing a user defined function. Consequently, for a small number of sites like this it is easier to use the manual subset function and copy the functions for the analyses to accomplish the task. I did; however, create a function to computerize the panel of diagnostic plots for the three main analyses to match those in Fig. 6 of Chapter 2.

Script 2.3 jb

```
#activate additional packages needed
options(scipen=7, contrasts=(c("contr.sum", "contr.poly")))
library(lme4)          #lmer()
library(car)           #Anova()
library(lsmeans)       #lsmeans() & contrasts()
library(EMSaov)        #EMSanova()
library(tidyverse)     #ggplot
library(gridExtra)     #put graphs on one page

#data management
#read in the variable names and data with read.csv function
mdat <- read.csv("Statbean-data.csv")
#attach as data frame
attach(mdat)
```

```
#check first & last 6 rows of data with head & tail functions
head(mdat)
tail(mdat)
#check 6 descriptive stats per variable with summary function
summary(mdat)

#statistical analyses
#function for 2x2 panel of diagnostic plots: lmer model object
#call with model object from lmer; returns a graphic object
diagPlots<-function(model){
      #predicted values & unscaled residuals in data frame
      predv <- predict(model)
      rres <- residuals(model, scaled=F)
      resdf <- data.frame(predv, rres)
      #plot 1: residuals versus predicted
            p1<-ggplot(model, aes(x=predv,
                  y=rres))+geom_point()+
            geom_hline(yintercept=0, linetype="dashed")+
            xlab("Predicted Values") +
            ylab("Residuals")+
            theme_bw()

      #plot 2: histogram of residuals with normal dist.
      plot.new()
      p2a <- ggplot(resdf, aes(x=rres,
            y=(..density..)*0.15))+
            geom_histogram(binwidth=(0.15), fill="cyan",
            colour="black", center=0.225) +
            xlab("Residuals")+ ylab("Density") +
            theme_bw()
      #construct data frame dfn for normal curve
      x <- seq(1.5*min(rres), 1.5*max(rres),
             length.out=100)
            dfn <- with(resdf, data.frame(x=x,
              y=dnorm(x, mean(rres), sd(rres))))
      p2 <-p2a + geom_line(data=dfn, aes(x=x, y=y*0.15),
             colour="red")

      #Q-Q plot as visual check on normality
      plot.new()
      p3<-ggplot(resdf)+ geom_qq(aes(sample=rres))+
             xlab("Normal Quantiles") +
      ylab("Residuals")+  theme_bw()

      #boxplot of residuals
      plot.new()
      p4<-ggplot(resdf, aes(x=1, y=rres))+
            geom_boxplot(width=0.1 )+
            scale_x_discrete(labels="")+ xlab("All
            Cases")+
            scale_y_continuous(limits=c(-0.5, 0.5))+
            ylab("Residuals") +   theme_bw()
```

```
      #put plots into panel and return
      grid.arrange(p1, p2, p3, p4, ncol=2,
            top="Diagnostics")
      Dplot <- arrangeGrob(grobs=list(p1, p2, p3, p4),
            ncol=2,
            top="Diagnostics")
      return(Dplot)
      }

#extract location = Central data
cdat <- subset(mdat, Loc=='Central')
#recreate location factor to eliminate ghost levels
cdat$Loc <- droplevels(cdat$Loc)
#pH anova with blocks random: Loc= Central
mod.pHc <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=cdat)
print(summary(mod.pHc))
(aov.pHc <- Anova(mod.pHc, type=3, test="F"))
(lsm.pHc <- lsmeans(mod.pHc, specs=c("Mulch", "Ca_Trt")))
#linear contrasts
contrast(lsm.pHc, list(
      "Gypsum-linear"   = c(-1,-1,0,0,1,1,0,0,0,0),
      "Gypsum-quad."    = c(1,1,-2,-2,1,1,0,0,0,0),
      "Lime-linear"    = c(-1,-1,0,0,0,0,0,0,1,1),
      "Lime-quad."     = c(1,1,0,0,0,0,-2,-2,1,1),
      "Lime vs. Gypsum"= c(0,0,1,1,1,1,-1,-1,-1,-1),
      "Interact: GLxM" = c(1,-1,0,0,-1,1,0,0,0,0),
      "Interact: GQxM" = c(-1,1,2,-2,-1,1,0,0,0,0),
      "Interact: LLxM" = c(1,-1,0,0,0,0,0,0,-1,1),
      "Interact: LQxM" = c(-1,1,0,0,0,0,2,-2,-1,1),
      "Interact:MxLvG" = c(0,0,-1,1,-1,1,1,-1,1,-1)))
#request diagnostic plots and save them to file
Dplotc <- diagPlots(mod.pHc)
ggsave("Ch2-Fig6c.png", Dplotc, dpi=600)

#Expected MS for pH anova with blocks random: Loc= Central
expms <- EMSanova(pH ~ Blk + Mulch + Ca_Trt + Mulch*Ca_Trt,
data=cdat, type=c("R", "F", "F"))
expms
pool.ID <- c("Blk:Mulch", "Blk:Ca_Trt", "Residuals")
(expmsP <- PooledANOVA(expms, pool.ID))

#copy Fig. 7-pH, Ch 2: bar chart of pH means by Ca_Trt
#compact letter display - a quick way to get means se
cld.pHc<-cld(lsmeans(mod.pHc, "Ca_Trt", adjust="none"))
#adjust data for baseline of pH=4
cld.pHc$lsmean <- cld.pHc$lsmean - 4
#calculate LSD and hLSD for error bars
(s <- sigma(mod.pHc))
(dfe <- aov.pHc$Df.res[3])
grpn <- aggregate(pH~Ca_Trt +Loc, data=cdat, length)
(n <- grpn$pH[1])
```

```
(LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
#add hLSD to data frame with means
cld.pHc <- cbind(cld.pHc, hLSD)
#make the bar chart with error bars
bg.pHc <- ggplot(cld.pHc, aes(x=Ca_Trt, y=lsmean)) +
      geom_bar(stat="identity", fill="cyan")+
      geom_errorbar(aes(ymin=lsmean-hLSD,
        ymax=lsmean+hLSD), position=position_dodge(0.9),
        width=0.2) + ylab("pH")+
      scale_x_discrete()+ scale_y_continuous(breaks=0:4,
      labels=0:4+4)+  theme_bw()
bg.pHc

#Ca anova with blocks random: Loc= Central
mod.Cac <- lmer(Ca ~ Mulch*Ca_Trt + (1|Blk), data=cdat)
print(summary(mod.Cac))
(aov.Cac <- Anova(mod.Cac, type=3, test="F"))
(lsm.Cac <- lsmeans(mod.Cac, specs=c("Mulch", "Ca_Trt")))
#linear contrasts
contrast(lsm.Cac, list(
      "Gypsum-linear"   = c(-1,-1,0,0,1,1,0,0,0,0),
      "Gypsum-quad."    = c(1,1,-2,-2,1,1,0,0,0,0),
      "Lime-linear"    = c(-1,-1,0,0,0,0,0,0,1,1),
      "Lime-quad."     = c(1,1,0,0,0,0,-2,-2,1,1),
      "Lime vs. Gypsum"= c(0,0,1,1,1,1,-1,-1,-1,-1),
      "Interact: GLxM" = c(1,-1,0,0,-1,1,0,0,0,0),
      "Interact: GQxM" = c(-1,1,2,-2,-1,1,0,0,0,0),
      "Interact: LLxM" = c(1,-1,0,0,0,0,0,0,-1,1),
      "Interact: LQxM" = c(-1,1,0,0,0,0,2,-2,-1,1),
      "Interact:MxLvG" = c(0,0,-1,1,-1,1,1,-1,1,-1)))
# copy Fig. 7-pH, Ch 2: bar chart of pH means by Ca_Trt
#compact letter display - a quick way to get means
cld.Cac<-cld(lsmeans(mod.Cac, "Ca_Trt", Letters=LETTERS,
adjust="none"))
#ggplot to imitate Fig. 7Ca (left side) in Chapter 2
#calculate LSD and hLSD for error bars
(s <- sigma(mod.Cac))
(dfe <- aov.Cac$Df.res[3])
grpn <- aggregate(Ca~Ca_Trt +Loc, data=cdat, length)
(n <- grpn$Ca[1])
(LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
#add hLSD to data frame with means
cld.Cac <- cbind(cld.Cac, hLSD)
#make the bar chart with error bars
bg.Cac <- ggplot(cld.Cac, aes(x=Ca_Trt, y=lsmean)) +
      geom_bar(stat="identity", fill="gold4")+
      geom_errorbar(aes(ymin=lsmean-hLSD,
        ymax=lsmean+hLSD), position=position_dodge(0.9),
        width=0.2) + ylab("Ca, mg/kg")+ ylim(0,1500)+
      scale_x_discrete()+ theme_bw()
bg.Cac
```

```
#put Ca and pH bar graphs on same page and save
bg.both <- grid.arrange(grobs=list(bg.pHc, bg.Cac), ncol=2)
ggsave("Ch2-Fig7.png", bg.both, dpi=600)


#extract location = East data
edat <- subset(mdat, Loc=='East')
#recreate location factor to eliminate ghost levels
edat$Loc <- droplevels(edat$Loc)
#pH anova with blocks random: Loc= East
mod.pHe <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=edat)
print(summary(mod.pHe))
Anova(mod.pHe, type=3, test="F")
Dplote <- diagPlots(mod.pHe)
ggsave("Ch2-Fig6e.png", Dplote, dpi=600)

#extract location = West data
wdat <- subset(mdat, Loc=='West')
#recreate location factor to eliminate ghost levels
wdat$Loc <- droplevels(wdat$Loc)
#pH anova with blocks random: Loc= West
mod.pHw <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=wdat)
print(summary(mod.pHw))
Anova(mod.pHw, type=3, test="F")
Dplotw <- diagPlots(mod.pHw)
ggsave("Ch2-Fig6w.png", Dplotw, dpi=600)
```

Output 2.3 jb

```
-------------------------------------------------------------------------------------------
> source("c2-s3.r", echo=T)

> #activate additional packages needed
> options(scipen=7, contrasts=(c("contr.sum", "contr.poly")))
> library(lme4)         #lmer()
Loading required package: Matrix
> library(car)          #Anova()
> library(lsmeans)      #lsmeans() & contrasts()
Loading required package: estimability
> library(EMSaov)       #EMSanova()
> library(tidyverse)    #ggplot
-- Attaching packages ------------- tidyverse 1.2.1 --
v ggplot2 2.2.1     v purrr   0.2.4
v tibble  1.3.4     v dplyr   0.7.4
v tidyr   0.7.2     v stringr 1.2.0
v readr   1.1.1     v forcats 0.2.0
-- Conflicts ---------------- tidyverse_conflicts() --
x tidyr::expand() masks Matrix::expand()
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x dplyr::recode() masks car::recode()
x purrr::some()   masks car::some()
```

```
> library(gridExtra)    #put graphs on one page
Attaching package: 'gridExtra'
The following object is masked from 'package:dplyr':
    combine

> #data management
> #read in the variable names and data with read.csv function
> mdat <- read.csv("Statbean-data.csv")
> #attach as data frame
> attach(mdat)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #function for 2x2 panel of diagnostic plots: lmer model object
> #call with model object from lmer; returns a graphic object ....
[TRUNCATED]
> #extract location = Central data
> cdat <- subset(mdat, Loc=='Central')
> #recreate location factor to eliminate ghost levels
> cdat$Loc <- droplevels(cdat$Loc)

> #pH anova with blocks random: Loc= Central
> mod.pHc <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=cdat)
> print(summary(mod.pHc))
Linear mixed model fit by REML ['lmerMod']
Formula: pH ~ Mulch * Ca_Trt + (1 | Blk)
   Data: cdat
REML criterion at convergence: 29

Scaled residuals:
'    Min       1Q   Median       3Q      Max
-1.50924 -0.50465 -0.09063  0.54987  1.31203
```

The residuals are scaled differently than in Fig. 5 of Chapter 2, but this five-number summary shows no potential outliers and good symmetry.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk      (Intercept) 0.006969 0.08348
 Residual             0.068774 0.26225
Number of obs: 30, groups:  Blk, 3
```

The variance component estimates for Blocks and Residual given under the "Variance" column above match the SAS estimates in Table 11 of Chapter 2. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
> Anova(mod.pHc, type=3, test="F")
```

```
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: pH
Item                 F Df Df.res    Pr(>F)
(Intercept)  7226.7093  1      2 0.0001383 ***
Mulch           0.6522  1     18 0.4298777
Ca_Trt          4.3511  4     18 0.0123106 *
Mulch:Ca_Trt    0.9431  4     18 0.4617728
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Table 11 of Chapter 2.

```
> (lsm.pHc <- lsmeans(mod.pHc, specs=c("Mulch", "Ca_Trt")))
Loading required namespace: lmerTest
 Mulch Ca_Trt    lsmean        SE    df lower.CL upper.CL
 no    control 5.423333 0.1588955 18.58 5.090257 5.756410
 yes   control 5.683333 0.1588955 18.58 5.350257 6.016410
 no    g0730   5.546667 0.1588955 18.58 5.213590 5.879743
 yes   g0730   5.863333 0.1588955 18.58 5.530257 6.196410
 no    g1460   5.693333 0.1588955 18.58 5.360257 6.026410
 yes   g1460   5.506667 0.1588955 18.58 5.173590 5.839743
 no    l0730   5.963333 0.1588955 18.58 5.630257 6.296410
 yes   l0730   5.953333 0.1588955 18.58 5.620257 6.286410
 no    l1460   6.056667 0.1588955 18.58 5.723590 6.389743
 yes   l1460   6.063333 0.1588955 18.58 5.730257 6.396410

Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
```

The above least squares means and standard errors equal those in SAS shown in the supplementary material available for Chapter 2.

```
> #linear contrasts
> contrast(lsm.pHc, list(
+       "Gypsum-linear"   = c(-1,-1,0,0,1,1,0,0,0,0),
+       "Gypsum-quad."    = c(1,1,-2,-2,1,1,0,0,0,0),
+       "Li ..." ... [TRUNCATED]
 contrast           estimate        SE df t.ratio p.value
 Gypsum-linear    0.09333333 0.3028189 18   0.308  0.7615
 Gypsum-quad.    -0.51333333 0.5244976 18  -0.979  0.3407
 Lime-linear      1.01333333 0.3028189 18   3.346  0.0036
 Lime-quad.      -0.60666667 0.5244976 18  -1.157  0.2625
 Lime vs. Gypsum -1.42666667 0.4282505 18  -3.331  0.0037
 Interact: GLxM  -0.44666667 0.3028189 18  -1.475  0.1575
 Interact: GQxM  -0.56000000 0.5244976 18  -1.068  0.2998
 Interact: LLxM  -0.25333333 0.3028189 18  -0.837  0.4138
 Interact: LQxM   0.28666667 0.5244976 18   0.547  0.5914
 Interact:MxLvG   0.13333333 0.4282505 18   0.311  0.7591
```

The p.values for the above linear contrasts match those reported for SAS in the left side of Table 13 of Chapter 2. R uses t-ratios rather the F-Values SAS uses, but because the F Distribution is the square of Student's t, the probablity values above are identical.

```
> #request diagnostic plots and save them to file
> Dplotc <- diagPlots(mod.pHc)
> ggsave("Ch2-Fig6c.png", Dplotc, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 2.2 of this appendix for the panel of diagnostic plots, which are the equivalent of the top section of Fig. 6 in Chapter 2.

```
> #Expected MS for pH anova with blocks random: Loc= Central
> expms <- EMSanova(pH ~ Blk + Mulch + Ca_Trt + Mulch*Ca_Trt,
data=cdat, type=c("R", "F", .... [TRUNCATED]

> expms
`              Df          SS          MS  Fvalue Pvalue Sig
Blk             2 0.27692667 0.13846333  1.2354 0.3408
Mulch           1 0.04485333 0.04485333  0.8087 0.4634
Blk:Mulch       2 0.11092667 0.05546333  0.4948 0.6272
Ca_Trt          4 1.19698000 0.29924500 10.3932  0.003  **
Blk:Ca_Trt      8 0.23034000 0.02879250  0.2569 0.9641
Mulch:Ca_Trt    4 0.25944667 0.06486167  0.5787 0.6867
Residuals       8 0.89667333 0.11208417

`              EMS
Blk            Error+10Blk
Mulch          Error+5Blk:Mulch+15Mulch
Blk:Mulch      Error+5Blk:Mulch
Ca_Trt         Error+2Blk:Ca_Trt+6Ca_Trt
Blk:Ca_Trt     Error+2Blk:Ca_Trt
Mulch:Ca_Trt   Error+3Mulch:Ca_Trt
Residuals      Error

> pool.ID <- c("Blk:Mulch", "Blk:Ca_Trt", "Residuals")
> (expmsP <- PooledANOVA(expms, pool.ID))
Item          Df     SS     MS Fvalue Pvalue Sig EMS
Blk            2 0.2769 0.1385 2.0133 0.1625     Error+10Blk
Mulch          1 0.0449 0.0449 0.6522 0.4299     Error+15Mulch
Ca_Trt         4 1.1970 0.2992 4.3511 0.0123   * Error+6Ca_Trt
Mulch:Ca_Trt   4 0.2594 0.0649 0.9431 0.4618     Error+3Mulch:Ca_Trt
Residuals     18 1.2379 0.0688                   Error
```

Table 12 in Chapter 2 contains the SAS version of the Expected Mean Squares; the R Expected Mean Squares shown above are equivalent. Both show the Residual Error is the correct term to test all the other effects. Note the R function assumes all combinations of the terms in the model will be examined separately (see further above), thus it was necessary to tell it to pool the block interactions with the Residual Error.

```
> # copy Fig. 7-pH, Ch 2: bar chart of pH means by Ca_Trt
> #compact letter display - a quick way to get means
> cld.pHc<-cld(lsmeans(m .... [TRUNCATED]
NOTE: Results may be misleading due to involvement in interactions
```

```
> #adjust data for baseline of pH=4
> cld.pHc$lsmean <- cld.pHc$lsmean - 4
> #calculate LSD and hLSD for error bars
> (s <- sigma(mod.pHc))
[1] 0.2622488
> (dfe <- aov.pHc$Df.res[3])
[1] 18
> grpn <- aggregate(pH~Ca_Trt +Loc, data=cdat, length)
> (n <- grpn$pH[1])
[1] 6
> (LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
[1] 0.3180994
[1] 0.1590497
> #add hLSD to data frame with means
> cld.pHc <- cbind(cld.pHc, hLSD)

> #make the bar chart with error bars
> bg.pHc <- ggplot(cld.pHc, aes(x=Ca_Trt, y=lsmean)) +
+       geom_bar(stat="identity", fill="cyan")+
+       geom_error .... [TRUNCATED]
> bg.pHc
```

The resulting bar chart of the pH means with error bars is displayed in Fig. 2.3 in this appendix.

```
> #Ca anova with blocks random: Loc= Central
> mod.Cac <- lmer(Ca ~ Mulch*Ca_Trt + (1|Blk), data=cdat)
> print(summary(mod.Cac))
Linear mixed model fit by REML ['lmerMod']
Formula: Ca ~ Mulch * Ca_Trt + (1 | Blk)
   Data: cdat
REML criterion at convergence: 313

Scaled residuals:
`    Min      1Q  Median      3Q     Max
-1.5378 -0.4982 -0.1309  0.6082  1.7929

Random effects:
 Groups   Name        Variance Std.Dev.
 Blk      (Intercept)      0     0.0
 Residual             108573   329.5
Number of obs: 30, groups:  Blk, 3
```

The scaled residuals above appear symmetric with no evidence of potential outliers. The variance components do not match those in the SAS PROC MIXED output in the supplemental material because SAS estimated the variance component for blocks to be -6620. R does not have an option to allow them to be negative so it estimated the block variance component to be 0. This caused R's to estimate of the Residual Error to be less than SAS's (108573 versus 115193.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
> Anova(mod.Cac, type=3, test="F")
```

```
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: Ca
Item                F Df Df.res    Pr(>F)
(Intercept)  252.9209  1      2 0.003931 **
Mulch          1.1920  1     18 0.289328
Ca_Trt         2.9978  4     18 0.046462 *
Mulch:Ca_Trt   0.4090  4     18 0.799764
---------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Due to the nearly 10% difference in Residual Errors because of the negative/non-negative variance components described above, the F-tests in this ANOVA do not quite agree with those in the SAS printout in the supplemental materials. Both R and SAS conclude the Mulch and Mulch-by-Ca_Trt effects are not significant. However, SAS also concludes Ca_Trt is not quite significant at the 5% level (p=5.6%), but R finds it to be just significant (p=4.6%).

```
> (lsm.Cac <- lsmeans(mod.Cac, specs=c("Mulch", "Ca_Trt")))
 Mulch Ca_Trt     lsmean       SE    df lower.CL upper.CL
 no    control  534.0667 190.2393 19.75 136.9113  931.222
 yes   control  709.5733 190.2393 19.75 312.4180 1106.729
 no    g0730    707.7833 190.2393 19.75 310.6280 1104.939
 yes   g0730   1092.0367 190.2393 19.75 694.8813 1489.192
 no    g1460    956.7867 190.2393 19.75 559.6313 1353.942
 yes   g1460    870.8100 190.2393 19.75 473.6547 1267.965
 no    l0730   1060.7267 190.2393 19.75 663.5713 1457.882
 yes   l0730   1184.7333 190.2393 19.75 787.5780 1581.889
 no    l1460   1195.9200 190.2393 19.75 798.7647 1593.075
 yes   l1460   1254.9333 190.2393 19.75 857.7780 1652.089

Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
```

The above least squares means and SEs equal those in SAS shown in the supplementary material available for Chapter 2. It may seem surprising that the SEs are equal because the Residual Errors are not. Nevertheless, they are because the appropriate error part of the SE for a single treatment mean is the sum of the variance components for Blocks and Residual Error. For SAS this is (-6620 + 115193), or 108573. For R this is (0 + 108573), or 108573, thus, SEs are equal.

```
> #linear contrasts
> contrast(lsm.Cac, list(
+       "Gypsum-linear"   = c(-1,-1,0,0,1,1,0,0,0,0),
+       "Gypsum-quad."    = c(1,1,-2,-2,1,1,0,0,0,0),
+       "Li ..." ... [TRUNCATED]
 contrast           estimate       SE    df t.ratio p.value
 Gypsum-linear     583.95667 380.4785 19.75   1.535  0.1407
 Gypsum-quad.     -528.40333 659.0082 19.75  -0.802  0.4322
 Lime-linear      1207.21333 380.4785 19.75   3.173  0.0048
 Lime-quad.       -796.42667 659.0082 19.75  -1.209  0.2411
 Lime vs. Gypsum -1068.89667 538.0779 19.75  -1.987  0.0610
 Interact: GLxM   -261.48333 380.4785 19.75  -0.687  0.4999
 Interact: GQxM   -678.97667 659.0082 19.75  -1.030  0.3153
```

```
 Interact: LLxM   -116.49333 380.4785 19.75  -0.306  0.7627
 Interact: LQxM    -13.49333 659.0082 19.75  -0.020  0.9839
 Interact:MxLvG    115.25667 538.0779 19.75   0.214  0.8326
```

The p.values for the above linear contrasts do not quite match those reported for SAS in the right side of Table 13 of Chapter 2. This descrepacy is a product of the difference between SAS and R in allowing or not allowing negative variance components discussed above. In spite of this difference, the conclusions on which contrasts are significant are the same.

```
> #match Fig. 7-Ca in Ch 2: bar chart of Ca_Trt means
#compact letter display - a quick way to get means
> cld.Cac<-cld(lsmeans(mod. .... [TRUNCATED]
NOTE: Results may be misleading due to involvement in interactions
>
> #calculate LSD and hLSD for error bars
> (s <- sigma(mod.Cac))
[1] 329.5041
> (dfe <- aov.Cac$Df.res[3])
[1] 18
> grpn <- aggregate(Ca~Ca_Trt +Loc, data=cdat, length)
> (n <- grpn$Ca[1])
[1] 6
> (LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
[1] 399.6779
[1] 199.8389
> #add hLSD to data frame with means
> cld.Cac <- cbind(cld.Cac, hLSD)

> #make the bar chart with error bars
> bg.Cac <- ggplot(cld.Cac, aes(x=Ca_Trt, y=lsmean)) +
+       geom_bar(stat="identity", fill="gold4")+
+       geom_erro .... [TRUNCATED]
> bg.Cac

> #put Ca and pH bar graphs on same page and save
> bg.both <- grid.arrange(grobs=list(bg.pHc, bg.Cac), ncol=2)
> ggsave("Ch2-Fig7.png", bg.both, dpi=600)
Saving 5.76 x 5.75 in image
```

The resulting bar charts of the soil pH and Ca means with error bars are displayed in Fig. 2.3 in this appendix.

```
> #extract location = East data
> edat <- subset(mdat, Loc=='East')
> #recreate location factor to eliminate ghost levels
> edat$Loc <- droplevels(edat$Loc)
> #pH anova with blocks random: Loc= East
> mod.pHe <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=edat)
> print(summary(mod.pHe))
Linear mixed model fit by REML ['lmerMod']
Formula: pH ~ Mulch * Ca_Trt + (1 | Blk)
   Data: edat

REML criterion at convergence: 4.6
```

```
Scaled residuals:
‘    Min       1Q   Median       3Q      Max
-1.53963 -0.47877 -0.01521  0.66209  1.52286
```
The residuals above are scaled differently than in Fig. 6 of Chapter 2, but this five-number summary shows no potential outliers and good symmetry.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk      (Intercept) 0.04108  0.2027
 Residual             0.01569  0.1253
Number of obs: 30, groups:  Blk, 3
```

The variance component estimates for Blocks and Residual given under the “Variance” column above match the SAS estimates in the SAS PROC MIXED output in the supplemental materials. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
> Anova(mod.pHe, type=3, test="F")
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: pH
Item                 F Df Df.res     Pr(>F)
(Intercept)  1204.3780  1      2  0.0008293 ***
Mulch           0.0688  1     18  0.7960241
Ca_Trt         11.1982  4     18 0.00009698 ***
Mulch:Ca_Trt    1.1682  4     18  0.3576655
---------------------------------------------
Sig. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Table 18 of Chapter 2 and the SAS table in the supplemental material.

```
> Dplote <- diagPlots(mod.pHe)
> ggsave("Ch2-Fig6e.png", Dplote, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 2.4 of this appendix for the panel of diagnostic plots, which are the equivalent of the middle section of Fig. 6 in Chapter 2.

```
> #extract location = West data
> wdat <- subset(mdat, Loc=='West')
> #recreate location factor to eliminate ghost levels
> wdat$Loc <- droplevels(wdat$Loc)
> #pH anova with blocks random: Loc= West
> mod.pHw <- lmer(pH ~ Mulch*Ca_Trt + (1|Blk), data=wdat)
> print(summary(mod.pHw))
Linear mixed model fit by REML ['lmerMod']
Formula: pH ~ Mulch * Ca_Trt + (1 | Blk)
   Data: wdat
```

```
REML criterion at convergence: 24.6

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.0092 -0.4215  0.0176  0.6118  1.3576
```

The residuals above are scaled differently than in Fig. 6 of Chapter 2, but this five-number summary shows no potential outliers, but only fair symmetry.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk      (Intercept) 0.18155  0.4261
 Residual             0.04038  0.2009
Number of obs: 30, groups:  Blk, 3
```

The variance component estimates for Blocks and Residual given under the "Variance" column above match the SAS estimates in the SAS PROC MIXED output in the supplemental materials. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
> Anova(mod.pHw, type=3, test="F")
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: pH
Item                F Df Df.res   Pr(>F)
(Intercept)  738.0587  1      2 0.001352 **
Mulch          1.4167  1     18 0.249411
Ca_Trt         5.9330  4     18 0.003161 **
Mulch:Ca_Trt   0.4069  4     18 0.801281
---------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Table 18 of Chapter 2 and the SAS table in the supplemental material..

```
> Dplotw <- diagPlots(mod.pHw)
> ggsave("Ch2-Fig6w.png", Dplotw, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 2.5 of this appendix for the panel of diagnostic plots, which are the equivalent of the bottom section of Fig. 6 in Chapter 2.

==========

Tables 15, 16, and 19 (Chapter 2). Mixed ANOVA across three locations

The following R-script mimics the SAS mixed model analyses as summarized in Tables 15, 16, and 19 of Chapter 2 for pH, Ca, and Ptotal from the Statbean data combined across all three locations. It also reproduces the additional information in Figures 8 and 9. Similar to the situation for analyzing

the data by location, it is possible to automate the analyses across multiple dependent variables; however, when there are only a few it is easier to copy and paste the relevant functions and insert the added response variables by hand. The data from the supplemental materials were stored in a file named Statbean-data.csv.

Script 2.4 jb

```
#activate additional packages needed
options(scipen=7, contrasts=(c("contr.sum", "contr.poly")))
library(lme4)          #lmer()
library(car)           #Anova()
library(lsmeans)       #lsmeans() & contrasts()
library(EMSaov)        #EMSanova()
library(tidyverse)     #ggplot
library(gridExtra)     #put graphs on one page

#data management
#read in the variable names and data with read.csv function
mdat <- read.csv("Statbean-data.csv")
#attach as data frame
attach(mdat)
#check first & last 6 rows of data with head & tail functions
head(mdat)
tail(mdat)
#check 6 descriptive stats per variable with summary function
summary(mdat)

#statistical analyses
#function for 2x2 panel of diagnostic plots: lmer model object
#call with model object from lmer; returns a graphic object
diagPlots<-function(model){
      #predicted values & unscaled residuals in data frame
      predv <- predict(model)
      rres <- residuals(model, scaled=F)
      resdf <- data.frame(predv, rres)
      #plot 1: residuals versus predicted
            p1<-ggplot(model, aes(x=predv,
                  y=rres))+geom_point()+
            geom_hline(yintercept=0, linetype="dashed")+
            xlab("Predicted Values") +
            ylab("Residuals")+ theme_bw()

      #plot 2: histogram of residuals with normal dist.
      plot.new()
      p2a <- ggplot(resdf, aes(x=rres,
            y=(..density..)*0.15))+
            geom_histogram(binwidth=(0.15), fill="cyan",
            colour="black", center=0.225) +
            xlab("Residuals")+ ylab("Density") +  theme_bw()
      #construct data frame dfn for normal curve
      x <- seq(1.5*min(rres), 1.5*max(rres),
```

```
            length.out=100)
            dfn <- with(resdf, data.frame(x=x,
              y=dnorm(x, mean(rres), sd(rres))))
      p2 <-p2a + geom_line(data=dfn, aes(x=x, y=y*0.15),
            colour="red")

      #Q-Q plot as visual check on normality
      plot.new()
      p3<-ggplot(resdf)+ geom_qq(aes(sample=rres))+
            xlab("Normal Quantiles") +
            ylab("Residuals")+  theme_bw()

      #boxplot of residuals
      plot.new()
      p4<-ggplot(resdf, aes(x=1, y=rres))+
            geom_boxplot(width=0.1 )+
            scale_x_discrete(labels="")+ xlab("All
              Cases")+
            scale_y_continuous(limits=c(-0.5, 0.5))+
            ylab("Residuals") + theme_bw()

      #put plots into panel and return
      grid.arrange(p1, p2, p3, p4, ncol=2,
            top="Diagnostics")
      Dplot <- arrangeGrob(grobs=list(p1, p2, p3, p4),
            ncol=2, top="Diagnostics")
      return(Dplot)
      }

#pH anova with blocks random: all 3 locations
mod.pH <- lmer(pH ~ Loc*Mulch*Ca_Trt + (1|Blk:Loc), data=mdat)
print(summary(mod.pH))
(aov.pH <- Anova(mod.pH, type=3, test="F"))
(lsm.pH <- lsmeans(mod.pH, ~Ca_Trt|Loc|Mulch))
#request diagnostic plots and save them to file
Dplotc <- diagPlots(mod.pH)
ggsave("Ch2-Fig8.png", Dplotc, dpi=600)

#Expected MS for pH anova with blocks random
expms <- EMSanova(pH ~ Loc + Blk + Mulch + Ca_Trt,
      data=mdat, type=c("F", "R", "F", "F"))
pool.ID <- c("Blk:Mulch", "Loc:Blk:Mulch",
      "Blk:Ca_Trt",    "Loc:Blk:Ca_Trt",
      "Blk:Mulch:Ca_Trt", "Residuals")
(expmsP <- PooledANOVA(expms, pool.ID))

#mimic Fig. 9pH in Ch 2: bar chart-pH means by Loc & Ca_Trt
#compact letter display - a quick way to get means
cld.pH<-cld(lsmeans(mod.pH, ~Loc|Ca_Trt, adjust="none"))
#adjust data for baseline of pH=3
cld.pH$lsmean <- cld.pH$lsmean - 3
#calculate LSD and hLSD for error bars
```

```
(s <- sigma(mod.pH))
(dfe <- aov.pH$Df.res[5])
grpn <- aggregate(pH~Ca_Trt +Loc, data=mdat, length)
(n <- grpn$pH[1])
(LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
#add hLSD to data frame with means
cld.pH <- cbind(cld.pH, hLSD)
#make the bar chart with error bars
bg.pH <- ggplot(cld.pH, aes(x=Loc, y=lsmean, fill=Ca_Trt)) +
      geom_bar(position="dodge", stat="identity")+
      geom_errorbar(aes(ymin=lsmean-hLSD,
        ymax=lsmean+hLSD), position=position_dodge(0.9),
        width=0.2) + ylab("pH")+
      scale_x_discrete(limits=c("East", "Central",
        "West")) +
      scale_y_continuous(breaks=0:3, labels=0:3+3)+
      guides(fill=FALSE) + theme_bw()
bg.pH

#Ca anova with blocks random: all 3 locations
mod.Ca <- lmer(Ca ~ Loc*Mulch*Ca_Trt + (1|Blk:Loc), data=mdat)
print(summary(mod.Ca))
(aov.Ca <- Anova(mod.Ca, type=3, test="F"))
(lsm.Ca <- lsmeans(mod.Ca, ~Ca_Trt|Loc|Mulch))
#mimic Fig. 9Ca in Ch 2: bar chart-Ca means by Loc & Ca_Trt
#compact letter display - a quick way to get means
cld.Ca<-cld(lsmeans(mod.Ca, ~Loc|Ca_Trt, adjust="none"))
#calculate LSD and hLSD for error bars
(s <- sigma(mod.Ca))
(dfe <- aov.Ca$Df.res[5])
grpn <- aggregate(Ca~Ca_Trt +Loc, data=mdat, length)
(n <- grpn$Ca[1])
(LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
#add hLSD to data frame with means
cld.Ca <- cbind(cld.Ca, hLSD)
#make the bar chart with error bars
#ggplot to imitate Fig. 9Ca (left side) in Chapter 2
bg.Ca <- ggplot(cld.Ca, aes(x=Loc, y=lsmean, fill=Ca_Trt)) +
      geom_bar(position="dodge", stat="identity")+
      geom_errorbar(aes(ymin=lsmean-SE,
        ymax=lsmean+SE), position=position_dodge(0.9),
        width=0.2) + ylab("Ca, mg/kg")+
      scale_x_discrete(limits=c("East", "Central",
      "West"))+ theme_bw() +
      theme(legend.position=c(0.2,0.8))
bg.Ca

#put Ca and pH bar graphs on same page and save
bg.both <- grid.arrange(grobs=list(bg.pH, bg.Ca), ncol=2)
ggsave("Ch2-Fig9.png", bg.both, dpi=600)

#Ptotal anova with blocks random: all 3 locations
```

```
mod.Ptot <-lmer(Ptotal ~Loc*Mulch*Ca_Trt +(1|Blk:Loc), data=mdat)
print(summary(mod.Ptot))
Anova(mod.Ptot, type=3, test="F")
```

Output 2.4 jb

```
> source("c2-s4.r", echo=T)

> #activate additional packages needed
> options(scipen=7, contrasts=(c("contr.sum", "contr.poly")))
> library(lme4)          #lmer()
> library(car)           #Anova()
> library(lsmeans)               #lsmeans() & contrasts()
> library(EMSaov)                #EMSanova()
> library(tidyverse)     #ggplot
> library(gridExtra)     #put graphs on one page

> #data management
> #read in the variable names and data with read.csv function
> mdat <- read.csv("Statbean-data.csv")

> #attach as data frame
> attach(mdat)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #function for 2x2 panel of diagnostic plots: lmer model object
> #call with model object from lmer; returns a graphic object ....
[TRUNCATED]

> #pH anova with blocks random: all 3 locations
> mod.pH <- lmer(pH ~ Loc*Mulch*Ca_Trt + (1|Blk:Loc), data=mdat)
> print(summary(mod.pH))
Linear mixed model fit by REML ['lmerMod']
Formula: pH ~ Loc * Mulch * Ca_Trt + (1 | Blk:Loc)
   Data: mdat
REML criterion at convergence: 92.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.04683 -0.39720 -0.01177  0.49109  1.84889
```

The residuals above are scaled differently than in Fig. 8 of Chapter 2, but this five-number summary shows no potential outliers and good symmetry.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk:Loc  (Intercept) 0.07653  0.2766
 Residual             0.04161  0.2040
Number of obs: 90, groups:  Blk:Loc, 9
```

The variance component estimates for Blocks and Residual given under the "Variance" column above match the SAS estimates in the SAS PROC MIXED output in Tables 15 and 19 of Chapter 2 and the supplemental materials. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
Correlation matrix not shown by default, as p = 30 > 12.
Use print(summary(mod.pH), correlation=TRUE)  or
         vcov(summary(mod.pH))    if you need it

> (aov.pH <- Anova(mod.pH, type=3, test="F"))
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: pH
Item                       F  Df  Df.res         Pr(>F)
(Intercept)       3443.8841   1       6 0.000000001645 ***
Loc                 65.0836   2       6 0.000085553012 ***
Mulch                1.8750   1      54          0.1766
Ca_Trt              16.2266   4      54 0.000000008731 ***
Loc:Mulch            0.3017   2      54          0.7408
Loc:Ca_Trt           0.4716   8      54          0.8708
Mulch:Ca_Trt         1.4229   4      54          0.2389
Loc:Mulch:Ca_Trt     0.4855   8      54          0.8612
---------------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Tables 15, 16, and 19 of Chapter 2 and the SAS table in the supplemental material.

```
> (lsm.pH <- lsmeans(mod.pH, ~Ca_Trt|Loc|Mulch))
Loc = Central:
 Ca_Trt Mulch   lsmean        SE    df lower.CL upper.CL
 0      no    5.423333 0.1984495 12.56 4.993083 5.853584
 G1X    no    5.546667 0.1984495 12.56 5.116416 5.976917
 G2X    no    5.693333 0.1984495 12.56 5.263083 6.123584
 L1X    no    5.963333 0.1984495 12.56 5.533083 6.393584
 L2X    no    6.056667 0.1984495 12.56 5.626416 6.486917
 0      yes   5.683333 0.1984495 12.56 5.253083 6.113584
 G1X    yes   5.863333 0.1984495 12.56 5.433083 6.293584
 G2X    yes   5.506667 0.1984495 12.56 5.076416 5.936917
 L1X    yes   5.953333 0.1984495 12.56 5.523083 6.383584
 L2X    yes   6.063333 0.1984495 12.56 5.633083 6.493584

Loc = East:
 Ca_Trt Mulch   lsmean        SE    df lower.CL upper.CL
 0      no    4.013333 0.1984495 12.56 3.583083 4.443584
 G1X    no    4.033333 0.1984495 12.56 3.603083 4.463584
 G2X    no    4.016667 0.1984495 12.56 3.586416 4.446917
 L1X    no    4.156667 0.1984495 12.56 3.726416 4.586917
 L2X    no    4.440000 0.1984495 12.56 4.009750 4.870250
```

```
 0       yes    4.103333 0.1984495 12.56 3.673083 4.533584
 G1X     yes    4.030000 0.1984495 12.56 3.599750 4.460250
 G2X     yes    3.933333 0.1984495 12.56 3.503083 4.363584
 L1X     yes    4.313333 0.1984495 12.56 3.883083 4.743584
 L2X     yes    4.340000 0.1984495 12.56 3.909750 4.770250

Loc = West:
 Ca_Trt Mulch   lsmean        SE    df lower.CL upper.CL
 0       no     6.583333 0.1984495 12.56 6.153083 7.013584
 G1X     no     6.596667 0.1984495 12.56 6.166416 7.026917
 G2X     no     6.516667 0.1984495 12.56 6.086416 6.946917
 L1X     no     6.936667 0.1984495 12.56 6.506416 7.366917
 L2X     no     6.933333 0.1984495 12.56 6.503083 7.363584
 0       yes    6.806667 0.1984495 12.56 6.376416 7.236917
 G1X     yes    6.730000 0.1984495 12.56 6.299750 7.160250
 G2X     yes    6.496667 0.1984495 12.56 6.066416 6.926917
 L1X     yes    6.920000 0.1984495 12.56 6.489750 7.350250
 L2X     yes    7.050000 0.1984495 12.56 6.619750 7.480250

Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
```

The above least squares means and standard errors equal those in SAS shown in the supplementary material available for Chapter 2.

```
> #request diagnostic plots and save them to file
> Dplotc <- diagPlots(mod.pH)
> ggsave("Ch2-Fig8.png", Dplotc, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 2.6 of this appendix for the panel of diagnostic plots, which are the equivalent of Fig. 8 in Chapter 2.

```
> pool.ID <- c("Blk:Mulch", "Loc:Blk:Mulch", "Blk:Ca_Trt",
"Loc:Blk:Ca_Trt", "Blk:Mulch:Ca_Trt", "Residuals")
> expms <- EMSanova(pH ~ Loc + Blk + Mulch + Ca_Trt, data=mdat,
type=c("F", "R", "F", "F"))

> (expmsP <- PooledANOVA(expms, pool.ID))
Item             Df       SS      MS Fvalue    Pvalue Sig
Loc               2 105.0369 52.5185 66.688    0.0008 ***
Blk               2   1.6915  0.8458 20.3239  <0.0001 ***
Loc:Blk           4   3.1501  0.7875 18.9244  <0.0001 ***
Mulch             1   0.0780  0.0780  1.875    0.1766
Loc:Mulch         2   0.0251  0.0126  0.3017   0.7408
Ca_Trt            4   2.7010  0.6753 16.2266  <0.0001 ***
Loc:Ca_Trt        8   0.1570  0.0196  0.4716   0.8708
Mulch:Ca_Trt      4   0.2369  0.0592  1.4229   0.2389
Loc:Mulch:Ca_Trt  8   0.1616  0.0202  0.4855   0.8612
Residuals        54   2.2472  0.0416

Item               EMS
Loc                Error+10Loc:Blk+30Loc
Blk                Error+30Blk
```

```
Loc:Blk             Error+10Loc:Blk
Mulch               Error+45Mulch
Loc:Mulch           Error+15Loc:Mulch
Ca_Trt              Error+18Ca_Trt
Loc:Ca_Trt          Error+6Loc:Ca_Trt
Mulch:Ca_Trt        Error+9Mulch:Ca_Trt
Loc:Mulch:Ca_Trt    Error+3Loc:Mulch:Ca_Trt
Residuals           Error
```

The Df, SS, MS, Fvalues, Pvalues, and expected mean squares (EMS) in the above ANOVA tabale match those in Table 16 of Chapter, except the R function did not allow a second pooling to combine Blk and Blk:Loc.

```
> #mimic Fig. 9pH in Ch 2: bar chart-pH means by Loc & Ca_Trt
> #compact letter display - a quick way to get means
> cld.pH<-cld(lsmeans(mod.pH .... [TRUNCATED]
NOTE: Results may be misleading due to involvement in interactions
> #adjust data for baseline of pH=3
> cld.pH$lsmean <- cld.pH$lsmean - 3
> #calculate LSD and hLSD for error bars
> (s <- sigma(mod.pH))
[1] 0.2039959
> (dfe <- aov.pH$Df.res[5])
[1] 54
> grpn <- aggregate(pH~Ca_Trt +Loc, data=mdat, length)
> (n <- grpn$pH[1])
[1] 6
> (LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
[1] 0.2361288
[1] 0.1180644
> #add hLSD to data frame with means
> cld.pH <- cbind(cld.pH, hLSD)

> #make the bar chart with error bars
> ggplot(cld.pH, aes(x=Loc, y=lsmean, fill=Ca_Trt)) +
+       geom_bar(position="dodge", stat="identity")+
+       geom_e .... [TRUNCATED]
```

See Fig. 2.7 of this appendix for the bar graph comparing the location-by-calcium treatment mean pH values, which is the equivalent of Fig. 9 in Chapter 2.

```
> #Ca anova with blocks random: all 3 locations
> mod.Ca <- lmer(Ca ~ Loc*Mulch*Ca_Trt + (1|Blk:Loc), data=mdat)
> print(summary(mod.Ca))
Linear mixed model fit by REML ['lmerMod']
Formula: Ca ~ Loc * Mulch * Ca_Trt + (1 | Blk:Loc)
   Data: mdat
REML criterion at convergence: 959.3

Scaled residuals:
`    Min       1Q   Median       3Q      Max
-1.70403 -0.45637  0.00976  0.37846  2.19976
```

The five-number summary of the scaled residuals above do not show any potential outliers, but do show reasonably good symmetry.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk:Loc  (Intercept) 51679    227.3
 Residual             86858    294.7
Number of obs: 90, groups:  Blk:Loc, 9
```

The variance component estimates for Blocks and Residual given under the "Variance" column above match the SAS estimates in the SAS PROC MIXED output in Table 19 of Chapter 2. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
Correlation matrix not shown by default, as p = 30 > 12.
Use print(summary(mod.Ca), correlation=TRUE)  or
         vcov(summary(mod.Ca))    if you need it

> (aov.Ca <- Anova(mod.Ca, type=3, test="F"))
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: Ca
Item                    F Df Df.res      Pr(>F)
(Intercept)      180.6665  1      6 0.00001051 ***
Loc               50.2011  2      6  0.0001793 ***
Mulch              0.4506  1     54  0.5049098
Ca_Trt             6.5171  4     54  0.0002355 ***
Loc:Mulch          0.6521  2     54  0.5250144
Loc:Ca_Trt         1.1566  8     54  0.3421309
Mulch:Ca_Trt       0.0329  4     54  0.9978586
Loc:Mulch:Ca_Trt   0.5037  8     54  0.8482302
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Table 19 of Chapter 2.

```
> (lsm.Ca <- lsmeans(mod.Ca, ~Ca_Trt|Loc|Mulch))
Loc = Central:
 Ca_Trt Mulch     lsmean       SE    df   lower.CL  upper.CL
 0      no     534.06667 214.8925 26.69   92.90721  975.2261
 G1X    no     707.78333 214.8925 26.69  266.62387 1148.9428
 G2X    no     956.78667 214.8925 26.69  515.62721 1397.9461
 L1X    no    1060.72667 214.8925 26.69  619.56721 1501.8861
 L2X    no    1195.92000 214.8925 26.69  754.76054 1637.0795
 0      yes    709.57333 214.8925 26.69  268.41387 1150.7328
Loc = West:
 L2X    no    2454.53000 214.8925 26.69 2013.37054 2895.6895
 0      yes   1996.03333 214.8925 26.69 1554.87387 2437.1928
 G1X    yes   1874.30667 214.8925 26.69 1433.14721 2315.4661
 G2X    yes   1948.04333 214.8925 26.69 1506.88387 2389.2028
```

```
 L1X    yes   2300.55333 214.8925 26.69 1859.39387 2741.7128
 L2X    yes   2626.21333 214.8925 26.69 2185.05387 3067.3728
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
```

Only the first and last six means are shown above.

```
> #mimic Fig. 9Ca in Ch 2: bar chart-Ca means by Loc & Ca_Trt
> #compact letter display - a quick way to get means
> cld.Ca<-cld(lsmeans(mod.Ca .... [TRUNCATED]
NOTE: Results may be misleading due to involvement in interactions
> #calculate LSD and hLSD for error bars
> (s <- sigma(mod.Ca))
[1] 294.7161
> (dfe <- aov.Ca$Df.res[5])
[1] 54
> grpn <- aggregate(Ca~Ca_Trt +Loc, data=mdat, length)
> (n <- grpn$Ca[1])
[1] 6
> (LSD <- abs(qt(0.025, dfe)*sqrt(2/n)*s)); (hLSD<-LSD/2)
[1] 341.1391
[1] 170.5696
> #add hLSD to data frame with means
> cld.Ca <- cbind(cld.Ca, hLSD)

> #make the bar chart with error bars
> #ggplot to imitate Fig. 9Ca (left side) in Chapter 2
> ggplot(cld.Ca, aes(x=Loc, y=lsmean, fill=Ca_Trt)) +
+  .... [TRUNCATED]
> bg.Ca

> #put Ca and pH bar graphs on same page and save
> bg.both <- grid.arrange(grobs=list(bg.pH, bg.Ca), ncol=2)
> ggsave("Ch2-Fig9.png", bg.both, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 2.7a of this appendix for the bar graph comparing the location-by-calcium treatment mean Ca values, which is the equivalent of Fig. 9 in Chapter 2.

```
> #Ptotal anova with blocks random: all 3 locations
> mod.Ptot <-lmer(Ptotal ~Loc*Mulch*Ca_Trt +(1|Blk:Loc), data=mdat)
> print(summary(mod.Ptot))
Linear mixed model fit by REML ['lmerMod']
Formula: Ptotal ~ Loc * Mulch * Ca_Trt + (1 | Blk:Loc)
   Data: mdat

REML criterion at convergence: 533.1

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.62531 -0.52638  0.07839  0.47593  2.08547
```

The five-number summary of the scaled residuals above do not show any potential outliers, but there is some chance of skewness.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk:Loc  (Intercept) 102.80   10.139
 Residual              65.49    8.093
Number of obs: 90, groups:  Blk:Loc, 9
```

The variance component estimates for Blocks and Residual given under the "Variance" column above match the SAS estimates in the SAS PROC MIXED output in Table 19 of Chapter 2. The Standard Deviations listed in these R tables are not the standard error of the estimate, but merely the square root of the variance component estimate.

The estimated fixed effect coefficients and their correlations have been omitted here and elsewhere to save space.

```
Correlation matrix not shown by default, as p = 30 > 12.
Use print(summary(mod.Ptot), correlation=TRUE)  or
         vcov(summary(mod.Ptot))         if you need it

> Anova(mod.Ptot, type=3, test="F")
Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)

Response: Ptotal
Item                    F Df Df.res      Pr(>F)
(Intercept)       24.7970  1      6   0.002503 **
Loc                3.8985  2      6   0.082244 .
Mulch              3.7910  1     54   0.056737 .
Ca_Trt             0.8687  4     54   0.488749
Loc:Mulch         12.0978  2     54 0.00004557 ***
Loc:Ca_Trt         1.3514  8     54   0.239000
Mulch:Ca_Trt       0.2970  4     54   0.878640
Loc:Mulch:Ca_Trt   0.3801  8     54   0.926651
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numerator and denominator df, F, and Pr(>F) in the ANOVA above equal those from SAS as shown in Table 19 of Chapter 2.

Chapter 3. Blocking Principles for Biological Experiments

Michael D. Casler

Box 1 (Chapter 3). Linear mixed model ANOVA from an augmented design.

This example illustrates the key feature of an augmented design; that is, using a few check treatments to estimate the error variance to compare a number of unreplicated treatments. By considering the unreplicated treatments to be nested within the replicated checks, the analysis can be performed in one ANOVA. The data lines for this example are in the wide (unstacked) format without column names. Commas were inserted for blank spaces to create a data file (Ch3Ex1dat.csv). This script enters the data into R with the read.csv() function because there are only a few columns; it then manually stacks the data. There are functions in R specifically for analyzing augmented design data, but the following R script uses the more generic linear mixed model analysis function, lmer(), because it has been more completely vetted and it allows more flexibility in considering factors fixed or random. Due to the severe imbalance in the design, R generates a large number of warning messages; the redundancies have been deleted to conserve space. See Example 1 of Chapter 13 in the text and this appendix for another illustration of an augmented design.

Script 3.1 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)
library(lsmeans)
library(dplyr)
library(plyr)

#data management
#read in the variable names and data with read.csv function
dat1 <- read.csv("Ch3Ex1dat.csv", na.strings=".")
#attach as data frame
attach(dat1)
#construct vectors w n=66 from n=22 data
enum <- c(entryno, entryno, entryno)
entry <- as.character(entry)
variety <- c(entry, entry, entry)
yield <- c(y1, y2, y3); yield <- as.numeric(yield)
b1 <- c(rep(1, times=22));  b2 <- c(rep(2, times=22))
b3 <- c(rep(3, times=22));  blk <- c(b1, b2, b3)
x <- entryno; x <- ifelse(x<5, 0,x)
c <- entryno; c <- ifelse(c>4, 0,c)
#set up factor (class) variables from numeric
x <- factor(x) ; c <- factor(c)
blk <- factor(blk)
#construct data frame from vectors
dat <- data.frame(enum, blk, variety, yield,x,c)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
```

```
summary(dat)

#statistical analyses
#analyze checks and experimental within checks, blocks random
m2 <- lmer(yield ~ c + x:c  + (1|blk), data=dat)
print(summary(m2, ddf="Satterthwaite"))
anova(m2, type=3,TEST="F", ddf="Satterthwaite")
tbla <- cld(lsmeans(m2, ~x|c))
#remove the many cases of NA due to missing cells
tbla <- na.omit(tbla)
#use only relevent columns and resort by x and c
tbla <- select(tbla, x, c, lsmean, SE, df)
tbla <- arrange(tbla, c, x)
print(tbla)
```

Output 3.1 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
> library(lsmeans)
> library(dplyr)
> library(plyr)

> #data management
> #read in the variable names and data with read.csv function
> dat1 <- read.csv("Ch3Ex1dat.csv", na.strings=".")
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #analyze checks and experimental within checks, blocks random
> m2 <- lmer(yield ~ c + x:c  + (1|blk), data=dat)
fixed-effect model matrix is rank deficient so dropping 69 columns /
coefficients

> print(summary(m2, ddf="Satterthwaite"))
fixed-effect model matrix is rank deficient so dropping 69 columns /
coefficients
Linear mixed model fit by REML t-tests use Satterthwaite
approximations to
  degrees of freedom [lmerMod]
Formula: yield ~ c + x:c + (1 | blk)
   Data: dat

REML criterion at convergence: 130.5

Scaled residuals:
   Min     1Q Median     3Q    Max
-1.641  0.000  0.000  0.000  1.571

Random effects:
```

```
 Groups   Name        Variance  Std.Dev.
 blk      (Intercept) 2.856e-12 1.690e-06
 Residual             6.886e+04 2.624e+02
Number of obs: 30, groups:  blk, 3
```

These variance components match those in SAS output.

```
> anova(m2, type=3,TEST="F", ddf="Satterthwaite")
fixed-effect model matrix is rank deficient so dropping 69 columns /
coefficients
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom
Term Sum Sq Mean Sq NumDF  DenDF F.value   Pr(>F)
c    994413  248603     4 9.0023  3.6105 0.050771 .
c:x 7164350  447772    16 9.0023  6.5030 0.003679 **
---------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Mean Squares, Numerator DF, and F values in the above ANOVA are the same as in the SAS output, but the Denominator (residual) DF and resulting Pr(>F) are not. This is likely because SAS used a containment method to get the denominator df in this unbalanced design. R does not have that option as far as I can tell, so I used the Satterthwaite option, which is a little more liberal (9 versus 7 denominator df). R does have the Kenward-Rodger option of determining the df, which is about equal to the containment method for the F-test of the check cultivars (7.2 versus 7.0 denominator df) and a little more conservative for the test of the unreplicated, experimental cultivars (4.0 versus 7.0 denominator df).

```
> tbla <- cld(lsmeans(m2, ~x|c))
fixed-effect model matrix is rank deficient so dropping 69 columns /
coefficients

> #remove the many cases of NA due to missing cells
> tbla <- na.omit(tbla)

> #use only relevent columns and resort by x and c
> tbla <- select(tbla, x, c, lsmean, SE, df)

> tbla <- arrange(tbla, c, x)

> print(tbla)
```

The following lsmeans and SEs match those in the SAS output, but the the df of do not –see above explanation.

```
c = 0:
 x    lsmean       SE df
 5  2169.000 262.4051  9
 6  3250.000 262.4051  9
 7  3807.000 262.4051  9
 8  4068.000 262.4051  9
 9  3871.000 262.4051  9
 10 3838.000 262.4051  9
 11 4244.000 262.4051  9
 12 3290.000 262.4051  9
 13 3019.000 262.4051  9
 14 3506.000 262.4051  9
 15 4384.000 262.4051  9
```

```
 16 4148.000 262.4051  9
 17 4167.000 262.4051  9
 18 4023.000 262.4051  9
 19 2435.000 262.4051  9
 20 4595.000 262.4051  9
 21 3957.000 262.4051  9

c = 1:
 x    lsmean       SE df
 0  4098.250 131.2026  9

c = 2:
 x    lsmean       SE df
 0  4001.667 151.4997  9

c = 3:
 x    lsmean       SE df
 0  4143.000 151.4997  9

c = 4:
 x    lsmean       SE df
 0  3799.667 151.4997  9

Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
P value adjustment: tukey method for comparing a family of 18
estimates
significance level used: alpha = 0.05
Warning message:
In createDesignMat(rho) :
  missing cells for some factors (combinations of factors)
 care must be taken with type III  hypothesis
```

==========

Box 2 (Chapter 3). Likelihood ratio test to help decide whether or not to retain random effects.

The data come from a 3 × 14, two-factor factorial in a split-plot arrangement with whole plots in a Randomized Complete Block Design with four blocks of one replication per block. Fourteen Italian ryegrass cultivars (subplots) were planted at three seeding rates (whole plots) and the percent survivorship determined on each of the 168 plots. The data come in two wide (unstacked) format files without column headings. I made two text files from the data lines for "data a" and "data b" in Box 2 and named them Ch3Ex2a.txt and Ch3Ex2b.txt. The first is the field map which contains the row number, block (aka rep), seeding rate, and 7 columns of cultivar numbers. The second contains the row number and 7 columns of survivorship (gc95). Similar to the SAS code, this R script assigns the column names after reading in the data, stacks the two tables into the long format (required by most statistic functions), then sorts and merges them by row and column. This script uses the generalized linear mixed model function, lmer(), to duplicate the SAS results. Because the rand() function calculates the likelihood ratio test directly from the full model, it was not necessary to run the reduced model explicitly.

Script 3.2 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)
library(lsmeans)
library(tidyr)
library(dplyr)

#data management
#read in the a data with read.csv function
a <- read.csv("Ch3Ex2a.txt", header=F)
#specify column names & change rowwise data to column vectors
colnames(a) <- c("row", "rep", "rate", "x1", "x2", "x3", "x4",
"x5", "x6", "x7")
a$row <- factor(a$row); a$rep <- factor(a$rep); a$rate <-
factor(a$rate)
aa <- gather(a, col, cultivar, x1:x7)
aa$cultivar <- factor(aa$cultivar)
#read in the b data with read.csv function
b <- read.csv("Ch3Ex2b.txt", header=F)
#specify column names & change rowwise data to column vectors
colnames(b) <- c("row", "x1", "x2", "x3", "x4", "x5", "x6", "x7")
b$row <- factor(b$row)
bb <- gather(b, col, gc95, x1:x7)
#sort data sets and merge them
aa <- arrange(aa, row, col); bb <- arrange(bb, row, col)
c <- merge(aa, bb)
attach(c); on.exit(detach)
#check first & last 6 rows of data with head & tail functions
head(c); tail(c)
#check stats for each variable
summary(c)

#statistical analyses
#analyze full model
mfull <- lmer(gc95 ~ rate*cultivar  + (1|rep)+ (1|rep:rate),
data=c)
print(summary(mfull, ddf="Satterthwaite"))
anova(mfull, type=3,TEST="F", ddf="Satterthwaite")
#run likelihood ratio test for random factors
rand(mfull)
#print lsmeans for cultivar
(lsmeans(mfull, "cultivar"))
```

Output 3.2 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
> library(lsmeans)
> library(tidyr)
```

```
> library(dplyr)

> #data management
> #read in the a data with read.csv function
> a <- read.csv("Ch3Ex2a.txt", header=F)

> #specify column names & change rowwise data to column vectors
> colnames(a) <- c("row", "rep", "rate", "x1", "x2", "x3", "x4", "x5",
"x6", "x7")

> a$row <- factor(a$row); a$rep <- factor(a$rep); a$rate <-
factor(a$rate)

> aa <- gather(a, col, cultivar, x1:x7)

> aa$cultivar <- factor(aa$cultivar)

> #read in the b data with read.csv function
> b <- read.csv("Ch3Ex2b.txt", header=F)

> #specify column names & change rowwise data to column vectors
> colnames(b) <- c("row", "x1", "x2", "x3", "x4", "x5", "x6", "x7")

> b$row <- factor(b$row)

> bb <- gather(b, col, gc95, x1:x7)

> #sort data sets and merge them
> aa <- arrange(aa, row, col); bb <- arrange(bb, row, col)
> c <- merge(aa, bb)
> attach(c); on.exit(detach)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #analyze full model
> mfull <- lmer(gc95 ~ rate*cultivar  + (1|rep)+ (1|rep:rate), data=c)

> print(summary(mfull, ddf="Satterthwaite"))
Linear mixed model fit by REML t-tests use Satterthwaite
approximations to
  degrees of freedom [lmerMod]
Formula: gc95 ~ rate * cultivar + (1 | rep) + (1 | rep:rate)
   Data: c
```

The residual loglikelihood value matches the one from SAS.

```
REML criterion at convergence: 1011.5

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.79362 -0.27805  0.00465  0.20804  2.84368

Random effects:
 Groups   Name        Variance Std.Dev.
 rep:rate (Intercept)   4.393   2.096
```

These variance components equal

```
 rep      (Intercept)   2.200   1.483
 Residual             108.387  10.411
Number of obs: 168, groups:  rep:rate, 12; rep, 4

> anova(mfull, type=3,TEST="F", ddf="Satterthwaite")
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom
term          Sum Sq Mean Sq NumDF DenDF Fvalue   Pr(>F)
rate            2933  1466.5     2     6 13.530 0.005978**
cultivar       41142  3164.7    13   117 29.199 <2.2e-16***
rate:cultivar   5524   212.4    26   117  1.960 0.008159 **
---------------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

those in SAS output.

The tests of the fixed effects in the above ANOVA are the same as in SAS – first ANOVA in Box 2 of Chapter 3.

```
> #run likelihood ratio test for random factors
> rand(mfull)
Analysis of Random effects Table:
term     Chi.sq Chi.DF p.value
rep       0.197      1     0.7
rep:rate  0.662      1     0.4
```

The likelihood ratio test of rep:rate is the same as calculated at the end of Box 2 in Chapter 3.

```
> #print lsmeans for cultivar
> (lsmeans(mfull, "cultivar"))
NOTE: Results may be misleading due to involvement in interactions
```

The least square means and SEs for cultivars below equal those from SAS in the first ANOVA in Box 2 of Chapter 3, but require careful use due to significant interaction.

```
 cultivar    lsmean       SE df   lower.CL upper.CL
 1         37.083333 3.154107 70 30.7926710 43.37400
 2         50.416667 3.154107 70 44.1260044 56.70733
 3          5.000000 3.154107 70 -1.2906623 11.29066
 4         31.666667 3.154107 70 25.3760044 37.95733
 5          6.666667 3.154107 70  0.3760044 12.95733
 6          5.833333 3.154107 70 -0.4573290 12.12400
 7         12.500000 3.154107 70  6.2093377 18.79066
 8         41.250000 3.154107 70 34.9593377 47.54066
 9          5.833333 3.154107 70 -0.4573290 12.12400
 10         5.000000 3.154107 70 -1.2906623 11.29066
 11         6.666667 3.154107 70  0.3760044 12.95733
 12         5.000000 3.154107 70 -1.2906623 11.29066
 13         6.250000 3.154107 70 -0.0406623 12.54066
 14        25.833333 3.154107 70 19.5426710 32.12400

Results are averaged over the levels of: rate
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
>
```

==========

Box 3 (Chapter 3). Arranging blocks and blocking patterns for future experiments.

This example combines the residuals from a previous experiment in various row-column combinations to determine the best blocking pattern for a future experiment. I made a data file, Ch3Ex3.txt, from the data lines for "data a" in Box 3 of Chapter 3. The file is in the wide format without headers so the R script adds the column names after reading in the data. The first column contains the row numbers and each of next six columns corresponds to a column in the field so the R script stacks the data into the long format (required by most statistic functions). After stacking the data, the SAS code uses modular arithmetic to generate blocks with 1, 2, 3, 4, or 6 rows per block and with 1, 2, 3, or 6 columns per block. It starts the calculations with the within block residual mean squares for a Completely Randomized Design (the whole study is essentially one large block), which is basically the mean squares of the residuals from the original study. Then it calculates the mean square within blocks for the 22 block dimensions created with the modular arithmetic of rows and columns. Finally, the combination(s) with minimal variation within blocks is identified graphically. The following R script tracks these same steps.

Script 3.3 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)
library(tidyr)
library(dplyr)
library(ggplot2)

#data management
#read in the a data with read.csv function
a <- read.csv("Ch3Ex3.txt", header=F)
#specify column names & change rowwise data to column vectors
colnames(a) <- c("row", "y1", "y2", "y3", "y4", "y5", "y6")
a$row <- factor(a$row)
b <- gather(a, col, yield, y1:y6)
#create group variables for row and make them factors
b$row <-as.numeric(as.character(b$row))
b$r1 <- factor(b$row)
b$r2 <- factor(floor((b$row+1)/2))
b$r3 <- factor(floor((b$row+2)/3))
b$r4 <- factor(floor((b$row+3)/4))
b$r6 <- factor(floor((b$row+5)/6))
#create group variables for columns and make them factors
b$col <- substring(b$col, 2, 2)
b$col <- as.numeric(as.character(b$col))
b$c1 <- factor(b$col)
b$c2 <- factor(floor((b$col+1)/2))
b$c3 <- factor(floor((b$col+2)/3))
#check first & last 6 rows of data with head & tail functions
head(b); tail(b)
#check stats for each variable
summary(b)
```

```
#statistical analyses
#analyze CRD model, rows/block=12 & cols/blk=6 & print results
mod <- lm(yield ~ r1:c1 , data=b)
summary(mod)
(crd <- anova(mod))
mscrd <- crd$`Mean Sq`[1]
#create data frame to store results
rcres <- data.frame(rows=numeric(23), cols=numeric(23),
residual= numeric(23))
#analyze r2-c1 combination and print results
mod <- lm(yield ~ r2:c1, data=b)
summary(mod)
anova(mod)
msresid <- deviance(mod)/df.residual(mod)
#store results for 2 rows/blk - 1 col/blk
rcres[1,] <- c(2, 1, msresid)
#repeat for all combinations - no print
mod <- lm(yield ~ r3:c1, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[2,] <- c(3, 1, msresid)
mod <- lm(yield ~ r4:c1, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[3,] <- c(4, 1, msresid)
mod <- lm(yield ~ r6:c1, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[4,] <- c(6, 1, msresid)
mod <- lm(yield ~ c1, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[5,] <- c(12, 1, msresid)
mod <- lm(yield ~ r1:c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[6,] <- c(1, 2, msresid)
mod <- lm(yield ~ r2:c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[7,] <- c(2, 2, msresid)
mod <- lm(yield ~ r3:c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[8,] <- c(3, 2, msresid)
mod <- lm(yield ~ r4:c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[9,] <- c(4, 2, msresid)
mod <- lm(yield ~ r6:c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[10,] <- c(6, 2, msresid)
mod <- lm(yield ~ c2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[11,] <- c(12, 2, msresid)
mod <- lm(yield ~ r1:c3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[12,] <- c(1, 3, msresid)
mod <- lm(yield ~ r2:c3, data=b)
```

```
msresid <- deviance(mod)/df.residual(mod)
rcres[13,] <- c(2, 3, msresid)
mod <- lm(yield ~ r3:c3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[14,] <- c(3, 3, msresid)
mod <- lm(yield ~ r4:c3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[15,] <- c(4, 3, msresid)
mod <- lm(yield ~ r6:c3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[16,] <- c(6, 3, msresid)
mod <- lm(yield ~ c3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[17,] <- c(12, 3, msresid)
mod <- lm(yield ~ r1, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[18,] <- c(1, 6, msresid)
mod <- lm(yield ~ r2, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[19,] <- c(2, 6, msresid)
mod <- lm(yield ~ r3, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[20,] <- c(3, 6, msresid)
mod <- lm(yield ~ r4, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[21,] <- c(4, 6, msresid)
mod <- lm(yield ~ r6, data=b)
msresid <- deviance(mod)/df.residual(mod)
rcres[22,] <- c(6, 6, msresid)
#add mean square for r1*c1 from completely randomized design
rcres[23,] <- c(12, 6, mscrd)
print(rcres)
#plot the data to see minimum msresidual
rcres <- arrange(rcres, cols, rows)
rcres$cols <- factor(rcres$cols)
ggplot(rcres, aes(x=rows, y=residual, colour=cols, group=cols))+
   geom_point(size=2)+
   geom_smooth(method="lm", formula=y~exp(1/x), se=F, size=2)+
   labs(x="Number of rows in each block", y="Expected residual
variance")+
   scale_colour_hue(name="Columns in each block")+
   theme_bw()+
   theme(legend.justification=c(0.95,0.05),
   legend.position=c(0.95,0.05), legend.direction="horizontal",
   panel.background=element_blank(),
   text=element_text(size=12))
ggsave("C3-Ex3plot.png", dpi=600, height=7, width=10.5, units="cm")
```

Output 3.3 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
> library(tidyr)
> library(dplyr)
> library(ggplot2)

> #data management
> #read in the a data with read.csv function
> a <- read.csv("Ch3Ex3.txt", header=F)
```

Checks on input data from head(), tail(), and summary() omitted

```
>
> #statistical analyses
> #analyze CRD model, rows/block=12 & cols/blk=6 & print results
> mod <- lm(yield ~ r1:c1 , data=b)

> summary(mod)

Call:
lm(formula = yield ~ r1:c1, data = b)

Residuals:
ALL 72 residuals are 0: no residual degrees of freedom!

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:    NaN
F-statistic:   NaN on 71 and 0 DF,  p-value: NA

> (crd <- anova(mod))
```

The residual sum of squares (0) and the ANOVA values for r1:c1 below match those from PROC GLM in SAS for the Completely Randomized Design results in Box 3 of Chapter 3.

```
Analysis of Variance Table
Response: yield
term       Df Sum Sq Mean Sq F value Pr(>F)
r1:c1      71 612.75  8.6302
Residuals   0   0.00

> mscrd <- crd$`Mean Sq`[1]

> #create data frame to store results
> rcres <- data.frame(rows=numeric(23), cols=numeric(23),
+ residual= numeric(23))

> #analyze r2-c1 combination and print results
> mod <- lm(yield ~ r2:c1, data=b)

> summary(mod)

Call:
lm(formula = yield ~ r2:c1, data = b)

Residuals:
```

```
`  Min      1Q Median      3Q     Max
-4.610 -1.183  0.000  1.183  4.610


Residual standard error: 3.005 on 36 degrees of freedom
Multiple R-squared:  0.4694,    Adjusted R-squared:  -0.04647
F-statistic: 0.9099 on 35 and 36 DF,  p-value: 0.6093

> anova(mod)
Analysis of Variance Table
```

Mean square residual for RCBD with 2 rows and 1 column per block in ANOVA below equals that from SAS in Box 3 of Chapter 3.

```
Response: yield
term       Df Sum Sq Mean Sq F value Pr(>F)
r2:c1      35 287.62  8.2177  0.9099 0.6093
Residuals 36 325.13  9.0313

> msresid <- deviance(mod)/df.residual(mod)
> #store results for 2 rows/blk - 1 col/blk
> rcres[1,] <- c(2, 1, msresid)

> #repeat for all combinations - no print
> mod <- lm(yield ~ r3:c1, data=b)
> msresid <- deviance(mod)/df.residual(mod)
> rcres[2,] <- c(3, 1, msresid)
> mod <- lm(yield ~ r4:c1, data=b)
> msresid <- deviance(mod)/df.residual(mod)
> rcres[3,] <- c(4, 1, msresid)
> mod <- lm(yield ~ r6:c1, data=b)
> msresid <- deviance(mod)/df.residual(mod)
> rcres[4,] <- c(6, 1, msresid)
> mod <- lm(yield ~ c1, data=b)
> msresid <- deviance(mod)/df.residual(mod)
> rcres[5,] <- c(12, 1, msresid)
```

Many redundant echoes of the R-script like those above were deleted.

```
> #add mean square for r1*c1 from completely randomized design
> rcres[23,] <- c(12, 6, mscrd)
```

The mean square residual for the row and block combinations in the table below are the same as in the SAS output summarized in Box 3 of Chapter 3; except, the mean square for the r1*c1 combination (CRD) was added to the end so itwould be included in the graph. See Fig. 3.1, which reproduces the graph in Box 3 of Chapter 3.

```
> print(rcres)
Obs rows cols residual
1     2    1 9.031288
2     3    1 8.065546
3     4    1 9.026714
4     6    1 8.011371
5    12    1 8.191936
6     1    2 5.111574
```

```
7      2    2 8.000500
8      3    2 7.597664
9      4    2 8.694907
10     6    2 8.089543
11    12    2 8.388074
12     1    3 6.695214
13     2    3 7.890012
14     3    3 7.855694
15     4    3 8.736928
16     6    3 8.197571
17    12    3 8.558527
18     1    6 6.547846
19     2    6 7.818142
20     3    6 8.033838
21     4    6 8.722255
22     6    6 8.158777
23    12    6 8.630242
```

```
> #plot the data to see minimum msresidual
> rcres <- arrange(rcres, cols, rows)

> rcres$cols <- factor(rcres$cols)

> ggplot(rcres, aes(x=rows, y=residual, colour=cols, group=cols))+
+    geom_point(size=2)+
+    geom_smooth(method="lm", formula=y~exp(1/x), se=F, si ....
[TRUNCATED]
> ggsave("C3-Ex3plot.png", dpi=600, height=7, width=10.5, units="cm")

Warning message:
In anova.lm(mod) :
  ANOVA F-tests on an essentially perfect fit are unreliable
>
```

Chapter 4. Power and Replication – Designing Powerful Experiments. Michael D. Casler

Exercise 1 (Chapter 4). Predicting the power of a future CRD experiment.

This exercise uses central and non-central F-distributions to predict and plan the power of a future Completely Randomized Design with sampling. Specifically, it proposes a trial with two treatments that differ by 5%, (means 95 and 100), an experimental error variance component of 5, a sampling error of 10, and a Type I error rate of $\alpha = 0.05$. It uses SAS's capability of outlining an ANOVA with only the design, treatment means, and variance components specified to obtain the simulated F-ratio and treatment df to compute the non-centrality parameter of the F-distribution needed for the power calculation. The recent addition of the EMSaov package to R provides a function (EMSanova()) to get most of the same information (Choe et al., 2017). This function doesn't give a direct estimate of the mean squares for the error terms from the variance components; however, it does provide the formulas. The correct error term and its simulated value, mean square reps within treatments, (msreps, assuming it is a random factor) are relatively easy to determine so this script calculates msreps and then verifies the formula in the EMSanova() function output. The SAS code estimates ncparm with

ncparm = numdf(Fvalue),

where numdf is the numerator df for the test of treatments and Fvalue is the ratio of the mean squares for treatments to msreps. This script uses an equivalent equation comprised of components that are a little easier to extract from the R output objects

ncparm = tSS/msreps,

where tSS is the sum of squares for treatments.

Script 4.1 jb

```
#activate additional packages needed
library(EMSaov)

#planning Completely Randomized Design with sampling
#enter planned treatment means in () following c
tmeans <- c(95, 100)
#enter planned no. of reps/trt (nr) and no. of samples/rep (ns)
nr <- 4
ns <- 2
#enter planned variance component for experimental error (vce)
vce <- 5
#enter planned variance component for sampling error (vcs)
vcs <- 10
#enter Type I error rate (a1) to be used as a decimal value
a1 <- 0.05

#statistical analyses
#determine the number of treatments (nt)
nt <- length(tmeans)
#estimate mean square for rep:trt (msrep) for testing trt
#for rep:trt a random factor: msrep = vcs + ns*(vce)
msrep <- vcs + ns*(vce); cat("msrep = ",msrep)
```

```
#create planning data set based on experiment size entered above
plan.dat <- expand.grid(smpl = factor(1:ns), reps = factor(1:nr), trt
= factor(1:nt))
#make vector of mock observations using treatment means
Y <- NULL
for (i in 1:length(tmeans)) Y <- c(Y, rep(tmeans[i], nr*ns))
plan.dat <- cbind(plan.dat,Y)
#print planning data with mock observations, Y
print(plan.dat); cat("\n")
#analyze mock Y for treatment ss, df, & expected mean squares
m1 <- EMSanova(Y ~ trt + reps, data=plan.dat, type = c("F", "R"),
nested=c(NA, "trt"), level=c(1,1))
tss <- m1$SS[1]
tSS <- round(tss, digits=6)
tMS <- round(m1$MS[1], digits = 6)
Fratio <- tMS/msrep
ProbF <- pf(Fratio, nt-1, nt*(nr-1), 0, lower.tail=FALSE)
cat("F ratio = ", Fratio, "\n")
cat("Prob[>F] = ", ProbF, "\n")
cat("\n")
m1$SS <- round(m1$SS, digits=6)
m1$MS <- round(m1$MS, digits = 6)
m1$Fvalue <- strtrim(m1$Fvalue, 8)
cat("Mock ANOVA: Df, trt SS & MS, and EMS are correct,  ", "\n")
print(m1, right=FALSE);  cat("\n")
#calculate the non-centrality parameter, ncparm
ncparm <- tss/msrep; cat("ncparm = ",ncparm)
#get critical value from cummulative F dist with numdf=(nt-1)=1
#dendf=nt(nr-1)=6 at prob level 1- a1
F_critical <- qf(1-a1, nt-1,nt*(nr-1), 0)
cat("F_critical = ",F_critical)
#calculate and print out power
power <- 1 - (pf(F_critical, nt-1, nt*(nr-1), ncparm))
cat("power = ", power)
```

Output 4.1 jb

```
----------------------------------------------------------------------------------------------
> #activate additional packages needed
> library(EMSaov)

> #planning Completely Randomized Design with sampling
> #enter planned treatment means in () following c
> tmeans <- c(95, 100)

> #enter planned no. of reps/trt (nr) and no. of samples/rep (ns)
> nr <- 4
> ns <- 2

> #enter planned variance component for experimental error (vce)
> vce <- 5

> #enter planned variance component for sampling error (vcs)
> vcs <- 10
```

```
> #enter Type I error rate (a1) to be used as a decimal value
> a1 <- 0.05

> #statistical analyses
> #determine the number of treatments (nt)
> nt <- length(tmeans)

> #estimate mean square for rep:trt (msrep) for testing trt
> #for rep:trt a random factor: msrep = vcs + ns*(vce)
> msrep <- vcs + ns*(vce); cat("ms ..." ... [TRUNCATED]
msrep =  20
> #create planning data set based on experiment size entered above
> plan.dat <- expand.grid(smpl = factor(1:ns), reps = factor(1:nr),
trt = factor(1: .... [TRUNCATED]

> #make vector of mock observations using treatment means
> Y <- NULL

> for (i in 1:length(tmeans)) Y <- c(Y, rep(tmeans[i], nr*ns))

> plan.dat <- cbind(plan.dat,Y)

> #print planning data with mock observations, Y
> print(plan.dat); cat("\n")
`  smpl reps trt   Y
1     1    1   1  95
2     2    1   1  95
3     1    2   1  95
4     2    2   1  95
5     1    3   1  95
6     2    3   1  95
7     1    4   1  95
8     2    4   1  95
9     1    1   2 100
10    2    1   2 100
11    1    2   2 100
12    2    2   2 100
13    1    3   2 100
14    2    3   2 100
15    1    4   2 100
16    2    4   2 100

> #analyze mock Y for treatment ss, df, & expected mean squares
> m1 <- EMSanova(Y ~ trt + reps, data=plan.dat, type = c("F", "R"),
nested=c(NA, "trt" .... [TRUNCATED]
> tss <- m1$SS[1]
> tSS <- round(tss, digits=6)
> tMS <- round(m1$MS[1], digits = 6)
> Fratio <- tMS/msrep
> ProbF <- pf(Fratio, nt-1, nt*(nr-1), 0, lower.tail=FALSE)
> cat("F ratio = ", Fratio, "\n")
F ratio =  5
> cat("Prob[>F] = ", ProbF, "\n")
```

```
Prob[>F] =  0.0667068
```

Prob[>F] for mock ANOVA matches that from SAS in Chapter 4

```
> cat("\n")
> m1$SS <- round(m1$SS, digits=6)
> m1$MS <- round(m1$MS, digits = 6)
> m1$Fvalue <- strtrim(m1$Fvalue, 8)
> cat("Mock ANOVA: Df, trt SS & MS, and EMS are correct,  ", "\n")
Mock ANOVA: Df, trt SS & MS, and EMS are correct,

> print(m1, right=FALSE);  cat("\n")
term      Df   SS  MS   Fvalue  Pvalue Sig Lvl EMS
trt        1  100 100 5.070602 <0.0001 *** 1   Error+2reps(trt)+8trt
reps(trt)  6    0   0 1        0.4852      1   Error+2reps(trt)
Residuals  8    0   0                      1   Error
```

From the EMS values it is clear the appropriate error term to test the effect of treatments (trt) is MS[rep:trt].

```
> #calculate the non-centrality parameter, ncparm
> ncparm <- tss/msrep; cat("ncparm = ",ncparm)
ncparm =  5
> #get critical value from cummulative F dist with numdf=(nt-1)=1
> #dendf=nt(nr-1)=6 at prob level 1- a1
> F_critical <- qf(1-a1, nt-1,nt*(nr-1), 0)

> cat("F_critical = ",F_critical)
F_critical =  5.987378
> #calculate and print out power
> power <- 1 - (pf(F_critical, nt-1, nt*(nr-1), ncparm))
> cat("power = ", power)
power =  0.4674093
> #
```

The the F_critical and power results above equal those computed with SAS in Chapter 4.

==========

Exercise 2 (Chapter 4). Predicting the power for a range of layouts for a future CRD experiment.

This exercise is an extension of the previous one and determines the power for a range of reps per treatment and samples per rep. Similar to the SAS code, this script puts the power calculations used in the previous example into a function and calls it for each combination of reps and samples. The complete output contains the mock data and analyses for all six combinations, but due to the redundancies the output for all but one of the combinations has been deleted here to save space.

Script 4.2 jb

```
#activate additional packages needed
library(EMSaov)
library(plyr)
```

```
#planning Completely Randomized Design with sampling
#enter planned treatment means in () following c
tmeans <- c(95, 100)
#enter min. (nrmin) & max. (nrmax) no. of reps/trt
nrmin <- 4
nrmax <- 6
#enter min. (nsmin) & max. (nsmax) no. of samples/rep
nsmin <- 2
nsmax <- 3
#enter planned variance component for experimental error (vce)
vce <- 5
#enter planned variance component for sampling error (vcs)
vcs <- 10
#enter Type I error rate (a1) to be used as a decimal value
a1 <- 0.05

#statistical analyses
#setup a function to calculate power from each design
pwrcalc <- function(x) {
#get ns and nr for current plan
ns <- x[1];  nr <- x[2]
cat("no. of samples/exp. unit, no. of reps/trt: ", ns, ",", nr, "\n")
#determine the number of treatments (nt)
nt <- length(tmeans)
#estimate mean square for rep:trt (msrep) for testing trt
#for rep:trt a random factor: msrep = vcs + ns*(vce)
msrep <- vcs + ns*(vce); cat("msrep = ",msrep, "\n")
#create planning data set based on experiment size entered above
plan.dat <- expand.grid(smpl = factor(1:ns), reps = factor(1:nr), trt
= factor(1:nt))
#make vector of mock observations using treatment means
Y <- NULL
for (i in 1:length(tmeans))   Y <- c(Y, rep(tmeans[i], nr*ns))
plan.dat <- cbind(plan.dat,Y)
#print planning data with mock observations, Y
print(plan.dat); cat("\n")
#analyze mock Y for treatment ss, df, & expected mean squares
m1 <- EMSanova(Y ~ trt + reps, data=plan.dat, type = c("F", "R"),
nested=c(NA, "trt"), level=c(1,1))
tss <- m1$SS[1]
tSS <- round(tss, digits=6)
tMS <- round(m1$MS[1], digits = 6)
Fratio <- tMS/msrep
ProbF <- pf(Fratio, nt-1,nt*(nr-1), 0, lower.tail=FALSE)
m1$SS <- round(m1$SS, digits = 6)
m1$MS <- round(m1$MS, digits = 6)
m1$Fvalue <- strtrim(m1$Fvalue, 8)
cat("Mock ANOVA: Df, trt SS & MS, and EMS are correct,  ", "\n")
print(m1, right=FALSE);  cat("\n")
#calculate the non-centrality parameter, ncparm
ncparm <- tss/msrep; cat("ncparm = ",ncparm, "\n")
#get critical value from cummulative F dist with numdf=(nt-1)
#dendf=nt(nr-1) at prob level 1- a1
F_critical <- qf(1-a1, nt-1,nt*(nr-1), 0)
```

```
cat("F_critical = ",F_critical, "\n")
#calculate and print out power
power <- 1 - (pf(F_critical, nt-1, nt*(nr-1), ncparm))
cat("power = ", power, "\n")
cat("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~", "\n")
cat("\n", "\n")
v <- c(ns, nr, nt, nt-1, nt*(nr-1), Fratio, ProbF, a1, ncparm,
F_critical, power)
names(v) <- c("ns", "nr", "nt", "numdf", "denomdf", "Fvalue",
"ProbF", "alpha", "ncparm", "F_critical", "power")
return(v)
}

#set up data frame for power results for each design
planpwr <- expand.grid(smpl=seq(nsmin,nsmax,1), reps =
seq(nrmin,nrmax,1))
#apply power calculation function to each row of planpwr
reslt <- apply(planpwr, 1, pwrcalc)
#transpose, format as data frame, sort, and print the results
treslt <- t(reslt)
treslt <- data.frame(treslt)
treslt <- arrange(treslt, ns, nr)
print.noquote(treslt)
```

Output 4.2 jb

```
> #activate additional packages needed
> library(EMSaov)
> library(plyr)

> #planning Completely Randomized Design with sampling
> #enter planned treatment means in () following c
> tmeans <- c(95, 100)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

no. of samples/exp. unit, no. of reps/trt:  3 , 5
msrep =  25
`  smpl reps trt   Y
1     1    1   1  95
2     2    1   1  95
3     3    1   1  95
4     1    2   1  95
5     2    2   1  95
6     3    2   1  95
7     1    3   1  95
8     2    3   1  95
9     3    3   1  95
10    1    4   1  95
11    2    4   1  95
12    3    4   1  95
13    1    5   1  95
14    2    5   1  95
15    3    5   1  95
```

```
16    1    1   2 100
17    2    1   2 100
18    3    1   2 100
19    1    2   2 100
20    2    2   2 100
21    3    2   2 100
22    1    3   2 100
23    2    3   2 100
24    3    3   2 100
25    1    4   2 100
26    2    4   2 100
27    3    4   2 100
28    1    5   2 100
29    2    5   2 100
30    3    5   2 100
```

```
Mock ANOVA: Df, trt SS & MS, and EMS are correct,
Term       Df    SS    MS   Fvalue  Pvalue Sig Lvl EMS
trt         1 187.5 187.5 2.056094 <0.0001 *** 1   Error+3reps(trt)+15trt
reps(trt)   8   0.0   0.0 0.9366   0.509       1   Error+3reps(trt)
Residuals  20   0.0   0.0                      1   Error
```

The above mock ANOVA gives the trt SS and verifies the df, error term (reps(trt), and formula for MS for reps(trt).

```
ncparm =  7.5
F_critical =  5.317655
power =  0.6708477
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
> #transpose, format as data frame, sort, and print the results
> treslt <- t(reslt)
> treslt <- data.frame(treslt)
> treslt <- arrange(treslt, ns, nr)
> print.noquote(treslt)
```

```
  ns nr nt numdf dendf Fvalue      ProbF alpha ncparm F_critical     power
1  2  4  2     1     6   5.00 0.06670680  0.05   5.00   5.987378 0.4674093
2  2  5  2     1     8   6.25 0.03694204  0.05   6.25   5.317655 0.5930781
3  2  6  2     1    10   7.50 0.02088241  0.05   7.50   4.964603 0.6949439
4  3  4  2     1     6   6.00 0.04982526  0.05   6.00   5.987378 0.5373402
5  3  5  2     1     8   7.50 0.02550387  0.05   7.50   5.317655 0.6708477
6  3  6  2     1    10   9.00 0.01334366  0.05   9.00   4.964603 0.7714014
> #
```

The power results above equal those in Box 2 of Chapter 4.

==========

Exercise 3 (Chapter 4). Predicting the power of a future RCBD experiment at several locations.

This exercise determines the power for a range of locations (2 to 6) and blocks (4 to 8) to detect a relatively small difference, 5%, between two treatments (means of 9.5 versus 10.0) when the variance components are 0 for blocks, 0.02 for the location-by-treatment interaction, and 0.2 for the

residual; and the Type 1 error is 0.05. In addition, the design includes one replication per block. As in the previous exercise, both the SAS code and this R-script place the power calculations in a macro or function that is called for each combination of locations and blocks within location. The full output contains the mock data and analyses for all combinations, but due to the redundancies, the output for all but one of the combinations has been deleted here to save space. Also note that the SAS code includes location as a fixed effect; however, in order for R's EMSanova() function to make the location-by-treatment interaction random, which is critical for the analysis, location must be a random factor in R. A graphical summary such as Fig. 3 of Chapter 4 is often useful and the last 14 lines of this script show how to emulate that figure and the result is displayed as Fig. 4.1 of this appendix.

Script 4.3 jb

```
-----------------------------------------------------------------------------------------------
#activate additional packages needed and set options
library(EMSaov)
library(plyr)
library(ggplot2)
options(scipen=4)

#planning Randomized Complete Block Design at multiple locations
#enter planned treatment meanl in () following c
tmeanl <- c(9.5, 10)
#enter min. (nbmin) & max. (nbmax) no. of blocks within a site
nbmin <- 4
nbmax <- 8           #set to 20 to redo Fig. 3 of Ch. 4
#enter min. (nlmin) & max. (nlmax) no. of locations (aka sites)
nlmin <- 2
nlmax <- 6
#enter planned variance component for blocks in location (vbl)
vbl <- 0
#enter planned variance component for trt x location  (vtl)
vtl <- 0.02
#enter planned variance component for residual error (vre)
vre <- 0.2
#enter Type I error rate (a1) to be used as a decimal value
a1 <- 0.05

#statistical analyses
#setup a function to calculate power from each design
pwrcalc <- function(x) {
#get nl and nb for current plan
nl <- x[1];  nb <- x[2]
cat("no. of locations, no. of blocks/site: ", nl, ",", nb, "\n")
#determine the number of treatments (nt)
nt <- length(tmeanl)
#estimate mean square for trt x locations (mstl) for testing trt
#with locations and trt x locations random: mstl = vre +nb*(vtl)
mstl <- vre + nb*(vtl); cat("mstl = ",mstl, "\n")
#create planning data set based on experiment size entered above
plan.dat <- expand.grid(site = factor(1:nl), blks = factor(1:nb),
trt = factor(1:nt))
```

```
#make vector of mock observationl using treatment means
Y <- NULL
for (i in 1:nt)   Y <- c(Y, rep(tmeanl[i], nb*nl))
plan.dat <- cbind(plan.dat,Y)
#print planning data with mock observations, Y
print(plan.dat); cat("\n")
#analyze mock Y for treatment ss, df, & expected mean squares
m1 <- EMSanova(Y ~ site + blks + trt, data=plan.dat, type = c("R",
"R", "F"), nested=c(NA, "site", NA), level=c(1,1,1))
tss <- m1$SS[3]
tSS <- round(tss, digits=6)
tMS <- round(m1$MS[3], digits=6)
dendf <- m1$Df[4]
Fratio <- tMS/mstl
ProbF <- pf(Fratio, nt-1, dendf, 0, lower.tail=FALSE)
m1$SS <- round(m1$SS, digits = 6)
m1$MS <- round(m1$MS, digits = 6)
m1$Fvalue <- strtrim(m1$Fvalue, 8)
cat("Mock ANOVA: Df, trt SS & MS, and EMS are correct,  ", "\n")
print(m1, right=FALSE);  cat("\n")
#calculate the non-centrality parameter, ncparm
ncparm <- tss/mstl; cat("ncparm = ",ncparm, "\n")
#get critical value from cummulative F dist with numdf=(nt-1)
#dendf from ANOVA at prob level (1 - a1)
F_critical <- qf(1-a1, nt-1,dendf, 0)
cat("F_critical = ",F_critical, "\n")
#calculate and print out power
power <- 1 - (pf(F_critical, nt-1, dendf, ncparm))
cat("power = ", power, "\n")
cat("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~", "\n")
cat("\n", "\n")
v <- c(nl, nb, nt, nt-1, dendf, Fratio, ProbF, a1, ncparm,
F_critical, power)
names(v) <- c("nl", "nb", "nt", "numdf", "dendf", "Fvalue",
"ProbF", "alpha", "ncparm", "F_critical", "power")
return(v)
}

#set up data frame for power results for each design and sort it
planpwr <- expand.grid(site=seq(nlmin, nlmax, 1), blks = seq(nbmin,
nbmax, 1))
planpwr <- arrange(planpwr, site, blks)
#apply power calculation function to each row of planpwr
reslt <- apply(planpwr, 1, pwrcalc)
#tranlpose and format as data frame
treslt <- t(reslt)
treslt <- data.frame(treslt)
#reformat several values so results fit on one line and print
treslt$Fvalue <- formatC(treslt$Fvalue, format="f", digits=4)
treslt$ProbF <- formatC(treslt$ProbF, format="f", digits=4)
treslt$ncparm <- formatC(treslt$ncparm, format="f", digits=4)
```

```
treslt$F_critical <- formatC(treslt$F_critical, format="f",
digits=4)
print.noquote(treslt)
treslt$nl <- factor(treslt$nl)
#graph results
plt <- ggplot(treslt, aes(x=nb, y=power, colour=nl, group=nl))+
   geom_point(size=2)+
   geom_smooth(span=1, se=F, size=2)+
   scale_y_continuous(breaks = seq(0, 1, by =0.1))+
   geom_hline(yintercept=0.80)+
   labs(x="Number of complete blocks per site", y="Estimated
power")+
   scale_colour_hue(name="Number of sites")+
   theme_bw()+
   theme(legend.justification=c(0,1), legend.position=c(0,1),
legend.direction="horizontal", panel.background=element_blank())
print(plt)
ggsave("C4-Ex3plot.png", dpi=600)
```

Output 4.3 jb

```
> #activate additional packages needed and set options
> library(EMSaov)
> library(plyr)
> library(ggplot2)
> options(scipen=4)

> #planning Randomized Complete Block Design at multiple locations
> #enter planned treatment meanl in () following c
> tmeanl <- c(9.5, 10)
> #enter min. (nbmin) & max. (nbmax) no. of blocks within a site
> nbmin <- 4
> nbmax <- 8           #set to 20 to redo Fig. 3 of Ch. 4

> #enter min. (nlmin) & max. (nlmax) no. of locations (aka sites)
> nlmin <- 2
> nlmax <- 6

> #enter planned variance component for blocks in location (vbl)
> vbl <- 0
> #enter planned variance component for trt x location  (vtl)
> vtl <- 0.02
> #enter planned variance component for residual error (vre)
> vre <- 0.2
> #enter Type I error rate (a1) to be used as a decimal value
> #setup a function to calculate power from each design
> pwrcalc <- function(x) {
+ #get nl and nb for current plan
+ nl <- x .... [TRUNCATED]

> #set up data frame for power results for each design and sort it
```

```
> planpwr <- expand.grid(site=seq(nlmin, nlmax, 1), blks = seq(nbmin,
nbmax, 1))
> planpwr <- arrange(planpwr, site, blks)
> #apply power calculation function to each row of planpwr
> reslt <- apply(planpwr, 1, pwrcalc)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

no. of locations, no. of blocks/site:  6 , 8
mstl =  0.36
`  site blks trt    Y
1     1    1   1  9.5
2     2    1   1  9.5
3     3    1   1  9.5
4     4    1   1  9.5
5     5    1   1  9.5
6     6    1   1  9.5
7     1    2   1  9.5
8     2    2   1  9.5
9     3    2   1  9.5
10    4    2   1  9.5
11    5    2   1  9.5
12    6    2   1  9.5
13    1    3   1  9.5
14    2    3   1  9.5
15    3    3   1  9.5
16    4    3   1  9.5
17    5    3   1  9.5
18    6    3   1  9.5
19    1    4   1  9.5
20    2    4   1  9.5
21    3    4   1  9.5
22    4    4   1  9.5
23    5    4   1  9.5
24    6    4   1  9.5
25    1    5   1  9.5
26    2    5   1  9.5
27    3    5   1  9.5
28    4    5   1  9.5
29    5    5   1  9.5
30    6    5   1  9.5
31    1    6   1  9.5
32    2    6   1  9.5
33    3    6   1  9.5
34    4    6   1  9.5
35    5    6   1  9.5
36    6    6   1  9.5
37    1    7   1  9.5
38    2    7   1  9.5
39    3    7   1  9.5
40    4    7   1  9.5
41    5    7   1  9.5
42    6    7   1  9.5
43    1    8   1  9.5
44    2    8   1  9.5
```

```
45    3    8   1  9.5
46    4    8   1  9.5
47    5    8   1  9.5
48    6    8   1  9.5
49    1    1   2 10.0
50    2    1   2 10.0
51    3    1   2 10.0
52    4    1   2 10.0
53    5    1   2 10.0
54    6    1   2 10.0
55    1    2   2 10.0
56    2    2   2 10.0
57    3    2   2 10.0
58    4    2   2 10.0
59    5    2   2 10.0
60    6    2   2 10.0
61    1    3   2 10.0
62    2    3   2 10.0
63    3    3   2 10.0
64    4    3   2 10.0
65    5    3   2 10.0
66    6    3   2 10.0
67    1    4   2 10.0
68    2    4   2 10.0
69    3    4   2 10.0
70    4    4   2 10.0
71    5    4   2 10.0
72    6    4   2 10.0
73    1    5   2 10.0
74    2    5   2 10.0
75    3    5   2 10.0
76    4    5   2 10.0
77    5    5   2 10.0
78    6    5   2 10.0
79    1    6   2 10.0
80    2    6   2 10.0
81    3    6   2 10.0
82    4    6   2 10.0
83    5    6   2 10.0
84    6    6   2 10.0
85    1    7   2 10.0
86    2    7   2 10.0
87    3    7   2 10.0
88    4    7   2 10.0
89    5    7   2 10.0
90    6    7   2 10.0
91    1    8   2 10.0
92    2    8   2 10.0
93    3    8   2 10.0
94    4    8   2 10.0
95    5    8   2 10.0
96    6    8   2 10.0
```

```
Mock ANOVA: Df, trt SS & MS, and EMS are correct,
`           Df SS MS Fvalue   Pvalue  Sig M.Level  EMS
site         5 0  0  0.9741   0.4447      1        Error+2blks(site)+16site
blks(site) 42 0  0  1.0002   0.4997      1        Error+2blks(site)
trt          1 6  6  1.985667 <0.0001 *** 1        Error+8site:trt+48trt
site:trt     5 0  0  1.0116   0.4229      1        Error+8site:trt
Residuals   42 0  0                      1        Error
```

The above mock ANOVA gives the trt SS and verifies the df, error term (site:trt), and formula for MS for site:trt. The font size has been reduced to fit data on one line.

```
ncparm =  16.66667
F_critical =  6.607891
power =  0.8987716
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

> #tranlpose and format as data frame
> treslt <- t(reslt)
> treslt <- data.frame(treslt)
> #reformat several values so results fit on one line and print
> treslt$Fvalue <- formatC(treslt$Fvalue, format="f", digits=4)
> treslt$ProbF <- formatC(treslt$ProbF, format="f", digits=4)
> treslt$ncparm <- formatC(treslt$ncparm, format="f", digits=4)
> treslt$F_critical <- formatC(treslt$F_critical, format="f",
digits=4)


> print.noquote(treslt)
`  nl nb nt numdf dendf   Fvalue  ProbF alpha  ncparm F_critical     power
1   2  4  2     1     1   3.5714 0.3098  0.05  3.5714   161.4476 0.1192977
2   2  5  2     1     1   4.1667 0.2900  0.05  4.1667   161.4476 0.1281937
3   2  6  2     1     1   4.6875 0.2755  0.05  4.6875   161.4476 0.1355639
4   2  7  2     1     1   5.1471 0.2643  0.05  5.1471   161.4476 0.1417814
5   2  8  2     1     1   5.5556 0.2554  0.05  5.5556   161.4476 0.1471043
6   3  4  2     1     2   5.3571 0.1467  0.05  5.3571    18.5128 0.2683507
7   3  5  2     1     2   6.2500 0.1296  0.05  6.2500    18.5128 0.2995139
8   3  6  2     1     2   7.0312 0.1176  0.05  7.0313    18.5128 0.3256909
9   3  7  2     1     2   7.7206 0.1088  0.05  7.7206    18.5128 0.3479747
10  3  8  2     1     2   8.3333 0.1020  0.05  8.3333    18.5128 0.3671636
11  4  4  2     1     3   7.1429 0.0755  0.05  7.1429    10.1280 0.4509447
12  4  5  2     1     3   8.3333 0.0632  0.05  8.3333    10.1280 0.5046193
13  4  6  2     1     3   9.3750 0.0549  0.05  9.3750    10.1280 0.5480540
14  4  7  2     1     3 10.2941 0.0490  0.05 10.2941    10.1280 0.5837086
15  4  8  2     1     3 11.1111 0.0446  0.05 11.1111    10.1280 0.6133674
16  5  4  2     1     4  8.9286 0.0404  0.05  8.9286     7.7086 0.6160059
17  5  5  2     1     4 10.4167 0.0321  0.05 10.4167     7.7086 0.6792799
18  5  6  2     1     4 11.7188 0.0267  0.05 11.7188     7.7086 0.7271846
19  5  7  2     1     4 12.8676 0.0230  0.05 12.8676     7.7086 0.7641582
20  5  8  2     1     4 13.8889 0.0204  0.05 13.8889     7.7086 0.7932202
21  6  4  2     1     5 10.7143 0.0221  0.05 10.7143     6.6079 0.7443690
22  6  5  2     1     5 12.5000 0.0166  0.05 12.5000     6.6079 0.8046044
23  6  6  2     1     5 14.0625 0.0133  0.05 14.0625     6.6079 0.8466094
24  6  7  2     1     5 15.4412 0.0111  0.05 15.4412     6.6079 0.8766878
25  6  8  2     1     5 16.6667 0.0095  0.05 16.6667     6.6079 0.8987716
```

The results above match those in the solution table for Exercise 3 of Chapter 4.

```
> treslt$nl <- factor(treslt$nl)

> #graph results
> plt <- ggplot(treslt, aes(x=nb, y=power, colour=nl, group=nl))+
+    geom_point(size=2)+
+    geom_smooth(span=1, se=F, size=2)+
+  .... [TRUNCATED]

> print(plt)
`geom_smooth()` using method = 'loess'
> ggsave("C4-Ex3plot.png", dpi=600)
Saving 5.76 x 5.75 in image
`geom_smooth()` using method = 'loess'
```

See Fig. 4.1 of this appendix.

Chapter 6. Linear Regression Techniques
Christel Richter and Hans-Peter Piepho

Example 1. Linear Regression: Read the data and do basic plots.

An analysis of the relationship between the weight and length of six-year old eels is conducted using a data set of 34 eels collected from similar habitats. The relationship is visulaized using a scatterplot with marginal boxplots for each of the variables. The relationship is quantified using linear regression and several measures of model fit are calculated and graphed. The data were stored in a comma delimited file named eel.csv for these scripts. The SAS code to generate the graphs, conduct the linear regression, and generate the measures of model fit have been translated to R-scripts below. There are multiple methods of conducting linear regression and drawing graphs in R but the following R scripts use the generic linear model analysis function lm() and plot() because they are included in base R and allow for flexibility. To save a high resolution plot of publication quality, describe and save the plot before it is actually created.

Script 6.1 kagc

```
#Set options
#Switch to scientific numbers after 7 digits
options(scipen=7)
#Check current directory
getwd()
#If necessary, set wd to correct one for your computer
setwd('~/…)
#Double check
getwd()

#Install and load needed packages
#These are the packages needed for chapter 6 analyses
#To install several packages at once, create a package list, then install
the whole list
package.list <- c('AICcmodavg', 'aTSA', 'car', 'emmeans',
'ggplot2', 'HH', 'investr', 'lme4', 'lsmeans', 'MASS', 'merTools','nlme',
'nlstools', 'nortest', 'plyr', 'rcompanion', 'reshape2', 'rsq',
'segmented','Stack', 'TSA','TTR')
install.packages(package.list, .Library)

#Data management
#Read in the data with read.csv function
#Give it a short unique name
#This code works if dataset is in working directory
#header=TRUE means column names are in the dataset
#For example and for all datasets in this chapter:
eel.dat <- read.csv ('eel.csv', header=TRUE)
print(eel.dat)
#Attach as data frame
attach (eel.dat)
#View the structure of the data
str(eel.dat)
#Variables can be integer, number or factor

#Correct variable type as needed for analysis
#To change variable types to factors
for ( var  in c('Eel_no'))
```

```
{eel.dat [,var] <- as.factor( eel.dat[,var])}
#To change variable types to number
for ( var  in c('Length', 'Weight'))
{eel.dat [,var] <- as.numeric(eel.dat [,var])}
#Double check
str (eel.dat)
#Check first & last 6 rows of data with head & tail
head (eel.dat)
tail (eel.dat)
#Calculate basic stats for each variable
summary( eel.dat)

#Save the high resolution plot
png('Ch6.Fig1.png', width = 8, height = 8,
units = 'in', res = 600)
#The dev.off function is here
#so that the save function can easily be turned on
#and off to send plots to the Plots tab on the
#lower right in RStudio
#Comment it out to send the plots to a file.
#dev.off()
#Use the par command to set each plot in its section
#Set margins and axes.
par (fig = c(0.0,0.8,0.25,1.0),
mar = c(3.1,3.1,3.1,2.1),
new = TRUE,
cex.axis = 0.8,
mgp = c(2,1,0))
#Plot a scatter plot, label X and Y axes,
#Select closed circles for characters (pch)
plot (eel.dat$Length, eel.dat$Weight,
xlab = 'Length', ylab = 'Weight', pch = 20,     panel.first = grid(lty =
1,lwd = 2))
#Locate a boxplot below the scatter plot
par(fig = c(0.0,0.8,0.0,0.40),
mar = c(3.1,3.1,3.1,2.1), new = TRUE)
#Plot and color a horizontal boxplot for the X variable
boxplot(eel.dat$Length, horizontal = TRUE,
col = 'lightpink1', axes = FALSE)
#Locate a boxplot to the right of the scatter plot
par (fig = c(0.65,1.0,0.25,1.0), new = TRUE)
#Plot and color a vertical boxplot for the Y variable
boxplot (eel.dat$Weight,
col = 'lightblue1', axes = FALSE)
#Create and place a title for the plot
mtext ('Fig. 1.  Example 1. Scatterplot and Boxplots',      side=3,
outer=TRUE, line=-1)
#Turn off the high resolution save plot
dev.off()
#See Fig 6.1 at end of this file for plot.
#This reproduces Fig. 6.1 in chapter 6.
#Fig 2 can be reproduced using similar methods
```

Output 6.1 kagc

```
> #Data management
> #Read in the data with read.csv function
> #Give it a short unique name
> #This code works if dataset is in working directory
> #header=TRUE means column names are in the dataset
> #For example and for all datasets in this chapter:
> eel.dat <- read.csv ('eel.csv', header=TRUE)
> print(eel.dat)
`  Eel_no Length Weight
1       1     33  108.6
2       2     34  114.1
3       3     36  120.4
4       4     36  128.6
5       5     37  137.5 . . . TRUNCATED
> #Attach as data frame
> attach (eel.dat)
> #View the structure of the data
> str(eel.dat)
'data.frame':    34 obs. of  3 variables:
 $ Eel_no: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Length: int  33 34 36 36 37 39 39 40 41 42 ...
 $ Weight: num  109 114 120 129 138 ...
> #Variables can be integer, number or factor
>
> #Correct variable type as needed for analysis
> #To change variable types to factors
> for ( var  in c('Eel_no')) . . . TRUNCATED
> str (eel.dat)
'data.frame':    34 obs. of  3 variables:
 $ Eel_no: Factor w/ 34 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9
10 ...
 $ Length: num  33 34 36 36 37 39 39 40 41 42 ...
 $ Weight: num  109 114 120 129 138 ...
> #Check first & last 6 rows of data with head & tail
> head (eel.dat)
` Eel_no Length Weight
1       1     33  108.6
2       2     34  114.1
3       3     36  120.4
4       4     36  128.6
5       5     37  137.5
6       6     39  144.2
> tail (eel.dat)
   Eel_no Length Weight
29     29     51  221.6
30     30     52  231.7
31     31     53  246.2
32     32     54  247.5
33     33     55  254.8
34     34     58  275.0
> #Calculate basic stats for each variable
> summary( eel.dat)
`     Eel_no        Length          Weight
 1       : 1   Min.   :33.00   Min.   :108.6
```

```
 2       : 1   1st Qu.:41.25   1st Qu.:161.1
 3       : 1   Median :46.00   Median :189.7
 4       : 1   Mean   :45.12   Mean   :187.2
 5       : 1   3rd Qu.:50.50   3rd Qu.:219.2
 6       : 1   Max.   :58.00   Max.   :275.0

> #Save the high resolution plot
> png('Ch6.Fig1.png', width = 8, height = 8,
+ units = 'in', res = 600)
> #The dev.off function is here
> #so that the save function can easily be turned on
> #and off to send plots to the Plots tab on the
> #lower right in RStudio
> #Comment it out to send the plots to a file.
> #dev.off()
> #Use the par command to set each plot in its section
> #Set margins and axes.
> par (fig = c(0.0,0.8,0.25,1.0),
+ mar = c(3.1,3.1,3.1,2.1),
+ new = TRUE,      . . .TRUNCATED
> #Plot a scatter plot, label X and Y axes,
> #Select closed circles for characters (pch)
> plot (eel.dat$Length, eel.dat$Weight,
+ xlab = 'Length', ylab = 'Weight', pch = 20, . .TRUNCATED
> #Locate a boxplot below the scatter plot
> par(fig = c(0.0,0.8,0.0,0.40),
+ mar = c(3.1,3.1,3.1,2.1), new = TRUE)
> #Plot and color a horizontal boxplot for the X variable
> boxplot(eel.dat$Length, horizontal = TRUE, . .TRUNCATED
> #Locate a boxplot to the right of the scatter plot
> par (fig = c(0.65,1.0,0.25,1.0), new = TRUE)
> #Plot and color a vertical boxplot for the Y variable
> boxplot (eel.dat$Weight, . . . TRUNCATED
1
> #This reproduces Fig. 6.1 in chapter 6.
> #Fig 2 can be reproduced using similar methods
```

Ignore the warning messages that will be generated as you run this code. They are for information only

==========

Example 1 (continued). Regression: diagnostic plots

In the SAS code, the PROC REG code requests that a dataset of statistics is saved as eel_out with variables labelled as P=yhat, r=yresid STUDENT=student, RSTUDENT=rstudent, H=h COOKD=cookd COVRATIO=covratio, DFFITS=dffits PRESS=PRESS. Similarly, the lm function in R generates a dataset containing the regression coefficients, residauls, effects, fitted values. The naming convention that we will use for the regression models generated for Chapter 6 is the example number, the type of model used and an index number, (Ex1lm1). We will use these statistics, calculate others, combine them into a new dataset and generate diagnostic plots.
Plots to replicate the SAS PROC REG output:
Residuals histogram with normal and kernel density plots [see below]
Residuals by predicted values [Plot 1 in plot(lm)

RStudent by predicted values for weight [see below]
Observed values by predicted values for weight [see below]
Cooks D for weight [see below]
Outlier and leverage diagnostics for weight
Rstudent against leverage with outlier and high #leverage datapoints labelled (Plot 4 in plot(lm))
QQ plot of residuals for weight (Plot 2 in plot(lm))
Resdiual fit spread plot for (Plot 3 in plot(lm))
Scale-location, also see below]
Resduals boxplot for weight [see below]
Influence diagnostics for weight (DFFITS against observation)
Influence diagnostics for weight (DFBETAS against observation for intercept and length)

Script 6.2 kagc

```
#Conduct linear regression analysis with lm
Ex1lm1 <- lm(Weight ~ Length, data=eel.dat)
#Print parameter estimates from this model
anova (Ex1lm1)
summary (Ex1lm1)

#In R, the plot function applied to the output of the #lm, plot(lm),
will produce four diagnostic plots.

#Save the high resolution plot
png('Ch6.Fig2.png', width = 8, height = 8,
units = 'in', res = 600)
#Comment the dev.off function in and out as needed
#to send the plots to a file.
#dev.off()
#The par function can be used to put all four
#graphs on one page.
par(mfrow=c(2,2))
plot(Ex1lm1, new=FALSE)
dev.off()

#Residuals histograms
#with kernel and normal density curves
#Save the high resolution plot
png('Ch6.Fig3a.png', width = 8, height = 8,
units = 'in', res = 600)
#Comment the dev.off function in and out as needed
#to send the plots to a file.
#dev.off()
par(mfrow=c(1,1))
#Note, the freq = F command
#changes the y axis to frequency
hist(Ex1lm1$res,breaks = 20, freq = F,
xlim = c(-10,10), ylim = c(0,.2),
xlab = 'Residuals', ylab = 'Probability',
col = 'lightblue', main = "")
mtext("Residuals Histogram, Normal and Kernel
Density Curve for Ex 1")
curve(dnorm(x, mean = mean(Ex1lm1$res),
```

```
sd = sd(Ex1lm1$res)), add = T)
lines(density(Ex1lm1$res), col = 'red')
legend('topright', c('Normal','Kernel'),
col = c('black','red'), lty = c(1,1),
lwd = c(2,2,2), cex = 0.5)
dev.off()

#Student residuals histogram
png('Ch6.Fig3b.png', width = 8, height = 8,
units = 'in', res = 600)
par(mfrow=c(1,1))
hist(rstudent(Ex1lm1), breaks = 20, freq = F,
xlim = c(-3.5,3.5),  ylim = c(0,1.0),
xlab = 'Student Residuals',
ylab = 'Probability',
col='lightgreen', main='')
mtext("Student Residuals Histogram, Normal
and Kernel Density Curve for Ex 1")
curve(dnorm (x, mean = mean(rstudent(Ex1lm1)),
sd = sd(rstudent(Ex1lm1))), add = T)
lines(density (rstudent(Ex1lm1)), col = 'red')
legend('topright', c('Normal','Kernel'),
col = c('black','red'),
lty = c(1,1), lwd = c(2,2,2), cex = 0.5)
dev.off()

#Frequency histogram for data
png('Ch6.Fig3c.png', width = 8, height = 8,
units = 'in', res = 600)
par(mfrow=c(1,1))
hist(eel.dat$Weight, breaks = 20, freq = F,
xlim = c(50,300), ylim = c(0, 0.015),
xlab = 'Weight', ylab = 'Probability',
col = 'blue', main = '')
mtext ("Histogram, Normal and Kernel
Density Curve for Ex 1")
curve(dnorm(x, mean = mean(eel.dat$Weight),
sd = sd(eel.dat$Weight)), add = T)
lines (density(eel.dat$Weight), col = 'red')
legend ('topright', c('Normal', 'Kernel'),
col = c('black','red'), lty=c(1,1),
lwd = c(2,2,2), cex = 0.5)
dev.off()
#this code is used again in Ex8.

#Several residuals plots
png('Ch6.Fig4.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par (mfrow = c(2,2))
plot (eel.dat$Weight, Ex1lm1$fitted, pch = 20,
xlab = 'Observed Weight',
ylab = 'Predicted Weight', main = '')
abline(lm(Ex1lm1$fitted ~ eel.dat$Weight))
```

```
plot (rstudent(Ex1lm1), Ex1lm1$fitted, pch=20,
xlab = 'Student Residuals',
ylab = 'Predicted Weight', main='')
abline(v=0)
plot (Ex1lm1$residuals, Ex1lm1$fitted, pch=20,
xlab = 'Residuals',
ylab = 'Predicted Weight', main='')
abline(v=0)
plot(Ex1lm1$residuals, eel.dat$length, pch=20,
xlab = 'Residuals',
ylab = 'Length', main='')
abline(v=0)
dev.off()

png('Ch6.Fig5.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par (mfrow = c(2,2))
boxplot(Ex1lm1$residuals, col = 'blue',
xlab = 'Residuals')
#Compute and plot Cook's distance and leverage
plot (cooks.distance (Ex1lm1),
xlab = 'Eel No', ylab = 'Cooks distances')
lev <- (hat(model.matrix(Ex1lm1)))
plot(lev)
#Draw gridlines and label datapoints on leverage plots
plot (lev, rstudent(Ex1lm1),
xlab = 'Leverage', ylab = 'Rstudent', new = TRUE)
abline(v = 0.12, col = 'blue')
abline(h = c(-2,2), col = 'red')
text(lev, rstudent(Ex1lm1),
rownames(eel.dat),pos = 3,
col = 'red',cex = 0.5)
#data for eel_No 1, 2 and 34 have high leverage
#data for  eel_No  3, 31 and 18 have high residuals
dev.off()

png('Ch6.Fig6.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par (mfrow = c(2,2))
#Calculate and plot DFFITS
plot(dffits(Ex1lm1))
abline(h = c(-.5,.5), col = 'red')
#Calculate and plot DFBETAS.
#Use rename function from plyr package to avoid similar names
#Graphs for intercept & length have reversed Y axis
library(plyr)
a <- as.data.frame (dfbetas(Ex1lm1))
dfb <- rename ( a,
c('(Intercept)'  = 'dfbetas_intercept',
'Length' = 'dfbetas_length'))
plot (dfb$dfbetas_intercept,
xlab = 'intercept',
```

```
ylab = 'DFBETAS',
ylim = c(-0.6, 0.6))
abline(h = c(-.35,.35), col = 'red')
plot (dfb$dfbetas_length,
xlab = 'length',
ylab = '',
ylim = c(0.6, -0.6))
abline(h = c(-.35,.35), col = 'red')
dev.off()

#Calculate the PRESS statistic with the PRESS function
PRESS  <- function(linear.model) {
#first calculate the predictive residuals
pr <- residuals(linear.model)/ (1-lm.influence(linear.model)$hat)
#calculate the PRESS
PRESS <- sum(pr^2)
return(PRESS)
}
Press <- PRESS(Ex1lm1)
print(Press)
```

Output 6.2 kagc

```
> #Conduct linear regression analysis with lm
> Ex1lm1 <- lm(Weight ~ Length, data=eel.dat)
> #Print parameter estimates from this model
> anova (Ex1lm1)
Analysis of Variance Table

Response: Weight
'          Df Sum Sq Mean Sq F value    Pr(>F)
Length      1  57466   57466  5560.8 < 2.2e-16 ***
Residuals 32    331      10
---------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> summary (Ex1lm1)

Call:
lm(formula = Weight ~ Length, data = eel.dat)

Residuals:
'    Min      1Q  Median      3Q     Max
-8.3125 -1.7115 -0.2393  1.9117  6.3219

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
(Intercept) -114.20443    4.07953  -28.00   <2e-16 ***
Length         6.68080    0.08959   74.57   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.215 on 32 degrees of freedom
Multiple R-squared:  0.9943, Adjusted R-squared:  0.9941
F-statistic:  5561 on 1 and 32 DF,  p-value: < 2.2e-16
```

```
>
> #In R, the plot function applied to the output of the #lm, plot(lm),
will produce four diagnostic plots.
>
> #Save the high resolution plot
> png('Ch6.Fig2.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #Comment the dev.off function in and out as needed
> #to send the plots to a file.
> #dev.off()
> #The par function can be used to put all four
> #graphs on one page.
> par(mfrow=c(2,2))
> plot(Ex1lm1, new=FALSE) . . .TRUNCATED
>
> #Residuals histograms
> #with kernel and normal density curves
> #Save the high resolution plot
> png('Ch6.Fig3a.png', width = 8, height = 8,
+     units = 'in', res = 600) . . .TRUNCATED
>
> #Student residuals histogram
> png('Ch6.Fig3b.png', width = 8, height = 8,
+     units = 'in', res = 600)
> par(mfrow=c(1,1)) ) . . .TRUNCATED
>
> #Frequency histogram for data
> png('Ch6.Fig3c.png', width = 8, height = 8,
+     units = 'in', res = 600)
> par(mfrow=c(1,1)) ) . . .TRUNCATED

> #this code is used again in Ex8.
>
> #Several residuals plots
> png('Ch6.Fig4.png', width = 8, height = 8,
+     units = 'in', res = 600) . . . TRUNCATED
> #dev.off()
> par (mfrow = c(2,2)) ) . . . TRUNCATED

>
> png('Ch6.Fig5.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par (mfrow = c(2,2)) . . . TRUNCATED

> png('Ch6.Fig6.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par (mfrow = c(2,2)). . . TRUNCATED
> #Graphs for intercept & length have reversed Y axis
> library(plyr)
> a <- as.data.frame (dfbetas(Ex1lm1))
> dfb <- rename ( a, . . . TRUNCATED
```

```
> #Calculate the PRESS statistic with the PRESS function
> PRESS  <- function(linear.model) {
+   #first calculate the predictive. . . TRUNCATED
+ }
> Press <- PRESS(Ex1lm1)
> print(Press)
[1] 371.7289
```

==========

Example 1 (continued): Chapter 6, Fig. 3.

Figure 3 in Chapter 6 is a scatter plot with a trend line and confidence intervals, a legend, the error DF, the root MSE, the R square, and the Adjusted R-square from the regression model. We will first calculate the confidence interval (CI) and the prediction interval (PI) for the regression line, combine these new data with the original dataset, rename some variables, and generate the plots with additional details from the regression model. Plots for Figure 3, for Examples 2-4 can be generated similarly.

Script 6.3 kagc

```
#Calculate the CI and PI on the regression line,
#Save these data into new data frames
CI <- as.data.frame( predict(Ex1lm1, Ex1lm1$Length,
interval='confidence',level = 0.95, type='response'))
PI <- as.data.frame(predict(Ex1lm1, Ex1lm1$Length,
interval='prediction',level = 0.95, type='response'))

#Some of the variables in CI and PI have the same name #Rename them
using the rename function in PLYR
library(plyr)
CInew <- rename(CI, c( 'lwr' = 'LCI', 'upr' = 'UCI'))
PInew <- rename(PI, c('lwr' = 'LPI', 'upr' = 'UPI'))

#Bind  CInew and PInew with with the original data.
#Add variables from the model output
#Rename variables as needed.
eel.dat2 <- cbind(eel.dat, CInew$fit, residuals(Ex1lm1),
rstudent(Ex1lm1),cooks.distance(Ex1lm1),
covratio(Ex1lm1), dffits(Ex1lm1), lev, dfb,
CInew$LCI, CInew$UCI, PInew$LPI, PInew$UPI)

eel.dat3 <- rename(eel.dat2,
c('residuals(Ex1lm1)' = 'Residual',
'rstudent(Ex1lm1)' = 'Rstudent',
'CInew$fit' = 'Fit',
'cooks.distance(Ex1lm1)' = 'CooksD',
'covratio(Ex1lm1)' = 'CovR',
'dffits(Ex1lm1)' = 'DFFITS',
'CInew$LCI' = 'LCI',
'CInew$UCI' = 'UCI',
'PInew$LPI' = 'LPI',
```

```
'PInew$UPI' = 'UPI'))
#Double check variables in dataset
str(eel.dat3)

#Generate the Confidence interval plot
png('Ch6.Fig7.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
plot(Weight~Length, eel.dat3, col =('black'),pch=1,
xlab = ('Length (cm)'),
ylab = ('Weight (g)'), new = FALSE )
lines(Fit~Length,eel.dat3,col='black',lwd=2,lty=1)
lines(LCI~Length,eel.dat3,col='blue',lwd=2,lty=3)
lines(UCI~Length,eel.dat3,col='blue',lwd=2,lty=3)
lines(LPI~Length,eel.dat3,col='grey',lwd=2,lty=1)
lines(UPI~Length,eel.dat3,col='grey',lwd=2,lty=1)
mtext('Fig. 3. Observed and fitted values with
95% confidence intervals for Ex1.', side=3,
outer=TRUE, line=-3)
#Print the regression equation on the graph
text(35,250,
expression ('yhat = -114.204 + 6.681 * x'),
adj=0, cex=0.8)
#Place the legends on the graph
legend('bottomright', c('Observed','Fitted',
'Conf.Interval','Pred. Interval'),
col = c('black','black','blue','grey'),
lty = c(0,1,3,1), lwd = c(NA,2,2,2),
pch = c(1,NA,NA,NA), cex = 0.5)
legend('right', c('Observations = 34',
'Parameters = 2', 'Error DF = 32',
'MSE = 10.334', 'R-Square = 0.9943',
'Adj. R-Square = 0.9941'), cex=0.5)
dev.off()

#Note, Have not been able to duplicate the confidence #ellipse in
Figure 4 left
#Call the ggplot2 package and use the
#ggsave command to save the plot
#prediction interval ellipse
library(ggplot2)
par(mfrow=c(1,1))
ggplot(eel.dat, aes(x=Weight,y=Length)) +         geom_point(size=2,
shape=19) +
geom_smooth(method='auto', se=TRUE, fullrange=FALSE, level=0.95) +
stat_ellipse(type='t') +
ggtitle('Scatter Plot with
95% Prediction   Ellipse') +
annotate('text', x=150, y=60,
label= 'Obs: 34 Corr: 0.9971 p <2 .2e-16')
ggelipse
ggsave('Ch6.Fig8.png', ggelipse, dpi=600)
dev.off()
```

Output 6.3 kagc

```
> #Calculate the CI and PI on the regression line,
> #Save these data into new data frames
> CI <- as.data.frame( predict(Ex1lm1, Ex1lm1$Length,
interval='confidence',
+  level = 0.95, type='response'))
> PI <- as.data.frame(predict(Ex1lm1, Ex1lm1$Length,
interval='prediction',
+ level = 0.95, type='response'))
>
> #Some of the variables in CI and PI have the same name #Rename
them using the rename function in PLYR
> library(plyr) . . .TRUNCATED
>
> #Bind  CInew and PInew with with the original data.
> #Add variables from the model output . . . TRUNCATED
> #Double check variables in dataset
> str(eel.dat3)
'data.frame':    34 obs. of  16 variables:
 $ Eel_no: Factor w/ 34 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8
9 10 ...
 $ Length: num  33 34 36 36 37 39 39 40 41 42 ...
 $ Weight: num  109 114 120 129 138 ...
 $ Fit: num  106 113 126 126 133 ...
 $ Residual: num  2.34 1.16 -5.9 2.3 4.51 ...
 $ Rstudent: num  0.781 0.38 -2.02 0.745 1.492 ...
 $ CooksD: num  0.0517 0.0106 0.1931 0.0292 0.094 ...
 $ CovR: num  1.196 1.207 0.918 1.135 1.009 ...
 $ DFFITS: num  0.32 0.144 -0.651 0.24 0.442 ...
 $ lev: num  0.1435 0.1254 0.094 0.094 0.0806 ...
 $ dfbetas_intercept: num  0.302 0.134 -0.584 0.215 0.385 ...
 $ dfbetas_length: num  -0.285 -0.126 0.539 -0.199 -0.352 ...
 $ LCI: num  104 111 124 124 131 ...
 $ UCI: num  109 115 128 128 135 ...
 $ LPI: num  99.3 106 119.5 119.5 126.2 ...
 $ UPI: num  113 120 133 133 140 ...
>
> #Generate the Confidence interval plot
> png('Ch6.Fig7.png', width = 8, height = 8,
+ units = 'in', res = 600)
> #dev.off() . . . TRUNCATED

> #Note, Have not been able to duplicate the confidence #ellipse in
Figure 4 left
> #Call the ggplot2 package and use the
> #ggsave command to save the plot
> #prediction interval ellipse
> library(ggplot2) . . . TRUNCATED
> ggsave('Ch6.Fig8.png', ggelipse, dpi=600)
```

```
Saving 5.31 x 4.71 in image
`geom_smooth()` using method = 'loess'. . . TRUNCATED
=======================================================
```

Example 1 (continued). Tests for normality and additional comments

The univariate procedure of SAS returns summary statistics, tests of normality, and QQ plots. The summary statistics and QQ plot of studentized residuals were printed above but the QQ plots are printed again with the tests for normality. The last two sections of SAS code in this example reproduce the residuals plots with critical values marked. We haven't reproduced those plots again here. Because both X and Y are numerical variables we can also calculate the linear regression for Example 1 using weight as the regressor (x) and length as the regressand (y).

Script 6.4 kagc

```
-------------------------------------------------------------------------------------------------
#Calculate shapiro.test, and ks.test for normality
#Reproduce qqtests
shapiro.test(eel.dat3$Rstudent)
ks.test(eel.dat3$Rstudent, 'pnorm')
png('Ch6.Fig9.png', width = 8, height = 8, units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
qqnorm(eel.dat3$Rstudent)
qqline(eel.dat3$Rstudent, col = 'red')
legend('bottomright',
c('W = 0.9732',
 'p-value = 0.5559',
 'D = 0.10041',
 'p-value = 0.8492'), cex=1)
dev.off()
Ex1lm2 <- lm(Length ~ Weight, data = eel.dat)
summary(Ex1lm2)
print(Ex1lm2)
```

Output 6.4 kagc: additional comments

```
-------------------------------------------------------------------------------------------------
> #Calculate shapiro.test, and ks.test for normality
> #Reproduce qqtests
> shapiro.test(eel.dat3$Rstudent)
      Shapiro-Wilk normality test

data:  eel.dat3$Rstudent
W = 0.97323, p-value = 0.5559

> ks.test(eel.dat3$Rstudent, 'pnorm')

      One-sample Kolmogorov-Smirnov test

data:  eel.dat3$Rstudent
D = 0.10041, p-value = 0.8492
alternative hypothesis: two-sided
```

```
> png('Ch6.Fig9.png', width = 8, height = 8, units = 'in', res = 600)
> #dev.off() . . . TRUNCATED
>
> Ex1lm2 <- lm(Length ~ Weight, data = eel.dat)
> summary(Ex1lm2)

Call:
lm(formula = Length ~ Weight, data = eel.dat)

Residuals:
`    Min       1Q   Median       3Q      Max
-0.89577 -0.30951  0.01777  0.26021  1.24216

Coefficients:
`            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.254758   0.382598   45.10   <2e-16 ***
Weight       0.148826   0.001996   74.57   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4798 on 32 degrees of freedom
Multiple R-squared:  0.9943, Adjusted R-squared:  0.9941
F-statistic:  5561 on 1 and 32 DF,  p-value: < 2.2e-16

> print(Ex1lm2)
Call:
lm(formula = Length ~ Weight, data = eel.dat)

Coefficients:
(Intercept)       Weight
`   17.2548       0.1488
```

==========

Example 2. Generate dataset, linear regression, and summary statistics

In Example 2, inverse regression is used with the investr package in R to predict the phosphorus content in soil that is accessible to plants based on a colorimetric measuring instrument. The intital dataset contains five phosphorus levels and their associated colorimetric extinction coefficients. We will create the data set using the code below, conduct a linear regression of these data and the use this calibration to condut the inverse regression. The cali_start dataset that is generated in the SAS code in not needed in the R investr package. In Chapter 6, plots in Figures 2, and 3 can be generated as in Example 1

Script 6.5 kagc

```
#Build a dataset named calib.dat from scratch,
#First generate a vector for each variable where
#c=the phosporus concentration
#ext=the colorimatric extinction value
obs_no <- c(1:5)
c <- c(0.5,1.0,1.5,3.0,5.0)
ext <- c(55.5,77.0,100.7,165.5,253.4)
```

```
#Bind the vectors into one dataframe.
calib.dat <- data.frame(cbind(obs_no, c, ext))
#Obtain basic statistics for the variables
summary(calib.dat)
#Linear regression for extintion on concentration
Ex2lm1 <- lm(ext ~ c, data = calib.dat)
summary(Ex2lm1)
anova(Ex2lm1)
#Calculate the mean for initial C values in calib.dat
#Save mean as a vector
cmean <- mean (calib.dat$c)
#Calculate the rootMSE & Regression function for Ex2lm1
#First, use the SS function to calculate Sum of squares of C
#Save the SSC as a vector
SS<-function(x) {
  VC<- var(x);  N<-length(x)
  SSN<-VC*(N-1)
}
SSC <- SS(calib.dat$c)
#Save regression coefficients from Ex2lm1
CF2  <-  round (coef(Ex2lm1),  digits = 2)
Intercept <- CF2[1]
Slope <- CF2[2]
#Calculate RMSE from regression model
RSS  <-  c(crossprod(residuals(Ex2lm1)))
#Calculate mean squared error for model:
MSE <-  RSS / (df.residual(Ex2lm1))
#Calculate Root MSE for model:
RMSE <-  sqrt(MSE)
#Pearson estimated residual variance using REML:
SIG2 <- RSS / (df.residual(Ex2lm1))
N <- length(calib.dat$c)
ext <- (230)
Ex2.stats <- rbind(Intercept, Slope, SSC, RSS, MSE, RMSE, SIG2, N,
ext)
print(Ex2.stats)
#Use Investr package to calculate inverse regression
library(investr)
#Enter lm model name, extinction reading, alpha level
#Enter upper and lower limits on concentration
(lims <- calibrate(Ex2lm1, y0=230,
         interval = 'inversion', level = 0.95))
print (lims)
```

Output 6.5 kagc

```
---------------------------------------------------------------------------------------------
> #Build a dataset named calib.dat from scratch
> #First generate a vector for each variable where
> #c=the phosporus concentration
> #ext=the colorimatric extinction value
> obs_no <- c(1:5)
> c <- c(0.5,1.0,1.5,3.0,5.0)
> ext <- c(55.5,77.0,100.7,165.5,253.4)
> #Bind the vectors into one dataframe.
```

```
> calib.dat <- data.frame(cbind(obs_no, c, ext))
> #Obtain basic statistics for the variables
> summary(calib.dat)
     obs_no         c             ext
 Min.   :1   Min.   :0.5   Min.   : 55.5
 1st Qu.:2   1st Qu.:1.0   1st Qu.: 77.0
 Median :3   Median :1.5   Median :100.7
 Mean   :3   Mean   :2.2   Mean   :130.4
 3rd Qu.:4   3rd Qu.:3.0   3rd Qu.:165.5
 Max.   :5   Max.   :5.0   Max.   :253.4
> #Linear regression for extintion on concentration
> Ex2lm1 <- lm(ext ~ c, data = calib.dat)
> summary(Ex2lm1)

Call:
lm(formula = ext ~ c, data = calib.dat)

Residuals:
      1       2       3       4       5
-0.18647 -0.66692  1.05263 -0.08872 -0.11053

Coefficients:
            Estimate Std. Error t value   Pr(>|t|)
(Intercept)  33.7060     0.5497   61.31 0.00000956 ***
c            43.9609     0.2007  219.00 0.00000021 ***
---------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.732 on 3 degrees of freedom
Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
F-statistic: 4.796e+04 on 1 and 3 DF,  p-value: 0.0000002099

> anova(Ex2lm1)
Analysis of Variance Table

Response: ext
          Df  Sum Sq Mean Sq F value       Pr(>F)
c          1 25703.1 25703.1   47963 0.0000002099 ***
Residuals  3     1.6     0.5
---------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> #Calculate the mean for initial C values in calib.dat
> #Save mean as a vector
> cmean <- mean (calib.dat$c)
> #Calculate the rootMSE & Regression function for Ex2lm1
> #First, use the SS function to calculate Sum of squares of C
> #Save the SSC as a vector . . . TRUNCATED
> Ex2.stats <- rbind(Intercept, Slope, SSC, RSS, MSE, RMSE, SIG2, N,
ext)
> print(Ex2.stats)
           (Intercept)
Intercept  33.7100000
Slope      43.9600000
SSC        13.3000000
```

```
RSS          1.6076692
MSE          0.5358897
RMSE         0.7320449
SIG2         0.5358897
N            5.0000000
ext        230.0000000
> #Use Investr package to calculate inverse regression
> library(investr)
> #Enter lm model name, extinction reading, alpha level
> #Enter upper and lower limits on concentration
> (lims <- calibrate(Ex2lm1, y0=230,
+  interval = 'inversion', level = 0.95))
estimate    lower    upper
4.465195 4.398929 4.532417
> print (lims)
estimate    lower    upper
4.465195 4.398929 4.532417
```

==========

Example 3. Generate Dataset, linear regression, summary statistics, model comparisons, plus Tables 2 and 3.

In a pasture survey, the dependency of the crude fiber content, g/kg biomass y, on the cutting date, x, needs to be analyzed. Both variables are quantitative, y is a continuous random variable, and the time points are mixed Model I. Five dates were fixed at intervals of d. The first date was set to zero. Four samples were randomly drawn from the field at each date so that for each x value n = observations exist, and these N = 4 ´ = observations can be assumed to be independent. Several models are run and model fits are compared in Tables 2 and 3. Plots for Example 3, Figures 2, 3, 6, and 7 can be generated as in Example 1, Chapter 6.

Script 6.6 kagc

```
#Read in dataset as and attach as dataframe as above
fibre.dat <- read.csv('Fibre.csv',
header = TRUE,sep=',')
attach(fibre.dat)

# Change variable types
for ( var  in c('Obs','Block') )
{fibre.dat [,var] <-as.factor(fibre.dat[,var]) }
for ( var  in c('Day', 'Content') )
{ fibre.dat[,var] <-as.numeric(fibre.dat[,var]) }
str(fibre.dat)
summary(fibre.dat)

#Table 2A. Analyses of variance (ANOVA), estimation and tests of fixed
parameters without lack of it test
#Simple regression without blocking variable
Ex3lm1 <- lm(Content ~ Day, data=fibre.dat)
print(Ex3lm1)
#Save the high resolution plot
png('Ch6.Fig10.png', width = 8, height = 8,
```

```
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
plot(Ex3lm1)
dev.off()

#Table 2 B';
#Use the lack fit function below.
#The results are the F and p values that in table 2B.
lack.fit <- function(mod) {
  stopifnot(class(mod) == 'lm')
  z <- data.matrix(mod$model, rownames.force=NA)
  y <- z[, 1]
  x <- z[, 2]
  n <- length (x)
  m <- length( unique(x))
  if (m == n) stop
  ('The test requires xs not all different')
  SSres <- sum(residuals(mod)^2)
  SSep <- sum(sapply(split(x=y, f=x),
                     function(w) sum((w-mean(w))^2)))
  SSfa <- SSres - SSep
  Fo <- ((SSfa / (m-2)) / (SSep / (n-m)))
  p.value <- pf(Fo, df1=m-2, df2=n-m, lower.tail=F)
  res <- list(SSep=SSep, SSres=SSres, SSfa=SSfa,
              Fo=Fo, p.value=p.value)
  class(res) <- 'lackfit'
  res
}

lof <- lack.fit(Ex3lm1)
print(lof)

#Table 2.C ANOVA table if Day is handled as a quanitative treatment
factor
Ex3lm2 <- lm(Content ~ factor(Day), data=fibre.dat)
#Ex3lm2 residual SE from = error estimate of true σ.
#The residual error differences between Ex3lm1 and #Ex3lm2 is the lack
of fit (LOF)
anova(Ex3lm2)
summary(Ex3lm2)

#Compare models
#Calculate the SS, df, MSE for the 'pure error' from
#the second model 'by hand':
SSPERR <- sum(Ex3lm2$res^2) #is the 'pure error'
DFPERR <- Ex3lm2$df # the df for the pure error
MSPERR <- (SSPERR/DFPERR)

#Calculate the SS, df, MSE for the error from the first #model 'by
hand':
SSERR <- sum(Ex3lm1$res^2)
DFERR <- Ex3lm1$df
MSERR  <- (SSERR/DFERR)
```

```
#Calculate the SS, MS and DF for LOF
SSLOF <-(sum(Ex3lm1$res^2) - sum(Ex3lm2$res^2))
DFLOF <- (Ex3lm1$df - Ex3lm2$df)
MSLOF <- SSLOF/DFLOF

#Calculate the test statistic (F) for LOF
#Test significance
f <- (SSLOF / DFLOF)/(MSPERR) # test statistic for LOF
list(f)
p <- 1-pf(f, DFLOF, DFPERR) # the p-value
print (f)
print (p)
#Somewhat easier way to get the same result
#Compute the ANOVA table for two models
anova(Ex3lm1, Ex3lm2)
#Because LOF is not significant, Model 1 is adequate

#List the regression parameters Table 2.D
summary(Ex3lm1)

#TITLE 'Example 3: Calculation of the means per day';
#Calculate means by Day for Content
#Add to original dataset \
#Create a new dataset with just the means
fibre.dat$mnContent <- with(fibre.dat,
                            ave(Content, Day, FUN=mean))
b1 <- subset (fibre.dat, Block = 1)
mnfibre.dat <- within(b1, rm(Block, Content))

#LM based on the means as in Table 3, P 105
Ex3lm3 <-lm( mnContent ~ Day, data = mnfibre.dat)
anova(Ex3lm3)
summary(Ex3lm3)

#Use the HH package to draw a graph with CI and PI
library(HH)
#Save the high resolution plot
png('Ch6.Fig11.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
ci.plot(Ex3lm3)
dev.off()
```

Output 6.6 kagc

```
> #Table 2A. Analyses of variance (ANOVA), estimation and tests of
fixed parameters without lack of it test
> #Simple regression without blocking variable
> Ex3lm1 <- lm(Content ~ Day, data=fibre.dat)
> print(Ex3lm1)

Call:
```

```
lm(formula = Content ~ Day, data = fibre.dat)

Coefficients:
(Intercept)          Day
`   227.400        5.645

> #Save the high resolution plot
> png('Ch6.Fig10.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(2,2))
> plot(Ex3lm1)
> dev.off()
>
> #Table 2 B';
> #Use the lack fit function below.
> #The results are the F and p values that in table 2B.
> lack.fit <- function(mod) {
+   stopifnot(class(mod) == 'lm')
+   z <- data.matrix(mod$model, rownames.force=NA)
+   y <- z[, 1]
+   x <- z[, 2]
+   n <- length (x)
+   m <- length( unique(x))
+   if (m == n) stop
+   ('The test requires xs not all different')
+   SSres <- sum(residuals(mod)^2)
+   SSep <- sum(sapply(split(x=y, f=x),
+     function(w) sum((w-mean(w))^2)))
+   SSfa <- SSres - SSep
+   Fo <- ((SSfa / (m-2)) / (SSep / (n-m)))
+   p.value <- pf(Fo, df1=m-2, df2=n-m, lower.tail=F)
+   res <- list(SSep=SSep, SSres=SSres, SSfa=SSfa,
+               Fo=Fo, p.value=p.value)
+   class(res) <- 'lackfit'
+   res
+ }
>
> lof <- lack.fit(Ex3lm1)
> print(lof)
Fo:
[1] 0.2099579
p.value:
[1] 0.8879234
>
> #Table 2.C ANOVA table if Day is handled as a quanitative treatment
factor
> Ex3lm2 <- lm(Content ~ factor(Day), data=fibre.dat)
> #Ex3lm2 residual SE from = error estimate of true σ.
> #The residual error differences between Ex3lm1 and #Ex3lm2 is the
lack of fit (LOF)
> anova(Ex3lm2)
Analysis of Variance Table
```

```
Response: Content
‘            Df Sum Sq Mean Sq F value       Pr(>F)
factor(Day)   4  32013  8003.3  34.229 0.0000002231 ***
Residuals    15   3507   233.8
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(Ex3lm2)

Call:
lm(formula = Content ~ factor(Day), data = fibre.dat)

Residuals:
‘   Min      1Q  Median      3Q     Max
-23.250  -9.188  -0.875  11.375  27.000

Coefficients:
‘             Estimate Std. Error t value Pr(>|t|)
(Intercept)    227.750      7.646  29.789 9.23e-15 ***
factor(Day)5    29.000     10.812   2.682  0.01706 *
factor(Day)10   51.750     10.812   4.786  0.00024 ***
factor(Day)15   88.250     10.812   8.162 6.72e-07 ***
factor(Day)20  111.500     10.812  10.312 3.33e-08 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.29 on 15 degrees of freedom
Multiple R-squared:  0.9013, Adjusted R-squared:  0.8749
F-statistic: 34.23 on 4 and 15 DF,  p-value: 0.0000002231

>
> #Compare models
> #Calculate the SS, df, MSE for the 'pure error' from
> #the second model 'by hand':
> SSPERR <- sum(Ex3lm2$res^2) #is the 'pure error'
> DFPERR <- Ex3lm2$df # the df for the pure error
> MSPERR <- (SSPERR/DFPERR)   . . . TRUNCATED
>
> #Calculate the test statistic (F) for LOF
> #Test significance
> f <- (SSLOF / DFLOF)/(MSPERR) # test statistic for LOF
> list(f)
[[1]]
[1] 0.2099579

> p <- 1-pf(f, DFLOF, DFPERR) # the p-value
> print (f)
[1] 0.2099579
> print (p)
[1] 0.8879234
> #Somewhat easier way to get the same result
> #Compute the ANOVA table for two models
> anova(Ex3lm1, Ex3lm2)
Analysis of Variance Table
```

```
Model 1: Content ~ Day
Model 2: Content ~ factor(Day)
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1     18 3654.5
2     15 3507.3  3    147.28 0.21 0.8879
> #Because LOF is not significant, Model 1 is adequate
>
> #List the regression parameters Table 2.D
> summary(Ex3lm1)

Call:
lm(formula = Content ~ Day, data = fibre.dat)

Residuals:
`    Min       1Q   Median       3Q      Max
-26.8500  -8.9875  -0.0125  10.9250  30.9250

Coefficients:
`           Estimate Std. Error t value       Pr(>|t|)
(Intercept) 227.4000     5.5185   41.21        < 2e-16 ***
Day           5.6450     0.4506   12.53 0.000000000252 ***
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 14.25 on 18 degrees of freedom
Multiple R-squared:  0.8971, Adjusted R-squared:  0.8914
F-statistic:   157 on 1 and 18 DF,  p-value: 0.0000000002515

>
> #TITLE 'Example 3: Calculation of the means per day';
> #Calculate means by Day for Content
> #Add to original dataset  . . . TRUNCATED
>
> #LM based on the means as in Table 3, P 105
> Ex3lm3 <-lm( mnContent ~ Day, data = mnfibre.dat)
> anova(Ex3lm3)
Analysis of Variance Table

Response: mnContent
`          Df Sum Sq Mean Sq F value    Pr(>F)
Day         1  31866   31866  3894.7 < 2.2e-16 ***
Residuals 18    147       8
-----------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> summary(Ex3lm3)

Call:
lm(formula = mnContent ~ Day, data = mnfibre.dat)

Residuals:
`   Min      1Q  Median      3Q     Max
-4.350 -1.050  0.350  1.125  3.925

Coefficients:
```

```
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) 227.40000    1.10783  205.27   <2e-16 ***
Day           5.64500    0.09045   62.41   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.86 on 18 degrees of freedom
Multiple R-squared:  0.9954, Adjusted R-squared:  0.9951
F-statistic:  3895 on 1 and 18 DF,  p-value: < 2.2e-16

>
> #Use the HH package to draw a graph with CI and PI
> library(HH)
> #Save the high resolution plot
> png('Ch6.Fig11.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> ci.plot(Ex3lm3)
> dev.off()
```

==========

Example 3 (continued). Fixed and random block models.

Later in Chapter 6, the Example 3 dataset is used to fit fixed and random models for Block. These analyses are described in Tables 18. R does not have a function for the standard error of a predicted random effect like SAS so these are bootstrapped intervals. They do not exactly match those from SAS as reported on the right side of Table 19B in Chapter 6; however, except for one limit, there is less than 7% discrepancy. Note that the prediction interval was for the actual observation so that the intercept and the average Day effects had to be subtracted (total of 283.85) to get the predicted block effects.

Script 6.7 kagc

```
#Further analyses of Example 3.
#Example 3 (modified): Table 18 A';
#See Example 7 for additional explanation
#Earlier analysis (Ex3lm1) considered this
#experiment as a CRD design
#Repeat of CRD analysis.
Ex3lm1 <- lm(Content ~ Day, data=fibre.dat)
anova(Ex3lm1)
print(summary(Ex3lm1)); confint(Ex3lm1)

#Run the model including Block as a fixed effect.
#This is the same as approach B for Ex 7
#Use the contr.SAS function to include contrasts for #the block
effects, calculated as in SAS
contrasts (fibre.dat$Block) = contr.SAS(4)
Ex3lm4 <- lm(Content ~ Day + Block, data=fibre.dat)
#This is Table 18A.
#Both effects are significant
summary(Ex3lm4)
#This model improves the R2, and the T value
```

```
#The parameter estimates are in Table 18B
confint (Ex3lm4)
anova(Ex3lm4)
(round (anova(Ex3lm4,  test ='F'), 2))

#Use lsmeans function to print out treatment means
lsmeans (Ex3lm1, 'Day',
         at = list(Day = c(0, 5, 10, 15, 20)))
#lsmeans is being depracated so can also use emmeans for the other
model
library(emmeans)
emmeans (Ex3lm4, 'Day',
         at = list(Day = c(0, 5, 10, 15, 20)))

#Example 3: Tables 19 and 20
#broad inference' with block random
library(lme4)
#Run mixed model with blocks as a random effect
Ex3lmer2 <- lmer(Content ~ Day + (1|Block),
                 data = fibre.dat)
anova(Ex3lmer2)
summary(Ex3lmer2)

#Random effects are BLUPs, fixed effects are BLUEs
#Print out means, fixed and random effects
lsmeans(Ex3lmer2, 'Day',
        at = list(Day = c(0, 5, 10, 15, 20)))
#Obtain CI for fixed effects with the confint function
fixef(Ex3lmer2)
confint(Ex3lmer2, method = 'profile')
ranef(Ex3lmer2)

#use bootstrapping to get PI for the BLUPs
p.int1 <- predictInterval(Ex3lmer2,
                          newdata = (fibre.dat), n.sims=500,
stat='mean',     include.resid.var=T, level=0.95)
p.int2 <- data.frame(fibre.dat, p.int1)
p.int2 %>%
  group_by(Block) %>%
  summarize(av_lwr=(mean(lwr)-283.85),
            av_upr=(mean(upr)-283.85))

#Example 3: Table 20;
#The treatment means on in the first column of
#table 20 were calculated above with emmeans package.
#To calculate the prediction interval merTools
library(merTools) #predictInterval()
p.int1 <- predictInterval(Ex3lmer2, newdata=(fibre.dat),
                          n.sims=500, stat='mean',
include.resid.var=T, level=0.95)
p.int2 <- data.frame(fibre.dat, p.int1)
p.int2 %>%
  group_by(Block) %>%
  summarize(av_lwr=(mean(lwr)-283.85),
```

```
            av_upr=(mean(upr)-283.85))

#The CI are calculated in each of the following calls of the emmeans
function:
#CRD model
emmeans (Ex3lm1, 'Day',
 at = list(Day = c(0, 5, 10, 15, 20)))
#RDBD, Fixed Blocks:
emmeans (Ex3lm4, 'Day',
   at = list(Day = c(0, 5, 10, 15, 20)))
#RCBD model, Random blocks
emmeans (Ex3lmer2, 'Day',
         at = list(Day = c(0, 5, 10, 15, 20)))
```

Output 6.7 kagc

```
#Further analyses of Example 3.
> #Example 3 (modified): Table 18 A';
> #See Example 7 for additional explanation
> #Earlier analysis (Ex3lm1) considered this
> #experiment as a CRD design
> #Repeat of CRD analysis.
> Ex3lm1 <- lm(Content ~ Day, data=fibre.dat)
> anova(Ex3lm1)
Analysis of Variance Table

Response: Content
`          Df Sum Sq Mean Sq F value           Pr(>F)
Day         1  31866   31866  156.95 0.0000000002515 ***
Residuals 18   3655     203
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> print(summary(Ex3lm1)); confint(Ex3lm1)

Call:
lm(formula = Content ~ Day, data = fibre.dat)

Residuals:
`    Min       1Q   Median       3Q      Max
-26.8500  -8.9875  -0.0125  10.9250  30.9250

Coefficients:
`            Estimate Std. Error t value       Pr(>|t|)
(Intercept) 227.4000     5.5185   41.21        < 2e-16 ***
Day           5.6450     0.4506   12.53 0.000000000252 ***
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 14.25 on 18 degrees of freedom
Multiple R-squared:  0.8971, Adjusted R-squared:  0.8914
F-statistic:   157 on 1 and 18 DF,  p-value: 0.0000000002515

`                2.5 %     97.5 %
```

```
(Intercept) 215.805960 238.994040
Day           4.698351   6.591649
>
> #Run the model including Block as a fixed effect.
> #This is the same as approach B for Ex 7
> #Use the contr.SAS function to include contrasts for #the block
effects, calculated as in SAS
> contrasts (fibre.dat$Block) = contr.SAS(4)
> Ex3lm4 <- lm(Content ~ Day + Block, data=fibre.dat)
> #This is Table 18A.
> #Both effects are significant
> summary(Ex3lm4)

Call:
lm(formula = Content ~ Day + Block, data = fibre.dat)

Residuals:
   Min     1Q Median     3Q    Max
-9.800 -4.562 -1.038  2.987 14.975

Coefficients:
`           Estimate Std. Error t value Pr(>|t|)
(Intercept) 243.3500     3.8719  62.851  < 2e-16 ***
Day           5.6450     0.2235  25.252 1.05e-13 ***
Block1      -33.0000     4.4709  -7.381 2.29e-06 ***
Block2      -19.6000     4.4709  -4.384 0.000534 ***
Block3      -11.2000     4.4709  -2.505 0.024260 *
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.069 on 15 degrees of freedom
Multiple R-squared:  0.9789, Adjusted R-squared:  0.9733
F-statistic:   174 on 4 and 15 DF,  p-value: 2.258e-12

> #This model improves the R2, and the T value
> #The parameter estimates are in Table 18B
> confint (Ex3lm4)
`                2.5 %     97.5 %
(Intercept) 235.097271 251.602729
Day           5.168528   6.121472
Block1      -42.529431 -23.470569
Block2      -29.129431 -10.070569
Block3      -20.729431  -1.670569
> anova(Ex3lm4)
Analysis of Variance Table

Response: Content
`         Df Sum Sq Mean Sq F value    Pr(>F)
Day        1  31866   31866 637.682 1.051e-13 ***
Block      3   2905     968  19.377 2.029e-05 ***
Residuals 15    750      50
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> (round (anova(Ex3lm4,  test ='F'), 2))
```

```
Analysis of Variance Table

Response: Content
`          Df Sum Sq Mean Sq F value    Pr(>F)
Day         1  31866   31866  637.68 < 2.2e-16 ***
Block       3   2905     968   19.38 < 2.2e-16 ***
Residuals 15    750      50
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
>
> #Use lsmeans function to print out treatment means
> lsmeans (Ex3lm1, 'Day',
+          at = list(Day = c(0, 5, 10, 15, 20)))
Day  lsmean       SE df lower.CL upper.CL
` 0 227.400 5.518548 18 215.8060 238.9940
` 5 255.625 3.902203 18 247.4268 263.8232
 10 283.850 3.186135 18 277.1562 290.5438
 15 312.075 3.902203 18 303.8768 320.2732
 20 340.300 5.518548 18 328.7060 351.8940

Confidence level used: 0.95
> #lsmeans is being depracated so can also use emmeans for the other
model
> library(emmeans)
> emmeans (Ex3lm4, 'Day',
+          at = list(Day = c(0, 5, 10, 15, 20)))
 Day  emmean       SE df lower.CL upper.CL
   0 227.400 2.737837 15 221.5644 233.2356
   5 255.625 1.935943 15 251.4986 259.7514
  10 283.850 1.580691 15 280.4808 287.2192
  15 312.075 1.935943 15 307.9486 316.2014
  20 340.300 2.737837 15 334.4644 346.1356

Results are averaged over the levels of: Block
Confidence level used: 0.95
>
> #Example 3: Tables 19 and 20
> #broad inference' with block random
> library(lme4)
> #Run mixed model with blocks as a random effect
> Ex3lmer2 <- lmer(Content ~ Day + (1|Block),
+                  data = fibre.dat)
> anova(Ex3lmer2)
Analysis of Variance Table
    Df Sum Sq Mean Sq F value
Day  1  31866   31866  637.68
> summary(Ex3lmer2)
Linear mixed model fit by REML ['lmerMod']
Formula: Content ~ Day + (1 | Block)
   Data: fibre.dat

REML criterion at convergence: 140.3

Scaled residuals:
```

```
`    Min      1Q  Median      3Q     Max
-1.5108 -0.5308 -0.1734  0.3226  2.2348

Random effects:
 Groups   Name        Variance Std.Dev.
 Block    (Intercept) 183.67   13.552
 Residual              49.97    7.069
Number of obs: 20, groups:  Block, 4

Fixed effects:
`           Estimate Std. Error t value
(Intercept) 227.4000     7.3084   31.11
Day           5.6450     0.2235   25.25

Correlation of Fixed Effects:
`    (Intr)
Day -0.306
>
> #Random effects are BLUPs, fixed effects are BLUEs
> #Print out means, fixed and random effects
> lsmeans(Ex3lmer2, 'Day',
+         at = list(Day = c(0, 5, 10, 15, 20)))
Day  lsmean       SE   df lower.CL upper.CL
` 0 227.400 7.308420 3.64 206.3010 248.4990
` 5 255.625 7.047349 3.16 233.8129 277.4371
 10 283.850 6.958149 3.00 261.7061 305.9939
 15 312.075 7.047349 3.16 290.2629 333.8871
 20 340.300 7.308420 3.64 319.2010 361.3990

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
> #Obtain CI for fixed effects with the confint function
> fixef(Ex3lmer2)
(Intercept)         Day
`   227.400       5.645
> confint(Ex3lmer2, method = 'profile')
Computing profile confidence intervals ...
`                2.5 %     97.5 %
.sig01        6.091591  29.395986
.sigma        5.016767  10.125054
(Intercept) 211.715010 243.085006
Day           5.193991   6.096009
> ranef(Ex3lmer2)
$Block
` (Intercept)
1  -16.170105
2   -3.461635
3    4.504868
4   15.126873

> #use bootstrapping to get PI for the BLUPs
> p.int1 <- predictInterval(Ex3lmer2,
+  newdata = (fibre.dat), n.sims=500, stat='mean',
      include.resid.var=T, level=0.95)
```

```
> p.int2 <- data.frame(fibre.dat, p.int1)
> p.int2 %>%
+   group_by(Block) %>%
+   summarize(av_lwr=(mean(lwr)-283.85),
+             av_upr=(mean(upr)-283.85))
# A tibble: 4 x 3
  Block av_lwr av_upr
  <fct>  <dbl>  <dbl>
1 1      -38.2   5.30
2 2      -25.4   18.3
3 3      -17.3   26.7
4 4      - 7.33  37.7
>
> #Example 3: Table 20;
> #The treatment means on in the first column of
> #table 20 were calculated above with emmeans package.
> #To calculate the prediction interval merTools
> library(merTools) #predictInterval()
> p.int1 <- predictInterval(Ex3lmer2,
+     newdata=(fibre.dat),
+   n.sims=500, stat='mean', include.resid.var=T, level=0.95)
> p.int2 <- data.frame(fibre.dat, p.int1)
> p.int2 %>%
+   group_by(Block) %>%
+   summarize(av_lwr=(mean(lwr)-283.85),
+             av_upr=(mean(upr)-283.85))
# A tibble: 4 x 3
  Block av_lwr av_upr
  <fct>  <dbl>  <dbl>
1 1      -38.4   5.76
2 2      -25.4   17.6
3 3      -19.1   25.3
4 4      - 7.15  35.6
>
> #The CI are calculated in each of the following calls of the emmeans
function:
> #CRD model
> emmeans (Ex3lm1, 'Day',
+  at = list(Day = c(0, 5, 10, 15, 20)))
 Day  emmean       SE df lower.CL upper.CL
   0 227.400 5.518548 18 215.8060 238.9940
   5 255.625 3.902203 18 247.4268 263.8232
  10 283.850 3.186135 18 277.1562 290.5438
  15 312.075 3.902203 18 303.8768 320.2732
  20 340.300 5.518548 18 328.7060 351.8940

Confidence level used: 0.95
> #RDBD, Fixed Blocks:
> emmeans (Ex3lm4, 'Day',
+    at = list(Day = c(0, 5, 10, 15, 20)))
 Day  emmean       SE df lower.CL upper.CL
   0 227.400 2.737837 15 221.5644 233.2356
   5 255.625 1.935943 15 251.4986 259.7514
  10 283.850 1.580691 15 280.4808 287.2192
```

```
  15 312.075 1.935943 15 307.9486 316.2014
  20 340.300 2.737837 15 334.4644 346.1356

Results are averaged over the levels of: Block
Confidence level used: 0.95
> #RCBD model, Random blocks
> emmeans (Ex3lmer2, 'Day',
+          at = list(Day = c(0, 5, 10, 15, 20)))
 Day  emmean       SE   df lower.CL upper.CL
   0 227.400 7.308420 3.64 206.3010 248.4990
   5 255.625 7.047349 3.16 233.8129 277.4371
  10 283.850 6.958149 3.00 261.7061 305.9939
  15 312.075 7.047349 3.16 290.2629 333.8871
  20 340.300 7.308420 3.64 319.2010 361.3990

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
==========================================================
```

Example 4. Read data, do correlations, and study transformations.

Example 4 examines the relationship between weed infestation of windgrass and plot yield. Both the regressor and the regressand are random variables. The windgrass data (x) are counts so they are integer rather than numerical variables. The data are in the data set named grass.dat. The windgrass counts (x) and plot yield (y) are also ranked (rx and ry) so that the Spearman's correlation can be calculated. We will not use the rx and ry variables because we will use the spearman function to calculate these correlations. Other data manipulation in this example includes a square root transformation applied to the windgrass counts and a log transformation applied to the plot yields. In the code below, the data are loaded and then renamed and transformed. The scatter plot of the data indicated that the relationship was nonlinear so the point of this example is to use transformation and nonlinear regression to model the relationship among the windgrass counts and the plot yield. As with previous examples, Fig. 2, 3, 6, and 7 are analogous to Example 1. We will look at correlations among the variables and residuals diagnosis as in Fig. 6 and 7.

Script 6.8 kagc

```
-------------------------------------------------------------------------------------------------
#Read in Ex 4 data and attach as previously
grass.dat <- read.csv('grass.csv',
                      header = TRUE,sep=',', )
attach(grass.dat)
head(grass.dat)
tail(grass.dat)
str(grass.dat)

# Change variable types
for ( var  in c('x') )
{ grass.dat[,var] <-as.integer(grass.dat[,var]) }
for ( var  in c('y') )
{ grass.dat[,var] <-as.numeric(grass.dat[,var]) }
str(grass.dat)
#rename and transform variables for clarity
grass.dat$wgrass <-grass.dat$x
grass.dat$x <- NULL
```

```
grass.dat$yield <- grass.dat$y
grass.dat$y  <- NULL
grass.dat$sqrtwgrass <- sqrt(grass.dat$wgrass)
grass.dat$logyield <- log(grass.dat$yield)

#Calculate basic statistics
summary(grass.dat)

#Run Spearman and Pearson correlations
cmats <- cor(grass.dat$wgrass, grass.dat$yield, method='spearman', use
= 'complete.obs')
list(cmats)
#test of signficance
cor.test(grass.dat$wgrass, grass.dat$yield,
method = c('spearman'), conf.level = 0.95)
#ignore the warning message

#Calculate correlations
cmatp<-cor(grass.dat$wgrass, grass.dat$yield, method='pearson', use =
'complete.obs')
list(cmatp)
#test of signficance for correlation
cor.test(grass.dat$wgrass, grass.dat$yield,
method = c('pearson'), conf.level = 0.95)
#The linearity of the relationship is improved with the
#transformations but still approximate
#Use the pairs command with the panel.cor function
#to visualize the pearson correlations

#print out a panel of the correlations
png('Ch6.Fig12.png', width = 8, height = 8, units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))

panel.cor <- function(x, y, digits = 2, prefix = '', cex.cor, ...)
{
  usr <- par('usr'); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if (missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs (~ wgrass + sqrtwgrass + yield + logyield, data=grass.dat,
lower.panel = panel.smooth,  upper.panel = panel.cor)
dev.off()

#Conduct simple linear regression
Ex4lm1 <- lm(yield ~ sqrtwgrass, data=grass.dat)
summary(Ex4lm1)
#Calculate shapiro.test, and ks.test for normality
#Reproduce qqtests
sw <-shapiro.test(rstudent(Ex4lm1))
```

```
ks <-ks.test(rstudent(Ex4lm1), 'pnorm')
print(sw)
print(ks)
#plot several residuals plots together
png('Ch6.Fig13.png', width = 8, height = 8, units =
'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
qqnorm(rstudent(Ex4lm1))
qqline(rstudent(Ex4lm1), col = 'red')
#The Shapiro-Wilks and the Kolmogorov-Smirnov test #statistic are
different than those in Fig. 6.
#The QQ plots look similar.
#Studentized residuals against X values as in Fig. 7.
plot(grass.dat$sqrtwgrass, rstudent(Ex4lm1),
ylim = c(-3,3), new = TRUE)
abline(h = 0, lty = 3)
abline(h = 2, lty = 1)
abline(h = -2, lty = 1)
lev <- hat(model.matrix(Ex4lm1))
plot(lev)
dev.off()
```

Output 6.8 kagc

```
> #Read in Ex 4 data and attach as previously
> grass.dat <- read.csv('grass.csv',
+                       header = TRUE,sep=',', )
> attach(grass.dat)
The following objects are masked from grass.dat (pos = 3):
Obs, Rx, Ry, x, y

The following objects are masked from grass.dat (pos = 4):
Obs, Rx, Ry, x, y

The following object is masked from fibre.dat:
Obs

> head(grass.dat)
  Obs  Rx x    y   Ry
1   1 1.5 0 9310 48.0
2   2 1.5 0 8460 43.0
3   3 4.0 1 9770 52.0
4   4 4.0 1 9320 49.0
5   5 4.0 1 8620 44.0
6   6 6.0 2 7850 36.5
> tail(grass.dat)
   Obs Rx    x    y Ry
47  47 47  311 4130 10
48  48 48  337 3050  7
49  49 49  901 1740  2
50  50 50  927 1750  3
51  51 51 1102 1980  4
```

```
52  52 52 1204  540  1
> str(grass.dat)
'data.frame':    52 obs. of  5 variables:
 $ Obs: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Rx : num  1.5 1.5 4 4 4 6 7.5 7.5 9 10 ...
 $ x  : int  0 0 1 1 1 2 3 3 4 5 ...
 $ y  : int  9310 8460 9770 9320 8620 7850 9520 9080 6340 9340 ...
 $ Ry : num  48 43 52 49 44 36.5 51 46 22 50 ...
>
> # Change variable types
> for ( var  in c('x') )
+ { grass.dat[,var] <-as.integer(grass.dat[,var]) }
> for ( var  in c('y') )
+ { grass.dat[,var] <-as.numeric(grass.dat[,var]) }
> str(grass.dat)
'data.frame':    52 obs. of  5 variables:
 $ Obs: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Rx : num  1.5 1.5 4 4 4 6 7.5 7.5 9 10 ...
 $ x  : int  0 0 1 1 1 2 3 3 4 5 ...
 $ y  : num  9310 8460 9770 9320 8620 7850 9520 9080 6340 9340 ...
 $ Ry : num  48 43 52 49 44 36.5 51 46 22 50 ...
> #rename and transform variables for clarity
> grass.dat$wgrass <-grass.dat$x . . . TRUNCATED
>
> #Calculate basic statistics
> summary(grass.dat)
Font size reduced here so whole table can be shown
`      Obs              Rx              Ry            wgrass          yield
 Min.   : 1.00   Min.   : 1.50   Min.   : 1.00   Min.   :   0.0   Min.   : 540
 1st Qu.:13.75   1st Qu.:13.75   1st Qu.:13.75   1st Qu.:  23.5   1st Qu.:4895
 Median :26.50   Median :26.75   Median :26.25   Median :  93.5   Median :6780
 Mean   :26.50   Mean   :26.50   Mean   :26.50   Mean   : 175.8   Mean   :6383
 3rd Qu.:39.25   3rd Qu.:39.25   3rd Qu.:39.25   3rd Qu.: 208.5   3rd Qu.:7985
 Max.   :52.00   Max.   :52.00   Max.   :52.00   Max.   :1204.0   Max.   :9770

`  sqrtwgrass        logyield
 Min.   : 0.000   Min.   :6.292
 1st Qu.: 4.847   1st Qu.:8.496
 Median : 9.665   Median :8.822
 Mean   :10.519   Mean   :8.653
 3rd Qu.:14.424   3rd Qu.:8.985
 Max.   :34.699   Max.   :9.187
>
> #Run Spearman and Pearson correlations
> cmats <- cor(grass.dat$wgrass, grass.dat$yield, method='spearman',
use = 'complete.obs')
> list(cmats)
[[1]]
[1] -0.904033

> #test of signficance
> cor.test(grass.dat$wgrass, grass.dat$yield,
+          method = c('spearman'),
+          conf.level = 0.95)

      Spearman's rank correlation rho
data:  grass.dat$wgrass and grass.dat$yield
```

```
S = 44604, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
-0.904033

Warning message:
In cor.test.default(grass.dat$wgrass, grass.dat$yield, method =
c("spearman"),  :
  Cannot compute exact p-value with ties
> #ignore the warning message, it's for information
>
> #Calculate correlations
> cmatp<-cor(grass.dat$wgrass, grass.dat$yield,    method='pearson',
use = 'complete.obs')
> list(cmatp)
[[1]]
[1] -0.8090715

> #test of signficance for correlation
> cor.test(grass.dat$wgrass, grass.dat$yield,
+          method = c('pearson'),
+          conf.level = 0.95)

      Pearson's product-moment correlation

data:  grass.dat$wgrass and grass.dat$yield
t = -9.7344, df = 50, p-value = 3.946e-13
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.886284 -0.688101
sample estimates:
       cor
-0.8090715

> #The linearity of the relationship is improved with
>  the transformations but still approximate
> #Use the pairs command with the panel.cor function
> #to visualize the pearson correlations
>
> #print out a panel of the correlations
> png('Ch6.Fig12.png', width = 8, height = 8, units =
> 'in', res = 600)
> #dev.off()
> par(mfrow=c(1,1))
>
> panel.cor <- function(x, y, digits = 2, prefix = '', cex.cor, ...)
+ {
+   usr <- par('usr'); on.exit(par(usr))
+   par(usr = c(0, 1, 0, 1))
+   r <- abs(cor(x, y)) . . . TRUNCATED

#Conduct simple linear regression
#with the transformed variable
```

```
Ex4lm1 <- lm(yield ~ sqrtwgrass, data=grass.dat)
> (Ex4lm1)

Call:
lm(formula = yield ~ sqrtwgrass, data = grass.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-2672.5  -538.2   149.5   498.8  2219.3

Coefficients:
`            Estimate Std. Error t value Pr(>|t|)
(Intercept)  9090.78     234.41   38.78   <2e-16 ***
sqrtwgrass   -257.42      17.68  -14.56   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 1029 on 50 degrees of freedom
Multiple R-squared:  0.8092, Adjusted R-squared:  0.8054
F-statistic:   212 on 1 and 50 DF,  p-value: < 2.2e-16
> #Calculate shapiro.test, and ks.test for normality
> #Reproduce qqtests
> sw <-shapiro.test(rstudent(Ex4lm1))
> ks <-ks.test(rstudent(Ex4lm1), 'pnorm')
> print(sw)

      Shapiro-Wilk normality test

data:  rstudent(Ex4lm1)
W = 0.97538, p-value = 0.3523

> print(ks)

      One-sample Kolmogorov-Smirnov test

data:  rstudent(Ex4lm1)
D = 0.072682, p-value = 0.9277
alternative hypothesis: two-sided

> #plot several residuals plots together
> png('Ch6.Fig13.png', width = 8, height = 8, units =
+       'in', res = 600) . . . TRUNCATED
> lev <- hat(model.matrix(Ex4lm1))
> plot(lev)
> dev.off()
#Datapoints 49-52 have high leverage.
#Variance heterogeneity and model inadequacy exist
#In Chapter 6, Table 4, the diagnositc statistics are printed for all
four examples.  We did not reproduce Table 4 here
=========================================================
```

Example 4. Regression with log of yield & Fig 8. non-linear regression

The square root transformation improved the model fit but did not correct for the variance heterogeneity. We will try two additional methods, use of log transformation and, finally we will use nonlinear regression.

Script 6.9 kagc

```
#Conduct linear regression with the
#log transformed yield data.
Ex4lm2 <- lm(logyield ~ wgrass, data = grass.dat)
str(grass.dat)
anova (Ex4lm2)
#Generate a few diagnostic plots
png('Ch6.Fig14.png', width = 8, height = 8, units =
'in', res = 600)
par(mfrow=c(2,2))
plot(Ex4lm2)
dev.off()

#the relationship between the two variables is nonlinear.
#The nonlinear regression formula recommended in Ch. 6 is:
#  yield = a*exp(b*wgrass), with additive errors;
#The nls function requires start values.
#We will use the start values from the Exlm2 model
parastart <- c(a = 8000, b = -0.01)
Ex4nls1 <- nls(y ~ (a*exp(b*wgrass)), data = grass.dat,
start = parastart,  trace=T)

#Viewing output of Nonlinear regression
prednls <-predict(Ex4nls1)
grass.dat2 <- cbind(grass.dat, prednls)
summary (Ex4nls1)
cor(grass.dat$yield, predict(Ex4nls1))
#This fit is better but there are outlying residuals
#Plot a series of fitted by actual data
#for the models we have tried.
png('Ch6.Fig15.png', width = 8, height = 8, units =
'in', res = 600)
par(mfrow=c(2,2))
plot(grass.dat$wgrass, grass.dat$yield,
xlab = 'wgrass', ylab = 'yield')
plot(grass.dat$yield, fitted.values(Ex4lm1),
xlab = 'yield', ylab = 'fit with sqrt(x)')
plot(grass.dat$logyield, fitted.values(Ex4lm2),
xlab = 'logyield', ylab = 'fit')
plot (grass.dat$yield, grass.dat$prednls,
xlab = 'yield', ylab = 'nlsfit')
dev.off()
png('Ch6.Fig16.png', width = 8, height = 8, units =
'in', res = 600)
plot (Ex4nls1)
dev.off()

#Change the start values based on the previous nls #model and
calculate confidence intervals.
```

```
parastart <- c(a = 7000, b = -0.01, c = 1200)
Ex4nls2 <- nls(y ~ ((a*exp(b*wgrass)) + c),
data = grass.dat, start = parastart, trace = T)
cor(grass.dat$y, predict(Ex4nls2))
png('Ch6.Fig17.png', width = 8, height = 8, units =
'in', res = 600)
par(mfrow=c(1,1))
#Compare two non linear regressions
plot(grass.dat$x, grass.dat$y, xlab = 'Windgrass no',
ylab = 'Predicted Yield with additive errors')
lines (grass.dat$x,predict(Ex4nls1),col = 'red' )
lines (grass.dat$x,predict(Ex4nls2),col = 'green')
dev.off()
#Plot CI for fitted values with ci.nls function.

grass.dat$x <- grass.dat$wgrass
# typical plot with confidence interval
grass.dat$pred <- predict(Ex4nls2)
se = summary(Ex4lm2)$sigma
ci = outer(grass.datf$pred, c(outer(se, c(-1,1), '*'))*1.96, '+')
ii = order(grass.dat$x)
png('Ch6.Fig18.png', width = 8, height = 8, units =
'in', res = 600)
par(mfrow=c(1,1))
# typical plot with confidence interval
with (grass.dat[ii,], plot(x, pred, ylim=range(ci), type='l' ))
matlines(df[ii,'x'], ci[ii,], lty=2, col=1)
# shaded area plot
low = ci[ii,1]; high = ci[ii,2]; base = grass.dat[ii,'x']
polygon(c(base,rev(base)), c(low,rev(high)), col='grey')
with(df[ii,], lines(x, pred, col='blue'))
with(grass.dat, points(x, yield))
dev.off()

#Modified from #https://stackoverflow.com/questions/32459480/r-
#confidence-bands-for-exponential-model-nls-in-basic-#graphics
#This was written as a function but the code wasn't #working so it has
been modified to work with renamed #datasets.
```

Output 6.9 kagc

```
#Conduct linear regression with the
> #log transformed yield data.
> Ex4lm2 <- lm(logyield ~ wgrass, data = grass.dat)
> str(grass.dat)
'data.frame':    52 obs. of  9 variables:
 $ Obs       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Rx        : num  1.5 1.5 4 4 4 6 7.5 7.5 9 10 ...
 $ Ry        : num  48 43 52 49 44 36.5 51 46 22 50 ...
 $ wgrass    : int  0 0 1 1 1 2 3 3 4 5 ...
 $ yield     : num  9310 8460 9770 9320 8620 7850 9520 9080 6340 9340
...
 $ sqrtwgrass: num  0 0 1 1 1 ...
```

```
 $ logyield  : num  9.14 9.04 9.19 9.14 9.06 ...
 $ x         : int  0 0 1 1 1 2 3 3 4 5 ...
 $ pred      : num  8804 8804 8778 8778 8778 ...
> anova (Ex4lm2)
Analysis of Variance Table

Response: logyield
`         Df  Sum Sq Mean Sq F value    Pr(>F)
wgrass     1 12.8696  12.870  238.53 < 2.2e-16 ***
Residuals 50  2.6976   0.054
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> #Generate a few diagnostic plots
> png('Ch6.Fig14.png', width = 8, height = 8, units =
+       'in', res = 600)
> par(mfrow=c(2,2))
> plot(Ex4lm2)
> dev.off()
null device
          1
> #the relationship between the two variables is nonlinear.
> #The nonlinear regression formula recommended in Ch. #6 is:
> #  yield = a*exp(b*wgrass), with additive errors;
> #The nls function requires start values.
> #We will use the start values from the Exlm2 model
> parastart <- c(a = 8000, b = -0.01)
> Ex4nls1 <- nls(y ~ (a*exp(b*wgrass)), data = grass.dat,
+                start = parastart,  trace=T)
484997310 :  8000.00   -0.01
76989719 :  8123.933742350   -0.003258723
48934034 :  8699.27072032   -0.00257744
48871879 :  8671.674995370   -0.002601013
48871718 :  8673.391104092   -0.002604146
48871716 :  8673.614111622   -0.002604538
48871716 :  8673.641950372   -0.002604587
>
> #Viewing output of Nonlinear regression
> prednls <-predict(Ex4nls1)
> grass.dat2 <- cbind(grass.dat, prednls)
> summary (Ex4nls1)

Formula: y ~ (a * exp(b * wgrass))

Parameters:
`     Estimate   Std. Error t value Pr(>|t|)
a 8673.6419504  230.5106602   37.63  < 2e-16 ***
b   -0.0026046    0.0002596  -10.03 1.45e-13 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 988.7 on 50 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 0.000003322
```

```
> cor(grass.dat$yield, predict(Ex4nls1))
[1] 0.909643
> #This fit is better but there are outlying residuals
> #Plot a series of fitted by actual data
> #for the models we have tried.
> png('Ch6.Fig15.png', width = 8, height = 8, units =
+       'in', res = 600)
par(mfrow=c(2,2)). . . TRUNCATED

> png('Ch6.Fig16.png', width = 8, height = 8, units =
+       'in', res = 600)
> plot (Ex4nls1)
> dev.off()
null device
          1
> #Change the start values based on the previous nls #model and
calculate confidence intervals.
> parastart <- c(a = 7000, b = -0.01, c = 1200)
> Ex4nls2 <- nls(y ~ ((a*exp(b*wgrass)) + c),
+   data = grass.dat, start = parastart, trace = T)
285637718 :  7000.00   -0.01 1200.00
107581726 :  6326.939503264   -0.001954413 2289.766208273
49518021 :  6592.147926679   -0.003806217 2168.165890243
45518672 :  7537.243901260   -0.003428736 1266.576815121
45477470 :  7551.659359456   -0.003475325 1252.456222986
45477468 :  7551.53608794   -0.00347602 1252.76805442
> cor(grass.dat$y, predict(Ex4nls2))
[1] 0.9143939
> png('Ch6.Fig17.png', width = 8, height = 8, units =
+       'in', res = 600)
> par(mfrow=c(1,1))
> #Compare two non linear regressions
> plot(grass.dat$x, grass.dat$y, . . . TRUNCATED

> #Plot CI for fitted values with ci.nls function.
>
> grass.dat$x <- grass.dat$wgrass
> # typical plot with confidence interval
> grass.dat$pred <- predict(Ex4nls2)
> se = summary(Ex4lm2)$sigma
> ci = outer(grass.datf$pred, c(outer(se, c(-1,1), '*'))*1.96, '+') .
. . TRUNCATED
> png('Ch6.Fig18.png', width = 8, height = 8, units =
+       'in', res = 600)
> par(mfrow=c(1,1)) . . . TRUNCATED
==========================================================
```

Example 5. Data input, calculation of the variables, Table 6 A TO D and Table 7.

Example 5 uses a dataset named space.dat to examine decisions about retaining variables in multiple regression. The variables are labelled X1, X2, R1, R2, R3, and R4. X1 and X2 refer to spacings of interseedling distances in cm in the row and column direction of the field experiment. The R

variables are the plot yields in kg for Replications 1-4. In order to use this dataset, it has to be reshaped. Several new variables are added: The X1 and X2 variables are combined into a single treatment variable, a plot area variable is calculated, the length to width ratio or the plot is calculated as the shape index 1 variable and shape index 2 variable reflects the deviation from a square to a rectangular shape.

Script 6.10 kagc '

```
#Read in data for Example 5 as previously
space.dat <- read.csv('space.csv',

#Calculate new variables.
space.dat$treatment <- (space.dat$x1*1000 + space.dat$x2)
space.dat$area <- (space.dat$x1*space.dat$x2)
space.dat$shape_ind1 = (space.dat$x1/space.dat$x2)
space.dat$shape_ind2 = ((space.dat$x1 +
space.dat$x2)/2/sqrt(space.dat$x1 * space.dat$x2))

#Reshape space.dat to list blocks vertically.
#Some renaming is needed after reshaping.
library(dplyr)
library(reshape2)
space.dat2 <- melt(space.dat, id = c('x1','x2','treatment', 'area',
'shape_ind1', 'shape_ind2'))
space.dat2$yield <-space.dat2$value
space.dat2$value=NULL
space.dat2$block <-space.dat2$variable
space.dat2$variable=NULL
#Check dataset
str(space.dat2)
# Ensure class and numerical variables are coded as such
# Change variable types
for ( var  in c('x1', 'x2','x1', 'treatment',
'area', 'shape_ind1','shape_ind2','yield') )
{ space.dat2[,var]
<- as.numeric(space.dat2[,var]) }
for ( var  in c('treatment') )
{ space.dat2[,var] <-as.factor(space.dat2[,var]) }
#Double check
str(space.dat2)

#Using the lm function to fit regressors sequentially.
#Table6A: Bocks, Treatments (spacing) as qualitative #and fixed
effects
Ex5lm1 <- lm(yield ~  block + treatment,
data = space.dat2)
#The anova function outputs the data in Table 6A.
anova (Ex5lm1)

#Table6B: Type I sum of squares, F tests of block, area #and lack of
fit
#In table 6B, the source named 'lack of fit' is named #treatment in
the output here and below because the #lack of fit reflects the
```

```
variation in treatments from #Ex5lm1 that is not explained by the plot
area.
Ex5lm2 <- lm(yield ~  block + area + treatment,
data = space.dat2)
#The anova function outputs the data in Table 6B.
anova (Ex5lm2)

#Table6C. Type 1 Sum of Squares,  F tests of block, #area, shape index
1 and lack of fit(treatment)
Ex5lm3 <- lm(yield ~  block + area + shape_ind1  + treatment, data =
space.dat2)
anova (Ex5lm3)

#Table6D. Type 1 Sum of Squares,  F tests of block, #area, shape index
2 and lack of fit(treatment)
Ex5lm4<-lm(yield ~  block + area + shape_ind2  + treatment, data =
space.dat2)
anova (Ex5lm4)
#Shape index 2 results in lower lackof fit

#Example 5, Table 7 to compare residuals
#In these models, area is included as a quantitative effect. The
treatment or lack of fit is not included in these models. If we were
to consider area as factor, Ex5lm5 would equal Ex5lm1.
Ex5lm5 <- lm(yield ~  block + area, data = space.dat2)
anova (Ex5lm5)
Ex5lm6 <- lm(yield ~  block + area + shape_ind1,
data = space.dat2)
anova(Ex5lm6)
Ex5lm7 <- lm(yield ~  block + area + shape_ind2,
data = space.dat2)
anova(Ex5lm7)
#Add the residuals from these models to space.dat2
space.dat2$residual_B <- resid(Ex5lm5)
space.dat2$residual_C <- resid(Ex5lm6)
space.dat2$residual_D <- resid(Ex5lm7)

#Mean yield and mean residuals for lm models
#attaching mwresi as dataframe and sorting by
#descending treatment
sort(space.dat2$treatment, decreasing = TRUE)
mwresi <- aggregate(space.dat2 [c('yield',
'residual_B', 'residual_C', 'residual_D')],
by = list(treatment = space.dat2$treatment),
mean, na.rm = TRUE)
attach(mwresi)
#Sort by treatment descending.  This is Table 7
mwresi2 <- arrange(mwresi, desc(treatment))
list (mwresi2)
```

Output 6.10 kagc

```
> for ( var  in c('x1', 'x2','x1', 'treatment',
```

```
+     'area', 'shape_ind1','shape_ind2','yield') )
+     { space.dat2[,var] <- as.numeric(space.dat2[,var]) }
>     for ( var  in c('treatment') )
+     { space.dat2[,var] <-as.factor(space.dat2[,var]) }
> #Double check
> str(space.dat2)
'data.frame':    40 obs. of  8 variables:
 $ x1        : num  30 30 30 30 24 24 24 20 20 15 ...
 $ x2        : num  30 24 20 15 24 20 15 20 15 15 ...
 $ treatment : Factor w/ 10 levels "1","2","3","4",..: 10 9 8 7 6 5 4
3 2 1 ...
 $ area      : num  900 720 600 450 576 480 360 400 300 225 ...
 $ shape_ind1: num  1 1.25 1.5 2 1 ...
 $ shape_ind2: num  1 1.01 1.02 1.06 1 ...
 $ yield     : num  5.95 7.1 7 8.1 8.85 7.65 7.8 8.05 9.3 9.35 ...
 $ block     : Factor w/ 4 levels "R1","R2","R3",..: 1 1 1 1 1 1 1 1 1
1 ...
> # Ensure class and numerical variables are coded as such
> # Change variable types
> for ( var  in c('x1', 'x2','x1', 'treatment',
+  'area', 'shape_ind1','shape_ind2','yield') )
+  { space.dat2[,var] <- as.numeric(space.dat2[,var]) }
>   for ( var  in c('treatment') )
+  { space.dat2[,var] <-as.factor(space.dat2[,var]) }
> #Double check
> str(space.dat2)
'data.frame':    40 obs. of  8 variables:
 $ x1        : num  30 30 30 30 24 24 24 20 20 15 ...
 $ x2        : num  30 24 20 15 24 20 15 20 15 15 ...
 $ treatment : Factor w/ 10 levels "1","2","3","4",..: 10 9 8 7 6 5 4
3 2 1 ...
 $ area      : num  900 720 600 450 576 480 360 400 300 225 ...
 $ shape_ind1: num  1 1.25 1.5 2 1 ...
 $ shape_ind2: num  1 1.01 1.02 1.06 1 ...
 $ yield     : num  5.95 7.1 7 8.1 8.85 7.65 7.8 8.05 9.3 9.35 ...
 $ block     : Factor w/ 4 levels "R1","R2","R3",..: 1 1 1 1 1 1 1 1 1
1 ...
> #Using the lm function to fit regressors sequentially.
> #Table6A: Bocks, Treatments (spacing) as qualitative #and fixed
effects
> Ex5lm1 <- lm(yield ~  block + treatment,
+     data = space.dat2)
> #The anova function outputs the data in Table 6A.
> anova (Ex5lm1)
Analysis of Variance Table

Response: yield
`          Df Sum Sq Mean Sq F value    Pr(>F)
block      3  5.880 1.96000  4.2630 0.0137514 *
treatment  9 23.142 2.57136  5.5927 0.0002275 ***
Residuals 27 12.414 0.45977
---------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
>
```

```
> #Table6B: Type I sum of squares, F tests of block, area #and lack of
fit
> #In table 6B, the source named 'lack of fit' is named #treatment in
the output here and below because the #lack of fit reflects the
variation in treatments from #Ex5lm1 that is not explained by the plot
area.
> Ex5lm2 <- lm(yield ~  block + area + treatment,
+     data = space.dat2)
> #The anova function outputs the data in Table 6B.
> anova (Ex5lm2)
Analysis of Variance Table

Response: yield
‘          Df Sum Sq Mean Sq F value       Pr(>F)
block       3  5.880  1.9600   4.263     0.01375 *
area        1 16.786 16.7863  36.510 0.000001892 ***
treatment   8  6.356  0.7945   1.728     0.13724
Residuals  27 12.414  0.4598
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #Table6C. Type 1 Sum of Squares,  F tests of block, #area, shape
index 1 and lack of fit(treatment)
> Ex5lm3 <- lm(yield ~  block + area + shape_ind1  +     treatment,
data = space.dat2)
> anova (Ex5lm3)
Analysis of Variance Table

Response: yield
‘           Df  Sum Sq Mean Sq F value       Pr(>F)
block        3  5.8800  1.9600  4.2630     0.01375 *
area         1 16.7863 16.7863 36.5103 0.000001892 ***
shape_ind1   1  1.9779  1.9779  4.3019     0.04773 *
treatment    7  4.3781  0.6254  1.3603     0.26188
Residuals   27 12.4138  0.4598
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #Table6D. Type 1 Sum of Squares,  F tests of block, #area, shape
index 2 and lack of fit(treatment)
> Ex5lm4<-lm(yield ~  block + area + shape_ind2  +  treatment, data =
space.dat2)
> anova (Ex5lm4)
Analysis of Variance Table

Response: yield
‘           Df  Sum Sq Mean Sq F value       Pr(>F)
block        3  5.8800  1.9600  4.2630     0.01375 *
area         1 16.7863 16.7863 36.5103 0.000001892 ***
shape_ind2   1  2.4937  2.4937  5.4239     0.02759 *
treatment    7  3.8622  0.5517  1.2001     0.33626
Residuals   27 12.4138  0.4598
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> #Shape index 2 results in lower lackof fit
>
> #Example 5, Table 7 to compare residuals
> #In these models, area is included as a quantitative effect. The
treatment or lack of fit is not included in these models. If we were
to consider area as factor, Ex5lm5 would equal Ex5lm1.
> Ex5lm5 <- lm(yield ~  block + area, data = space.dat2)
> anova (Ex5lm5)
Analysis of Variance Table

Response: yield
`          Df Sum Sq Mean Sq F value       Pr(>F)
block       3  5.880  1.9600  3.6548      0.02159 *
area        1 16.786 16.7863 31.3015 0.000002643 ***
Residuals 35 18.770  0.5363
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> Ex5lm6 <- lm(yield ~  block + area + shape_ind1,
+     data = space.dat2)
> anova(Ex5lm6)
Analysis of Variance Table

Response: yield
`           Df  Sum Sq Mean Sq F value       Pr(>F)
block        3  5.8800  1.9600  3.9686       0.0158 *
area         1 16.7863 16.7863 33.9888 0.000001432 ***
shape_ind1   1  1.9779  1.9779  4.0048       0.0534 .
Residuals   34 16.7918  0.4939
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> Ex5lm7 <- lm(yield ~  block + area + shape_ind2,
+     data = space.dat2)
> anova(Ex5lm7)
Analysis of Variance Table

Response: yield
`           Df  Sum Sq Mean Sq F value       Pr(>F)
block        3  5.8800  1.9600  4.0944      0.01388 *
area         1 16.7863 16.7863 35.0660 0.000001088 ***
shape_ind2   1  2.4937  2.4937  5.2093      0.02884 *
Residuals   34 16.2760  0.4787
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> #Add the residuals from these models to space.dat2
> space.dat2$residual_B <- resid(Ex5lm5)
> space.dat2$residual_C <- resid(Ex5lm6)
> space.dat2$residual_D <- resid(Ex5lm7)
>
> #Mean yield and mean residuals for lm models
> #attaching mwresi as dataframe and sorting by #descending treatment
> sort(space.dat2$treatment, decreasing = TRUE)
 [1] 10 10 10 10 9  9  9  9  8  8  8  8  7  7  7  7  6  6  6  6  5  5
5  5  4  4
[27] 4  4  3  3  3  3  2  2  2  2  1  1  1  1
```

```
Levels: 1 2 3 4 5 6 7 8 9 10
> mwresi <- aggregate(space.dat2 [c('yield',
+     'residual_B', 'residual_C', 'residual_D')],
+     by = list(treatment = space.dat2$treatment),
+     mean, na.rm = TRUE)
>
> attach(mwresi)
> #Sort by treatment descending.  This is Table 7
> mwresi2 <- arrange(mwresi, desc(treatment))
> print (mwresi2)
`  treatment  yield  residual_B residual_C  residual_D
1         10 6.0250 -0.07072002 -0.1983165 -0.16444439
2          9 6.4750 -0.22505794 -0.2092044 -0.27131136
3          8 7.1875  0.08455011  0.2557248  0.21141505
4          7 6.9250 -0.68156482 -0.1819410 -0.03353807
5          6 7.8500  0.66647172  0.4747710  0.50319868
6          5 7.7750  0.26915817  0.2017144  0.14274355
7          4 7.5000 -0.40873378 -0.2134183 -0.23499201
8          3 7.7125 -0.06193647 -0.2884593 -0.26298903
9          2 8.7000  0.58982024  0.5822635  0.51052222
10         1 8.2000 -0.16198722 -0.4231342 -0.40060465
=========================================================
```

Example 6: Potato – read data and create variables.

An experiment analyzed the dependency of the weight of tubers y on their size (x) for a given potato (Solanum tuberosum L.) variety. The potato.dat dataset has three variables, size, weight, and a sample index; which numbered the potatoes of each size from 1 to maximum (around 90 potatoes per size). Several additional variables are created for both the regressor and regressand and will be used in the following analyses.

Script 6.11 kagc

```
-----------------------------------------------------------------------------------------------
#Read in datatset and create new variables, size 2, #size3, log_size,
size_reciprocal, log_weight, and #weight1_3.
potato.dat <- read.csv ('potato.csv',
header = TRUE,sep = ',')
potato.dat$size2 = potato.dat$size*potato.dat$size
potato.dat$size3 = potato.dat$size2*potato.dat$size
potato.dat$log_size = log(potato.dat$size)
potato.dat$size_reciprocal = 1/potato.dat$size
potato.dat$log_weight = log(potato.dat$weight)
potato.dat$weight1_3=3*potato.dat$weight**(1/3)
str(potato.dat)
#change weight from int to num
for ( var  in c('weight'))
{potato.dat[,var] <-as.numeric(potato.dat[,var]) }
str (potato.dat)

#We are going to use this dataset for several analyses #so the scatter
plot and boxplots are designed below.  #These are not included in the
SAS code but plotting #them here helps to explain decisions made about
the #analyses below.
```

```
#Use the par command to locate the plot in each section of the figure
and set margins and axes.
png('Ch6.Fig19.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par(fig = c(0.0,0.8,0.25,1.0),
mar = c(3.1,3.1,3.1,2.1),
cex.axis=0.8, mgp= c(2,1,0))
#Plot a scatter plot
plot(potato.dat$size, potato.dat$weight,
xlab = 'Size', ylab = 'Weight',new = TRUE, pch = 20,
panel.first = grid(lty = 1,lwd = 2))
# Fit a loess line
loess_fit <- loess(potato.dat$weight~potato.dat$size)
lines(potato.dat$size, predict(loess_fit),
col = 'blue', lwd=4)
#Boxplot for size
par(fig = c(0.0,0.8,0.0,0.40),
mar = c(3.1,3.1,3.1,2.1), new = TRUE)
boxplot(potato.dat$size, horizontal = TRUE,
col = 'lightpink1', axes = FALSE)
#Boxplot for weight
par(fig = c(0.65,1.0,0.25,1.0), new = TRUE)
boxplot(potato.dat$weight,
col ='lightblue1' , axes = FALSE)
#Title to the plot
mtext('Fig. 1.  Example 6. Scatterplot and Boxplots', side=3,
outer=TRUE, line=-1)
#There is variance heterogeneity present.
#As the sizes increase, the variances increase
dev.off()
```

Output 6.11 kagc

```
> #Read in datatset and create new variables, size 2, #size3,
log_size, size_reciprocal, log_weight, and #weight1_3.
> potato.dat <- read.csv ('potato.csv',
+    header = TRUE,sep = ',') . . . TRUNCATED
> str(potato.dat)
The font is changed so that the table is easier to read
'data.frame':    524 obs. of  9 variables:
 $ size            : num  32.5 32.5 32.5 32.5 32.5 32.5 32.5 32.5 32.5
32.5 ...
 $ weight          : int  16 17 18 18 18 18 18 18 18 18 ...
 $ Sample          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ size2           : num  1056 1056 1056 1056 1056 ...
 $ size3           : num  34328 34328 34328 34328 34328 ...
 $ log_size        : num  3.48 3.48 3.48 3.48 3.48 ...
 $ size_reciprocal: num  0.0308 0.0308 0.0308 0.0308 0.0308 ...
 $ log_weight      : num  2.77 2.83 2.89 2.89 2.89 ...
 $ weight1_3       : num  7.56 7.71 7.86 7.86 7.86 ...

> #change weight from int to num
```

```
> for ( var  in c('weight'))
+ {potato.dat[,var] <-as.numeric(potato.dat[,var]) }
> str (potato.dat)
'data.frame':    524 obs. of  9 variables:
 $ size           : num  32.5 32.5 32.5 32.5 32.5 32.5 32.5 32.5 32.5
32.5 ...
 $ weight         : num  16 17 18 18 18 18 18 18 18 18 ...
 $ Sample         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ size2          : num  1056 1056 1056 1056 1056 ...
 $ size3          : num  34328 34328 34328 34328 34328 ...
 $ log_size       : num  3.48 3.48 3.48 3.48 3.48 ...
 $ size_reciprocal: num  0.0308 0.0308 0.0308 0.0308 0.0308 ...
 $ log_weight     : num  2.77 2.83 2.89 2.89 2.89 ...
 $ weight1_3      : num  7.56 7.71 7.86 7.86 7.86 ...
>
> #We are going to use this dataset for several analyses so the
scatter plot and boxplots are designed below. These are not included
in the SAS code but plotting them here helps to explain decisions made
about the analyses below.
> #Use the par command to locate the plot in each section of the
figure and set margins and axes.
> png('Ch6.Fig19.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(fig = c(0.0,0.8,0.25,1.0),
+     mar = c(3.1,3.1,3.1,2.1),
+     cex.axis=0.8, mgp= c(2,1,0))
#Plot a scatter plot . . . TRUNCATED
#There is variance heterogeneity present.
#As the sizes increase, the variances increase
dev.off()
```

==========

Example 6 (continued). Sequential model fitting - Table 8 A, B, and C

Model sum of square in Table 8A is sum of model effects. Model mean square is Model Sum of squares divided by degrees of freedom. The model effects from Anova function match the left side of Table 8B, sequential sum of squares X1->X2->X3. Rearrange the order of the size variables X3->X1->X2

Script 6.12 kagc

```
#Sequential model fitting with sequence x1→ x2 → x3;
#The lm function fits effects sequentially
Ex6lm1 <- lm(weight ~  size + size2 + size3,
             data = potato.dat)
anova(Ex6lm1)
#Model sum of square in Table 8A is sum of model #effects.  Model mean
square is Model Sum of squares #divided by df.
#The model effects from Anova function match the left #side of Table
8B, sequential sum of squares X1->X2->X3
#Rearrange the order of the size variables X3->X1->X2
Ex6lm2 <- lm(weight ~  size3 + size + size2,
```

```
             data = potato.dat)
#Ex6lm2 sequential sum of squares is Table 8B center
anova(Ex6lm2)
#Use the CAR package and its Anova command to calculate
#the partial (not sequential) sum of squares to match #the right side
of table 8B
library(car)
Ex6aov1 <- Anova(Ex6lm1, type=c(3))
print(Ex6aov1)
#Estimates of the fixed effects in Table 8C.
summary(Ex6lm1)
#They are the same regardless of sequence
summary(Ex6lm2)
#Use the rsq package to calculate the partial Rsquare values in Table
8C
library(rsq)
rsq(Ex6lm1)
rsq.partial(Ex6lm1)
#Use the the vif function to calculate the variance #inflation factor,
inverse of Tolerance in Table 8C.
vif <- vif(Ex6lm1)
TOL=1/vif(Ex6lm1)
list(TOL)
```

Output 6.12 kagc

```
---------------------------------------------------------------------------------------
> #Sequential model fitting with sequence x1→ x2 → x3;
> #The lm function fits effects sequentially
> Ex6lm1 <- lm(weight ~  size + size2 + size3,
+              data = potato.dat)
> anova(Ex6lm1)
Analysis of Variance Table

Response: weight
`          Df Sum Sq Mean Sq   F value Pr(>F)
size        1 767104  767104 6514.8040 <2e-16 ***
size2       1  16475   16475  139.9147 <2e-16 ***
size3       1    250     250    2.1196  0.146
Residuals 520  61229     118
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> #Model sum of square in Table 8A is sum of model #effects.  Model
mean square is Model Sum of squares #divided by df.
> #The model effects from Anova function match the left #side of Table
8B, sequential sum of squares X1->X2->X3
> #Rearrange the order of the size variables X3->X1->X2
> Ex6lm2 <- lm(weight ~  size3 + size + size2,
+              data = potato.dat)
> #Ex6lm2 sequential sum of squares is Table 8B center
> anova(Ex6lm2)
Analysis of Variance Table

Response: weight
`          Df Sum Sq Mean Sq   F value Pr(>F)
```

```
size3        1 783609  783609 6654.9797 <2e-16 ***
size         1    123     123    1.0441 0.3074
size2        1     96      96    0.8145 0.3672
Residuals 520  61229     118
---------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> #Use the CAR package and its Anova command to calculate
> #the partial (not sequential) sum of squares to match #the right
side of table 8B
> library(car)
> Ex6aov1 <- Anova(Ex6lm1, type=c(3))
> print(Ex6aov1)
Anova Table (Type III tests)

Response: weight
'           Sum Sq  Df F value Pr(>F)
(Intercept)    118   1  1.0027 0.3171
size           107   1  0.9061 0.3416
size2           96   1  0.8145 0.3672
size3          250   1  2.1196 0.1460
Residuals    61229 520
> #Estimates of the fixed effects in Table 8C.
> summary(Ex6lm1)

Call:
lm(formula = weight ~ size + size2 + size3, data = potato.dat)

Residuals:
'   Min      1Q  Median      3Q     Max
-45.462  -6.068  -0.095   5.910  41.538

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.013e+02  1.012e+02  -1.001    0.317
size         6.681e+00  7.019e+00   0.952    0.342
size2       -1.435e-01  1.591e-01  -0.902    0.367
size3        1.716e-03  1.179e-03   1.456    0.146

Residual standard error: 10.85 on 520 degrees of freedom
Multiple R-squared:  0.9275, Adjusted R-squared:  0.9271
F-statistic:  2219 on 3 and 520 DF,  p-value: < 2.2e-16

> #They are the same regardless of sequence
> summary(Ex6lm2)

Call:
lm(formula = weight ~ size3 + size + size2, data = potato.dat)

Residuals:
'    Min      1Q  Median      3Q     Max
-45.462  -6.068  -0.095   5.910  41.538

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -1.013e+02  1.012e+02  -1.001    0.317
size3        1.716e-03  1.179e-03   1.456    0.146
size         6.681e+00  7.019e+00   0.952    0.342
size2       -1.435e-01  1.591e-01  -0.902    0.367

Residual standard error: 10.85 on 520 degrees of freedom
Multiple R-squared:  0.9275, Adjusted R-squared:  0.9271
F-statistic:  2219 on 3 and 520 DF,  p-value: < 2.2e-16

> #Use the rsq package to calculate the partial Rsquare values in
Table 8C
> library(rsq)
> rsq(Ex6lm1)
[1] 0.9275447
> rsq.partial(Ex6lm1)
$adjustment
[1] FALSE

$variable
[1] "size"  "size2" "size3"

$partial.rsq
[1] 0.001739553 0.001563869 0.004059529

> #Use the the vif function to calculate the variance #inflation
factor, inverse of Tolerance in Table 8C.
> vif <- vif(Ex6lm1)
> TOL=1/vif(Ex6lm1)
> list(TOL)
[[1]]
`       size        size2        size3
6.108466e-05 1.451714e-05 5.435331e-05
==========================================================
```

Example 6. Table 9 A and B.

Run linear models using a sequential approach for several combinations of the size, size2 and size3, with and without intercept. These models are named as Ex6, int means intercept included, m(for model) followed by numbers that refer to the size variables included. The two functions (model_fit_stats_AIC and rowout) create table 9. Model_fit_stats_AIC based on the model_fit_stats function.

Script 6.13 kagc

```
-----------------------------------------------------------------------------------------------
#Models with intercept
Ex6intm123 <- lm(weight ~ size + size2 + size3,
                 data = potato.dat)
summary(Ex6intm123)
Ex6intm12 <- lm(weight ~ size + size2,
                data = potato.dat)
summary(Ex6intm12)
Ex6intm13 <- lm(weight ~ size + size3,
                data = potato.dat)
summary(Ex6intm13)
Ex6intm23 <- lm(weight ~ size2 + size3,
```

```
                data = potato.dat)
summary(Ex6intm23)
Ex6intm1 <- lm(weight ~ size, data = potato.dat)
summary(Ex6intm1)
Ex6intm2 <- lm(weight ~ size2, data = potato.dat)
summary(Ex6intm2)
Ex6intm3 <- lm(weight ~ size3, data = potato.dat)
summary(Ex6intm3)

#Models without intercept
Ex6m123 <- lm(weight ~ 0 + size + size2 + size3,
              data = potato.dat)
summary(Ex6m123)
Ex6m12 <- lm(weight ~ size + size2, data = potato.dat)
summary(Ex6m12)
Ex6m13 <- lm(weight ~ size + size3, data = potato.dat)
summary(Ex6m13)
Ex6m23 <- lm(weight ~ size2 + size3, data = potato.dat)
summary(Ex6m23)
Ex6m1 <- lm(weight ~  size, data = potato.dat)
summary(Ex6m1)
Ex6m2 <- lm(weight ~  size2, data = potato.dat)
summary(Ex6m2)
Ex6m3 <- lm(weight ~  size3, data = potato.dat)
summary(Ex6m3)

#Use the two functions (model_fit_stats_AIC and rowout)
#to create table 9. Model_fit_stats_AIC is based on the
#model_fit_stats function.
#For these codes to work, the package dplyr has be loaded and
unloaded, then reloaded
library(dplyr)
n <- nrow(potato.dat)
model_fit_stats_AIC <- function(linear.model, ds) {
  coef <-linear.model[['coefficients']]
  r.sqr <- summary(linear.model)$r.squared
  adj.r.sqr <- summary(linear.model)$adj.r.squared
  rss <- sum((linear.model$residuals)^2)
  dferr <- (nrow(ds)-(linear.model$rank))
  rank=(linear.model$rank)
  mserr <- (rss/dferr)
  pr <- residuals(linear.model)/(1-     influence(linear.model)$hat)
  PRESS <- sum(pr^2)
  rsspressdiff <- abs(rss-PRESS)
  AICout <-extractAIC(linear.model, scale=0, k = 2)
  AICrank <-AICout[1]
  AIC <- AICout[2]

  return.df <- data.frame (coef=coef,
  r.squared = r.sqr,
  adj.r.squared = adj.r.sqr,
  s2 = mserr, modelrank=rank,
  press = PRESS,
  rsspressdiff = rsspressdiff,
```

```
  AIC=AIC, AICrank=AICrank)
  return(return.df)
}

rowout <- function(ds, dsout) {
  new <- select(ds,coef)
  newt <- t(new)
  new2 <- select(ds, -coef)
  rownames(new2) <- NULL
  newu <-unique(new2)
  dsout <- cbind.data.frame(newt,newu)
}

#Call the function for each model
mfs1 <- model_fit_stats_AIC(Ex6intm123, potato.dat)
mfs1out <-rowout(mfs1,mfs1out)
mfs2 <- model_fit_stats_AIC(Ex6intm12, potato.dat)
mfs2out <- rowout(mfs2,mfs2out)
mfs3 <- model_fit_stats_AIC(Ex6intm13, potato.dat)
mfs3out <-rowout(mfs3,mfs3out)
mfs4 <- model_fit_stats_AIC(Ex6intm23, potato.dat)
mfs4out <- rowout(mfs4,mfs4out)
mfs5 <- model_fit_stats_AIC(Ex6intm1, potato.dat)
mfs5out <- rowout(mfs5,mfs5out)
mfs6 <- model_fit_stats_AIC(Ex6intm2, potato.dat)
mfs6out <- rowout(mfs6,mfs6out)
mfs7 <- model_fit_stats_AIC(Ex6intm3, potato.dat)
mfs7out <- rowout(mfs7,mfs7out)

#below are the models without intercepts
mfs8 <- model_fit_stats_AIC(Ex6m123, potato.dat)
mfs8out <- rowout(mfs8,mfs8out)
mfs9 <- model_fit_stats_AIC(Ex6m12, potato.dat)
mfs9out <- rowout(mfs9,mfs9out)
mfs10 <- model_fit_stats_AIC(Ex6m13, potato.dat)
mfs10out <- rowout(mfs10,mfs10out)
mfs11 <- model_fit_stats_AIC(Ex6m23, potato.dat)
mfs11out <- rowout(mfs11,mfs11out)
mfs12 <- model_fit_stats_AIC(Ex6m1, potato.dat)
mfs12out <- rowout(mfs12,mfs12out)
mfs13 <- model_fit_stats_AIC(Ex6m2, potato.dat)
mfs13out <- rowout(mfs13,mfs13out)
mfs14 <- model_fit_stats_AIC(Ex6m3, potato.dat)
mfs14out <- rowout(mfs14,mfs14out)

#Combine the model fit statistics for the different #models into one
dataset.
#The AIC values don't completely match SAS and R
#The relative AIC values for various models are #important, not the
absolute values

#Use the stack package to combine the different models
library(Stack)
```

```
#Note that stack will keep adding to the dataset so if #this code is
run more than once be sure to delete the #mfsall dataset first.
rm(mfsall)
mfsall <- Stack(mfs1out, mfs2out)
mfsall <- Stack(mfsall, mfs3out)
mfsall <- Stack(mfsall, mfs4out)
mfsall <- Stack(mfsall, mfs5out)
mfsall <- Stack(mfsall, mfs6out)
mfsall <- Stack(mfsall, mfs7out)
mfsall <- Stack(mfsall, mfs8out)
mfsall <- Stack(mfsall, mfs9out)
mfsall <- Stack(mfsall, mfs10out)
mfsall <- Stack(mfsall, mfs11out)
mfsall <- Stack(mfsall, mfs12out)
mfsall <- Stack(mfsall, mfs13out)
mfsall <- Stack(mfsall, mfs14out)
str(mfsall)
#Use the plyr and dplyr packages to rearrange MFSALL
#Match table 9 A & B
#In order to use the rename function, the dplyr package has to be
unloaded
detach(package:dplyr)
library(plyr)
mfsall2 <- rename (mfsall,
c('(Intercept)' = 'Intercept'))
#Reload the dplyr package
library(dplyr)
mfsall3 <- mfsall2 %>% select(Intercept,
size, size2, size3, everything())
#The best model in Table 9A is Ex6intm3.
#The best model overall is Ex6m3
#Print diagnostic plots for Ex6m3
png('Ch6.Fig20.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
plot(Ex6m3)
dev.off()
#This model fits better than others but
#there are still problems with fit
```

Output 6.13 kagc

```
> #Models with intercept
> Ex6intm123 <- lm(weight ~ size + size2 + size3,
+                  data = potato.dat)
> summary(Ex6intm123)

Call:
lm(formula = weight ~ size + size2 + size3, data = potato.dat)

Residuals:
```

```
`    Min      1Q  Median      3Q     Max
-45.462  -6.068  -0.095   5.910  41.538

Coefficients:
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.013e+02  1.012e+02  -1.001    0.317
size         6.681e+00  7.019e+00   0.952    0.342
size2       -1.435e-01  1.591e-01  -0.902    0.367
size3        1.716e-03  1.179e-03   1.456    0.146

Residual standard error: 10.85 on 520 degrees of freedom
Multiple R-squared:  0.9275, Adjusted R-squared:  0.9271
F-statistic:  2219 on 3 and 520 DF,  p-value: < 2.2e-16

> Ex6intm12 <- lm(weight ~ size + size2,
+                 data = potato.dat)
> summary(Ex6intm12)

Call:
lm(formula = weight ~ size + size2, data = potato.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-44.917  -6.226  -0.618   5.904  42.083

Coefficients:
`            Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.421653  14.770600   3.007  0.00276 **
size        -3.490245   0.672357  -5.191    3e-07 ***
size2        0.087768   0.007428  11.816  < 2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.86 on 521 degrees of freedom
Multiple R-squared:  0.9272, Adjusted R-squared:  0.927
F-statistic:  3320 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6intm13 <- lm(weight ~ size + size3,
+                 data = potato.dat)
> summary(Ex6intm13)

Call:
lm(formula = weight ~ size + size3, data = potato.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-45.149  -6.322  -0.410   5.708  41.851

Coefficients:
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.047e+01  1.017e+01  -1.029    0.304
size         3.547e-01  3.470e-01   1.022    0.307
size3        6.534e-04  5.498e-05  11.886   <2e-16 ***
---------------------------------------------------
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.85 on 521 degrees of freedom
Multiple R-squared:  0.9274, Adjusted R-squared:  0.9272
F-statistic:  3329 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6intm23 <- lm(weight ~ size2 + size3,
+                 data = potato.dat)
> summary(Ex6intm23)

Call:
lm(formula = weight ~ size2 + size3, data = potato.dat)

Residuals:
‘   Min      1Q  Median      3Q     Max
-45.155  -6.297  -0.403   5.739  41.845

Coefficients:
‘             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.1297179  5.2212726  -0.982    0.326
size2        0.0076766  0.0078654   0.976    0.330
size3        0.0005992  0.0001128   5.313  1.6e-07 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.85 on 521 degrees of freedom
Multiple R-squared:  0.9274, Adjusted R-squared:  0.9271
F-statistic:  3329 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6intm1 <- lm(weight ~ size, data = potato.dat)
> summary(Ex6intm1)

Call:
lm(formula = weight ~ size, data = potato.dat)

Residuals:
‘   Min      1Q  Median      3Q     Max
-38.005  -8.486   0.618   7.687  48.995

Coefficients:
‘             Estimate Std. Error t value Pr(>|t|)
(Intercept) -127.58696    2.81284  -45.36   <2e-16 ***
size           4.42769    0.06178   71.67   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.22 on 522 degrees of freedom
Multiple R-squared:  0.9078, Adjusted R-squared:  0.9076
F-statistic:  5137 on 1 and 522 DF,  p-value: < 2.2e-16

> Ex6intm2 <- lm(weight ~ size2, data = potato.dat)
> summary(Ex6intm2)

Call:
```

```
lm(formula = weight ~ size2, data = potato.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-42.188  -7.384  -0.180   5.853  44.812

Coefficients:
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.194e+01  1.377e+00  -23.19   <2e-16 ***
size2        4.934e-02  6.216e-04   79.38   <2e-16 ***
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 11.13 on 522 degrees of freedom
Multiple R-squared:  0.9235, Adjusted R-squared:  0.9233
F-statistic:  6300 on 1 and 522 DF,  p-value: < 2.2e-16

> Ex6intm3 <- lm(weight ~ size3, data = potato.dat)
> summary(Ex6intm3)

Call:
lm(formula = weight ~ size3, data = potato.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-45.642  -6.210  -0.293   5.742  41.358

Coefficients:
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.253e-01  9.853e-01  -0.127    0.899
size3        7.089e-04  8.689e-06  81.589   <2e-16 ***
---------------------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 10.85 on 522 degrees of freedom
Multiple R-squared:  0.9273, Adjusted R-squared:  0.9271
F-statistic:  6657 on 1 and 522 DF,  p-value: < 2.2e-16

>
> #Models without intercept
> Ex6m123 <- lm(weight ~ 0 + size + size2 + size3,
+               data = potato.dat)
> summary(Ex6m123)

Call:
lm(formula = weight ~ 0 + size + size2 + size3, data = potato.dat)

Residuals:
`    Min      1Q  Median      3Q     Max
-45.160  -6.276  -0.393   5.763  41.840

Coefficients:
`       Estimate Std. Error t value Pr(>|t|)
size  -0.3375284  0.3621761  -0.932   0.3518
```

```
size2  0.0149172  0.0159812   0.933   0.3510
size3  0.0005484  0.0001719   3.191   0.0015 **
---------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.85 on 521 degrees of freedom
Multiple R-squared:  0.9822, Adjusted R-squared:  0.9821
F-statistic:  9560 on 3 and 521 DF,  p-value: < 2.2e-16

> Ex6m12 <- lm(weight ~ size + size2, data = potato.dat)
> summary(Ex6m12)

Call:
lm(formula = weight ~ size + size2, data = potato.dat)

Residuals:
    Min      1Q  Median      3Q     Max
-44.917  -6.226  -0.618   5.904  42.083

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.421653  14.770600   3.007  0.00276 **
size        -3.490245   0.672357  -5.191    3e-07 ***
size2        0.087768   0.007428  11.816  < 2e-16 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.86 on 521 degrees of freedom
Multiple R-squared:  0.9272, Adjusted R-squared:  0.927
F-statistic:  3320 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6m13 <- lm(weight ~ size + size3, data = potato.dat)
> summary(Ex6m13)

Call:
lm(formula = weight ~ size + size3, data = potato.dat)

Residuals:
    Min      1Q  Median      3Q     Max
-45.149  -6.322  -0.410   5.708  41.851

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.047e+01  1.017e+01  -1.029    0.304
size         3.547e-01  3.470e-01   1.022    0.307
size3        6.534e-04  5.498e-05  11.886   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.85 on 521 degrees of freedom
Multiple R-squared:  0.9274, Adjusted R-squared:  0.9272
F-statistic:  3329 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6m23 <- lm(weight ~ size2 + size3, data = potato.dat)
```

```
> summary(Ex6m23)

Call:
lm(formula = weight ~ size2 + size3, data = potato.dat)

Residuals:
'    Min      1Q  Median      3Q     Max
-45.155  -6.297  -0.403   5.739  41.845

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.1297179  5.2212726  -0.982    0.326
size2        0.0076766  0.0078654   0.976    0.330
size3        0.0005992  0.0001128   5.313  1.6e-07 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.85 on 521 degrees of freedom
Multiple R-squared:  0.9274, Adjusted R-squared:  0.9271
F-statistic:  3329 on 2 and 521 DF,  p-value: < 2.2e-16

> Ex6m1 <- lm(weight ~  size, data = potato.dat)
> summary(Ex6m1)

Call:
lm(formula = weight ~ size, data = potato.dat)

Residuals:
'    Min      1Q  Median      3Q     Max
-38.005  -8.486   0.618   7.687  48.995

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
(Intercept) -127.58696    2.81284  -45.36   <2e-16 ***
size           4.42769    0.06178   71.67   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.22 on 522 degrees of freedom
Multiple R-squared:  0.9078, Adjusted R-squared:  0.9076
F-statistic:  5137 on 1 and 522 DF,  p-value: < 2.2e-16

> Ex6m2 <- lm(weight ~  size2, data = potato.dat)
> summary(Ex6m2)

Call:
lm(formula = weight ~ size2, data = potato.dat)

Residuals:
'    Min      1Q  Median      3Q     Max
-42.188  -7.384  -0.180   5.853  44.812

Coefficients:
'             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -3.194e+01  1.377e+00  -23.19   <2e-16 ***
size2        4.934e-02  6.216e-04   79.38   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.13 on 522 degrees of freedom
Multiple R-squared:  0.9235, Adjusted R-squared:  0.9233
F-statistic:  6300 on 1 and 522 DF,  p-value: < 2.2e-16

> Ex6m3 <- lm(weight ~  size3, data = potato.dat)
> summary(Ex6m3)

Call:
lm(formula = weight ~ size3, data = potato.dat)

Residuals:
‘   Min      1Q  Median      3Q     Max
-45.642  -6.210  -0.293   5.742  41.358

Coefficients:
‘             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.253e-01  9.853e-01  -0.127    0.899
size3        7.089e-04  8.689e-06  81.589   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.85 on 522 degrees of freedom
Multiple R-squared:  0.9273, Adjusted R-squared:  0.9271
F-statistic:  6657 on 1 and 522 DF,  p-value: < 2.2e-16

>
> #Use the two functions (model_fit_stats_AIC and rowout)
> #to create table 9. Model_fit_stats_AIC is based on the
> #model_fit_stats function.
> #For these codes to work, the package dplyr has be loaded and
unloaded, then reloaded
> library(dplyr)
> n <- nrow(potato.dat)
> model_fit_stats_AIC <- function(linear.model, ds) {
+   coef <-linear.model[['coefficients']]
+   r.sqr <- summary(linear.model)$r.squared
+   adj.r.sqr <- summary(linear.model)$adj.r.squared
+   rss <- sum((linear.model$residuals)^2)
+   dferr <- (nrow(ds)-(linear.model$rank))
+   rank=(linear.model$rank)
+   mserr <- (rss/dferr)
+   pr <- residuals(linear.model)/(1-   influence(linear.model)$hat)
+   PRESS <- sum(pr^2)
+   rsspressdiff <- abs(rss-PRESS)
+   AICout <-extractAIC(linear.model, scale=0, k = 2)
+   AICrank <-AICout[1]
+   AIC <- AICout[2]
+
+   return.df <- data.frame (coef=coef,
```

```
+                     r.squared = r.sqr,
+                     adj.r.squared = adj.r.sqr,
+                     s2 = mserr,modelrank=rank,
+                     press = PRESS,
+                     rsspressdiff = rsspressdiff,
+                     AIC=AIC, AICrank=AICrank)
+   return(return.df)
+ }
>
> rowout <- function(ds, dsout) {
+   new <- select(ds,coef)
+   newt <- t(new)
+   new2 <- select(ds, -coef)
+   rownames(new2) <- NULL
+   newu <-unique(new2)
+   dsout <- cbind.data.frame(newt,newu)
+ }
>
> #Call the function for each model
> mfs1 <- model_fit_stats_AIC(Ex6intm123, potato.dat)
> mfs1out <-rowout(mfs1,mfs1out)
> mfs2 <- model_fit_stats_AIC(Ex6intm12, potato.dat)
> mfs2out <- rowout(mfs2,mfs2out)
> mfs3 <- model_fit_stats_AIC(Ex6intm13, potato.dat)
> mfs3out <-rowout(mfs3,mfs3out)
> mfs4 <- model_fit_stats_AIC(Ex6intm23, potato.dat)
> mfs4out <- rowout(mfs4,mfs4out)
> mfs5 <- model_fit_stats_AIC(Ex6intm1, potato.dat)
> mfs5out <- rowout(mfs5,mfs5out)
> mfs6 <- model_fit_stats_AIC(Ex6intm2, potato.dat)
> mfs6out <- rowout(mfs6,mfs6out)
> mfs7 <- model_fit_stats_AIC(Ex6intm3, potato.dat)
> mfs7out <- rowout(mfs7,mfs7out)
>
> #below are the models without intercepts
> mfs8 <- model_fit_stats_AIC(Ex6m123, potato.dat)
> mfs8out <- rowout(mfs8,mfs8out)
> mfs9 <- model_fit_stats_AIC(Ex6m12, potato.dat)
> mfs9out <- rowout(mfs9,mfs9out)
> mfs10 <- model_fit_stats_AIC(Ex6m13, potato.dat)
> mfs10out <- rowout(mfs10,mfs10out)
> mfs11 <- model_fit_stats_AIC(Ex6m23, potato.dat)
> mfs11out <- rowout(mfs11,mfs11out)
> mfs12 <- model_fit_stats_AIC(Ex6m1, potato.dat)
> mfs12out <- rowout(mfs12,mfs12out)
> mfs13 <- model_fit_stats_AIC(Ex6m2, potato.dat)
> mfs13out <- rowout(mfs13,mfs13out)
> mfs14 <- model_fit_stats_AIC(Ex6m3, potato.dat)
> mfs14out <- rowout(mfs14,mfs14out)
>
> #Combine the model fit statistics for the different #models into one
dataset.
> #The AIC values don't completely match SAS and R
```

```
> #The relative AIC values for various models are #important, not the
absolute values
>
> #Use the stack package to combine the different models
> library(Stack)
> #Note that stack will keep adding to the dataset so if #this code is
run more than once be sure to delete the #mfsall dataset first.
> rm(mfsall)
> mfsall <- Stack(mfs1out, mfs2out)
> mfsall <- Stack(mfsall, mfs3out)
> mfsall <- Stack(mfsall, mfs4out)
> mfsall <- Stack(mfsall, mfs5out)
> mfsall <- Stack(mfsall, mfs6out)
> mfsall <- Stack(mfsall, mfs7out)
> mfsall <- Stack(mfsall, mfs8out)
> mfsall <- Stack(mfsall, mfs9out)
> mfsall <- Stack(mfsall, mfs10out)
> mfsall <- Stack(mfsall, mfs11out)
> mfsall <- Stack(mfsall, mfs12out)
> mfsall <- Stack(mfsall, mfs13out)
> mfsall <- Stack(mfsall, mfs14out)
> str(mfsall)
'data.frame':    14 obs. of  12 variables:
 $ r.squared    : num  0.928 0.927 0.927 0.927 0.908 ...
 $ adj.r.squared: num  0.927 0.927 0.927 0.927 0.908 ...
 $ s2           : num  118 118 118 118 149 ...
 $ modelrank    : int  4 3 3 3 2 2 2 3 3 3 ...
 $ press        : num  62219 62237 62093 62115 78603 ...
 $ rsspressdiff : num  990 758 768 779 650 ...
 $ AIC          : num  2503 2503 2502 2502 2625 ...
 $ AICrank      : num  4 3 3 3 2 2 2 3 3 3 ...
 $ (Intercept)  : num  -101.33 44.42 -10.47 -5.13 -127.59 ...
 $ size3        : num  0.001716 NA 0.000653 0.000599 NA ...
 $ size2        : num  -0.14355 0.08777 NA 0.00768 NA ...
 $ size         : num  6.681 -3.49 0.355 NA 4.428 ...
> #Use the plyr and dplyr packages to rearrange MFSALL
> #Match table 9 A & B
> #In order to use the rename function, the dplyr package has to be
unloaded
> detach(package:dplyr)
> library(plyr)
> mfsall2 <- rename (mfsall,
+                    c('(Intercept)' = 'Intercept'))
> #Reload the dplyr package
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:plyr':
arrange, count, desc, failwith, id, mutate, rename, summarise,
summarize

The following object is masked from 'package:car':
recode
```

```
The following objects are masked from 'package:stats':
filter, lag

The following objects are masked from 'package:base':
intersect, setdiff, setequal, union

> mfsall3 <- mfsall2 %>% select(Intercept,
+        size, size2, size3, everything())
> #The best model in Table 9A is Ex6intm3.
> #The best model overall is Ex6m3
> #Print diagnostic plots for Ex6m3
> png('Ch6.Fig20.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(2,2))
> plot(Ex6m3)
> dev.off()
> #This model fits better than others but
> #there are still problems with fit but we will this model for
additional analysis below
```

==========

Example 6. Second step. Table 12

Box Cox transformations to remove variance heterogeneity with log(size) as regressor using lambda=0, 1/3, or 1 if it is in confidence interval of the optimal lambda from Box Cox.

Script 6.14 kagc

```
library(car)
library(rcompanion)

#A few additional diagnostic plots
png('Ch6.Fig21.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
plotNormalHistogram    (potato.dat$weight)
qqnorm (potato.dat$weight,
        ylab     = 'Sample Quantiles for potato weight')
qqline(potato.dat$weight, col='red')
dev.off()

library(MASS)
#Conduct boxcox transformation of weight
#These are data presented in Table 12 of the Chapter 6.
#Calculate original lambda for values -0.5 to 1 by 0.01
#Using log_size as regressor
#This boxcox function outputs a plot of the
#log likliehood by the lambda values
png('Ch6.Fig22.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
```

```
par(mfrow=c(1,1))
Box = boxcox(potato.dat$weight ~ potato.dat$log_size,
             lambda = seq(-.5,1,0.01))
dev.off()
#Create a data frame with the results
Cox = data.frame(Box$x, Box$y)
#Order the new data frame by decreasing y
Cox2 = Cox[with(Cox, order(-Cox$Box.y)),]
#Display the lambda with the greatest log likliehood
#This is the result on the second line of Table 12 of Chapter 6.
print (Cox2[1,]) #Original lambda is 0.14
#display 95% CI for best lambda
print(range(Box$x[Box$y > max(Box$y)-qchisq(0.95,1)/2]))
#The confidence intervals are 0.08 and 0.2

#Calculation of the correct -2LL for the model
#(-2LL= log(weight)=b0+3*log(size))
#Calculate means and sums for log_size and log_weight
log_size_mean <-(mean(potato.dat$log_size))
log_weight_mean <-(mean(potato.dat$log_weight))
log_size_sum <-(sum(potato.dat$log_size))
log_weight_sum <-(sum(potato.dat$log_weight))
intercept <- cbind(log_size_mean,log_weight_mean,
                   log_size_sum,log_weight_sum)
#Calculate b0 and combine means, sums and
b0 with #original data into new potato.1 dataset
b0 <- (mean(potato.dat$log_weight))- 3*(mean(potato.dat$log_size))
potato.1 <-cbind(potato.dat, intercept, b0)
#Calculate resid resid=log_weight-b0-3*log_size;
potato.1$resid = potato.1$log_weight-potato.1$b0 -
(3*potato.1$log_size)
#Add n to dataset
potato.1$n = length(potato.1$size)
#Calculate the variance of the resid value and add
potato.1$MSresid <-(var(potato.1$resid))
#LLnew is -2LL, corrected by the Jacobian
#note: PI=3.1415926536
LLvalue = -(potato.1$n/2*log(potato.1$MSresid)) -
  (potato.1$n/2*log(2*3.1415926536)) -
  ((potato.1$n - 1)/2)
head(LLnew <- -2*(LLvalue-potato.1$log_weight_sum), n=1)

#Rerun BoxCox transformation with size not log_size as regressor
png('Ch6.Fig23.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
Box <- boxcox(potato.dat$weight ~ potato.dat$size,
              lambda = seq(-.5,1,0.01))
dev.off()
#As above, create a data frame with the results
Cox <-data.frame(Box$x, Box$y)
#As above, order the new data frame by decreasing y
Cox2 <- Cox[with(Cox, order(-Cox$Box.y)),]
```

```
#As above, display the greatest log likliehood
print(Cox2[1,]) #Best lambda is 0.34
#As above, display 95% CI for best lambda
print(range(Box$x[Box$y > max(Box$y)-qchisq(0.95,1)/2]))
#Now the CI ranges from .28 to 0.4 which includes 3
#This is the result on the third line of Table 12 of Chapter 6.
#Fourth line of Table 12 is from last line of Table 9B in Chapter 6.
#Extract that lambda, transform weight data and add
#transformed variable to original dataset
lambda = Cox2[1, 'Box.x']
potato.dat$W_box = (potato.dat$weight ^ lambda - 1)/lambda
#Calculate new -2LL for this model as above.

#We don't exactly match Fig 10 but these plots of
#the transformed and untransformed data illustrate
#that the transformed data meet the model expectations better.
#Compare plots for untransformed and transformed weight
#Lack of normality and variance heterogeneity are #evident in
untransformed data
library(rcompanion)
png('Ch6.Fig24.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
plotNormalHistogram(potato.dat$weight, main='Weight Frequency',
                    xlab="Size", ylab='Weight')
plotNormalHistogram(potato.dat$W_box, main='W_box Frequency',
                    xlab="Size", ylab='BC_weight')
boxplot(potato.dat$weight ~ potato.dat$size,
        ylab='Weight',
        xlab='Size')
boxplot(potato.dat$W_box ~ potato.dat$size,
        ylab='BC_weight',
        xlab='Size')
dev.off()

#We have previously determined that size*3 is the best model for the
data.
#Run BoxCox transformation again
#with size as regressor, using lambda=0.33 (or optimal) but focus on
# more data points between 0.3 and 0.4.
png('Ch6.Fig25.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
box2 = boxcox(potato.dat$weight ~ potato.dat$size,
              lambda = seq(0.3,0.4,0.001))
dev.off()
#Create a data frame with the results
cox <-data.frame(box2$x, box2$y)
# Order the new data frame by decreasing y
cox2 <- cox[with(cox, order(-cox$box2.y)),]
#Display the lambda with the greatest log likliehood
print(cox2[1,])
```

```
#Best lambda is 0.34
#display 95% CI for best lambda
print(range(box2$x[box2$y > max(box2$y)-qchisq(0.95,1)/2]))
#now the CI range from .28 to 0.4
#Extract that lambda, transform weight data and
#add transformed variable to original dataset
lambda = cox2[1, 'box2.x']
potato.dat$W_box2 <-
  (potato.dat$weight ^ lambda - 1)/lambda

#Now run regression using  size as regressor and W_box2 as regressand.
Ex6lmboxcox <- lm(W_box2 ~ size, data=potato.dat)
summary(Ex6lmboxcox)
png('Ch6.Fig26.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(2,2))
plot(Ex6lmboxcox)
dev.off()
#This model has a lower AIC and better adj R2
#than other models (See Table 8).
#Run the model_fit_stats_AIC function from script 6.17.
mfs15 <- model_fit_stats_AIC(Ex6lmboxcox, potato.dat)
print(mfs15)

#But even with the transformation the
#residuals plots showed increasing variance
#with increasing size (variance heterogeneity),
#therefore we will use the reciprocal of the size
#variable(size_reciprocal) for weighted least squares analysis

#For comparison, repeat the unweighted regression using
#our best model from above (Table 9b)
#These data match line 1 in Table 14
#This model doesn't have random effects.
#The variance component listed in Table 14 is the residual variance.
Ex6m3 <- lm(weight ~  0 + size3 , data=potato.dat)
AIC(Ex6m3)
summary(Ex6m3)
anova(Ex6m3)

#Weighted regression with weight = 1/size
#This analysis matches second line of Table 14
#Use the glm function from nlme to  perform weighted regression.
#Note that glm is using maximum likeliehood estimation
library(nlme)
Ex6wglm <- glm(weight ~ size3 - 1,
               weights = size_reciprocal, data = potato.dat)
summary(Ex6wglm)

#Using the weights has corrected for some of the
#variance heterogeneity in the model
#The glm above matches the SAS output,
#SAS uses REML but the result is the same if method=ML
```

```
#is specified in SAS.
#In REML, SAS includes an F value rather than
#T for the significance of the Fixed effect.
#T2 from the Ex6wglm equals F = 27489.64
#Not using KR DDFM because using ML estimation.

#Correcting for heterogeneous variances using nlme
#weighted by the reciprocal of size3
#This model doesn't directly match any of those in
#Table 14 but the AIC value would place it at line 3
Ex6wlme1 <- lme(weight~size3 - 1, data=potato.dat,
                random= ~1|size, method='REML',
                weights = varIdent(form = ~1|size3))
summary(Ex6wlme1)
print(VarCorr(Ex6wlme1))

#Correcting for heterogeneous variances using nlme with
#the reciprocal of size. Use the best model from our analysise in
table 9,
#weighted by the reciprocal of size using individual variance per
size.
#This is not identical but quite similar to the last line of Table 14
Ex6lme2 <- lme(weight ~ size3 - 1, data=potato.dat,
               random= ~1|size,
               weights = varIdent(form = ~1|size))
summary(Ex6lme2)
print(VarCorr(Ex6lme2) )

#The estimated variance components for each size aren't
#printed out in the model summary but rather scaled
#from the inital one and printed as standard deviations
#so they have to be calculated as below
summary(Ex6lme2$modelStruct)
(c(1.0000000, coef( Ex6lme2$modelStruct$varStruct,
                    unconstrained=F))*Ex6lme2$sigma)^2

#Compare these models,
#weighted regression with individual weights (best model)
png('Ch6.Fig27a.png', width = 8, height = 8,
    units = 'in', res = 600)
par(mfrow=c(1,1))
plot(Ex6lme2)
dev.off()
#Compare this plot to the same from the unweighted analysis
png('Ch6.Fig27b.png', width = 8, height = 8,
    units = 'in', res = 600)
par(mfrow=c(1,1), new=TRUE)
plot (Ex6m3$fitted, rstudent(Ex6m3), pch=1, col='blue',
      xlab = 'Student Residuals',
      ylab = 'Predicted',main='')
abline(h=0)
dev.off()

#Histogram plots of residuals, unweighted analysis
```

```
png('Ch6.Fig28a.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
hist(Ex6m3$res, breaks = 20, freq = F,
     xlim = c(-30,30), ylim = c(0,.06),
     xlab = 'Residuals', ylab = 'Probability', col='lightblue',
main='')
mtext('Residuals Histogram for unweighted model,
      Normal and Kernel Density Curve for Ex 6,
      unweighted model',
      side=3, outer=TRUE, line= -3)
curve(dnorm(x, mean = mean(Ex6m3$res), sd = sd(Ex6m3$res)), add = T)
lines(density (Ex6m3$res), col = 'red')
legend('topright', c('Normal','Kernel'),
       col = c('black','red'), lty=c(1,1),
       lwd=c(2,2,2), cex=0.5)
dev.off()

png('Ch6.Fig28b.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
hist(Ex6lme2$res, breaks = 20, freq = F,
     xlim = c(-30,30), ylim = c(0,.06),
     xlab = 'Residuals', ylab = 'Probability', col='lightblue',
main='')
mtext('Residuals Histogram for best model,
   Normal and Kernel Density Curve
    for Ex 6, best model, weighted',
      side=3, outer=TRUE, line= -3)
curve(dnorm(x, mean = mean(Ex6lme2$res), sd = sd(Ex6lme2$res)), add =
T)
lines(density (Ex6lme2$res), col = 'red')
legend('topright', c('Normal','Kernel'),
       col = c('black','red'), lty=c(1,1),
       lwd=c(2,2,2), cex=0.5)
dev.off()

#Note: The model below did not converge.  It didn't converge #in SAS
either. See comments about this in Chapter 6 #text.Since it wasn't the
best model, we didn't find an #R solution
Ex6HVexp <- lme(weight~size3 - 1,
                random= ~1|size,
                weights = varExp(form = ~size3|size), data=potato.dat,
)
summary(Ex6HVexp)
summary(Ex6HVexp$modelStruct)
```

Output 6.14 kagc

```
------------------------------------------------------------------------------------------
> library(car)
> library(rcompanion)
```

```
>
> #A few additional diagnostic plots
> png('Ch6.Fig21.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(2,2))
> plotNormalHistogram  (potato.dat$weight)
> qqnorm (potato.dat$weight,. . . TRUNCATED
>
> library(MASS)
> #Conduct boxcox transformation of weight
> #These are data presented in Table 12 of Chapter 6.
> #Calculate original lambda for values -0.5 to 1 by 0.01
> #Using log_size as regressor
> #This boxcox function outputs a plot of the
> #log likliehood by the lambda values
> png('Ch6.Fig22.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(1,1))
> Box = boxcox(potato.dat$weight ~ potato.dat$log_size, + lambda =
seq(-.5,1,0.01))  . . . TRUNCATED

> #Display the lambda with the greatest log likliehood
> #This is the result on the second line of Table 12 of Chapter 6.
> print (Cox2[1,]) #Original lambda is 0.14
   Box.x     Box.y
65  0.14 -662.2657
> #display 95% CI for best lambda
> print(range(Box$x[Box$y > max(Box$y)-qchisq(0.95,1)/2]))
[1] 0.08 0.20
> #The confidence intervals are 0.08 and 0.2
>
> #Calculation of the correct -2LL for the model
> #(-2LL= log(weight)=b0+3*log(size)) . . .TRUNCATED
> LLvalue = -(potato.1$n/2*log(potato.1$MSresid)) -
+   (potato.1$n/2*log(2*3.1415926536)) -
+   ((potato.1$n - 1)/2)
> head(LLnew <- -2*(LLvalue-potato.1$log_weight_sum), n=1)
[1] 3821.923
>
> #Rerun BoxCox transformation with size not log_size as regressor
> png('Ch6.Fig23.png', width = 8. . . TRUNCATED >

> #As above, create a data frame with the results
> Cox <-data.frame(Box$x, Box$y)  . . . TRUNCATED > print(Cox2[1,])
#Best lambda is 0.34
   Box.x     Box.y
85  0.34 -664.5704
> #As above, display 95% CI for best lambda
> print(range(Box$x[Box$y > max(Box$y)-qchisq(0.95,1)/2]))
[1] 0.28 0.40
> #Now the CI ranges from .28 to 0.4 which includes 3
> #This is the result on the third line of Table 12 of Chapter 6.
```

```
> #Fourth line of Table 12 is from last line of Table 9B in Chapter 6.
> #Extract that lambda, transform weight data and add
> #transformed variable to original dataset
> lambda = Cox2[1, 'Box.x'. . . TRUNCATED
> #Calculate new -2LL for this model as above.
>
> #We don't exactly match Fig 10 but these plots of
> #the transformed and untransformed data illustrate
> #that the transformed data meet the model expectations better.
> #Compare plots for untransformed and transformed weight
> #Lack of normality and variance heterogeneity are #evident in
untransformed data
> library(rcompanion)
> png('Ch6.Fig24.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(2,2))
> plotNormalHistogram(potato.dat$weight, main='Weight Frequency',
+   xlab="Size", ylab='Weight') . . . TRUNCATED
>
> #We have previously determined that size*3 is the best model for the
data.
> #Run BoxCox transformation again
> #with size as regressor, using lambda=0.33 (or optimal) but focus on
more data points between 0.3 and 0.4.
> png('Ch6.Fig25.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off(). . . TRUNCATED

> #Create a data frame with the results
> cox <-data.frame(box2$x, box2$y)  . . . TRUNCATED > #Display the
lambda with the greatest log likliehood
> print(cox2[1,])
    box2.x    box2.y
40   0.339 -664.5699
> #Best lambda is 0.34
> #display 95% CI for best lambda
> print(range(box2$x[box2$y > max(box2$y)-qchisq(0.95,1)/2]))
[1] 0.3 0.4
> #now the CI range from .28 to 0.4
> #Extract that lambda, transform weight data and
> #add transformed variable to original dataset
> lambda = cox2[1, 'box2.x']  . . . TRUNCATED

> #Now run regression using  size as regressor and W_box2 as
regressand.
> Ex6lmboxcox <- lm(W_box2 ~ size, data=potato.dat)
> summary(Ex6lmboxcox)

Call:
lm(formula = W_box2 ~ size, data = potato.dat)

Residuals:
`     Min       1Q   Median       3Q      Max
```

```
-2.02953 -0.44343  0.02029  0.43499  1.95204

Coefficients:
`            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.337442   0.142460  -23.43   <2e-16 ***
size         0.276987   0.003129   88.53   <2e-16 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6189 on 522 degrees of freedom
Multiple R-squared:  0.9376, Adjusted R-squared:  0.9374
F-statistic:  7837 on 1 and 522 DF,  p-value: < 2.2e-16

> png('Ch6.Fig26.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #dev.off()
> par(mfrow=c(2,2))   . . . TRUNCATED

> #This model has a lower AIC and better adj R2
> #than other models (See Table 8).
> #Run the model_fit_stats_AIC function from script 6.17.
> mfs15 <- model_fit_stats_AIC(Ex6lmboxcox, potato.dat)
> print(mfs15)
                coef r.squared adj.r.squared        s2 modelrank
press
(Intercept) -3.337442 0.9375536     0.9374339 0.3830544         2
201.4174
size         0.276987 0.9375536     0.9374339 0.3830544         2
201.4174
`           rsspressdiff       AIC AICrank
(Intercept)     1.462959 -500.8228       2
size            1.462959 -500.8228       2

> #But even with the transformation the
> #residuals plots showed increasing variance
> #with increasing size (variance heterogeneity),
> #therefore we will use the reciprocal of the size
> #variable(size_reciprocal) for weighted least squares analysis
>
> #For comparison, repeat the unweighted regression using
> #our best model from above (Table 9b)
> #These data match line 1 in Table 14
> #This model doesn't have random effects.
> #The variance component listed in Table 14 is the residual variance.
> Ex6m3 <- lm(weight ~  0 + size3 , data=potato.dat)
> AIC(Ex6m3)
[1] 3987.636
> summary(Ex6m3)

Call:
lm(formula = weight ~ 0 + size3, data = potato.dat)

Residuals:
`   Min      1Q  Median      3Q     Max
```

```
-45.583  -6.302  -0.344   5.668  41.417

Coefficients:
`      Estimate Std. Error t value Pr(>|t|)
size3 7.079e-04  4.176e-06   169.5   <2e-16 ***
---------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.84 on 523 degrees of freedom
Multiple R-squared:  0.9821, Adjusted R-squared:  0.9821
F-statistic: 2.874e+04 on 1 and 523 DF,  p-value: < 2.2e-16

> anova(Ex6m3)
Analysis of Variance Table

Response: weight
`           Df  Sum Sq Mean Sq F value    Pr(>F)
size3        1 3376891 3376891   28741 < 2.2e-16 ***
Residuals 523   61450     117
---------------------------------------------------
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #Weighted regression with weight = 1/size
> #This analysis matches second line of Table 14
> #Use the glm function from nlme to  perform weighted regression.
> #Note that glm is using maximum likeliehood estimation
> library(nlme)
> Ex6wglm <- glm(weight ~ size3 - 1,
+  weights = size_reciprocal, data = potato.dat)
> summary(Ex6wglm)

Call:
glm(formula = weight ~ size3 - 1, data = potato.dat, weights =
size_reciprocal)

Deviance Residuals:
`   Min       1Q   Median       3Q      Max
-6.0123  -0.9973  -0.0545   0.8929   5.4610

Coefficients:
`      Estimate Std. Error t value Pr(>|t|)
size3 7.080e-04  4.271e-06   165.8   <2e-16 ***
---------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 2.376507)

    Null deviance: 66545.0  on 524  degrees of freedom
Residual deviance:  1242.9  on 523  degrees of freedom
AIC: 3924.9

Number of Fisher Scoring iterations: 2

>
```

```
> #Using the weights has corrected for some of the
> #variance heterogeneity in the model
> #The glm above matches the SAS output,
> #SAS uses REML but the result is the same if method=ML
> #is specified in SAS.
> #In REML, SAS includes an F value rather than
> #T for the significance of the Fixed effect.
> #T2 from the Ex6wglm equals F = 27489.64
> #Not using KR DDFM because using ML estimation.
>
> #Correcting for heterogeneous variances using nlme
> #weighted by the reciprocal of size3
> #This model doesn't directly match any of those in
> #Table 14 but the AIC value would place it at line 3
> Ex6wlme1 <- lme(weight~size3 - 1, data=potato.dat,
+                 random= ~1|size, method='REML',
+                 weights = varIdent(form = ~1|size3))
> summary(Ex6wlme1)
Linear mixed-effects model fit by REML
 Data: potato.dat
`      AIC      BIC  logLik
  3838.799 3872.876 -1911.4

Random effects:
 Formula: ~1 | size
`       (Intercept) Residual
StdDev:   0.8743277 3.673435

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | size3
 Parameter estimates:
 34328.125  52734.375  76765.625 107171.875 144703.125 190109.375
  1.000000   2.039875   2.570281   3.022735   3.503492   4.425590
Fixed effects: weight ~ size3 - 1
`            Value    Std.Error DF  t-value p-value
size3 0.0007078289 5.910486e-06  5 119.7582       0

Standardized Within-Group Residuals:
`       Min          Q1         Med          Q3         Max
-2.79834772 -0.71671936 -0.04200641  0.68897783  3.22455942

Number of Observations: 524
Number of Groups: 6
> print(VarCorr(Ex6wlme1))
size = pdLogChol(1)
`           Variance  StdDev
(Intercept)  0.764449 0.8743277
Residual    13.494124 3.6734350
>
> #Correcting for heterogeneous variances using nlme with
> #the reciprocal of size. Use the best model from our analyses in
table 9,
```

```
> #weighted by the reciprocal of size using individual variance per
size.
> #This is not identical but quite similar to the last line of Table
14
> Ex6lme2 <- lme(weight ~ size3 - 1, data=potato.dat,
+                random= ~1|size,
+                weights = varIdent(form = ~1|size))
> summary(Ex6lme2)
Linear mixed-effects model fit by REML
 Data: potato.dat
`      AIC      BIC  logLik
  3838.799 3872.876 -1911.4

Random effects:
 Formula: ~1 | size
`        (Intercept) Residual
StdDev:   0.8743277 3.673435

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | size
 Parameter estimates:
`   32.5     37.5     42.5     47.5     52.5     57.5
1.000000 2.039875 2.570281 3.022735 3.503492 4.425590
Fixed effects: weight ~ size3 - 1
`            Value    Std.Error DF  t-value p-value
size3 0.0007078289 5.910486e-06  5 119.7582       0

Standardized Within-Group Residuals:
`       Min          Q1         Med          Q3         Max
-2.79834772 -0.71671936 -0.04200641  0.68897783  3.22455942

Number of Observations: 524
Number of Groups: 6
> print(VarCorr(Ex6lme2) )
size = pdLogChol(1)
`           Variance  StdDev
(Intercept)  0.764449 0.8743277
Residual    13.494124 3.6734350
>
> #The estimated variance components for each size aren't
> #printed out in the model summary but rather scaled
> #from the inital one and printed as standard deviations
> #so they have to be calculated as below
> summary(Ex6lme2$modelStruct)
Random effects:
 Formula: ~1 | size
`        (Intercept) Residual
StdDev:   0.2380137        1

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | size
 Parameter estimates:
```

```
`   32.5     37.5     42.5     47.5     52.5     57.5
1.000000 2.039875 2.570281 3.022735 3.503492 4.425590
> (c(1.0000000, coef( Ex6lme2$modelStruct$varStruct,
+                     unconstrained=F))*Ex6lme2$sigma)^2
`              37.5      42.5      47.5      52.5      57.5
 13.49412  56.15029  89.14681 123.29484 165.63305 264.29381
>
> #Compare these models,
> #weighted regression with individual weights (best model)
> png('Ch6.Fig27a.png', width = 8, height = 8,
+     units = 'in', res = 600). . . TRUNCATED

> #Compare this plot to the same from the unweighted analysis
> png('Ch6.Fig27b.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . . TRUNCATED
>
> #Histogram plots of residuals, unweighted analysis
> png('Ch6.Fig28a.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . . TRUNCATED
> hist(Ex6m3$res, breaks = 20, freq = F,
+      xlim = c(-30,30), ylim = c(0,.06),
+      xlab = 'Residuals', ylab = ). . . TRUNCATED
> png('Ch6.Fig28b.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . . TRUNCATED
>
> #Note: The model below did not converge.  It didn't converge #in SAS
either. See comments about this in Chapter 6 #text.Since it wasn't the
best model, we didn't find an R solution
> Ex6HVexp <- lme(weight~size3 - 1,
+                 random= ~1|size,
+                 weights = varExp(form = ~size3|size),
data=potato.dat, )
Error in lme.formula(weight ~ size3 - 1, random = ~1 | size, weights =
varExp(form = ~size3 |  :
  nlminb problem, convergence error code = 1
  message = false convergence (8) ). . . TRUNCATED
>
> # Reference that helped me through this.
> #Clark, M., Mixed Models in R.  CSCAR, ARC, Univ. of MI.  2018-02-
08,  https://m-clark.github.io/mixed-models-with-R/
==========================================================
```

Example 7. Table 15 and Fig. 14 in Chapter 6.

In this example the authors use regression to determine if the first four years of apple yields can be used to predict the total yield over ten years. There are two varieties (A and B) and 30 rootstocks for each. In the following script, I have simply copied and pasted the steps to conduct the analyses for each variety. There are ways to automate this in R, but it would require writing a user defined function. That is not a big problem, but for small numbers it is easier to copy and paste. In the code that follows, the regression model names are according to jb's conventions rather than kagc's.

Script 6.15 jb

```
#Activate additional packages needed
options(scipen=7 )
library(ggplot2) #filter()

#Data management
#Read in dataset, change to numeric, and attach
ap.dat<-read.csv('apple.csv',header=TRUE,sep=',')
ap.dat$Year1_4 <- as.numeric(ap.dat$Year1_4)
ap.dat$Year1_10 <- as.numeric(ap.dat$Year1_10)
attach(ap.dat)
#check data has entered correctly
head(ap.dat); tail(ap.dat)
summary(ap.dat)

#Statistical analyses
#extract variety A data and run regression
Adat <- filter(ap.dat, variety=='A')
mA <- lm(Year1_10~Year1_4, data=Adat)
print(summary(mA)); confint(mA)
#extract variety B data and run regression
Bdat <- filter(ap.dat, variety=='B')
mB <- lm(Year1_10~Year1_4, data=Bdat)
print(summary(mB)); confint(mB)

#plot like Fig. 14 in Chapter 6
#add prediction interval limits to data frame
pA <- predict(mA, interval='prediction')
pB <- predict(mB, interval='prediction')
p_int<- rbind(pA, pB)
ap.dat14<- data.frame(ap.dat, p_int)
#plot regression lines + confidence & prediction intervals
gg1 <-ggplot(ap.dat14,       aes(x=Year1_4,y=Year1_10,shape=variety))+
      geom_ribbon(aes(ymin=lwr,
      ymax=upr, fill='predict.'),
      alpha=0.3) +  geom_point() +
      stat_smooth(method=lm,
      aes(fill='conf.')) + theme_bw() +
      labs(title='cumulative yield after 10 years
      against after 4 years',
      x='yield (kg/ha) year 1 to 4')+
      theme(legend.position='bottom')
gg1
ggsave('Ch6.Fig29.png', gg1, dpi=600)
```

Output 6.15 jb

```
> #Activate additional packages needed
> options(scipen=7 )

> library(ggplot2)      #ggplot()
> library(dplyr)        #filter()

> #Data management
```

```
> #Read in dataset, change to numeric, and attach
> ap.dat<-read.csv('apple.csv',header=TRUE,sep=',')
> ap.dat$Year1_4 <- as.numeric(ap.dat$Year1_4)
> ap.dat$Year1_10 <- as.numeric(ap.dat$Year1_10)
> attach(ap.dat)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #Statistical analyses
> #extract variety A data and run regression
> Adat <- filter(ap.dat, variety=='A')
> mA <- lm(Year1_10~Year1_4, data=Adat)
> print(summary(mA)); confint(mA)
Call:
lm(formula = Year1_10 ~ Year1_4, data = Adat)

Residuals:
`   Min      1Q  Median      3Q     Max
-11.846  -4.420  -0.132   4.260  14.008

Coefficients:
Term        Estimate Std. Error t value Pr(>|t|)
(Intercept)  51.4332     6.9155   7.437 4.23e-08 ***
Year1_4       2.4282     0.1758  13.815 5.01e-14 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.475 on 28 degrees of freedom
Multiple R-squared:  0.8721,    Adjusted R-squared:  0.8675
F-statistic: 190.9 on 1 and 28 DF,  p-value: 5.01e-14
```

The above estimated intercept, slope for Year1_4, SEs, t-values, and Pr(>|t|) match those from SAS for Variety A as reported in Table 15 of Chapter 6. Also the Residual standard error, degrees of freedom, R-square, and Adj. R-square match those from SAS for Variety A as stated below Table 15 and to the right of Fig. 14 in Chapter 6.

```
term            2.5 %    97.5 %
(Intercept) 37.267532 65.598851
Year1_4      2.068178  2.788235
```

The above lower and upper limits of the 95% confidence intervals on the parameter estimates equal those from SAS for Variety A as reported in Table 15 of Chapter 6.

```
> #extract variety B data and run regression
> Bdat <- filter(ap.dat, variety=='B')
> mB <- lm(Year1_10~Year1_4, data=Bdat)
> print(summary(mB)); confint(mB)

Call:
lm(formula = Year1_10 ~ Year1_4, data = Bdat)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9471 -2.8533  0.0959  2.2091  7.1544
```

```
Coefficients:
Term        Estimate Std. Error t value Pr(>|t|)
(Intercept) 46.68211    2.81354   16.59 5.11e-16 ***
Year1_4      1.10155    0.06925   15.91 1.49e-15 ***
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.505 on 28 degrees of freedom
Multiple R-squared:  0.9004,    Adjusted R-squared:  0.8968
F-statistic: 253.1 on 1 and 28 DF,  p-value: 1.489e-15
```

The above estimated intercept, slope for Year1_4, SEs, t-values, and Pr(>|t|) match those from SAS for Variety B as reported in Table 15 of Chapter 6. Also the Residual standard error, degrees of freedom, R-square, and Adj. R-square match those from SAS for Variety B as stated below Table 15 and to the right of Fig. 14 in Chapter 6.

```
Term              2.5 %    97.5 %
(Intercept) 40.9188262 52.445393
Year1_4      0.9597038  1.243391
```

The above lower and upper limits of the 95% confidence intervals on the parameter estimates equal those from SAS for Variety B as reported in Table 15 of Chapter 6.

```
> #plot like Fig. 14 in Chapter 6
> #add prediction interval limits to data frame
> pA <- predict(mA, interval='prediction')
> pB <- predict(mB, interval='prediction')
> p_int<- rbind(pA, pB)
> ap.dat14<- data.frame(ap.dat, p_int)

> #plot regression lines + confidence & prediction intervals
> gg1 <-ggplot(ap.dat14,     aes(x=Year1_4,y=Year1_10,shape=variety))+
      geom_ribbon(aes(ymin .... [TRUNCATED]
> gg1
> ggsave('Ch6.Fig29.png', gg1, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.29 in this appendix.

```
Warning messages:
1: In predict.lm(mA, interval = 'prediction') :
  predictions on current data refer to _future_ responses

2: In predict.lm(mB, interval = 'prediction') :
  predictions on current data refer to _future_ responses

3: In data.row.names(row.names, rowsi, i) :
  some row.names duplicated:
31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,5
4,55,56,57,58,59,60 --> row.names NOT used
>
```

The warning messages are informational and don't indicate any problems.

==========

Example 7. Tables 16 and 17 plus Fig. 15 and 16.

The authors continued the use of the apple yield data in Example 7 to examine four approaches to modeling when there are both qualitative and quantitative explanatory variables. Briefly, these four approaches are (a) intercepts and slopes equal model, (b) both intercepts and slopes different among groups model, (c) only slopes are different model, and (d) only the intercepts are different model. Their SAS analyses concluded that Approach c gave the best model, but that the variances between groups were not equal so they modeled that, too. The following scripts reconstruct their SAS analyses and related graphs.

Script 6.16 jb

```
#Activate additional packages needed
options(scipen=7 )
library(ggplot2)      #ggplot()
library(car)          #Anova()
library(nlme)         #gls()
library(AICcmodavg)   #predictSE()

#Data management
#Read in dataset, change to numeric, and attach
ap.dat<-read.csv("apple.csv",header=TRUE,sep=",")
ap.dat$Year1_4 <- as.numeric(ap.dat$Year1_4)
ap.dat$Year1_10 <- as.numeric(ap.dat$Year1_10)
attach(ap.dat)
#check data has entered correctly
head(ap.dat); tail(ap.dat)
summary(ap.dat)

#Statistical analyses
#run regression on all 60 cases - Approach A
mAB <- lm(Year1_10~Year1_4, data=ap.dat)
Anova(mAB, type=3)
print(summary(mAB)); confint(mAB)

#plot like upper one in Fig. 15 in Chapter 6
#add prediction interval limits to data frame
pAB <- predict(mAB, interval="prediction")
ap.datu<- data.frame(ap.dat, pAB)
#plot regression lines + confidence & prediction intervals
gg2u <-ggplot(ap.datu, aes(x=Year1_4,y=Year1_10))+
  geom_ribbon(aes(ymin=lwr, ymax=upr, fill="predict."),
   alpha=0.3)+
  geom_point(aes(shape=variety)) + stat_smooth(method=lm,
   aes(fill="conf."))+theme_bw()+
  labs(title="cumulative yield after 10 years against after 4
years", x="yield (kg/ha) year 1 to 4")+
    theme(legend.position="bottom")
gg2u
ggsave("Ch6-Fig6.30a.png", gg2u, dpi=600)
```

```
#run regression on all 60 cases -Approach b
contrasts(ap.dat$variety) <- contr.SAS
mABv <- lm(Year1_10 ~ Year1_4*variety, data=ap.dat)
print(summary(mABv)); confint(mABv)
Anova(mABv, type=2)

#run regression on all 60 cases,
#Approach c - homogeneous variance model
contrasts(ap.dat$variety) <- contr.SAS
mh <- gls(Year1_10 ~ Year1_4 + Year1_4:variety, data=ap.dat)
print(summary(mh)); confint(mh)
Anova(mh, type=3)

#plot like one on lower left in Fig. 15 in Chapter 6
#add confidence interval limits to data frame
#standard predict() doesn't give SEs for gls models
n <- length(ap.dat$variety)
tval <- qt(0.975, n-2); tval
ph <- predictSE(mh, ap.dat, se.fit=T)
lwr <- ph$fit-tval*ph$se.fit
upr <- ph$fit+tval*ph$se.fit
ap.datllz <- data.frame(ap.dat, ph, lwr, upr)
#ap.datllz            #remove # to print CIs
#plot regression lines + confidence intervals
ggllz <-ggplot(ap.datllz, aes(x=Year1_4,
   y=Year1_10,shape=variety))+
   geom_ribbon(ymin=lwr, ymax=upr, fill="red", alpha=0.3)+
   geom_point() + geom_line(aes(x=Year1_4, y=fit))+
   theme_bw()+
   labs(title="cumulative yield after 10 years against after 4
years", x="yield (kg/ha) year 1 to 4")+
      theme(legend.position="bottom")
ggllz
ggsave("Ch6-Fig6.30b.png", ggllz, dpi=600)

#make data frame & plot residuals versus predicted values
fith <- fitted(mh); resh <- residuals(mh, type="pearson")
dfh <- data.frame(fith, resh)
gglr <-ggplot(dfh, aes(x=fith, y=resh, shape=variety))+
   geom_point() + theme_bw()+
   labs(title="Internally Studentized Residuals",
     x="Predicted value year 1 to 10",y="Studentized Residuals")+
   geom_hline(yintercept=0, linetype="dashed")+
   theme(legend.position="bottom")
gglr
ggsave("Ch6-Fig6.31.png", gglr, dpi=600)

#run regression on all 60 cases,
#Approach c - unequal variance model
contrasts(ap.dat$variety) <- contr.SAS
m.un <- gls(Year1_10 ~ Year1_4 + Year1_4:variety,
  weights=varIdent(form=~1|variety), data=ap.dat)
```

```
print(summary(m.un)); confint(m.un)
Anova(m.un, type=3)
#likelihood ratio test of parameter for unequal variance
anova(mh, m.un)
cat("common intercept: ", coef(m.un)[1])
cat("slope Variety A: ", coef(m.un)[2]+coef(m.un)[3])
cat("slope Variety B: ", coef(m.un)[2])

#plot like one on left in Fig. 16 in Chapter 6
#add confidence interval limits to data frame
#even predictSE() doesn't include unequal variance estimates
#so it is done it manually
#also no extractor for difference std dev factors found
sdfb <- 0.545227      #std dev factor for variety B
nun <- length(ap.dat$variety)
se.un <- ap.datllz$se.fit*(m.un$sigma/mh$sigma)
lwr.un <- ifelse(variety=="A", ap.datllz$fit-tval*se.un,
ap.datllz$fit-tval*se.un*sdfb)
upr.un <- ifelse(variety=="A", ap.datllz$fit+tval*se.un,
ap.datllz$fit+tval*se.un*sdfb)
ap.dat.un.z <- data.frame(ap.dat, ph$fit, se.un, lwr.un, upr.un)
#ap.dat.un.z          #remove # to print CIs
#plot regression lines + confidence intervals
gg.un <-ggplot(ap.dat.un.z, aes(x=Year1_4,
   y=Year1_10,shape=variety))+
   geom_ribbon(ymin=lwr.un, ymax=upr.un, fill="red", alpha=0.3)+
   geom_point() + geom_line(aes(x=Year1_4, y=ph.fit))+
   geom_point() + theme_bw()+
   labs(title="cumulative yield after 10 years against after 4
years",
     x="yield (kg/ha) year 1 to 4")+
   theme(legend.position="bottom")
gg.un
ggsave("Ch6-Fig6.32a.png", gg.un, dpi=600)

#plot like one on right in Fig. 16 in Chapter 6
#make data frame & plot residuals versus predicted values
fit.un <- fitted(m.un); res.un <- residuals(m.un, type="pearson")
df.un <- data.frame(fit.un, res.un)
gglr <-ggplot(df.un, aes(x=fit.un, y=res.un, shape=variety))+
   geom_point() + theme_bw()+
   labs(title="Internally Studentized Residuals",
     x="Predicted value year 1 to 10",y="Studentized Residuals")+
   geom_hline(yintercept=0, linetype="dashed")+
   theme(legend.position="bottom")
gglr
ggsave("Ch6-Fig6.32b.png", gglr, dpi=600)
```

Output 6.16 jb

```
--------------------------------------------------------------------------------------------
> #Activate additional packages needed
```

```
> options(scipen=7 ) . . . TRUNCATED

> #Data management . . . TRUNCATED
```

Checks on input data from head(), tail(), and summary() omitted

```
> #Statistical analyses
> #run regression on all 60 cases - Approach a
> mAB <- lm(Year1_10~Year1_4, data=ap.dat)
> Anova(mAB, type=3)
Anova Table (Type III tests)

Response: Year1_10
Term         Sum Sq Df F value    Pr(>F)
(Intercept)    9916  1  11.292 0.001381 **
Year1_4        7572  1   8.623 0.004753 **
Residuals     50929 58
---------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above df, sums of squares, mean squares F-value, and Pr(>F) agree with those from SAS as stated in Table 16A in Chapter 6.

```
> print(summary(mAB)); confint(mAB)

Call:
lm(formula = Year1_10 ~ Year1_4, data = ap.dat)

Residuals:
`   Min     1Q Median     3Q    Max
-37.51 -28.36  -4.69  27.58  51.49

Coefficients:
Term         Estimate Std. Error t value Pr(>|t|)
(Intercept)  63.5403    18.9084   3.360  0.00138 **
Year1_4       1.3883     0.4728   2.936  0.00475 **
---------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 29.63 on 58 degrees of freedom
Multiple R-squared:  0.1294,    Adjusted R-squared:  0.1144
F-statistic: 8.623 on 1 and 58 DF,  p-value: 0.004753
```

The above estimates, SEs, t-values, Pr(>|t|), residual standard error ($\sqrt{878.08} = 29.63$), error df, and adjusted R-squared equal those from SAS as reported in Table 16A in Chapter 6.

```
Term             2.5 %     97.5 %
(Intercept) 25.6909341 101.389677
Year1_4      0.4419505   2.334715
```

The above 95% confidence intervals on the parameter estimates match those from SAS as reported in lower right section of Table 16A of Chapter 6.

```
> #plot like upper one in Fig. 15 in Chapter 6
>                              . . . TRUNCATED
> gg2u
> ggsave('Ch6.Fig30a.png', gg2u, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.30a in this appendix.

```
> #run regression on all 60 cases -Approach b
> contrasts(ap.dat$variety) <- contr.SAS
> mABv <- lm(Year1_10 ~ Year1_4*variety, data=ap.dat)
> print(summary(mABv)); confint(mABv)

Call:
lm(formula = Year1_10 ~ Year1_4 * variety, data = ap.dat)

Residuals:
`    Min       1Q   Median       3Q      Max
-11.8461  -3.7932   0.0959   2.7256  14.0078

Coefficients:
Term             Estimate Std. Error t value Pr(>|t|)
(Intercept)       46.6821     4.1799  11.168 7.17e-16 ***
Year1_4            1.1015     0.1029  10.708 3.62e-15 ***
variety1           4.7511     6.9561   0.683    0.497
Year1_4:variety1   1.3267     0.1748   7.590 3.67e-10 ***
---------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.206 on 56 degrees of freedom
Multiple R-squared:  0.9741,    Adjusted R-squared:  0.9727
F-statistic: 700.7 on 3 and 56 DF,  p-value: < 2.2e-16
```

The above estimates, SEs, t-values, Pr(>|t|), residual standard error ($\sqrt{27.11} = 5.207$), error df, and adjusted R-squared equal those from SAS as reported in Table 16B in Chapter 6.

```
Term                  2.5 %    97.5 %
(Intercept)      38.3088476 55.055372
Year1_4           0.8954681  1.307627
variety1         -9.1835691 18.685733
Year1_4:variety1  0.9765056  1.676813
```

The above 95% confidence intervals on the parameter estimates match those from SAS as reported in lower right section of Table 16B of Chapter 6.

```
> Anova(mABv, type=3)

Anova Table (Type III tests)

Response: Year1_10
Term            Sum Sq Df  F value    Pr(>F)
(Intercept)     3381.1  1 124.7320 7.168e-16 ***
Year1_4         3108.0  1 114.6579 3.617e-15 ***
variety           12.6  1   0.4665    0.4974
```

```
Year1_4:variety 1561.5  1  57.6060 3.668e-10 ***
Residuals       1518.0 56
---------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above df, sums of squares, mean squares, F-value, and Pr(>F) for variety, Year1_4:variety, and Residuals agree with those from SAS as stated in Table 16A in Chapter 6; however, the statistics for the other terms do not match.

```
> #run regression on all 60 cases,
> #Approach c - homogeneous variance model
> contrasts(ap.dat$variety) <- contr.SAS

> mh <- gls(Year1_10 ~ Year1_4 + Year1_4:variety,   data=ap.dat)

> print(summary(mh)); confint(mh)

Generalized least squares fit by REML
  Model: Year1_10 ~ Year1_4 + Year1_4:variety
  Data: ap.dat
`      AIC      BIC    logLik
` 379.7536 387.9258 -185.8768

Coefficients:
Term                Value Std.Error  t-value p-value
(Intercept)      48.39761  3.325405 14.55390       0
Year1_4           1.06043  0.083031 12.77154       0
Year1_4:variety1  1.44379  0.033662 42.89045       0
```

The above estimates, SEs, t-values, Pr(>|t|), equal those from SAS as reported in Table 16C in Chapter 6.

```
 Correlation:
`                (Intr) Yer1_4
Year1_4          -0.960
Year1_4:variety1 -0.106 -0.092

Standardized residuals:
`       Min         Q1          Med          Q3         Max
-2.33099545 -0.77998473  0.03835789  0.59069218  2.79022039

Residual standard error: 5.181972
```

The above residual standard error matches the square root of the MSE from SAS as given in Table 16C of Chapter 6 to three decimal places.

```
Degrees of freedom: 60 total; 57 residual
Term                 2.5 %    97.5 %
(Intercept)      41.879931 54.915281
Year1_4           0.897695  1.223170
Year1_4:variety1  1.377813  1.509766
```

The above 95% confidence intervals on the parameter estimates match those from SAS as reported in lower right section of Table 16C of Chapter 6 for Year1_4 and Year1_4:variety. The limits for the intercept are slightly different resulting in an interval that is 0.21% narrower.

```
> Anova(mh, type=3)
Analysis of Deviance Table (Type III tests)

Response: Year1_10
Term            Df   Chisq Pr(>Chisq)
(Intercept)      1  211.82  < 2.2e-16 ***
Year1_4          1  163.11  < 2.2e-16 ***
Year1_4:variety  1 1839.59  < 2.2e-16 ***
---------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The function changed from lm() to gls() so we can test the fit of this model assuming equal variances against one assuming unequal variances the Anova() function produces Chi-square tests instead of F-tests. Consequently, these values don't match any of those in Table 16C, but the test results agree that all three effects are highly significant.

```
> #plot like one on lower left in Fig. 15 in Chapter 6
> #add prediction interval limits to data frame
#standard predict() doesn't give SEs for gls  .... [TRUNCATED]
> tval <- qt(0.975, n-2); tval
[1] 2.001717
> ph <- predictSE(mh, ap.dat, se.fit=T)
> lwr <- ph$fit-tval*ph$se.fit
> upr <- ph$fit+tval*ph$se.fit
> ap.datllz <- data.frame(ap.dat, ph, lwr, upr)
> #plot regression lines + confidence intervals
> ggllc <-ggplot(ap.datllc, aes(x=Year1_4,
      +    y=Year1_10,shape=variety))+
      +    geom_point() + stat_s .... [TRUNCATED]
> ggllc
> ggsave('Ch6.Fig30b.png', ggllc, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.30b in this appendix.

```
> #make data frame & plot residuals versus predicted values
> fith <- fitted(mh); resh <- residuals(mh, type='pearson')
> dfh <- data.frame(fith, resh)
> gglr <-ggplot(dfh, aes(x=fith, y=resh, shape=variety))+
      +    geom_point() + theme_bw()+
      +    labs(title='Internally Studentized Residuals',
      +      x .... [TRUNCATED]
> gglr
> ggsave('Ch6.Fig31.png', gglr, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.31 in this appendix.

```
> #run regression on all 60 cases,
> #Approach c - unequal variance model
> contrasts(ap.dat$variety) <- contr.SAS
> m.un <- gls(Year1_10 ~ Year1_4 + Year1_4:variety,
```

```
      + weights=varIdent(form=~1|variety), data=ap.dat)
> print(` (m.un)); confint(m.un)

Generalized least squares fit by REML
  Model: Year1_10 ~ Year1_4 + Year1_4:variety
  Data: ap.dat
     AIC      BIC  logLik
  371.88 382.0953 -180.94

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | variety
 Parameter estimates:
'       A         B
1.0000000 0.5452272
```

R provides the estimates of the relative standard deviations for each group whereas SAS gives the actual variance component estimates. Thus, the above values don't look like they match SAS, but they do. When you square the residual standard error stated below times the squared factor for Variety A above you get 41.90, the same as the SAS value in the upper right portion of Table 17A in Chapter 6. And when you the same for Variety B you get 12.2447, which also equals the SAS value.

```
Coefficients:
Term                 Value Std.Error  t-value p-value
(Intercept)      47.36548 2.5994245 18.22152       0
Year1_4           1.08517 0.0642533 16.88893       0
Year1_4:variety1  1.44490 0.0337925 42.75801       0
```

The point estimates for the parameters match those from SAS as given in Table 17B of Chapter 6, but the SEs and t-values are slightly different. For the intercept, the standard error is 1.7% smaller and the t-value 1.7% greater. For the Variety A by Year1_4 interaction, the SE is less than 0.1% smaller and the t-value 1.6% larger. For the interaction with Variety B, both values agree to 3 or 4 digits. All three parameter estimates are highly significant in both R and SAS.

```
 Correlation:
`                (Intr) Yer1_4
Year1_4          -0.970
Year1_4:variety1 -0.083 -0.034

Standardized residuals:
`       Min          Q1         Med          Q3         Max
-1.89443830 -0.75549886  0.04074676  0.64429787  2.27677348

Residual standard error: 6.417946
```

The residual standard error for the reference group as discussed above.

```
Degrees of freedom: 60 total; 57 residual
```

```
`                       2.5 %    97.5 %
(Intercept)       42.2706968 52.460253
Year1_4            0.9592352  1.211103
Year1_4:variety1   1.3786667  1.511131

> Anova(m.un, type=3)

Analysis of Deviance Table (Type III tests)

Response: Year1_10
Term             Df    Chisq Pr(>Chisq)
(Intercept)       1   332.02  < 2.2e-16 ***
Year1_4           1   285.24  < 2.2e-16 ***
Year1_4:variety   1 1828.25  < 2.2e-16 ***
---------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> #likelihood ratio test of parameter for unequal variance
> anova(mh, m.un)
'    Model df      AIC      BIC    logLik   Test  L.Ratio p-value
mh       1  4 379.7536 387.9258 -185.8768
m.un     2  5 371.8800 382.0953 -180.9400 1 vs 2 9.873608  0.0017
```

R calculates a log Likelihood, which is exactly half of the value SAS calculates. The penalty R imposes for the AIC value is more than the penalty SAS uses; however, the difference in AIC within each system is the same so the likelihood ratio test results are identical (Table 17A Chapter 6).

```
> cat('common intercept: ', coef(m.un)[1])
common intercept:  47.36548
> cat('slope Variety A: ', coef(m.un)[2]+coef(m.un)[3])
slope Variety A:  2.530068
> cat('slope Variety B: ', coef(m.un)[2])
slope Variety B:  1.085169
```

The above statements use R functions to combine the coefficient (parameter) estimates into the more useable intercepts and slopes for the linear equations for each variety as shown at the top of Table 17B.

```
> #plot like one on left in Fig. 16 in Chapter 6
> #add prediction interval limits to data frame
> #even predictSE() doesn't include unequal variance  .... [TRUNCATED]
> nun <- length(ap.dat$variety)
> se.un <- ap.datllz$se.fit*(m.un$sigma/mh$sigma)
> lwr.un <- ifelse(variety=="A", ap.datllz$fit-tval*se.un,
ap.datllz$fit-tval*se.un*sdfb)
> upr.un <- ifelse(variety=="A", ap.datllz$fit+tval*se.un,
ap.datllz$fit+tval*se.un*sdfb)
> ap.dat.un.z <- data.frame(ap.dat, ph$fit, se.un, lwr.un, upr.un)
> #plot regression lines + confidence intervals
> gg.un <-ggplot(ap.dat.un, aes(x=Year1_4,
      + y=Year1_10,shape=variety))+
```

```
      + geom_point() + stat_s .... [TRUNCATED]
> gg.un
> ggsave('Ch6.Fig32a.png', gg.un, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.32a in this appendix.

```
> #plot like one on right in Fig. 16 in Chapter 6
> #make data frame & plot residuals versus predicted values
> fit.un <- fitted(m.un); res.un <- resi .... [TRUNCATED]
> df.un <- data.frame(fit.un, res.un)
> gglr <-ggplot(df.un, aes(x=fit.un, y=res.un, shape=variety))+
      + geom_point() + theme_bw()+
      + labs(title='Internally Studentized Residuals',
      +  .... [TRUNCATED]
> gglr
> ggsave('Ch6.Fig32b.png', gglr, dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 6.32b in this appendix.

```
Warning messages:
1: In predict.lm(mAB, interval = 'prediction') :
  predictions on current data refer to _future_ responses
2: In predict.lm(mABv, interval = 'prediction') :
  predictions on current data refer to _future_ responses
```

The above warnings are simply informationa

==========

Example 8. Analysis as autoregressive model.

Example 8 consists of two quantitative variables, the year and the average temperature. The data are contained in a dataset named airtemp.dat. The data are used to examine additional model fitting strategies including nonlinear regression and segmented regression. This section also serves as an introduction to time series analysis.

Script 6.17 kagc

```
#Load the packages
library(lmtest)
library(aTSA)
library(TSA)
library(nlme)
library(rsq)
library(HH)
#Ex8 airtemp Load and attach data as usual.

#Run a simple regression analysis of this data
#This matches Table 21
Ex8lm1 <- lm(temp ~ year, data=airtemp.dat)

#Table 21A
```

```
anova(Ex8lm1)
#Table 21B
summary(Ex8lm1)
#Calculate and print R sqr and adj r square for Ex8lm1.
library(rsq)
r.sqr <- summary(Ex8lm1)$r.squared
adj.r.sqr <- summary(Ex8lm1)$adj.r.squared
print(r.sqr)
print(adj.r.sqr)
#The model fit is not great
#Figure 17 left
library(HH)
png('Ch6.Fig33a.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
ci.plot(Ex8lm1)
dev.off()

#Figure 17 right
#Plot residuals against the year
#Add residuals & studentized residuals to airtemp.dat
airtemp.dat$resid <- Ex8lm1$residuals
airtemp.dat$rst <- rstudent(Ex8lm1)
png('Ch6.Fig33b.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
xyplot(resid ~ year, data=airtemp.dat, ylab = 'residuals', xlab =
'Year', type = 'o',
       main='Trend analysis of residuals from
       simple linear regression (Ex8lm1) for Airtemp data')
#Scatter plots of the raw data and studentized
#residuals also indicates a trend
dev.off()
png('Ch6.Fig33c.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
xyplot(temp ~ year, data=airtemp.dat, ylab = 'Temp',
       xlab = 'Year', type = 'o',
       main='Trend analysis of raw data from
       simple linear regression for Airtemp data')
dev.off()
png('Ch6.Fig33d.png', width = 8, height = 8,
    units = 'in', res = 600)
#dev.off()
xyplot(rst ~ year, data=airtemp.dat, ylab = 'rst',
       xlab = 'Year', type = 'o',
       main='Trend analysis of student residuals from
       simple linear regression for Airtemp data')
dev.off()

#This function (from Ex1) plots residuals histograms
Residuals_hist <- function(res)   {
```

```
  par(mfrow=c(1,1))
  hist(res, breaks = 20, freq = F, xlim = c(-4,4),
       ylim = c(0,1.0),
       xlab = 'res', ylab = 'Probability', col='lightblue', main='')
  mtext('Residuals Histogram, Normal and Kernel
        Density Curve ', side=3, outer=TRUE, line= -3)
  curve(dnorm(x, mean = mean(res),
              sd = sd(res)), add = T)
  lines(density (res), col = 'red')
  legend('topright', c('Normal','Kernel'),
         col = c('black','red'), lty=c(1,1),  lwd=c(2,2,2), cex=0.5)
}

#Call function for residuals plots
png('Ch6.Fig34a.png', width = 8, height = 8,
    units = 'in', res = 600)
Residuals_hist(airtemp.dat$res)
dev.off()
png('Ch6.Fig34b.png', width = 8, height = 8,
    units = 'in', res = 600)
#Call function for students residuals plots
Residuals_hist(airtemp.dat$rst)
dev.off()
png('Ch6.Fig34c.png', width = 8, height = 8,
    units = 'in', res = 600)
#Box plot for the residuals
boxplot(airtemp.dat$resid, col = 'blue', xlab = 'Residuals')
dev.off()
png('Ch6.Fig34d.png', width = 8, height = 8,
    units = 'in', res = 600)
boxplot(airtemp.dat$rst, col='red', xlab = 'Student residuals')
dev.off()

#Other plots for the lm model to match SAS Proc Reg
#output are as described in Ex1 using the plot function
#but are not included here
#plot(Ex8lm1)

#Use generalized least squares (gls) function
#with the maximum likliehood (ML) method
#to facilitate model comparisons below
Ex8gls1 <- gls(temp ~ year, data=airtemp.dat,
               method = 'ML')
summary(Ex8gls1)
#Note: Method = ML reduces the AIC and the standard #error of the
residuals
#The model with method='REML' (Ex8gls2) matches the earlier lm model
(Ex8lm1)
Ex8gls2 <- gls(temp ~ year,  data=airtemp.dat,
               method = 'REML')
summary(Ex8gls2)
summary(Ex8lm1)
```

```
#Use the DurbinWatson Test from the CAR package to #calculate the DW
and autocorrelation statistics for #all possible lags in the dataset
(12)
library(car)
Ex8lm1.DW <- durbinWatsonTest(Ex8lm1, max.lag=12)
print(Ex8lm1.DW)
#The 1st, 3rd and 4th lags are signficantly autocorrelated.

#Based on these results we use the update command to
#run several models based on Ex8gls1
#but with differenct correlation functions
Ex8AR1 <- update(Ex8gls1,
                 correlation = corAR1(value = 0.235,form = ~1))
summary(Ex8AR1)
#Compare the two models.
#Our AIC values are slightly higher than reported by #SAS but the
relationship is the same
anova(Ex8gls1,Ex8AR1)
#AR1 is the better model at the p=0.0796 level

library(TSA)  #ts
#Convert the temp, residuals and student residuals from
#the first model into time series data and plot
temp_ts <- ts(airtemp.dat$temp, start = 1960,
              end = 2013)
resid_ts <- ts(airtemp.dat$resid, start = 1960,
               end = 2013 )
rst_ts <- ts(airtemp.dat$rst, start = 1960,
             end = 2013 )
plot.ts(temp_ts)
plot.ts(resid_ts)
plot.ts(rst_ts)
#These the same as our earlier xy plots
#so they are not reproduced here.
#The TS data will be used below.

#Use these datapoints to run a stepwise forward #analysis to fit an AR
model
library(aTSA)
stepar(temp_ts, trend = c('linear'),
       order = NULL, lead = 1,  output = TRUE)
#Based on these and our previous results, a lag of 4 seems reasonable.
#In the SAS code, PROC AUTOREG is used with a backwards algorithim to
fit
#the model.  We simply ran all the possile models here.
#Run models with lags of 1 through 12.
#note that the ARMA function with a p=1 is a synonym
#for the corAR1 function above.
Ex8AR1 = update(Ex8gls1,
                correlation = corARMA(p=1))
Ex8AR2 = update(Ex8gls1,
                correlation = corARMA(p=2))
Ex8AR3 = update(Ex8gls1,
                correlation = corARMA(p=3))
```

```
Ex8AR4 = update(Ex8gls1,
                correlation = corARMA(p=4))
Ex8AR5 = update(Ex8gls1,
                correlation = corARMA(p=5))
Ex8AR6 = update(Ex8gls1,
                correlation = corARMA(p=6))
Ex8AR7 = update(Ex8gls1,
                correlation = corARMA(p=7))
Ex8AR8 = update(Ex8gls1,
                correlation = corARMA(p=8))
Ex8AR9 = update(Ex8gls1,
                correlation = corARMA(p=9))
Ex8AR10 = update (Ex8gls1,
                  correlation = corARMA(p=10))
Ex8AR11 = update(Ex8gls1,
                 correlation = corARMA(p=11))
Ex8AR12 = update(Ex8gls1,
                 correlation = corARMA(p=12))

summary(Ex8AR1)
summary(Ex8AR2)
summary(Ex8AR3)
summary(Ex8AR4)
summary(Ex8AR5)
summary(Ex8AR6)
summary(Ex8AR7)
summary(Ex8AR8)
summary(Ex8AR9)
summary(Ex8AR10)
summary(Ex8AR11)
summary(Ex8AR12)

#Compare to the model without autocorrelation
anova(Ex8gls1, Ex8AR1)
anova(Ex8gls1, Ex8AR2)
anova(Ex8gls1, Ex8AR3)
anova(Ex8gls1, Ex8AR4)
anova(Ex8gls1, Ex8AR5)
anova(Ex8gls1, Ex8AR6)
anova(Ex8gls1, Ex8AR7)
anova(Ex8gls1, Ex8AR8)
anova(Ex8gls1, Ex8AR9)
anova(Ex8gls1, Ex8AR10)
anova(Ex8gls1, Ex8AR11)
anova(Ex8gls1, Ex8AR12)

#Models AR1 though AR6 are significantly better
#than the model without autocorrelation (P< 0.10).
#The AIC values are lower than the model without #autocorrelation for
models AR1-AR5.
#The lowest AIC is for model AR4.

#Compare the model with AR1 to AR4)
anova(Ex8AR1, Ex8AR4)
```

```
#Ex8AR4 is better than Ex9AR1.
#To calculate 95% CI for model parameters
intervals(Ex8AR1)
intervals(Ex8AR4)

#Can't use Ci.plot from the HH package for gls models so calculate by
hand
#Obtain predicted (fitted) values with predictSE #function from
AICmodavg package
library(AICcmodavg)
#Create a new data set with the fit and
#standard error for the model parameters
AR1_PI <- data.frame(predictSE.gls(Ex8AR1, newdata = airtemp.dat))
head(AR1_PI)
#Calculate the predication intervals
AR1_PI$UPI <- AR1_PI$fit - qnorm(AR1_PI$se.fit)
AR1_PI$LPI <- AR1_PI$fit + qnorm(AR1_PI$se.fit)
head(AR1_PI)

#Repeat for the EX8AR4 model
AR4_PI <- data.frame(predictSE.gls(Ex8AR4, newdata = airtemp.dat))
AR4_PI$UPI <- AR4_PI$fit - qnorm(AR4_PI$se.fit)
AR4_PI$LPI <- AR4_PI$fit + qnorm(AR4_PI$se.fit)
head(AR4_PI)

#To plot, add these new prediction intervals to the airtemp dataset,
#While we are at it, add resdiuals from the models
airtemp.dat$AR1_resid=Ex8AR1$residuals
airtemp.dat$AR1_fit=AR1_PI$fit
airtemp.dat$AR1_se.fit=AR1_PI$se.fit
airtemp.dat$AR1_LPI=AR1_PI$LPI
airtemp.dat$AR1_UPI=AR1_PI$UPI

airtemp.dat$AR4_resid=Ex8AR4$residuals
airtemp.dat$AR4_fit=AR4_PI$fit
airtemp.dat$AR4_se.fit=AR4_PI$se.fit
airtemp.dat$AR4_LPI=AR4_PI$LPI
airtemp.dat$AR4_UPI=AR4_PI$UPI

#Plot 18 top two graphs just the prediction intervals
png('Ch6.Fig35a.png', width = 8, height = 8,
    units = 'in', res = 600)
par(mfrow=c(1,1) )
plot(temp~year, data=airtemp.dat, col =('black'),pch=1, xlab =
('Year'),
     ylab = ('Temp'), new = FALSE )
lines(AR1_fit~year, data=airtemp.dat,col='black',lwd=2,lty=1)
lines(AR1_LPI~year, data=airtemp.dat,col='blue',lwd=2,lty=3)
lines(AR1_UPI~year, data=airtemp.dat,col='blue',lwd=2,lty=3)
mtext('Fig. 18. Observed and fitted values with
      95% prediction intervals for AR1.', side=3,  outer=TRUE, line=-
3)
#placing the legend on the graph
```

```
legend('topleft',
       c('Observed','Fitted','PI'),
       col = c('black','black','blue'),
       lty=c(0,1,3), lwd=c(NA,2,2),
       pch=c(1,NA,NA), cex=0.7)
dev.off()

png('Ch6.Fig35b.png', width = 8, height = 8,
    units = 'in', res = 600)
par(mfrow=c(1,1) )
plot(temp~year, data=airtemp.dat, col =('black'),pch=1, xlab =
('Year'),
     ylab = ('Temp'), new = FALSE )
lines(AR4_fit~year, data=airtemp.dat,col='black',lwd=2,lty=1)
lines(AR4_LPI~year, data=airtemp.dat,col='blue',lwd=2,lty=3)
lines(AR4_UPI~year, data=airtemp.dat,col='blue',lwd=2,lty=3)
mtext('Fig. 17. Observed and fitted values with
      95% prediction intervals for AR4.', side=3,  outer=TRUE, line=-
3)
#placing the legend on the graph
legend('topleft',
       c('Observed','Fitted','Pred. Interval'),
       col = c('black','black','blue'),            lty=c(0,1,3),
lwd=c(NA,2,2),
       pch=c(1,NA,NA), cex=0.7)
dev.off()

#Plot the original data and then use moving
#averages over 4, 6, and 8 years
#to smooth the plots and look at large trends
#The temperatures fluctuate but are increasing
#over time since 1990.  The increasing temperatures
#began in the between 1975 and 1985

library(TTR)
png('Ch6.Fig36a.png', width = 8, height = 8,
    units = 'in', res = 600)
par(mfrow=c(1,1) )
plot.ts(temp_ts)
dev.off()
temp_tsSMA4 <- SMA(temp_ts,n=4)
png('Ch6.Fig36b.png', width = 8, height = 8,
    units = 'in', res = 600)
plot.ts(temp_tsSMA4)
dev.off()
temp_tsSMA6 <- SMA(temp_ts,n=6)
png('Ch6.Fig36c.png', width = 8, height = 8,
    units = 'in', res = 600)
plot.ts(temp_tsSMA6)
dev.off()
temp_tsSMA8 <- SMA(temp_ts,n=8)
png('Ch6.Fig36d.png', width = 8, height = 8,
    units = 'in', res = 600)
plot.ts(temp_tsSMA8)
```

```
dev.off()

#Plotting the sutocorrelation function (ACF) and
#the partial autocoreelation function (PACF)
#for the studentized residuals
#from the original model indicates that it the
#autocorrelation alternates sign
#and the first 4 lags are the largest.
png('Ch6.Fig37a.png', width = 8, height = 8,
    units = 'in', res = 600)
acf(airtemp.dat$rst, lag.max = 12, plot = TRUE)
dev.off()
png('Ch6.Fig37b.png', width = 8, height = 8,
    units = 'in', res = 600)
pacf(airtemp.dat$rst, lag.max = 12, plot = TRUE)
dev.off()
#Plotting the sutocorrelation function (ACF) and
#the partial autocoreelation function (PACF)
#for the studentized residuals
#from the original model indicates that the
#autocorrelation alternates in sign
#and the first 4 lags are the largest.
#A sine curve might model these results
#Can we model this using nonlinear regression?
```

Output 6.17 kagc

```
> #Ex8 airtemp Load and attach data as usual.
> # Load the packages
>
> #Run a simple regression analysis of this data
> #This matches Table 21
> Ex8lm1 <- lm(temp ~ year, data=airtemp.dat)
>
> #Table 21A
> anova(Ex8lm1)
Analysis of Variance Table

Response: temp
`          Df  Sum Sq Mean Sq F value    Pr(>F)
year        1  8.0857  8.0857  16.209 0.000185 ***
Residuals 52 25.9399  0.4988
---------------------------------------------
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> #Table 21B
> summary(Ex8lm1)

Call:
lm(formula = temp ~ year, data = airtemp.dat)

Residuals:
`     Min       1Q   Median       3Q      Max
-1.86938 -0.40904 -0.01404  0.55821  1.10131
```

```
Coefficients:
`             Estimate Std. Error t value Pr(>|t|)
(Intercept) -40.346351  12.250650  -3.293 0.001785 **
year          0.024828   0.006167   4.026 0.000185 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7063 on 52 degrees of freedom
Multiple R-squared:  0.2376, Adjusted R-squared:  0.223
F-statistic: 16.21 on 1 and 52 DF,  p-value: 0.000185

> #Calculate and print R sqr and adj r square for Ex8lm1.
> library(rsq)
> r.sqr <- summary(Ex8lm1)$r.squared
> adj.r.sqr <- summary(Ex8lm1)$adj.r.squared
> print(r.sqr)
[1] 0.2376356
> print(adj.r.sqr)
[1] 0.2229748
> #The model fit is not great

> #Figure 17 left
> library(HH)
> png('Ch6.Fig33a.png', width = 8, height = 8,
+     units = 'in', res = 600). . .TRUNCATED
> #dev.off()
>
> #Figure 17 right
> #Plot residuals against the year
> #Add residuals & studentized residuals to airtemp.dat
> airtemp.dat$resid <- Ex8lm1$residuals
> airtemp.dat$rst <- rstudent(Ex8lm1)
> png('Ch6.Fig33b.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . .TRUNCATED
> png('Ch6.Fig33c.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . .TRUNCATED
> png('Ch6.Fig33d.png', width = 8, height = 8,
+     units = 'in', res = 600) ). . .TRUNCATED
>
> #This function (from Ex1) plots residuals histograms
> Residuals_hist <- function(res) {
+   par(mfrow=c(1,1))
+   hist(res, breaks = 20, freq = F, xlim = c(-4,4),
+        ylim = c(0,1.0), . . .TRUNCATED

> #Call function for residuals plots
> png('Ch6.Fig34a.png', width = 8, height = 8,
+     units = 'in', res = 600)
> Residuals_hist(airtemp.dat$res) . . .TRUNCATED
> png('Ch6.Fig34b.png', width = 8, height = 8,
+     units = 'in', res = 600)
> #Call function for students residuals plots..TRUNCATED
> png('Ch6.Fig34c.png', width = 8, height = 8,
```

```
+     units = 'in', res = 600)
> #Box plot for the residuals. . .TRUNCATED
> png('Ch6.Fig34d.png', width = 8, height = 8,
+       units = 'in', res = 600) . . .TRUNCATED
> #Other plots for the lm model to match SAS Proc Reg
> #output are as described in Ex1 using the plot function
> #but are not included here
> #plot(Ex8lm1)
>
> #Use generalized least squares (gls) function
> #with the maximum likliehood (ML) method
> #to facilitate model comparisons below
> Ex8gls1 <- gls(temp ~ year, data=airtemp.dat,
+                method = 'ML')
> summary(Ex8gls1)
Generalized least squares fit by maximum likelihood
  Model: temp ~ year
  Data: airtemp.dat
`      AIC      BIC    logLik
  119.6525 125.6195 -56.82626

Coefficients:
`               Value Std.Error   t-value p-value
(Intercept) -40.34635 12.250650 -3.293405  0.0018
year          0.02483  0.006167  4.026022  0.0002

 Correlation:
`    (Intr)
year -1

Standardized residuals:
`      Min         Q1        Med         Q3        Max
-2.6971809 -0.5901664 -0.0202500  0.8053925  1.5889933

Residual standard error: 0.6930866
Degrees of freedom: 54 total; 52 residual
> #Note: Method = ML reduces the AIC and the standard #error of the
residuals
> #The model with method='REML' (Ex8gls2) matches the earlier lm model
(Ex8lm1)
> Ex8gls2 <- gls(temp ~ year,  data=airtemp.dat,
+                method = 'REML')
> summary(Ex8gls2)
Generalized least squares fit by REML
  Model: temp ~ year
  Data: airtemp.dat
`      AIC      BIC    logLik
  130.8764 136.7301 -62.43818

Coefficients:
`               Value Std.Error   t-value p-value
(Intercept) -40.34635 12.250650 -3.293405  0.0018
year          0.02483  0.006167  4.026022  0.0002
```

```
 Correlation:
‘    (Intr)
year -1

Standardized residuals:
‘       Min          Q1         Med          Q3         Max
-2.64676185 -0.57913428 -0.01987146  0.79033710  1.55928983

Residual standard error: 0.7062894
Degrees of freedom: 54 total; 52 residual
> summary(Ex8lm1)

Call:
lm(formula = temp ~ year, data = airtemp.dat)

Residuals:
‘    Min       1Q   Median       3Q      Max
-1.86938 -0.40904 -0.01404  0.55821  1.10131

Coefficients:
‘             Estimate Std. Error t value Pr(>|t|)
(Intercept) -40.346351  12.250650  -3.293 0.001785 **
year          0.024828   0.006167   4.026 0.000185 ***
---------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7063 on 52 degrees of freedom
Multiple R-squared:  0.2376, Adjusted R-squared:  0.223
F-statistic: 16.21 on 1 and 52 DF,  p-value: 0.000185

>
> #Use the DurbinWatson Test from the CAR package to #calculate the DW
and autocorrelation statistics for #all possible lags in the dataset
(12)
> library(car)
> Ex8lm1.DW <- durbinWatsonTest(Ex8lm1, max.lag=12)
> print(Ex8lm1.DW)
 lag Autocorrelation D-W Statistic p-value
   1      0.23493934      1.511816   0.072
   2     -0.13771776      2.213469   0.468
   3     -0.28629705      2.492894   0.046
   4     -0.28035284      2.376058   0.084
   5     -0.10494001      2.024829   0.640
   6      0.08204596      1.632289   0.432
   7      0.15598224      1.442182   0.172
   8      0.13942158      1.428723   0.226
   9      0.08921885      1.527836   0.494
  10     -0.08118943      1.841222   0.608
  11     -0.05452713      1.772181   0.638
  12     -0.17899644      2.009403   0.130
 Alternative hypothesis: rho[lag] != 0
> #The 1st, 3rd and 4th lags are signficantly autocorrelated.
>
> #Based on these results we use the update command to
```

```
> #run several models based on Ex8gls1
> #but with differenct correlation functions
> Ex8AR1 <- update(Ex8gls1,
+   correlation = corAR1(value = 0.235,form = ~1))
> summary(Ex8AR1)
Generalized least squares fit by maximum likelihood
  Model: temp ~ year
  Data: airtemp.dat
`      AIC      BIC    logLik
  118.5794 126.5354 -55.28971

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
`     Phi
0.2353419

Coefficients:
`               Value Std.Error   t-value p-value
(Intercept) -39.19711 15.301123 -2.561715  0.0134
year          0.02425  0.007702  3.148398  0.0027

 Correlation:
'    (Intr)
year -1

Standardized residuals:
`       Min          Q1         Med          Q3         Max
-2.69329852 -0.60171190 -0.01536679  0.81123797  1.59832378

Residual standard error: 0.6927451
Degrees of freedom: 54 total; 52 residual
> #Compare the two models.
> #Our AIC values are slightly higher than reported by #SAS but the
relationship is the same
> anova(Ex8gls1,Ex8AR1)
        Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR1      2  4 118.5794 126.5354 -55.28971 1 vs 2 3.073104  0.0796
> #AR1 is the better model at the p=0.0796 level
>
> library(TSA)  #ts
> #Convert the temp, residuals and student residuals from
> #the first model into time series data and plot
> temp_ts <- ts(airtemp.dat$temp, start = 1960,
+               end = 2013)
> resid_ts <- ts(airtemp.dat$resid, start = 1960,
+                end = 2013 )
> rst_ts <- ts(airtemp.dat$rst, start = 1960,
+              end = 2013 )
> plot.ts(temp_ts)
> plot.ts(resid_ts)
> plot.ts(rst_ts)
> #These the same as our earlier xy plots
```

```
> #so they are not reproduced here.
> #The TS data will be used below.
>
> #Use these datapoints to run a stepwise forward #analysis to fit an
AR model
> library(aTSA)
> stepar(temp_ts, trend = c('linear'),
+        order = NULL, lead = 1,  output = TRUE)
Parameter of estimates for stepwise AR model:
`           Estimate Std. Error t value Pr(>|t|)
(Intercept)    8.291      0.195   42.53 4.47e-42
t              1.341      0.333    4.03 1.85e-04
AR1            0.192      0.140    1.37 1.76e-01
AR2           -0.172      0.140   -1.22 2.27e-01
AR3           -0.170      0.140   -1.22 2.30e-01
AR4           -0.209      0.140   -1.50 1.41e-01
---------------------------------------------
sigma^2 estimated as: 0.4894326 ; R.squared = 0.2376356> #Based on
these and our previous results, a lag of 4 seems reasonable.
> #In the SAS code, PROC AUTOREG is used with a backwards algorithim
to fit
> #the model.  We simply ran all the possile models here.
> #Run models with lags of 1 through 12.
> #note that the ARMA function with a p=1 is a synonym
> #for the corAR1 function above.
> Ex8AR1 = update(Ex8gls1,
+          correlation = corARMA(p=1))
> Ex8AR2 = update(Ex8gls1,. . .TRUNCATED
>
>
> summary(Ex8AR1)
Generalized least squares fit by maximum likelihood
  Model: temp ~ year
  Data: airtemp.dat
`      AIC      BIC    logLik
  118.5794 126.5354 -55.28971

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
`    Phi
0.235342

Coefficients:
`               Value Std.Error   t-value p-value
(Intercept) -39.19711 15.301124 -2.561715  0.0134
year          0.02425  0.007702  3.148398  0.0027

 Correlation:
`    (Intr)
year -1

Standardized residuals:
`       Min          Q1         Med          Q3         Max
```

```
-2.69329850 -0.60171190 -0.01536679  0.81123796  1.59832377

Residual standard error: 0.6927451
Degrees of freedom: 54 total; 52 residual
> summary(Ex8AR2) . . . TRUNCATED
>
> #Compare to the model without autocorrelation
> anova(Ex8gls1, Ex8AR1)
'       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR1      2  4 118.5794 126.5354 -55.28971 1 vs 2 3.073104  0.0796
> anova(Ex8gls1, Ex8AR2)
'       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR2      2  5 118.3731 128.3181 -54.18658 1 vs 2 5.279374  0.0714
> anova(Ex8gls1, Ex8AR3)
'       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR3      2  6 117.4622 129.3961 -52.73110 1 vs 2 8.190319  0.0422
> anova(Ex8gls1, Ex8AR4)
'       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR4      2  7 116.8401 130.7630 -51.42007 1 vs 2 10.81238  0.0288
> anova(Ex8gls1, Ex8AR5)
'       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8gls1     1  3 119.6525 125.6195 -56.82626
Ex8AR5      2  8 118.3286 134.2405 -51.16431 1 vs 2 11.32391  0.0453
> anova(Ex8gls1, Ex8AR6) . . . TRUNCATED
>
> #Models AR1 though AR6 are significantly better than the model
without autocorrelation (P< 0.10). The AIC values are lower than
> #the model without autocorrelation for models AR1-AR5.
> #The lowest AIC is for model AR4.
> #Compare the model with AR1 to AR4)
> anova(Ex8AR1, Ex8AR4)
'      Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Ex8AR1     1  4 118.5794 126.5354 -55.28971
Ex8AR4     2  7 116.8401 130.7630 -51.42007 1 vs 2 7.739278  0.0517
> #Ex8AR4 is better than Ex9AR1.
> #To calculate 95% CI for model parameters
> intervals(Ex8AR1)
Approximate 95% confidence intervals

 Coefficients:
'                  lower         est.       upper
(Intercept) -69.901064199 -39.19711317 -8.49316214
year          0.008794122   0.02424994  0.03970576
attr(,"label")
[1] "Coefficients:"

 Correlation structure:
'         lower     est.     upper
Phi -0.03358736 0.235342 0.4724925
attr(,"label")
```

```
[1] "Correlation structure:"

 Residual standard error:
`   lower      est.     upper
0.5678017 0.6927451 0.8451820
> intervals(Ex8AR4)
Approximate 95% confidence intervals

 Coefficients:
`                  lower         est.        upper
(Intercept) -60.59871180 -43.78589652 -26.97308124
year          0.01809509   0.02655839   0.03502169
attr(,"label")
[1] "Coefficients:"

 Correlation structure:
`         lower       est.      upper
Phi1 -0.4500021  0.1746912 0.41377739
Phi2 -0.6569078 -0.1755320 0.02615572
Phi3 -0.5730004 -0.1660596 0.03004165
Phi4 -0.4970473 -0.2391822 0.05750034
attr(,"label")
[1] "Correlation structure:"

 Residual standard error:
`   lower      est.     upper
0.5468838 0.6906276 0.8721533
>
> #Can't use Ci.plot from the HH package for gls models so calculate
by hand
> #Obtain predicted (fitted) values with predictSE #function from
AICmodavg package
> library(AICcmodavg)
> #Create a new data set with the fit and
> #standard error for the model parameters
> AR1_PI <- data.frame(predictSE.gls(Ex8AR1, newdata = airtemp.dat))
> head(AR1_PI)
`      fit    se.fit
1 8.332771 0.2374929
2 8.357021 0.2309068
3 8.381271 0.2243918
4 8.405521 0.2179542
5 8.429771 0.2116012
6 8.454021 0.2053405
> #Calculate the predication intervals
> AR1_PI$UPI <- AR1_PI$fit - qnorm(AR1_PI$se.fit)
> AR1_PI$LPI <- AR1_PI$fit + qnorm(AR1_PI$se.fit)
> head(AR1_PI)
`      fit    se.fit      UPI      LPI
1 8.332771 0.2374929 9.047162 7.618381
2 8.357021 0.2309068 9.092885 7.621158
3 8.381271 0.2243918 9.138716 7.623827
4 8.405521 0.2179542 9.184642 7.626400
5 8.429771 0.2116012 9.230649 7.628893
```

```
6 8.454021 0.2053405 9.276717 7.631325
>
> #Repeat for the EX8AR4 model
> AR4_PI <- data.frame(predictSE.gls(Ex8AR4, newdata = airtemp.dat))
> AR4_PI$UPI <- AR4_PI$fit - qnorm(AR4_PI$se.fit)
> AR4_PI$LPI <- AR4_PI$fit + qnorm(AR4_PI$se.fit)
> head(AR4_PI)
`      fit    se.fit      UPI      LPI
1 8.268551 0.1282573 9.403219 7.133884
2 8.295110 0.1245991 9.447409 7.142811
3 8.321668 0.1209774 9.491783 7.151553
4 8.348227 0.1173954 9.536335 7.160118
5 8.374785 0.1138570 9.581054 7.168516
6 8.401343 0.1103663 9.625926 7.176761
>
> #To plot, add these new prediction intervals to the airtemp dataset,
> #While we are at it, add resdiuals from the models
> airtemp.dat$AR1_resid=Ex8AR1$residuals
> airtemp.dat$AR1_fit=AR1_PI$fit . . . TRUNCATED
>
> #Plot 18 top two graphs just the prediction intervals
> png('Ch6.Fig35a.png', width = 8, height = 8,
+     units = 'in', res = 600)
> par(mfrow=c(1,1) )
> plot(temp~year, data=airtemp.dat, col =('black'),pch=1,      xlab =
('Year'),  .  . TRUNCATED
>
> png('Ch6.Fig35b.png', width = 8, height = 8,
+     units = 'in', res = 600)
> par(mfrow=c(1,1) )
> plot(temp~year, data=airtemp.dat, col =('black'),pch=1,      xlab =
('Year'), .  . TRUNCATED
>
> #Plot the original data and then use moving averages over 4, 6, and
8 years to smooth the plots and look at large trends.  The
temperatures fluctuate but are increasing over time since 1990
> #The increasing temperatures began in the between 1975 and 1985
>
> require(TTR)
> png('Ch6.Fig36a.png', width = 8, height = 8,
+     units = 'in', res = 600)
> par(mfrow=c(1,1) ) .  . TRUNCATED
> temp_tsSMA4 <- SMA(temp_ts,n=4)
> png('Ch6.Fig36b.png', width = 8, height = 8,
+     units = 'in', res = 600) .  . TRUNCATED
> temp_tsSMA6 <- SMA(temp_ts,n=6)
> png('Ch6.Fig36c.png', width = 8, height = 8,
+     units = 'in', res = 600) .  . TRUNCATED
> temp_tsSMA8 <- SMA(temp_ts,n=8)
> png('Ch6.Fig36d.png', width = 8, height = 8,
+     units = 'in', res = 600) .  . TRUNCATED

> #Plotting the autocorrelation function (ACF) and
```

```
> #the partial autocoreelation function (PACF) for the studentized
residuals
> #from the original model indicates that it the autocorrelation
alternates sign
> #and the first 4 lags are the largest.
> png('Ch6.Fig37a.png', width = 8, height = 8,
+     units = 'in', res = 600) .  . TRUNCATED
> acf(airtemp.dat$rst, lag.max = 12, plot = TRUE)
> png('Ch6.Fig37b.png', width = 8, height = 8,
+     units = 'in', res = 600) .  . TRUNCATED
> #Plotting the sutocorrelation function (ACF) and
> #the partial autocoreelation function (PACF)
> #for the studentized residuals
> #from the original model indicates that the #autocorrelation
alternates in sign
> #and the first 4 lags are the largest.
> #A sine curve might model these results
```

==========

Example 8. Nonlinear and piecewise regression

Based on the previous analysis, it looks like modelling the relationship between time and temperature can best be done using nonlinear regression. Examination of the moving average plots above (6.36) indicates there may be a break in the regression line where the initial portion is horizontal followed by increasing temperatures. Thus, piecewise (also called segmented) regression may be appropriate. Both approaches are run in the script below. For the piecewise regression we ensured the lines were continuous at the breakpoint with the formulas Ryan and Porth (2007) summarized. The SAS code uses PROC NLIN to estimate the parameters for the piecewise regression, but we chose to use an iterative method with the linear regression function lm() refined from Lemoine (2012) to make the estimates because it is more transparent and less reliant on starting values.

Script 6.18 kagc and jb

```
#Example 8 (con't): Load and attach data in airtemp.csv.

#Nonlinear regression script kagc
#Parameters
x=airtemp.dat$year
y=airtemp.dat$temp
a_start <- 7           #param a is the y value when x=0
b_start <- 1
c_start <- 1960
d_start <- 11
e_start <- 0.03

#regression model
Ex8nl1<-nls(y~a-b*sin((x-c)*2*3.14/d)+e*x,
start=list(a=a_start, b=b_start, c=c_start, d=d_start, e=e_start))
#Calculate goodness of fit
cor(y,predict(Ex8nl1))
#Calculate confidence intervals for the parameters
```

```
library(nlstools)
confint2(Ex8nl1, level = 0.95, method = c( 'profile'))

#plot the data  This matches Chapter 6, Figure 18 bottom
png('Ch6.Fig35c.png', width = 8, height = 8,
units = 'in', res = 600)
#dev.off()
par(mfrow=c(1,1))
plot(temp~year, data=airtemp.dat, col =('black'),pch=1, xlab =
('Year'), ylab = ('Temp'), new = FALSE,
main='Predicted values from
nonlinear regression (Red)' )
lines(x,predict(Ex8nl1),col='red',lty=2,lwd=3)
#Prediction intervals are not included
dev.off()

#Piecewise/segmented regression script jb
library(dplyr)               #rename()
library(ggplot2)             #ggplot()

#Set up search for best breakpoint.
#Create dataset, breaks, of potential breakpoints; i.e. years 1961-
1995)
breaks <- seq(1961, 1995, 1)
mse <- numeric(length(breaks))
for (i in 1:length(breaks)) {
  airtemp.dat$ind2 <-
    ifelse(airtemp.dat$year>=breaks[i],1,0)
  #get mse for each breakpoint in search range, minus 1 to
  #df for error for estimating breakpoint from same data
  mseg <-lm(temp~ I(ind2*(year-breaks[i])),
    data=airtemp.dat)
  mse[i] <- sum((mseg$residuals)^2)/(mseg$df.residual - 1)
}

print(breaks[which(mse==min(mse))])

#plot MSE vs breakpoint year
png('Ch6.Fig38.png', width = 8, height = 8,
units = 'in', res = 600)
plot(breaks, mse, ylab="MSE", xlab="Breakpoint year")
lines(breaks, mse)
dev.off()
#Min. MSE was for breakpoint yr 1979 based on plot & min()

#Run model using 1979 as the breakpoint
airtemp.dat$ind2 <- ifelse(airtemp.dat$year>=1979,1,0)
piecewise <-lm(temp~ I(ind2*(year-1979)), data=airtemp.dat)
summary(piecewise)

#Plot these results against the original data
line2 <- function(m,b,x){m*(x-1979)+b}
png('Ch6.Fig39.png', width = 8, height = 8,
```

```
  units = 'in', res = 600)
plot (temp ~ year, data=airtemp.dat)
abline(h=piecewise$coefficients[1], col="blue", lwd=2)
curve(line2(m=piecewise$coefficients[2],
  b=piecewise$coefficients[1], x=x), from=1960, to=2013,
  col="red", lwd=2, add=T)

dev.off()

#plot Fig. 19 in Chapter 6
#add prediction interval limits to data frame
#Calculate prediction and confidence intervals
pP <- predict(piecewise, interval='prediction')
pP <- as.data.frame(pP)
pP <- rename(pP, plwr=lwr, pupr=upr)
pC <- predict(piecewise, interval='confidence')
temp.datu<- data.frame(airtemp.dat, pP, pC)

#Use ggplot2 to generate the plot.
#plot regression lines + confidence &
#prediction intervals
ggseg <-ggplot(temp.datu, aes(x=year,y=temp))+
geom_ribbon(aes(ymin=plwr, ymax=pupr, fill='predict.'),
alpha=0.3) + geom_ribbon(aes(ymin=lwr,
ymax=upr, fill='conf.'),
alpha=0.3) + geom_point() + geom_line(aes(x=year,y=fit)) +
theme_bw() +
labs(title='segmented or piecewise regression', x='Year',
y= 'Temperature, °C' ) +  theme(legend.position='bottom')
ggseg
ggsave('Ch6-Fig40.png', ggseg, dpi=600)
```

Output 6.18 kagc and jb

```
---------------------------------------------------------------------------------------------
> #Nonlinear regression script kagc
> #Parameters
> x=airtemp.dat$year
> y=airtemp.dat$temp
> a_start<-7 #param a is the y value when x=0
> b_start<-1
> c_start<-1960
> d_start<-11
> e_start<-0.03
>
> #model
> Ex8nl1<-nls(y~a-b*sin((x-c)*2*3.14/d)+e*x,
+     start=list(a=a_start,b=b_start, c=c_start,    d=d_start,
e=e_start))
> #Calculate goodness of fit
> cor(y,predict(Ex8nl1))
[1] 0.68436
> #Calculate confidence intervals for the parameters
>
```

```
> library(nlstools)
> confint2(Ex8nl1, level = 0.95, method = c( 'profile'))
Waiting for profiling to be done...
`          2.5%         97.5%
a  -63.07341072  -20.52715294
b    0.30770432    0.78314049
c 1968.04388987 1969.73533987
d    7.73335921    8.33206385
e    0.01486039    0.03627686
>
> #plot the data  This matches Chapter 6, Figure 18 bottom
> png('Ch6.Fig35c.png', width = 8, height = 8,
+     units = 'in', res = 600). . . TRUNCATED
>
> #Piecewise/segmented regression script jb
> library(dplyr)       #rename()
> library(ggplot2)     #ggplot()
>
> #Our visual inspection of the moving average plots in Fig. 6.36
> #above supports a flat regression line between year and
> #temperature from 1960 to about 1985, followed
> #by increasing temperatures so we will fit a
> #piecewise (aka segmented) regression
>
> #Set up search for best breakpoint.
> #Create dataset, breaks, of potential breakpoints; i.e. years 1961 -
19955)
>
> breaks <- seq(1961, 1995, 1)
> mse <- numeric(length(breaks))
> for (i in 1:length(breaks)) {
+   airtemp.dat$ind2 <- ifelse(airtemp.dat$year>=breaks[i],1,0)
+   #get mse for each breakpoint in search range, minus 1 to
+   #df for error for estimating breakpoint from same data
+   mseg <-lm(temp~ I(ind2*(year-breaks[i])), data=airtemp.dat)
+   mse[i] <- sum((mseg$residuals)^2)/(mseg$df.residual - 1)
+ }
> print(breaks[which(mse==min(mse))])
[1] 1979
>
```

The minimum MSE occurred when the breakpoint year was 1979 compared to the SAS PROC NLIN estimated breakpoint of 1972 as reported on Page 149 of Chapter 6.

```
> #plot MSE vs breakpoint year
> png('Ch6.Fig38.png', width = 8, height = 8,
+ units = 'in', res = 600)
> plot(breaks, mse, ylab="MSE", xlab="Breakpoint year")
> lines(breaks, mse)
> dev.off()
null device
          1
> #Min. MSE was for breakpoint yr 1979 based on plot & min()
```

See Fig. 6.38 in this appendix.

```
>
> #Run model using 1979 as the breakpoint
> airtemp.dat$ind2 <- ifelse(airtemp.dat$year>=1979,1,0)
> piecewise <-lm(temp~ I(ind2*(year-1979)), data=airtemp.dat)
> summary(piecewise)

Call:
lm(formula = temp ~ I(ind2 * (year - 1979)), data = airtemp.dat)

Residuals:
`    Min       1Q   Median       3Q      Max
-1.84209 -0.43349 -0.04787  0.57821  1.13713
```

The five-number summary of the scaled residuals above do not show any potential outliers, but there is some chance of skewness.

```
Coefficients:
Term                Estimate Std. Error t value Pr(>|t|)
(Intercept)         8.589304   0.131663  65.237  < 2e-16 ***
I(ind2*(year-1979)) 0.034870   0.008271   4.216 9.94e-05 ***
---------------------------------------------------------
Sig. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6983 on 52 degrees of freedom
Multiple R-squared:  0.2548,  Adjusted R-squared:  0.2404
F-statistic: 17.78 on 1 and 52 DF,  p-value: 9.943e-05
```

The above estimate for the intercept (first line segment) is temp = 8.59, which is close to the SAS estimate of 8.50 (Page 149 of Chapter 6). The esimate for the second line segment is temp = 0.0349(year – 1979) + 8.59. Note that the slope is about 20% steeper than the SAS value. The MSE, 0.4876 (0.6983^2), is slightly less than for the SAS model (0.5002).

```
>
> #Plot these results against the original data
> line2 <- function(m,b,x){m*(x-1979)+b}
> png('Ch6.Fig39.png', width = 8, height = 8,
+     units = 'in', res = 600)
> plot (temp ~ year, data=airtemp.dat)
> abline(h=piecewise$coefficients[1], col="blue", lwd=2)
> curve(line2(m=piecewise$coefficients[2],
+    b=piecewise$coefficients[1], x=x), from=1960, to=2013,
+    col="red", lwd=2, add=T)
> dev.off()
null device
          1
```

See Fig. 6.39 in this appendix.

```
> #plot Fig. 19 in Chapter 6
> #add prediction interval limits to data frame
> #Calculate prediction and confidence intervals
> pP <- predict(piecewise, interval='prediction')
Warning message:
In predict.lm(piecewise, interval = "prediction") :
  predictions on current data refer to _future_ responses
```

```
> pP <- as.data.frame(pP)
> pP <- rename(pP, plwr=lwr, pupr=upr)
> pC <- predict(piecewise, interval='confidence')
> temp.datu<- data.frame(airtemp.dat, pP, pC)
>
> #Use ggplot2 to generate the plot.
> #plot regression lines + confidence &
> #prediction intervals
> ggseg <-ggplot(temp.datu, aes(x=year,y=temp))+
+ geom_ribbon(aes(ymin=plwr, ymax=pupr, fill='predict.'),
+ alpha=0.3) + geom_ribbon(aes(ymin=lwr,
+ ymax=upr, fill='conf.'),
+ alpha=0.3) + geom_point() + geom_line(aes(x=year,y=fit)) +
+ theme_bw() +
+ labs(title='segmented or piecewise regression', x='Year',
+ y= 'Temperature, °C' ) +  theme(legend.position='bottom')
> ggseg
> ggsave('Ch6-Fig40.png', ggseg, dpi=600)
Saving 5.76 x 5.75 in image
>
```

See Fig. 6.40 in this appendix.

==========

Chapter 7. Analysis and Interpretation of Interactions of Fixed and Random Effects

by Mateo Vargas, Barry Glaz, Jose Crossa, and Alex Morgounov

Table 1.1 (Chapter 7). Summary of ANOVA of wheat grain yield (Mg ha^{-1}) for complete model.

This example is a full model analysis of a 2 × 4 factorial (0 and 30 lbs. kg N ha^{-1} by 0, 50, 150, and 250 kg P ha^{-1}) trial conducted on two soils (Black and Chestnut) in two years (2007 and 2008). The data came in a file named, "Experiment 1 Data Wheat.csv", which was shortened to "Ch7Wheat.csv" for this script. In its initial form the data contained two anomalies for R: 1) there was a blank line between the first row (variable names) and the actual data, and 2) there was a trailing blank for each of the soil names for 2008. The first anomaly prevented the read.csv() function from reading the file and the second caused R to act as if there were four soils (Black, Black_, Chestnut, and Chestnut_ where the underscore indicates the blank character). A spreadsheet was used to remove both these problems, but R's ifelse() function could have taken care of the blank character on input. The details of the SAS code and output the chapter authors used are in their Appendices 1 and 2.

Script 7.1 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Wheat.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(Year) ; Nf <- factor(N)
Pf <- factor(P) ; Blk <- factor(Rep)

#statistical analyses
#anova of four factor factorial with blocks as random effect
#nested within year and soil
aov1 <- lmer(Yield ~ Yrf*Soil*Nf*Pf + (1|Blk:Yrf:Soil), data=dat)
print(summary(aov1))
anova(aov1, type=3, TEST="F")
```

Output 7.1 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)

> #data management
> #read in the variable names and data with read.csv function
> dat <- read.csv("Ch7Wheat.csv")
> #attach as data frame
> attach(dat)
```

Checks on input data from head() and tail() omitted

```
> #check stats for each variable
> summary(dat)
      Year            Soil          N            P              Rep
 Min.   :2007   black   :32   Min.   : 0   Min.   :  0.0   Min.   :1.0
 1st Qu.:2007   chestnut:32   1st Qu.: 0   1st Qu.: 37.5   1st Qu.:1.0
 Median :2008                 Median :15   Median :100.0   Median :1.5
 Mean   :2008                 Mean   :15   Mean   :112.5   Mean   :1.5
 3rd Qu.:2008                 3rd Qu.:30   3rd Qu.:175.0   3rd Qu.:2.0
 Max.   :2008                 Max.   :30   Max.   :250.0   Max.   :2.0
     Yield
 Min.   :0.750
 1st Qu.:1.025
 Median :1.290
 Mean   :1.471
 3rd Qu.:1.698
 Max.   :2.980
```

The summary function gives a 6 statistic description of numeric variables and counts for factor levels. This is where the problem with the Soil input first showed up. Initially R read the trailing blanks for the soil levels in 2008 as a real character and listed four levels with 16 cases for each. A spreadsheet was used to remove the trailing blanks.

```
> print(summary(aov1))
Linear mixed model fit by REML t-tests use Satterthwaite
approximations to
  degrees of freedom [lmerMod]
Formula: Yield ~ Yrf * Soil * Nf * Pf + (1 | Blk:Yrf:Soil)
   Data: dat

REML criterion at convergence: -40.3

Scaled residuals:
`    Min      1Q  Median      3Q     Max
-2.0849 -0.3705  0.0000  0.3705  2.0849
```

The scaled residuals show excellent symmetry and no suggestion of outliers.

```
Random effects:
 Groups       Name        Variance Std.Dev.
 Blk:Yrf:Soil (Intercept) 0.000292 0.01709
 Residual                 0.008036 0.08964
Number of obs: 64, groups:  Blk:Yrf:Soil, 8
```

The above variance components match those in Table 1.1 of Chapter 7. Note, R gives the square root of each component, not the standard error of the variance component.

```
> anova(aov1, type=3, TEST="F")
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom

Term           Sum Sq Mean Sq NumDF DenDF F.value    Pr(>F)
Yrf            9.6821  9.6821     1     4 1204.82 4.111e-06 ***
Soil           4.4722  4.4722     1     4  556.51 1.914e-05 ***
Nf             0.0977  0.0977     1    28   12.15 0.0016349 **
Pf             0.9733  0.3244     3    28   40.37 2.667e-10 ***
Yrf:Soil       1.0069  1.0069     1     4  125.30 0.0003626 ***
Yrf:Nf         0.0812  0.0812     1    28   10.11 0.0035888 **
Soil:Nf        0.1702  0.1702     1    28   21.17 8.239e-05 ***
Yrf:Pf         0.1650  0.0550     3    28    6.84 0.0013313 **
Soil:Pf        0.3536  0.1179     3    28   14.67 6.256e-06 ***
Nf:Pf          0.0358  0.0119     3    28    1.49 0.2400017
Yrf:Soil:Nf    0.2862  0.2862     1    28   35.62 1.995e-06 ***
Yrf:Soil:Pf    0.2048  0.0683     3    28    8.50 0.0003596 ***
Yrf:Nf:Pf      0.0157  0.0052     3    28    0.65 0.5891575
Soil:Nf:Pf     0.1299  0.0433     3    28    5.39 0.0046922 **
Yrf:Soil:Nf:Pf 0.0515  0.0172     3    28    2.13 0.1182779
---------------------------------------------------------------
Sig. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The NumDF, DenDF, and p-values in the above ANOVA table match those in Table 1.1 and Page 5 of Appendix 2 of Chapter 7.

==========

Table 1.2 (Chapter 7). Summary of ANOVA of wheat grain yield (Mg ha^{-1}) for reduced model.

The authors reduced the complete model by omitting the three nonsignificant terms for this analysis. R did not allow removal of the N × P term, ostensibly because there was a higher order interaction that contained it in the model (Soil × N × P). The results in this table also decomposed the P factor into linear, quadratic, and cubic responses; which required use of SAS's ORPOL function because the levels of P were not equally spaced. Because of the unequal spacing R's automatic CONTRAST.POLY function could not be used, so a customized set of orthogonal polynomial contrasts was created using the residual method published by Landram and Alidaee (1997). The R documentation does not fully explain how to enter customized contrast coefficients into the model, but Maier (2015) provided an excellent clarification of the process. When the built-in effect contrasts (contr.sum) were applied to the Year, Soil, and N factors along with the orthogonal polynomials for P, most of the results of Table 1.2 were replicated; however, the denominator degrees of freedom, the inclusion of the N × P interaction, and the test of the Soil × N × P interaction did not match.

Script 7.2 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Wheat.csv")
```

```
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(Year) ; Nf <- factor(N); Soil <- factor(Soil)
Pf <- factor(P) ; Blk <- factor(Rep)

#statistical analyses
#determine orthogonal contrast coefficients for four levels of P
lvls <- c(0,50,150,250)
lcr <- resid(lm(lvls~1)) ; lc <- lcr/min(abs(lcr))
qcr <- resid(lm(lvls^2 ~ 1 + lc)) ; qc <- qcr/min(abs(qcr))
ccr <- resid(lm(lvls^3 ~ 1 + lc + qc)); cc <- ccr/min(abs(ccr))
mat1 <- rbind(lc, qc, cc)
print(mat1,digits=4)
#check that the coefficients sum to zero
print(round(rowSums(mat1), digits=6))
#check the contrasts are orthogonal; i.e., sum of products= zero
round(lcqc<-crossprod(lc,qc), digits=6)
round(lccc<-crossprod(lc,cc), digits=6)
round(qccc<-crossprod(qc,cc), digits=6)
#add constant row, get inverse, & drop first -> contrast matrix
mat1 <- rbind(constant=0.25, mat1)
print(mat1)
imat <- solve(mat1)
cmat <- imat[ , -1]
#anova of four factor factorial with blocks as random effect
#nested within Year and Soil - reduced model; effects contrasts
#for Year, Soil, and N; & orthogonal polynomial contrasts for P
aov2 <- lmer(Yield ~ Yrf*Soil*Nf*Pf –Nf:Pf -Yrf:Nf:Pf -
Yrf:Soil:Nf:Pf + (1|Blk:Yrf:Soil), data=dat,
contrasts=list(Yrf=contr.sum, Soil=contr.sum, Nf=contr.sum,
Pf=cmat))
print(summary(aov2))
anova(aov2, type=3,TEST="F")
```

Output 7.2 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)

> #data management
> #read in the variable names and data with read.csv function
> dat <- read.csv("Ch7Wheat.csv")
> #set up factor (class) variables from numeric variables
> Yrf <- factor(Year) ; Nf <- factor(N); Soil <- factor(Soil)
```

```
> Pf <- factor(P) ; Blk <- factor(Rep)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #determine orthogonal contrast coefficients for four levels of P
> lvls <- c(0,50,150,250)
> lcr <- resid(lm(lvls~1)) ; lc <- lcr/min(abs(lcr))
> qcr <- resid(lm(lvls^2 ~ 1 + lc)) ; qc <- qcr/min(abs(qcr))
> ccr <- resid(lm(lvls^3 ~ 1 + lc + qc)); cc <- ccr/min(abs(ccr))
> mat1 <- rbind(lc, qc, cc)
> print(mat1,digits=4)
c        1      2      3     4
lc -3.000 -1.667  1.000 3.667
qc  1.767 -1.000 -2.419 1.651
cc -2.667  5.000 -3.333 1.000
```

The above coefficients for the orthogonal polynomials match those given on Page 3 of Appendix 3 in Chapter 7 when the latter are divided by the minimum absolute value coefficient for each term as these have been.

```
> #check that the coefficients sum to zero
> print(round(rowSums(mat1), digits=6))
lc qc cc
 0  0  0

> #check the contrasts are orthogonal; i.e., sum of products= zero
> round(lcqc<-crossprod(lc,qc), digits=6)
     [,1]
[1,]    0

> round(lccc<-crossprod(lc,cc), digits=6)
     [,1]
[1,]    0

> round(qccc<-crossprod(qc,cc), digits=6)
     [,1]
[1,]    0

> #add constant row, get inverse, & drop first -> contrast matrix
> mat1 <- rbind(constant=0.25, mat1)
> print(mat1)
c                 1         2         3        4
constant  0.250000  0.250000  0.250000 0.250000
lc       -3.000000 -1.666667  1.000000 3.666667
qc        1.767442 -1.000000 -2.418605 1.651163
cc       -2.666667  5.000000 -3.333333 1.000000

> imat <- solve(mat1)
> cmat <- imat[ , -1]

> #anova of four factor factorial with blocks as random effect
> #nested within year and soil - reduced model; effects contrasts
> #for Year, Soil, an .... [TRUNCATED]
```

```
> print(summary(aov2))
Linear mixed model fit by REML t-tests use Satterthwaite
approximations to
  degrees of freedom [lmerMod]
Formula: Yield ~ Yrf * Soil * Nf * Pf - Yrf:Nf:Pf - Yrf:Soil:Nf:Pf +
(1 |
    Blk:Yrf:Soil)
   Data: dat

REML criterion at convergence: -46.6

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.30859 -0.52386 -0.00076  0.53493  1.68829
```

The above scaled residuals show a slight left skewness, but no likely outliers.

```
Random effects:
 Groups       Name        Variance  Std.Dev.
 Blk:Yrf:Soil (Intercept) 0.0002224 0.01491
 Residual                 0.0085930 0.09270
Number of obs: 64, groups:  Blk:Yrf:Soil, 8
```

These variance components match those in Table 1.2 of Chapter 7. Note, R gives the square root of each component, not the standard error of the variance component.

```
Fixed effects:
Term            Estimate Std. Error       df t value Pr(>|t|)
(Intercept)      1.47094    0.01273  4.00000 115.546 3.36e-08 ***
Yrf1             0.44188    0.01273  4.00000  34.710 4.11e-06 ***
Soil1            0.30031    0.01273  4.00000  23.590 1.91e-05 ***
Nf1             -0.03906    0.01159 34.00000  -3.371 0.001878 **
Pflc             1.18542    0.11867 34.00000   9.989 1.20e-11 ***
Pfqc            -0.28352    0.08259 34.00000  -3.433 0.001587 **
Pfcc             0.20083    0.15411 34.00000   1.303 0.201272
Yrf1:Soil1       0.14250    0.01273  4.00000  11.194 0.000363 ***
Yrf1:Nf1        -0.03562    0.01159 34.00000  -3.074 0.004140 **
Soil1:Nf1       -0.05156    0.01159 34.00000  -4.450 8.77e-05 ***
Yrf1:Pflc        0.44750    0.11867 34.00000   3.771 0.000621 ***
Yrf1:Pfqc       -0.18017    0.08259 34.00000  -2.182 0.036145 *
Yrf1:Pfcc       -0.07292    0.15411 34.00000  -0.473 0.639134
Soil1:Pflc       0.75958    0.11867 34.00000   6.401 2.61e-07 ***
Soil1:Pfqc      -0.03512    0.08259 34.00000  -0.425 0.673368
Soil1:Pfcc      -0.00375    0.15411 34.00000  -0.024 0.980729
Yrf1:Soil1:Nf1  -0.06688    0.01159 34.00000  -5.771 1.71e-06 ***
Yrf1:Soil1:Pflc  0.57333    0.11867 34.00000   4.831 2.84e-05 ***
Yrf1:Soil1:Pfqc  0.01230    0.08259 34.00000   0.149 0.882519
Yrf1:Soil1:Pfcc -0.10583    0.15411 34.00000  -0.687 0.496907
Soil1:Nf1:Pflc   0.19875    0.11867 34.00000   1.675 0.103152
Soil1:Nf1:Pfqc   0.02183    0.08259 34.00000   0.264 0.793111
Soil1:Nf1:Pfcc  -0.53917    0.15411 34.00000  -3.499 0.001326 **
---------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above tests of the linear (Pflc), quadratic (Pfqc), and cubic (Pfcc) contrasts agree with those in the right three columns of Table 1.2 of Chapter 7.

```
Correlation matrix not shown by default, as p = 26 > 12.
Use print(summary(aov2), correlation=TRUE)  or
         vcov(summary(aov2))     if you need it

> anova(aov2, type=3,TEST="F")
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom

Term          Sum Sq Mean Sq NumDF DenDF F.value    Pr(>F)
Yrf         10.3530 10.3530     1     4 1204.82 4.111e-06 ***
Soil         4.7821  4.7821     1     4  556.51 1.914e-05 ***
Nf           0.0977  0.0977     1    34   11.36 0.0018781 **
Pf           0.9733  0.3244     3    34   37.75 6.326e-11 ***
Yrf:Soil     1.0767  1.0767     1     4  125.30 0.0003626 ***
Yrf:Nf       0.0812  0.0812     1    34    9.45 0.0041404 **
Soil:Nf      0.1702  0.1702     1    34   19.80 8.768e-05 ***
Yrf:Pf       0.1650  0.0550     3    34    6.40 0.0014791 **
Soil:Pf      0.3536  0.1179     3    34   13.72 5.015e-06 ***
Yrf:Soil:Nf  0.2862  0.2862     1    34   33.31 1.706e-06 ***
Yrf:Soil:Pf  0.2048  0.0683     3    34    7.94 0.0003792 ***
Soil:Nf:Pf   0.1657  0.0276     6    34    3.21 0.0131132 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above ANOVA results match those in the SAS output on Page 11 of Appendix 2 in Chapter 7.

==========

Figures 1.1, 1.2, and 1.3 (Chapter 7). Graphical interpretation of three-factor interactions.

All the three-factor-interactions were significant in this example and the authors used three graphical displays to help interpret them. The R script below, which will need to be appended to Script 7.2, runs the lsmean function and other statistics to do the calculations and the ggplot function to actually construct the plots. The results are shown in Output 7.3 and Figures 7.1, 7.2, and 7.3.

Script 7.3 jb

```
#use lsmeans and ggplot to imitate Fig. 1.1 of Ch. 7
#open file for graphs
win.metafile("c7-fig1.%02d.wmf")
#get lsmeans with compact letter display
lsmdat <- cld(lsmeans(aov2, ~Yrf:Soil:Nf), reversed=T,
Letters=LETTERS, adjust="none")
print(lsmdat)
#omit the extra lines with all NA values
lsmdat.df <- data.frame(lsmdat)
lsmdat.df <- na.omit(lsmdat.df)
#calculate & print half the LSD (HLSD) to use for error bars
(resvar <- attr(VarCorr(aov2), "sc")^2)
(resdf <- 34)
(HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/8))/2)
lsmdat.df <- cbind(lsmdat.df, HLSD)
#plot the lsmeans with error bars and connecting lines
```

```
ggplot(lsmdat.df, aes(x=Nf, y=lsmean, color= Yrf:Soil,
group=Yrf:Soil)) + geom_point(size=2) +
     geom_errorbar(aes(ymin=lsmean-HLSD, ymax=lsmean+HLSD), width =
0.1) +
labs(y="Grain yield (Mg ha-1)", x= "N fertilizer rate (kg ha-1)")+
geom_smooth(method="lm", formula=y~x, se=F, size=1.5) +
theme_bw()
#
#use lsmeans and ggplot to imitate Fig. 1.2 of Chapter 7
lsmP<- cld(lsmeans(aov2, ~Yrf:Soil:Pf), reversed=T,
Letters=LETTERS, adjust="none")
print(lsmP)
lsmP <- data.frame(lsmP)
lsmP <- na.omit(lsmP)
(HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/4))/2)
lsmP <- cbind(lsmP, HLSD)
ggplot(lsmP, aes(x=Pf, y=lsmean, color= Yrf:Soil, group=Yrf:Soil))
+ geom_point(size=2) +
     geom_errorbar(aes(ymin=lsmean-HLSD, ymax=lsmean+HLSD), width =
0.1) +
labs(y="Grain yield (Mg ha-1)", x= "P fertilizer rate (kg ha-1)")+
geom_smooth(method="glm", formula=y~poly(x,2), se=F, size=1.5)+
theme_bw()
#
#use lsmeans and ggplot to imitate Fig. 1.3 of Chapter 7
lsmSNP<- cld(lsmeans(aov2, ~Soil:Nf:Pf), reversed=T,
Letters=LETTERS, adjust="none")
print(lsmSNP)
lsmSNP <- data.frame(lsmSNP)
lsmSNP <- na.omit(lsmSNP)
(HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/4))/2)
lsmSNP <- cbind(lsmSNP, HLSD)
ggplot(lsmSNP, aes(x=Pf, y=lsmean, color= Soil:Nf, group=Soil:Nf))
+ geom_point(size=2) +
     geom_errorbar(aes(ymin=lsmean-HLSD, ymax=lsmean+HLSD), width =
0.1) +
labs(y="Grain yield (Mg ha-1)", x= "P fertilizer rate (kg ha-1)")+
geom_smooth(method="glm", formula=y~poly(x,3), se=F, size=1.5)+
theme_bw()
dev.off()
```

Output 7.3 jb

```
-------------------------------------------------------------------------------------------
> #use lsmeans and ggplot to imitate Fig. 1.1 of Ch. 7
> #open file for graphs
> win.metafile("c7-fig1.%02d.wmf")

> #get lsmeans with compact letter display
> lsmdat <- cld(lsmeans(aov2, ~Yrf:Soil:Nf), reversed=T,
Letters=LETTERS, adjust="none")
```

```
> print(lsmdat)

 Yrf  Soil     Nf  lsmean         SE    df  lower.CL  upper.CL .group
 2007 black    30 2.54875 0.03442827 12.37 2.4739881 2.6235119  A
 2007 black    0  2.16250 0.03442827 12.37 2.0877381 2.2372619   B
 2007 chestnut 0  1.51375 0.03442827 12.37 1.4389881 1.5885119    C
 2007 chestnut 30 1.42625 0.03442827 12.37 1.3514881 1.5010119    C
 2008 black    0  1.19875 0.03442827 12.37 1.1239881 1.2735119     D
 2008 black    30 1.17500 0.03442827 12.37 1.1002381 1.2497619     D
 2008 chestnut 30 0.89000 0.03442827 12.37 0.8152381 0.9647619      E
 2008 chestnut 0  0.85250 0.03442827 12.37 0.7777381 0.9272619      E

Results are averaged over the levels of: Pf
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
significance level used: alpha = 0.05
```

The order of the above lsmeans and the compact letter display match those in Fig. 1.1 of Chapter 7.

```
> #omit the extra lines with all NA values
> lsmdat.df <- data.frame(lsmdat)
> lsmdat.df <- na.omit(lsmdat.df)

> #calculate & print half the LSD (HLSD) to use for error bars
> (resvar <- attr(VarCorr(aov2), "sc")^2)
[1] 0.008593015
> (resdf <- 34)
[1] 34
> (HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/8))/2)
[1] 0.04709651
> lsmdat.df <- cbind(lsmdat.df, HLSD)

> #plot the lsmeans with error bars and connecting lines
> ggplot(lsmdat.df, aes(x=Nf, y=lsmean, color= Yrf:Soil,
group=Yrf:Soil)) + geom_point(size=2 .... [TRUNCATED]
```

See Fig. 7.1 in this appendix.

```
> #use lsmeans and ggplot to imitate Fig. 1.2 of Chapter 7
> lsmP<- cld(lsmeans(aov2, ~Yrf:Soil:Pf), reversed=T, Letters=LETTERS,
adjust="none")

> print(lsmP)
Yrf  Soil     Pf  lsmean         SE    df  lower.CL  upper.CL .group
2007 black    250 2.7075 0.04753349 28.14 2.6101541 2.8048459  A
2007 black    150 2.5600 0.04753349 28.14 2.4626541 2.6573459   B
2007 black    50  2.2075 0.04753349 28.14 2.1101541 2.3048459    C
2007 black    0   1.9475 0.04753349 28.14 1.8501541 2.0448459     D
2007 chestnut 150 1.5475 0.04753349 28.14 1.4501541 1.6448459      E
2007 chestnut 50  1.5125 0.04753349 28.14 1.4151541 1.6098459      E
2007 chestnut 250 1.4600 0.04753349 28.14 1.3626541 1.5573459      EF
2007 chestnut 0   1.3600 0.04753349 28.14 1.2626541 1.4573459       FG
2008 black    250 1.3050 0.04753349 28.14 1.2076541 1.4023459        GH
2008 black    150 1.2225 0.04753349 28.14 1.1251541 1.3198459        GH
2008 black    50  1.1825 0.04753349 28.14 1.0851541 1.2798459         H
2008 black    0   1.0375 0.04753349 28.14 0.9401541 1.1348459          I
2008 chestnut 250 0.9450 0.04753349 28.14 0.8476541 1.0423459          IJ
2008 chestnut 150 0.8900 0.04753349 28.14 0.7926541 0.9873459           JK
2008 chestnut 50  0.8600 0.04753349 28.14 0.7626541 0.9573459           JK
```

```
2008 chestnut 0   0.7900 0.04753349 28.14 0.6926541 0.8873459           K

Results are averaged over the levels of: Nf
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
significance level used: alpha = 0.05
```

The above lsmeans match those on Page 12 of App. 2 in Chapter 7. The compact letter displays also match except that R assigns the Black Soil with P=150 kg ha^{-1} in 2008 to both groups G and H whereas SAS assigned it only to H.

```
> lsmP <- data.frame(lsmP)
> lsmP <- na.omit(lsmP)

> (HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/4))/2)
[1] 0.06660453
> lsmP <- cbind(lsmP, HLSD)

> ggplot(lsmP, aes(x=Pf, y=lsmean, color= Yrf:Soil, group=Yrf:Soil)) +
geom_point(size=2) +
+        geom_errorbar(aes(ymin=lsmean-HLSD, ymax=lsmean+HLSD) ....
[TRUNCATED]
```

See Fig. 7.2 in this appendix.

```
> #use lsmeans and ggplot to imitate Fig. 1.3 of Chapter 7
> lsmSNP<- cld(lsmeans(aov2, ~Soil:Nf:Pf), reversed=T,
Letters=LETTERS, adjust="none")

> print(lsmSNP)
 Soil     Nf Pf  lsmean          SE    df  lower.CL upper.CL .group
 black    30 250 2.1075 0.04694511 37.31 2.0124067 2.202593  A
 black    30 150 1.9475 0.04694511 37.31 1.8524067 2.042593   B
 black    0  250 1.9050 0.04694511 37.31 1.8099067 2.000093   B
 black    0  150 1.8350 0.04694511 37.31 1.7399067 1.930093   B
 black    30 50  1.8250 0.04694511 37.31 1.7299067 1.920093   B
 black    30 0   1.5675 0.04694511 37.31 1.4724067 1.662593    C
 black    0  50  1.5650 0.04694511 37.31 1.4699067 1.660093    C
 black    0  0   1.4175 0.04694511 37.31 1.3224067 1.512593     D
 chestnut 0  50  1.3100 0.04694511 37.31 1.2149067 1.405093     DE
 chestnut 30 150 1.2600 0.04694511 37.31 1.1649067 1.355093      EF
 chestnut 30 250 1.2375 0.04694511 37.31 1.1424067 1.332593      EF
 chestnut 0  150 1.1775 0.04694511 37.31 1.0824067 1.272593      EFG
 chestnut 0  250 1.1675 0.04694511 37.31 1.0724067 1.262593       FG
 chestnut 0  0   1.0775 0.04694511 37.31 0.9824067 1.172593        G
 chestnut 30 0   1.0725 0.04694511 37.31 0.9774067 1.167593        G
 chestnut 30 50  1.0625 0.04694511 37.31 0.9674067 1.157593        G

Results are averaged over the levels of: Yrf
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
significance level used: alpha = 0.05
```

The order of the above lsmeans and the compact letter display agrees with that in Fig. 1.3 of Chapter 7.

```
> lsmSNP <- data.frame(lsmSNP)
> lsmSNP <- na.omit(lsmSNP)

> (HLSD <- abs(qt(0.025, resdf)*sqrt(2*resvar/4))/2)
```

```
[1] 0.06660453
> lsmSNP <- cbind(lsmSNP, HLSD)

> ggplot(lsmSNP, aes(x=Pf, y=lsmean, color= Soil:Nf, group=Soil:Nf)) +
geom_point(size=2) +
+       geom_errorbar(aes(ymin=lsmean-HLSD, ymax=lsmean+HLSD) ....
[TRUNCATED]
```

See Fig. 7.3 in this appendix.

```
> dev.off()
null device
          1
```

==========

Table 2.1 (Ch. 7). Results from six mixed models using two years of a four-factor factorial.

In Example 1 of Section 2 in Chapter 7 on Mixed Models, the authors used two years of a 2 × 2 × 2 × 3 (tillage, summer crop, manure, N rate) factorial experiment to illustrate the problems of estimating variance components with a small number of levels (in this case year) and the effects on other tests. Because Models 3 and 4 are simple reductions of the random terms in Model 5, the scripts for them have not been included here. The data file name was shortened to Ch7Ex1.csv.

Script 7.4 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex1.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head() & tail() functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)

#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
m1 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf), data=dat)
print(summary(m1))
anova(m1, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.4 jb

```
> source("c7-ex1m1.r", echo=T)

> #activate additional packages needed
> library(lme4)
> library(lmerTest)
> library(lsmeans)

> #set up factor (class) variables from numeric variables
> Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
> Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
> Manf <- factor(MANURE) ; Blk <- factor(REP)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m1 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf ....
[TRUNCATED]

> print(summary(m1))
Linear mixed model fit by REML
t-tests use  Satterthwaite approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Tillf * SCf * Manf * Nf + (1 | Yrf) + (1 | Blk:Yrf)
   Data: dat

REML criterion at convergence: 2006.4

Scaled residuals:
`    Min       1Q   Median       3Q      Max
-2.91839 -0.57458  0.07622  0.57995  2.02026
```

The above scaled residuals show a slight left skewnes, but no likely outliers.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk:Yrf  (Intercept)  56421   237.5
 Yrf      (Intercept) 237357   487.2
 Residual             701195   837.4
Number of obs: 144, groups:  Blk:Yrf, 6; Yrf, 2
```

The above variance components equal those in the Model 1 column of Table 2.1 in Chapter 7.

```
> anova(m1, type=3,TEST="F", ddf="Kenward-Roger")
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom

Term                 Sum Sq  Mean Sq NumDF DenDF F.value    Pr(>F)
Tillf             13349280 13349280     1   115  19.038 2.811e-05 ***
SCf                5586920  5586920     1   115   7.968  0.005613 **
```

```
Manf               33798720 33798720     1   115  48.202 2.429e-10 ***
Nf                101421331 50710665     2   115  72.320 < 2.2e-16 ***
Tillf:SCf             4203     4203     1   115   0.006  0.938420
Tillf:Manf          655290   655290     1   115   0.935  0.335716
SCf:Manf            825372   825372     1   115   1.177  0.280218
Tillf:Nf            225765   112883     2   115   0.161  0.851496
SCf:Nf             4347039  2173519     2   115   3.100  0.048847 *
Manf:Nf           29082887 14541444     2   115  20.738 2.039e-08 ***
Tillf:SCf:Manf      155499   155499     1   115   0.222  0.638592
Tillf:SCf:Nf        426277   213139     2   115   0.304  0.738478
Tillf:Manf:Nf      1527450   763725     2   115   1.089  0.339939
SCf:Manf:Nf        1282088   641044     2   115   0.914  0.403724
Tillf:SCf:Manf:Nf  1423846   711923     2   115   1.015  0.365518
---------------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above p-values match those in the Model 1 column of Table 2.1 of Chapter 7.

==========

Table 2.1: Model 2 (Chapter 7).

The following R-script reproduces the results in the Model 2 column of Table 2.1 in Chapter 7.

Script 7.5 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex1.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)

#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
m2 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf)+
(1|Yrf:Tillf)+ (1|Yrf:SCf)+ (1|Yrf:Manf)+ (1|Yrf:Nf), data=dat)
print(summary(m2))
anova(m2, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.5 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
> #set up factor (class) variables from numeric variables
> Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
> Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
> Manf <- factor(MANURE) ; Blk <- factor(REP)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m2 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf ....
[TRUNCATED]

> print(summary(m2))
Linear mixed model fit by REML
t-tests use  Satterthwaite approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Tillf * SCf * Manf * Nf + (1 | Yrf) + (1 | Blk:Yrf) +
    (1 | Yrf:Tillf) + (1 | Yrf:SCf) + (1 | Yrf:Manf) + (1 | Yrf:Nf)
   Data: dat

REML criterion at convergence: 1955.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-3.11454 -0.50155  0.04711  0.50235  2.18476
```

The above scaled residuals are skewed left and show at least one potential outlier (-3.11).

```
Random effects:
 Groups    Name        Variance Std.Dev.
 Yrf:Nf    (Intercept) 389605   624.2
 Blk:Yrf   (Intercept)  68607   261.9
 Yrf:Manf  (Intercept) 155899   394.8
 Yrf:SCf   (Intercept)  38679   196.7
 Yrf:Tillf (Intercept) 200557   447.8
 Yrf       (Intercept)      0     0.0
 Residual               398432   631.2
Number of obs: 144, groups:  Yrf:Nf, 6; Blk:Yrf, 6; Yrf:Manf, 4;
Yrf:SCf, 4; Yrf:Tillf, 4; Yrf, 2
```

There are some small discrepancies between the above variance components and those in the Model 2 column of Table 2.1 of Chapter 7. However, all agree to at least 3 significant digits.

```
> anova(m2, type=3,TEST="F", ddf="Kenward-Roger")
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom
Term              Sum Sq  Mean Sq NumDF DenDF F.value    Pr(>F)
Tillf             698142   698142     1     1   1.752  0.411880
SCf              1242982  1242982     1     1   3.120  0.327968
Manf             2240386  2240386     1     1   5.623  0.254064
Nf               4145014  2072507     2     2   5.202  0.161247
Tillf:SCf           4203     4203     1   110   0.011  0.918379
```

```
Tillf:Manf          655290   655290     1   110   1.645  0.202382
SCf:Manf            825372   825372     1   110   2.072  0.152908
Tillf:Nf            225765   112883     2   110   0.283  0.753829
SCf:Nf             4347039  2173519     2   110   5.455  0.005509 **
Manf:Nf           29082887 14541444     2   110  36.497  6.961e-13 ***
Tillf:SCf:Manf      155499   155499     1   110   0.390  0.533448
Tillf:SCf:Nf        426277   213139     2   110   0.535  0.587218
Tillf:Manf:Nf      1527450   763725     2   110   1.917  0.151953
SCf:Manf:Nf        1282088   641044     2   110   1.609  0.204777
Tillf:SCf:Manf:Nf  1423846   711923     2   110   1.787  0.172320
---------------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above p-values match those in the Model 2 column of Table 2.1 of Chapter 7.

==========

Table 2.1: Model 5 (Chapter 7).

The following script reproduces the results in the Model 5 column of Table 2.1 in Chapter 7

Script 7.6 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex1.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)

#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
m5 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Blk:Yrf)+ (1|Yrf)+
(1|Yrf:Tillf)+ (1|Yrf:SCf)+ (1|Yrf:Manf)+ (1|Yrf:Nf)+
(1|Yrf:Tillf:SCf)+ (1|Yrf:Tillf:Manf)+ (1|Yrf:Tillf:Nf)+
(1|Yrf:SCf:Manf)+ (1|Yrf:SCf:Nf)+ (1|Yrf:Manf:Nf)+
(1|Yrf:Tillf:SCf:Manf)+ (1|Yrf:Tillf:SCf:Nf)+ (1|Yrf:Tillf:Manf:Nf)+
(1|Yrf:SCf:Manf:Nf)+ (1|Yrf:Tillf:SCf:Manf:Nf), data=dat)
print(summary(m5, ddf="Kenward-Roger"))
anova(m5, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.6 jb

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m5 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Blk:Yrf)+ (1|Yrf ....
[TRUNCATED]

> print(summary(m5, ddf="Kenward-Roger"))
Linear mixed model fit by REML
t-tests use  Kenward-Roger approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Tillf * SCf * Manf * Nf + (1 | Blk:Yrf) + (1 | Yrf) +
    (1 | Yrf:Tillf) + (1 | Yrf:SCf) + (1 | Yrf:Manf) + (1 | Yrf:Nf) +
    (1 | Yrf:Tillf:SCf) + (1 | Yrf:Tillf:Manf) + (1 | Yrf:Tillf:Nf) +
    (1 | Yrf:SCf:Manf) + (1 | Yrf:SCf:Nf) + (1 | Yrf:Manf:Nf) +
    (1 | Yrf:Tillf:SCf:Manf) + (1 | Yrf:Tillf:SCf:Nf) + (1 |
    Yrf:Tillf:Manf:Nf) + (1 | Yrf:SCf:Manf:Nf) + (1 |
Yrf:Tillf:SCf:Manf:Nf)
   Data: dat

REML criterion at convergence: 1942.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.87851 -0.51372 -0.08716  0.63739  2.48120
```

The above, scaled residuals show very good symmetry and no likely outliers.

```
Random effects:
 Groups                 Name        Variance  Std.Dev.
 Yrf:Tillf:SCf:Manf:Nf (Intercept) 4.704e-09 6.858e-05
 Yrf:SCf:Manf:Nf       (Intercept) 8.703e-03 9.329e-02
 Yrf:Tillf:Manf:Nf     (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:SCf:Nf      (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:SCf:Manf    (Intercept) 0.000e+00 0.000e+00
 Yrf:Manf:Nf           (Intercept) 2.741e+05 5.236e+02
 Yrf:SCf:Nf            (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:Nf          (Intercept) 0.000e+00 0.000e+00
 Yrf:SCf:Manf          (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:Manf        (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:SCf         (Intercept) 0.000e+00 0.000e+00
 Yrf:Nf                (Intercept) 2.599e+05 5.098e+02
 Blk:Yrf               (Intercept) 7.118e+04 2.668e+02
 Yrf:Manf              (Intercept) 6.863e+04 2.620e+02
 Yrf:SCf               (Intercept) 4.055e+04 2.014e+02
 Yrf:Tillf             (Intercept) 2.061e+05 4.540e+02
 Yrf                   (Intercept) 1.032e-04 1.016e-02
 Residual                          3.394e+05 5.826e+02
Number of obs: 144, groups:
Yrf:Tillf:SCf:Manf:Nf, 48; Yrf:SCf:Manf:Nf, 24; Yrf:Tillf:Manf:Nf, 24;
Yrf:Tillf:SCf:Nf, 24; Yrf:Tillf:SCf:Manf, 16; Yrf:Manf:Nf, 12;
Yrf:SCf:Nf, 12; Yrf:Tillf:Nf, 12; Yrf:SCf:Manf, 8; Yrf:Tillf:Manf, 8;
```

```
Yrf:Tillf:SCf, 8; Yrf:Nf, 6; Blk:Yrf, 6; Yrf:Manf, 4; Yrf:SCf, 4;
Yrf:Tillf, 4; Yrf, 2


convergence code: 0
unable to evaluate scaled gradient
Model failed to converge: degenerate  Hessian with 1 negative
eigenvalues
```

Except for minor discrepancies in Year × Summer Crop and Year × Summer Crop × Manure × N these variance components match those in the Model 5 column of Table 2.1 in Chapter 7.

```
> anova(m5, type=3,TEST="F", ddf="Kenward-Roger")
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom

Term                Sum Sq Mean Sq NumDF DenDF F.value Pr(>F)
Tillf               584048  584048     1     1  1.7207 0.4147
SCf                1053995 1053995     1     1  3.1052 0.3286
Manf               1880815 1880815     1     1  5.5411 0.2557
Nf                 3488741 1744371     2     2  5.1391 0.1629
Tillf:SCf             4203    4203     1     1  0.0124 0.9294
Tillf:Manf          655290  655290     1     1  1.9306 0.3971
SCf:Manf            824373  824373     1     1  2.4287 0.3632
Tillf:Nf            225765  112883     2     2  0.3326 0.7504
SCf:Nf             4341778 2170889     2     2  6.3957 0.1352
Manf:Nf            2720215 1360107     2     2  4.0070 0.1997
Tillf:SCf:Manf      155499  155499     1     1  0.4581 0.6212
Tillf:SCf:Nf        426277  213139     2     2  0.6279 0.6143
Tillf:Manf:Nf      1527450  763725     2     2  2.2500 0.3077
SCf:Manf:Nf        1280537  640268     2     2  1.8863 0.3465
Tillf:SCf:Manf:Nf 1423846  711923     2     2  2.0974 0.3229
```

Despite the warning messages, the Pr(>F)-values match those in the Model 5 column of Table 2.1 of Chapter 7.

```
Warning messages:
1: In checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv,  :
  unable to evaluate scaled gradient
2: In checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv,  :
  Model failed to converge: degenerate  Hessian with 1 negative
eigenvalues
```

==========

Table 2.1: Model 6 (Chapter 7).

The following script reproduces the results in the Model 6 column of Table 2.1 in Chapter 7

Script 7.7 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex1.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)

#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
m6 <- lmer(YIELD ~ Yrf*Tillf*SCf*Manf*Nf+ (1|Blk:Yrf), data=dat)
print(summary(m6))
anova(m6, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.7 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m6 <- lmer(YIELD ~ Yrf*Tillf*SCf*Manf*Nf+ (1|Blk:Yrf), da ....
[TRUNCATED]

> print(summary(m6))
Linear mixed model fit by REML
t-tests use  Satterthwaite approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Yrf * Tillf * SCf * Manf * Nf + (1 | Blk:Yrf)
   Data: dat

REML criterion at convergence: 1561.1

Scaled residuals:
`    Min       1Q   Median       3Q      Max
-2.57868 -0.43595  0.00534  0.47650  2.61644
```

The above, scaled residuals show excellent symmetry and no likely outliers.

```
Random effects:
 Groups   Name        Variance Std.Dev.
 Blk:Yrf  (Intercept)  70512   265.5
 Residual             363020   602.5
Number of obs: 144, groups:  Blk:Yrf, 6
```

These variance components equal those in the Model 6 column of Table 2.1 in Chapter 7.

```
> anova(m6, type=3,TEST="F", ddf="Kenward-Roger")
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom
Term                     Sum Sq  Mean Sq NumDF DenDF F.value    Pr(>F)
Yrf                     3381496  3381496     1     4   9.315 0.0379528 *
Tillf                  13349280 13349280     1    92  36.773 2.900e-08 ***
SCf                     5586920  5586920     1    92  15.390 0.0001684 ***
Manf                   33798720 33798720     1    92  93.104 1.249e-15 ***
Nf                    101421331 50710665     2    92 139.691 < 2.2e-16 ***
Yrf:Tillf              8191044  8191044     1    92  22.564 7.458e-06 ***
Yrf:SCf                1822500  1822500     1    92   5.020 0.0274575 *
Tillf:SCf                 4203     4203     1    92   0.012 0.9145433
Yrf:Manf               6367211  6367211     1    92  17.540 6.442e-05 ***
Tillf:Manf              655290   655290     1    92   1.805 0.1824010
SCf:Manf                825372   825372     1    92   2.274 0.1350182
Yrf:Nf                20435463 10217732     2    92  28.146 2.902e-10 ***
Tillf:Nf                225765   112883     2    92   0.311 0.7335149
SCf:Nf                 4347039  2173519     2    92   5.987 0.0035940 **
Manf:Nf               29082887 14541444     2    92  40.057 3.067e-13 ***
Yrf:Tillf:SCf            95172    95172     1    92   0.262 0.6098617
Yrf:Tillf:Manf           99751    99751     1    92   0.275 0.6014039
Yrf:SCf:Manf             23358    23358     1    92   0.064 0.8003240
Tillf:SCf:Manf          155499   155499     1    92   0.428 0.5144339
Yrf:Tillf:Nf            105956    52978     2    92   0.146 0.8644119
Yrf:SCf:Nf                5753     2876     2    92   0.008 0.9921086
Tillf:SCf:Nf            426277   213139     2    92   0.587 0.5579920
Yrf:Manf:Nf            7163249  3581624     2    92   9.866 0.0001312 ***
Tillf:Manf:Nf          1527450   763725     2    92   2.104 0.1278219
SCf:Manf:Nf            1282088   641044     2    92   1.766 0.1767867
Yrf:Tillf:SCf:Manf      661511   661511     1    92   1.822 0.1803586
Yrf:Tillf:SCf:Nf        209081   104541     2    92   0.288 0.7504541
Yrf:Tillf:Manf:Nf       200863   100431     2    92   0.277 0.7589448
Yrf:SCf:Manf:Nf        1669258   834629     2    92   2.299 0.1060852
Tillf:SCf:Manf:Nf      1423846   711923     2    92   1.961 0.1465398
Yrf:Tillf:SCf:Manf:Nf   189429    94714     2    92   0.261 0.7709210
---------------------------------------------------------------------------
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The above Pr(>F)-values match those in the Model 6 column of Table 2.1 of Chapter 7.

==========

Table 2.2 (Chapter 7). Results from two mixed models using all ten years of a four-factor factorial.

In Example 2 of Section 2 on Mixed Models the authors analyzed all ten years of the 2 × 2 × 2 × 3 (tillage, summer crop, manure, N rate) factorial experiment instead of just two years as they did in Example 1. The data file name was shortened to Ch7Ex2.csv for use in the R scripts below.

Script 7.8 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex2.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)


#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
m1 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Blk:Yrf)+ (1|Yrf)+
(1|Yrf:Tillf)+ (1|Yrf:SCf)+ (1|Yrf:Manf)+ (1|Yrf:Nf)+
(1|Yrf:Tillf:SCf)+ (1|Yrf:Tillf:Manf)+ (1|Yrf:Tillf:Nf)+
(1|Yrf:SCf:Manf)+ (1|Yrf:SCf:Nf)+ (1|Yrf:Manf:Nf)+
(1|Yrf:Tillf:SCf:Manf)+ (1|Yrf:Tillf:SCf:Nf)+ (1|Yrf:Tillf:Manf:Nf)+
(1|Yrf:SCf:Manf:Nf)+ (1|Yrf:Tillf:SCf:Manf:Nf), data=dat)
print(summary(m1, ddf="Kenward-Roger"))
anova(m1, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.8 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m1 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Blk:Yrf)+ (1|Yrf ....
[TRUNCATED]
```

```
> print(summary(m1, ddf="Kenward-Roger"))
Linear mixed model fit by REML
t-tests use  Kenward-Roger approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Tillf * SCf * Manf * Nf + (1 | Blk:Yrf) + (1 | Yrf) +
    (1 | Yrf:Tillf) + (1 | Yrf:SCf) + (1 | Yrf:Manf) + (1 | Yrf:Nf) +
    (1 | Yrf:Tillf:SCf) + (1 | Yrf:Tillf:Manf) + (1 | Yrf:Tillf:Nf) +
    (1 | Yrf:SCf:Manf) + (1 | Yrf:SCf:Nf) + (1 | Yrf:Manf:Nf) +
    (1 | Yrf:Tillf:SCf:Manf) + (1 | Yrf:Tillf:SCf:Nf) + (1 |
    Yrf:Tillf:Manf:Nf) + (1 | Yrf:SCf:Manf:Nf) + (1 |
Yrf:Tillf:SCf:Manf:Nf)
   Data: dat

REML criterion at convergence: 10945.6

Scaled residuals:
`    Min      1Q  Median      3Q     Max
-3.3968 -0.5770  0.0254  0.5836  3.3373
```

The above, scaled residuals show excellent symmetry, but there are several potential outliers.

```
Random effects:
 Groups                 Name        Variance  Std.Dev.
 Yrf:Tillf:SCf:Manf:Nf (Intercept) 5.057e+03 7.112e+01
 Yrf:SCf:Manf:Nf       (Intercept) 7.251e+03 8.515e+01
 Yrf:Tillf:Manf:Nf     (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:SCf:Nf      (Intercept) 7.394e+03 8.599e+01
 Yrf:Tillf:SCf:Manf    (Intercept) 0.000e+00 0.000e+00
 Yrf:Manf:Nf           (Intercept) 1.202e+05 3.468e+02
 Yrf:SCf:Nf            (Intercept) 5.449e+04 2.334e+02
 Yrf:Tillf:Nf          (Intercept) 0.000e+00 0.000e+00
 Yrf:SCf:Manf          (Intercept) 0.000e+00 0.000e+00
 Yrf:Tillf:Manf        (Intercept) 1.273e-07 3.568e-04
 Yrf:Tillf:SCf         (Intercept) 7.371e+02 2.715e+01
 Yrf:Nf                (Intercept) 1.920e+05 4.382e+02
 Blk:Yrf               (Intercept) 1.312e+04 1.146e+02
 Yrf:Manf              (Intercept) 6.751e+04 2.598e+02
 Yrf:SCf               (Intercept) 8.203e-10 2.864e-05
 Yrf:Tillf             (Intercept) 5.632e+04 2.373e+02
 Yrf                   (Intercept) 4.116e+05 6.416e+02
 Residual                          2.410e+05 4.909e+02
Number of obs: 720, groups:
Yrf:Tillf:SCf:Manf:Nf, 240; Yrf:SCf:Manf:Nf, 120; Yrf:Tillf:Manf:Nf,
120; Yrf:Tillf:SCf:Nf, 120; Yrf:Tillf:SCf:Manf, 80; Yrf:Manf:Nf, 60;
Yrf:SCf:Nf, 60; Yrf:Tillf:Nf, 60; Yrf:SCf:Manf, 40; Yrf:Tillf:Manf,
40; Yrf:Tillf:SCf, 40; Yrf:Nf, 30; Blk:Yrf, 30; Yrf:Manf, 20; Yrf:SCf,
20; Yrf:Tillf, 20; Yrf, 10
```

These variance components equal those in the Model 1 column of Table 2.2 in Chapter 7.

```
> anova(m1, type=3,TEST="F", ddf="Kenward-Roger")
```

```
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom

Term                 Sum Sq Mean Sq NumDF DenDF F.value    Pr(>F)
Tillf              1663161 1663161     1     9   6.900 0.0275001 *
SCf                4473918 4473918     1     9  18.562 0.0019664 **
Manf               5696807 5696807     1     9  23.636 0.0008938 ***
Nf                18325315 9162658     2    18  38.015 3.452e-07 ***
Tillf:SCf           158921  158921     1     9   0.659 0.4377367
Tillf:Manf         1126334 1126334     1     9   4.673 0.0589008 .
SCf:Manf            359168  359168     1     9   1.490 0.2532064
Tillf:Nf            776364  388182     2    18   1.611 0.2272703
SCf:Nf             4295313 2147657     2    18   8.911 0.0020427 **
Manf:Nf           10788132 5394066     2    18  22.380 1.313e-05 ***
Tillf:SCf:Manf      147313  147313     1     9   0.611 0.4544134
Tillf:SCf:Nf       1712180  856090     2    18   3.552 0.0500963 .
Tillf:Manf:Nf      1729268  864634     2    18   3.587 0.0488408 *
SCf:Manf:Nf         149797   74898     2    18   0.311 0.7367509
Tillf:SCf:Manf:Nf  1423301  711650     2    18   2.953 0.0778074 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above Pr(>F)-values match those in the Model 1 column of Table 2.2 of Chapter 7.

==========

Table 2.2 Model 2 (Chapter 7).

This script reproduces the results in the Model 2 column of Table 2.2 in Chapter 7

Script 7.9 jb

```
#activate additional packages needed
library(lme4)
library(lmerTest)

#data management
#read in the variable names and data with read.csv function
dat <- read.csv("Ch7Ex2.csv")
#attach as data frame
attach(dat)
#check first & last 6 rows of data with head & tail functions
head(dat)
tail(dat)
#check stats for each variable
summary(dat)
#set up factor (class) variables from numeric variables
Yrf <- factor(YEAR) ; Nf <- factor(NLEVEL)
Tillf <- factor(TILLAGE); SCf <- factor(SUMCRP)
Manf <- factor(MANURE) ; Blk <- factor(REP)

#statistical analyses
#anova of four factor factorial, YEAR & blK(year) random effectS
```

```
m2 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf)+
(1|Yrf:Tillf)+ (1|Yrf:SCf)+ (1|Yrf:Manf)+ (1|Yrf:Nf)+
(1|Yrf:Tillf:SCf)+ (1|Yrf:Tillf:Nf)+ (1|Yrf:SCf:Manf)+ (1|Yrf:SCf:Nf)+
(1|Yrf:Manf:Nf), data=dat)
print(summary(m2, ddf="Kenward-Roger"))
anova(m2, type=3,TEST="F", ddf="Kenward-Roger")
```

Output 7.9 jb

```
> #activate additional packages needed
> library(lme4)
> library(lmerTest)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #anova of four factor factorial, YEAR & blK(year) random effectS
> m2 <- lmer(YIELD ~ Tillf*SCf*Manf*Nf+ (1|Yrf)+ (1|Blk:Yrf ....
[TRUNCATED]

> print(summary(m2, ddf="Kenward-Roger"))
Linear mixed model fit by REML
t-tests use  Kenward-Roger approximations to degrees of freedom
['lmerMod']
Formula: YIELD ~ Tillf * SCf * Manf * Nf + (1 | Yrf) + (1 | Blk:Yrf) +
    (1 | Yrf:Tillf) + (1 | Yrf:SCf) + (1 | Yrf:Manf) + (1 | Yrf:Nf) +
    (1 | Yrf:Tillf:SCf) + (1 | Yrf:Tillf:Nf) + (1 | Yrf:SCf:Manf) +
(1 | Yrf:SCf:Nf) + (1 | Yrf:Manf:Nf)
   Data: dat

REML criterion at convergence: 10947

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.3976 -0.5868  0.0235  0.5661  3.2575
```

The above, scaled residuals show excellent symmetry, but there are several potential outliers.

```
Random effects:
 Groups        Name        Variance Std.Dev.
 Yrf:Manf:Nf   (Intercept) 124539   352.90
 Yrf:SCf:Nf    (Intercept)  61329   247.65
 Yrf:Tillf:Nf  (Intercept)   2228    47.20
 Yrf:SCf:Manf  (Intercept)      0     0.00
 Yrf:Tillf:SCf (Intercept)   3496    59.13
 Yrf:Nf        (Intercept) 189353   435.15
 Blk:Yrf       (Intercept)  12829   113.27
 Yrf:Manf      (Intercept)  67505   259.82
 Yrf:SCf       (Intercept)      0     0.00
 Yrf:Tillf     (Intercept)  55658   235.92
 Yrf           (Intercept) 411380   641.39
 Residual                  248082   498.08
```

```
Number of obs: 720, groups:
Yrf:Manf:Nf, 60; Yrf:SCf:Nf, 60; Yrf:Tillf:Nf, 60; Yrf:SCf:Manf, 40;
Yrf:Tillf:SCf, 40; Yrf:Nf, 30; Blk:Yrf, 30; Yrf:Manf, 20; Yrf:SCf, 20;
Yrf:Tillf, 20; Yrf, 10
```

The above variance components come close to matching those in the Model 2 column of Table 2.2 in Chapter 7; they are all within 0.5% of the SAS values.

```
> anova(m2, type=3,TEST="F", ddf="Kenward-Roger")
Analysis of Variance Table of type III  with  Kenward-Roger
approximation for degrees of freedom

Term                Sum Sq Mean Sq NumDF DenDF F.value    Pr(>F)
Tillf              1711857 1711857     1     9   6.900 0.0275001 *
SCf                4447693 4447693     1     9  17.928 0.0021930 **
Manf               5863595 5863595     1     9  23.636 0.0008938 ***
Nf                18861834 9430917     2    18  38.015 3.452e-07 ***
Tillf:SCf           165057  165057     1     9   0.665 0.4357383
Tillf:Manf         1197236 1197236     1   550   4.826 0.0284505 *
SCf:Manf            446606  446606     1     9   1.800 0.2125522
Tillf:Nf            873964  436982     2    18   1.761 0.2001466
SCf:Nf             4483605 2241803     2    18   9.037 0.0019178 **
Manf:Nf           11104004 5552002     2    18  22.380 1.313e-05 ***
Tillf:SCf:Manf      156586  156586     1   550   0.631 0.4272630
Tillf:SCf:Nf       2135128 1067564     2   550   4.303 0.0139825 *
Tillf:Manf:Nf      1838124  919062     2   550   3.705 0.0252245 *
SCf:Manf:Nf         186265   93132     2   550   0.375 0.6871836
Tillf:SCf:Manf:Nf  1512896  756448     2   550   3.049 0.0481995 *
---------------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Pr(>F)-values match those in the Model 2 column of Table 2.2 of Chapter 7.

==========

Chapter 9. Analysis of Covariance

Kevin S. McCarter

Tables 1 - 3 and Fig. 1 - 4 of Chapter 9. Read in data and do descriptive statistics.

The data in this example consist of a two-treatment Completely Randomized Design with 10 replications of each treatment. Response measurements were taken on each of the 20 experimental units, Y, along with an independent variable, X. This R script emulates the SAS code in Fig. 1 and 2 of Chapter 9, in which the data are read and descriptive statistics performed. Table 1 and Fig. 1 display the data such that the treatment ID, Y, and X are repeated 20 times. This format is ideally suited for input with the scan() function in R. In addition to the tabular output shown below, Fig. 9.1 and 9.2 show the box plots analogous to Fig. 3 and 4 in Chapter 9. The SAS code makes a scatter plot as well, but it is not shown in the text, ostensibly because a very similar graph is presented as part of the ANCOVA output. The same procedure is followed in this appendix.

Script 9.1 jb

```
-------------------------------------------------------------------------------------------------
#activate additional packages needed
library(lsmeans)
library(plyr)
library(ggplot2)

#data management
#read in the data with scan function
dat0 <- scan("Ch9-Ex1dat.txt", what=list(trt="", y=0, x=0))
#combine vectors into data frame
dat1 <- as.data.frame(dat0)
#check first & last 6 rows of data with head & tail functions
head(dat1); tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#summarize data with mean and sd of x and y by trt
dat2 <- ddply(dat1, "trt", summarise, meany = round(mean(y), 1),
sdy = round(sd(y), 1), meanx = round(mean(x),1), sdx = round(sd(x),
1))
print(dat2)
#make boxplots of y and x versus trt like figs 3 & 4 then save
bp1 <- qplot(trt, y, data=dat1, geom="boxplot")+ theme_bw()
ggsave("Ch9-f3.png", width=13.5, height=9, units="cm", dpi=600)
bp2 <- qplot(trt, x, data=dat1, geom="boxplot")+ coord_flip() +
theme_bw()
ggsave("Ch9-f4.png", width=13.5, height=9, units="cm", dpi=600)
#scatter plot of y vs x grouped by treatment
ggplot(dat1, aes(x=x, y=y, shape=trt))+ geom_point(size=3)+
theme_bw()
ggsave("Ch9-fx.png", width=13.5, height=9, units="cm", dpi=600)
```

Output 9.1 jb

```
> #activate additional packages needed
> library(lsmeans)
> library(plyr)
> library(ggplot2)

> #data management
> #read in the data with scan function
> dat0 <- scan("Ch9-Ex1dat.txt", what=list(trt="", y=0, x=0))
Read 20 records

> #combine vectors into data frame
> dat1 <- as.data.frame(dat0)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #summarize data with mean and sd of x and y by trt
> dat2 <- ddply(dat1, "trt", summarise, meany = round(mean(y), 1), sdy
=  .... [TRUNCATED]

> print(dat2)
  trt meany  sdy meanx sdx
1   A  61.3 12.4   9.7 2.8
2   B  74.2 10.2  11.2 3.2
```

These means and standard deviations match those in Table 2 of Chapter 9, except for small difference in 10.2 versus 10.1 for sdy for B, likely because R and SAS use different rules for rounding (also see Output 2.2).

```
> #make boxplots of y and x versus trt like figs 3 & 4 then save
> bp1 <- qplot(trt, y, data=dat1, geom="boxplot")+ theme_bw()
> ggsave("Ch9-f3.png", width=13.5, height=9, units="cm", dpi=600)
```

See Fig. 9.1 of this appendix.

```
> bp2 <- qplot(trt, x, data=dat1, geom="boxplot")+ coord_flip() +
theme_bw()
> ggsave("Ch9-f4.png", width=13.5, height=9, units="cm", dpi=600)
```

See Fig. 9.2 of this appendix.

```
> #scatter plot of y vs x grouped by treatment
> ggplot(dat1, aes(x=x, y=y, shape=trt))+ geom_point(size=3)+
theme_bw()
> ggsave("Ch9-fx.png", width=13.5, height=9, units="cm", dpi=600)
>
```

==========

Tables 3, 4 & 5 and Fig. 5 & 6 of Chapter 9. ANOVA and ANCOVA with Example 1 data.

This script mimics the SAS code in Fig. 5 of Chapter 9, which runs the ANOVA of the effect of treatment group on y for the Example 1 data, then demonstrates the advantage of using ANCOVA to account for the variation associated with x, and finally shows there was no significant treatment difference in the x values. It also reproduces Fig. 6 of Chapter 9 illustrating the linear relationship between y and x for the two treatments (Fig. 9.3).

Script 9.2 jb

```
#activate additional packages needed
library(lsmeans)
library(plyr)
library(ggplot2)
library(gmodels)
library(car)

#data management
options(digits=5, scipen=10)
#read in the data with scan function
dat0 <- scan("Ch9-Ex1dat.txt", what=list(trt="", y=0, x=0))
#combine vectors into data frame
dat1 <- as.data.frame(dat0)
#check first & last 6 rows of data with head & tail functions
head(dat1); tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#use "effects" coding for ANOVA
contrasts(dat1$trt) <- contr.sum
#ANOVA for effect of trt on y with pairwise comparison
m1 <- lm(y ~ trt , data=dat1)
anova(m1)
fit.contrast(m1, trt, c(-1,1), conf.int=0.95)
#ANCOVA for effect of trt on y adjusting for x & pairwise diff.
m2 <- lm(y ~ x + trt , data=dat1)
anova(m2)
#get Type 3 analysis from Anova in car package; note cap A
Anova(m2, type=3)
fit.contrast(m2, trt, c(-1, 1), conf.int=0.95)
#ANOVA for effect of trt on x with pairwise comparison
m3 <- lm(x ~ trt , data=dat1)
anova(m3)
#add predicted values from ANCOVA output (m2) to data frame
pval <- data.frame(dat1, fitted(m2))
#scatter plot of y vs x grouped by treatment
sp <-ggplot(pval, aes(x=x, y=y, shape=trt,  group=trt))+
geom_point(size=3)+ theme_bw()
#add predicted line to scatterplot
```

```
finalplot <- sp + geom_line(aes(x=x, y=fitted.m2., linetype=trt))
finalplot
ggsave("c9-fig6p%02d.png", dpi=600)
```

Output 9.2 jb

```
> #activate additional packages needed
> library(lsmeans)
Loading required package: estimability
> library(plyr)
> library(ggplot2)
> library(gmodels)
> library(car)

> #data management
> options(digits=5, scipen=10)
> #read in the data with scan function
> dat0 <- scan("Ch9-Ex1dat.txt", what=list(trt="", y=0, x=0))
Read 20 records
> #combine vectors into data frame
> dat1 <- as.data.frame(dat0)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #use "effects" coding for ANOVA
> contrasts(dat1$trt) <- contr.sum

> #ANOVA for effect of trt on y with pairwise comparison
> m1 <- lm(y ~ trt , data=dat1)

> anova(m1)
Analysis of Variance Table

Response: y
Term      Df Sum Sq Mean Sq F value Pr(>F)
trt        1    832     832    6.48   0.02 *
Residuals 18   2310     128
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results in the above ANOVA match those in Table 3 of Chapter 9.

```
> fit.contrast(m1, trt, c(-1,1), conf.int=0.95)
Term          Estimate Std. Error t value Pr(>|t|) lower CI upper CI
trt c=( -1 1 )     12.9     5.0668   2.546 0.020268   2.2551   23.545
```

The above point estimate and the 95% CI for the treatment difference are the same as in Table 3 of Ch. 9.

```
> #ANCOVA for effect of trt on y adjusting for x & pairwise diff.
> m2 <- lm(y ~ x + trt , data=dat1)
```

```
> anova(m2)
Analysis of Variance Table

Response: y
Term       Df Sum Sq Mean Sq F value       Pr(>F)
x           1   2353    2353   80.43 0.000000075 ***
trt         1    292     292    9.98       0.0057 **
Residuals 17    497      29
---------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA results above match those in Table 4 of Chapter 9 for trt, but those for x do not because R defaults to Type 1 values and SAS defaults to Type 3 values. See next ANOVA

```
> #get Type 3 analysis from Anova in car package; note cap A
> Anova(m2, type=3)
Anova Table (Type III tests)

Response: y
Term         Sum Sq Df F value      Pr(>F)
(Intercept)   1507  1   51.52 0.00000155 ***
x             1813  1   61.97 0.00000045 ***
trt            292  1    9.98      0.0057 **
Residuals      497 17
-----------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Anova() function from the car package can do Type 3 analyses so the above ANOVA results match those in Table 4 of Chapter 9 for both x and trt.

```
> fit.contrast(m2, trt, c(-1, 1), conf.int=0.95)
Term          Estimate Std. Error t value  Pr(>|t|) lower CI upper CI
trt c=( -1 1 )      7.9      2.501  3.1587 0.0057344   2.6234   13.177
```

The above point estimate and the 95% CI for the treatment difference are the same as in Table 4 of Ch. 9.

```
> #ANOVA for effect of trt on x with pairwise comparison
> m3 <- lm(x ~ trt , data=dat1)

> anova(m3)
Analysis of Variance Table

Response: x
Term       Df Sum Sq Mean Sq F value Pr(>F)
trt         1   11.3   11.25    1.24   0.28
Residuals 18  163.2    9.07
```

The results in the above ANOVA match those in Table 5 of Chapter 9.

```
> #add predicted values from ANCOVA output (m2) to data frame
> pval <- data.frame(dat1, fitted(m2))
> #scatter plot of y vs x grouped by treatment
```

```
> sp <-ggplot(pval, aes(x=x, y=y, shape=trt,  group=trt))+
geom_point(size=3)+ theme_bw()
> #add predicted line to scatterplot
> (finalplot <- sp + geom_line(aes(x=x, y=fitted.m2., linetype=trt)))

> ggsave("c9-fig6p%02d.png", dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 9.3 of this appendix.

==========

Examples 2 and 3 of Chapter 9. ANOVAs and ANCOVAs .

These examples use the same SAS code as for Example 1; the only change is the input of different data. Consequently, the two R scripts for Example 1 apply as well when the corresponding data are entered.

==========

Example 4 of Chapter 9. ANOVAs and ANCOVAs .

The initial portion of this example uses the same SAS code as for Example 1; the only change being the new data. Consequently, the two R scripts for Example 1 apply when the corresponding data are entered. The last part of this example examines the possibility that the slope of the relationship between y and x differs between treatments. If the lines are nonparallel, then the differences between treatments at a range of x values are determined. The SAS code in Fig. 17 and 19 of Chapter 9 includes an interaction term to account for any disparity in slopes and the SAS code in Fig. 19 shows how to test the difference between treatment means at a specific value of x. The following R script illustrates how to include the interaction and the full range of x values presented in Table 23 of Chapter 9. Fig. 9.4, which mimics Fig. 18 of Chapter 9, illustrates the drastically different x-y relationships between treatments.

Script 9.3 jb

```
#activate additional packages needed
library(plyr)
library(ggplot2)
library(car)
library(lsmeans)

#data management
options(scipen=10)
#read in the data with scan function
dat0 <- scan("Ch9-Ex4dat.txt", what=list(trt="", y=0, x=0))
#combine vectors into data frame
dat1 <- as.data.frame(dat0)
#check first & last 6 rows of data with head & tail functions
head(dat1); tail(dat1)
#check stats for each variable
summary(dat1)
```

```
#statistical analyses
#use "effects" coding for ANOVA
contrasts(dat1$trt) <- contr.sum
#ANCOVA for effect of trt on y adjusting for x
#allow for separate slopes for each treatment
m2 <- lm(y ~ x + trt + trt:x, data=dat1)
summary(m2)
anova(m2)
#get Type 3 analysis from Anova in car package; note cap A
Anova(m2, type=3)
#least sq mean difference at mean of x
lsm1 <- lsmeans(m2, revpairwise ~ trt)
summary(lsm1, infer = c(T,T))
#least sq mean difference at several values of x
lsm2 <- lsmeans(m2, revpairwise ~ trt|x, at=list(x=c(8.0, 8.5, 9.0,
9.5, 10.0, 10.12, 10.5, 11.0, 11.5)))
summary(lsm2, infer = c(T,T))
#add predicted values from ANCOVA output (m2) to data frame
pval <- data.frame(dat1, fitted(m2))
#scatter plot of y vs x grouped by treatment
sp <-ggplot(pval, aes(x=x, y=y, shape=trt,  group=trt))+
geom_point(size=3)+ theme_bw()
#add predicted line to scatterplot
finalplot <- sp + geom_line(aes(x=x, y=fitted.m2., linetype=trt))
finalplot
ggsave("c9-fig18p.png", dpi=600)
```

Output 9.3 jb

```
> #activate additional packages needed
> library(plyr)
> library(ggplot2)
> library(car)
> library(lsmeans)
> #data management
> options(scipen=10)
> #read in the data with scan function
> dat0 <- scan("Ch9-Ex4dat.txt", what=list(trt="", y=0, x=0))
Read 20 records
> #combine vectors into data frame
> dat1 <- as.data.frame(dat0)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #use "effects" coding for ANOVA
> contrasts(dat1$trt) <- contr.sum

> #ANCOVA for effect of trt on y adjusting for x
> #allow for separate slopes for each treatment
```

```
> m2 <- lm(y ~ x + trt + trt:x, data=dat1)

> summary(m2)

Call:
lm(formula = y ~ x + trt + trt:x, data = dat1)

Residuals:
'    Min      1Q  Median      3Q     Max
-5.9104 -2.8443  0.0854  3.2937  5.2009
```

The residuals show very good symmetry. They are not scaled or standardized so there is no information about possible outliers.

```
Coefficients:
Term        Estimate Std. Error t value        Pr(>|t|)
(Intercept)  66.3896     3.9988  16.602 0.0000000000165 ***
x            -0.2501     0.3777  -0.662           0.517
trt1        -35.2617     3.9988  -8.818 0.0000001534029 ***
x:trt1        3.5192     0.3777   9.318 0.0000000727226 ***
---------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.75 on 16 degrees of freedom
Multiple R-squared:  0.8715,    Adjusted R-squared:  0.8474
F-statistic: 36.17 on 3 and 16 DF,  p-value: 0.0000002335
```

The Type 1 ANOVA has been omitted

```
> #get Type 3 analysis from Anova in car package; note cap A
> Anova(m2, type=3)
Anova Table (Type III tests)

Response: y
Term         Sum Sq Df  F value           Pr(>F)
(Intercept) 3875.3  1 275.6330 0.00000000001653 ***
x              6.2  1   0.4385           0.5173
trt         1093.2  1  77.7565 0.00000015340289 ***
x:trt       1220.7  1  86.8243 0.00000007272263 ***
Residuals    225.0 16
---------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Type 3 ANOVA from the car package is above. These results match the results in Table 22 of Chapter 9.

```
> #least sq mean difference at mean of x
> lsm1 <- lsmeans(m2, revpairwise ~ trt)
NOTE: Results may be misleading due to involvement in interactions

> summary(lsm1, infer = c(T,T))
$lsmeans
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
```

```
 A    64.21109 1.197838 16 61.67179 66.75039  53.606  <.0001
 B    63.50618 1.221746 16 60.91619 66.09616  51.980  <.0001

Confidence level used: 0.95
```

The means above plus the point estimate and CI below for the difference between treatments match those from SAS (bottom of Table 22 in Chapter 9).

```
$contrasts
 contrast   estimate       SE df  lower.CL upper.CL t.ratio p.value
 B - A    -0.7049114 1.710988 16 -4.332043  2.92222  -0.412  0.6858

Confidence level used: 0.95

> #least sq mean difference at several values of x
> lsm2 <- lsmeans(m2, revpairwise ~ trt|x, at=list(x=c(8.0, 8.5, 9.0,
9.5, 10.0, 10.12, 10.5, 11.0, .... [TRUNCATED]

> summary(lsm2, infer = c(T,T))
$lsmeans
x =  8.00:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    57.28063 1.342848 16 54.43392 60.12734  42.656  <.0001
 B    71.49706 2.057577 16 67.13519 75.85892  34.748  <.0001

x =  8.50:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    58.91517 1.265293 16 56.23287 61.59747  46.562  <.0001
 B    69.61241 1.800110 16 65.79635 73.42848  38.671  <.0001

x =  9.00:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    60.54971 1.212400 16 57.97954 63.11989  49.942  <.0001
 B    67.72777 1.568830 16 64.40200 71.05355  43.171  <.0001

x =  9.50:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    62.18426 1.187471 16 59.66693 64.70158  52.367  <.0001
 B    65.84313 1.376996 16 62.92403 68.76223  47.817  <.0001

x = 10.00:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    63.81880 1.192259 16 61.29132 66.34628  53.528  <.0001
 B    63.95849 1.243006 16 61.32344 66.59355  51.455  <.0001

x = 10.12:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    64.21109 1.197838 16 61.67179 66.75039  53.606  <.0001
 B    63.50618 1.221746 16 60.91619 66.09616  51.980  <.0001

x = 10.50:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    65.45334 1.226418 16 62.85345 68.05323  53.370  <.0001
 B    62.07385 1.186620 16 59.55833 64.58937  52.311  <.0001

x = 11.00:
 trt   lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    67.08788 1.287612 16 64.35827 69.81750  52.103  <.0001
 B    60.18921 1.218658 16 57.60577 62.77265  49.390  <.0001

x = 11.50:
```

```
 trt    lsmean       SE df lower.CL upper.CL t.ratio p.value
 A    68.72243 1.372229 16 65.81343 71.63142  50.081  <.0001
 B    58.30457 1.332759 16 55.47924 61.12989  43.747  <.0001

Confidence level used: 0.95
```

The means for TRT A and TRT B above match those from SAS for the respective values of x (left side of Table 23 in Chapter 9).

```
$contrasts
x =  8.00:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A     14.2164282 2.457003 16   9.0078145 19.4250418   5.786  <.0001

x =  8.50:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A     10.6972443 2.200310 16   6.0327959 15.3616927   4.862  0.0002

x =  9.00:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A      7.1780604 1.982711 16   2.9749005 11.3812204   3.620  0.0023

x =  9.50:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A      3.6588766 1.818297 16  -0.1957414  7.5134946   2.012  0.0613

x = 10.00:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A      0.1396927 1.722367 16  -3.5115615  3.7909470   0.081  0.9364

x = 10.12:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A     -0.7049114 1.710988 16  -4.3320433  2.9222205  -0.412  0.6858

x = 10.50:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A     -3.3794911 1.706508 16  -6.9971267  0.2381444  -1.980  0.0651

x = 11.00:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A     -6.8986750 1.772872 16 -10.6569949 -3.1403551  -3.891  0.0013

x = 11.50:
 contrast    estimate       SE df    lower.CL   upper.CL t.ratio p.value
 B - A    -10.4178589 1.912919 16 -14.4730655 -6.3626522  -5.446  0.0001

Confidence level used: 0.95
```

The point estimate of the treatment differences, CI limits, and p-values below match those from SAS for the respective values of x (right side of Table 23 in Chapter 9).

```
> #add predicted values from ANCOVA output (m2) to data frame
> pval <- data.frame(dat1, fitted(m2))

> #scatter plot of y vs x grouped by treatment
> sp <-ggplot(pval, aes(x=x, y=y, shape=trt,  group=trt))+
geom_point(size=3)+ theme_bw()
> #add predicted line to scatterplot
> finalplot <- sp + geom_line(aes(x=x, y=fitted.m2., linetype=trt))
> finalplot

> ggsave("c9-fig18p.png", dpi=600)
Saving 5.76 x 5.75 in image
```

See Fig. 9.4 of this appendix.

==========

Chapter 12. Spatial Analysis of Field Experiments

Juan Burgueño

Spatial analysis in SAS and R

There are two common approaches to analyzing spatial field trials: 1) using coordinate positions as covariates, and 2) using covariance structures in a linear, mixed model analysis. R and SAS follow similar algorithms for linear, mixed models so spatial analysis with coordinates as covariates would likely yield similar results. However, their routines for covariance structures are somewhat different and they only share half of the eight preprogrammed covariate structures (Table 3) so their results with the second approach will not likely match well. Because these two methods can be used together and there are many possible functions of the coordinate covariates, it is difficult to illustrate all the conceivable combinations; but this also allows tailoring the analysis to the data at hand. Such tailoring coupled with the differences between R and SAS decreases the probability of getting an exact match between R and SAS output. The real-world examples in this chapter demonstrate how the two approaches can be mixed. Thus, where the analysis focuses on various formulations of the row and column positions as covariates, the R and SAS results are very close. However, where there is more emphasis on covariance structures, the results are not so comparable. In spite of the different paths, the conclusions are often in agreement as they are in the following example, but certainly there is no guarantee that they will end up with the same conclusions. In summary, if one can get access to SAS, it is the better alternative because it has a wider range of options for agricultural trials that include these spatial analysis questions; however, R contains functions that can also provide adequate results. Whichever program is employed, the user needs to be aware of the differences so that the conclusions are suitable.

Example 1 (Chapter 12). Variety trial in a row & column design: Part 1

This example is a variety trial with 64 varieties planted in two blocks; there were 4 rows per block for a total of 8 rows and 16 columns. For the initial standard ANOVA, the R functions match SAS as shown in Script 14.1 and Output 14.1. R and SAS take different paths for the rest of the analyses, but end up with similar results (Script 14.2 and Output 14.2). That is, both R and SAS conclude there was an advantage to using a quadratic, row-column model, but no additional benefit for including a spatial covariance structure. Although the results were comparable in this case, it is not possible to ensure that the outcomes will be in agreement with other data sets.

Script 12.1 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(tidyverse)
library(broom)
library(car)
library(lsmeans)
```

```
#data management
#read in the data with read.csv function
dat0 <- read.csv("Ch14-Ex1dat.csv", header=F)
#add column names because they were not in data set
colnames(dat0) <- c("blk", "row", "col", "entry", "yield")
#attach as modern data frame; i.e. tibble
dat1 <- as.tibble(dat0)
#set up factor (class) variables from numeric variables
dat1$blkf <- factor(dat1$blk) ; dat1$rowf <- factor(dat1$row)
dat1$colf <- factor(dat1$col) ; dat1$entryf <- factor(dat1$entry)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#conduct anova and lsmeans analysis
aov1 <- lm(yield ~ rowf + colf + entryf , data=dat1)
fit1 <- glance(aov1)
str(fit1)
(MSE <- fit1$sigma^2)
round(Anova(aov1, type=3), 3)
lsmeans(aov1, "entryf")
```

Output 12.1 jb

```
---------------------------------------------------------------------------------------------
> source("c14-s1.r", echo=T)

> #set options & activate additional packages needed
> options(scipen=7)

> library(tidyverse)
-- Attaching packages -------------- tidyverse 1.2.1 --
v ggplot2 2.2.1     v purrr   0.2.4
v tibble  1.3.4     v dplyr   0.7.4
v tidyr   0.7.2     v stringr 1.2.0
v readr   1.1.1     v forcats 0.2.0
-- Conflicts ----------------------------------------
tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
> library(broom)
> library(car)
Attaching package: 'car'
The following object is masked from 'package:dplyr':
    recode
The following object is masked from 'package:purrr':
    some
> library(lsmeans)
```

```
Loading required package: estimability

> #data management
> #read in the data with read.csv function
> dat0 <- read.csv("Ch14-Ex1dat.csv", header=F)
> #add column names because they were not in data set
> colnames(dat0) <- c("blk", "row", "col", "entry", "yield")
> #attach as modern data frame; i.e. tibble
> dat1 <- as.tibble(dat0)
> #set up factor (class) variables from numeric variables
> dat1$blkf <- factor(dat1$blk) ; dat1$rowf <- factor(dat1$row)
> dat1$colf <- factor(dat1$col) ; dat1$entryf <- factor(dat1$entry)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #conduct anova and lsmeans analysis
> aov1 <- lm(yield ~ rowf + colf + entryf , data=dat1)

> fit1 <- glance(aov1)
> str(fit1)
'data.frame':   1 obs. of  11 variables:
 $ r.squared    : num 0.926
 $ adj.r.squared: num 0.775
 $ sigma        : num 0.377
 $ statistic    : num 6.16
 $ p.value      : num 0.00000000209
 $ df           : int 86
 $ logLik       : num 14.4
 $ AIC          : num 145
 $ BIC          : num 393
 $ deviance     : num 5.98
 $ df.residual  : int 42

> (MSE <- fit1$sigma^2)
[1] 0.1423837

> round(Anova(aov1, type=3), 3)
Anova Table (Type III tests)

Response: yield
term        Sum Sq Df F value Pr(>F)
(Intercept)  2.050  1  14.395 <2e-16 ***
rowf        24.932  7  25.015 <2e-16 ***
colf        17.507 15   8.197 <2e-16 ***
entryf       8.882 63   0.990  0.521
Residuals    5.980 42
---------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above ANOVA results match those in Table 1 of Chapter 14. The text does not include the lsmeans, but one of the supplemental files does include all 64 of them. As a comprise, the first six and last six are shown below.

```
> lsmeans(aov1, "entryf")
 entryf   lsmean        SE df lower.CL upper.CL
 1      2.083329 0.3074712 42 1.462827 2.703831
 2      2.772005 0.3098593 42 2.146684 3.397327
 3      2.740553 0.3150006 42 2.104856 3.376249
 4      2.596241 0.3155197 42 1.959497 3.232986
 5      1.719442 0.3133455 42 1.087086 2.351799
 6      2.761954 0.3162514 42 2.123733 3.400175

 59     2.913206 0.3146770 42 2.278162 3.548249
 60     2.064269 0.3149883 42 1.428597 2.699941
 61     2.498864 0.3542876 42 1.783883 3.213846
 62     2.959348 0.3503938 42 2.252225 3.666471
 63     3.090856 0.3122749 42 2.460659 3.721052
 64     2.560769 0.3086219 42 1.937945 3.183593

Results are averaged over the levels of: rowf, colf
Confidence level used: 0.95
```

==========

Example 1 (Chapter 12). Variety trial in a row & column design: Part 2

R and SAS take different paths for the rest of the analyses, but end up with similar results (Script 14.2 and Output 14.2).

Script 12.2 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(tidyverse)          #for tibble() etc.
library(nlme)               #for gls() etc.
library(geoR)               #for geodata(), variog4 etc.
library(vegan)              #for mantel()
library(lsmeans)            #for lsmeans()

#data management
#read in the data with read.csv function
dat0 <- read.csv("Ch14-Ex1dat.csv", header=F)
#add column names because they were not in data set
colnames(dat0) <- c("blk", "row", "col", "entry", "yield")
#attach as modern data frame; i.e. tibble
dat1 <- as.tibble(dat0)
#set up factor (class) variables from numeric variables
blkf <- factor(dat1$blk); rowf <- factor(dat1$row)
colf <- factor(dat1$col); entryf <- factor(dat1$entry)
dat1 <- add_column(dat1, blkf, rowf, colf, entryf)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)
```

```
#statistical analyses
#open PNG graphics device to save graphs
png("Ch14-F%d.png")
#conduct RCB anova (assumes autocorrelation=0)
m1 <- gls(yield ~ blkf + entryf, data=dat1)
round(anova(m1, type= "marginal"), 3)
(MSE1 <- m1$sigma^2)
m1
lsmeans(m1, "entryf")
#calculate distances and check for spatial correlation
res1 <- residuals(m1); dat1 <- add_column(dat1, res1)
dist1 <- dist(cbind(dat1$row, dat1$col))
dist2 <- dist(dat1$yield)
mantel(dist1, dist2)
#variogram of autocorrelations of residuals by rows & cols
#cols measure distance along rows (x-axis) & rows measure y
distance
dat2 <- tibble(dat1$col, dat1$row, res1)
datgeo <- as.geodata(dat2)
vario4a <- variog4(datgeo, uvec=1:17, max.dist=17, option="bin")
plot(vario4a)
vario4b <- variog4(datgeo, uvec=1:17, max.dist=17, option="bin",
direction=c(20, 40, 60, 80), unit.angle = "degrees", tolerance=10 )
plot(vario4b)
#compare models to account for spatial correlations
#quadratic polynomial model without spatial correlations
dat1$row <- as.double(dat1$row); dat1$col <- as.double(dat1$col);
m2ns <- gls(yield ~ row + I(row^2) + col + I(col^2) + entryf,
data=dat1, method="REML")
m2ns
round(anova(m2ns, type= "marginal"), 3)
m2ns$sigma^2
lsmeans(m2ns, "entryf")
#quadratic polynomial model + exponential correlation
mExp <- gls(yield ~ row + I(row^2) + col + I(col^2) + entryf,
data=dat1, corr=corExp(c(5, 0.04), form = ~row+ col, nugget=T),
method="REML")
mExp
round(anova(mExp, type= "marginal"), 3)
mExp$sigma^2
lsmeans(mExp, "entryf")
#compare AICs for nonspatial and spatial models
sapply(list(m2ns,mExp), AIC)
#mantel test of autocorrelation of residuals: nonspatial model
res.ns <- residuals(m2ns); dat1 <- add_column(dat1, res.ns)
dist.ns <- dist(dat1$res.ns)
mantel(dist1, dist.ns)
#mantel test of autocorrelation of residuals: Exp. spatial model
resExp <- residuals(mExp); dat1 <- add_column(dat1, resExp)
distExp <- dist(dat1$resExp)
mantel(dist1, distExp)
#narrow angle variogram for quadratic nonspatial model residuals
```

```
datNS <- tibble(dat1$col, dat1$row, dat1$res.ns)
datNSgeo <- as.geodata(datNS)
vario4ns <- variog4(datNSgeo, uvec=1:17, max.dist=17, option="bin",
direction=c(20, 40, 60, 80), unit.angle = "degrees", tolerance=10 )
plot(vario4ns)
plot(vario4ns$omnidirectional)
#narrow angle variogram for Exp. spatial model residuals
datExp <- tibble(dat1$col, dat1$row, dat1$resExp)
datEXPgeo <- as.geodata(datExp)
vario4Exp <- variog4(datEXPgeo, uvec=1:17, max.dist=17,
option="bin", direction=c(20, 40, 60, 80), unit.angle = "degrees",
tolerance=10 )
plot(vario4Exp)
plot(vario4Exp$omnidirectional)
dev.off()
```

Output 14.2 jb

```
> source("c14-s2.r", echo=T)

> #set options & activate additional packages needed
> options(scipen=7)
> library(tidyverse)       #for tibble() etc.
> library(nlme)            #for gls() etc.
> library(geoR)            #for geodata(), variog4 etc.
> library(vegan)           #for mantel()
> library(lsmeans)         #for lsmeans()

> #data management
> #read in the data with read.csv function
> dat0 <- read.csv("Ch14-Ex1dat.csv", header=F)
> #add column names because they were not in data set
> colnames(dat0) <- c("blk", "row", "col", "entry", "yield")
> #attach as modern data frame; i.e. tibble
> dat1 <- as.tibble(dat0)
> #set up factor (class) variables from numeric variables
> blkf <- factor(dat1$blk); rowf <- factor(dat1$row)
> colf <- factor(dat1$col); entryf <- factor(dat1$entry)
> dat1 <- add_column(dat1, blkf, rowf, colf, entryf)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #open PNG graphics device to save graphs
> png("Ch14-F%d.png")
> #conduct RCB anova (assumes autocorrelation=0)
> m1 <- gls(yield ~ blkf + entryf, data=dat1)
> round(anova(m1, type= "marginal"), 3)
Denom. DF: 63
term        numDF F-value p-value
(Intercept)     1  31.407  <.0001
blkf            1  30.958  <.0001
```

```
entryf          63   0.989   0.518

> (MSE1 <- m1$sigma^2)
[1] 0.5153285
```

MSE

Standard Randomized Complete Block with blocks as a fixed factor ANOVA above.

```
> m1
Generalized least squares fit by REML
  Model: yield ~ blkf + entryf
  Data: dat1
  Log-restricted-likelihood: -92.42376
```

The coefficient section that has little meaning here has been omitted.

```
Degrees of freedom: 128 total; 63 residual
Residual standard error: 0.7178638

> lsmeans(m1, "entryf")
 entryf  lsmean        SE df  lower.CL upper.CL
 1      3.21990 0.5076064 63 2.2055296  4.23427
 2      2.31740 0.5076064 63 1.3030296  3.33177
 3      2.43000 0.5076064 63 1.4156296  3.44437
 4      3.44160 0.5076064 63 2.4272296  4.45597
 5      1.56655 0.5076064 63 0.5521796  2.58092
 6      3.05635 0.5076064 63 2.0419796  4.07072

 59     3.15785 0.5076064 63 2.1434796  4.17222
 60     1.73805 0.5076064 63 0.7236796  2.75242
 61     2.65900 0.5076064 63 1.6446296  3.67337
 62     2.11970 0.5076064 63 1.1053296  3.13407
 63     3.11505 0.5076064 63 2.1006796  4.12942
 64     2.12320 0.5076064 63 1.1088296  3.13757

Results are averaged over the levels of: blkf
Confidence level used: 0.95
```

To save space, I've only shown the first and last six means for entries. These means match the arithmetic means, but not the lsmeans obtained in Script 14.1 above because of the different parameterization.

```
> #calculate distances and check for spatial correlation
> res1 <- residuals(m1); dat1 <- add_column(dat1, res1)
> dist1 <- dist(cbind(dat1$row, dat1$col))
> dist2 <- dist(dat1$yield)
> mantel(dist1, dist2)
Mantel statistic based on Pearson's product-moment correlation
Call:
mantel(xdis = dist1, ydis = dist2)

Mantel statistic r: 0.2471
      Significance: 0.001
```

The Mantel test concludes that the residuals from the standard RCBD analysis are spatially autocorrelated in spite of the blocking removing a significant amount of variation (see previous ANOVA).

```
Upper quantiles of permutations (null model):
   90%    95%  97.5%    99%
0.0352 0.0433 0.0515 0.0627
Permutation: free
Number of permutations: 999
> #variogram of autocorrelations of residuals by rows & cols
> #cols measure distance along rows (x-axis) & rows measure y distance
> dat2 <- tibble(d .... [TRUNCATED]
> datgeo <- as.geodata(dat2)
> vario4a <- variog4(datgeo, uvec=1:17, max.dist=17, option="bin")
variog: computing variogram for direction = 0 degrees (0 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 45 degrees (0.785 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 135 degrees (2.356
radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing omnidirectional variogram

> plot(vario4a)      See Fig. 14.1

> vario4b <- variog4(datgeo, uvec=1:17, max.dist=17, option="bin",
direction=c(20, 40, 60, 80), unit.angle = "degrees", tolerance=10 )
variog: computing variogram for direction = 20 degrees (0.349 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 40 degrees (0.698 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 60 degrees (1.047 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 80 degrees (1.396 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing omnidirectional variogram

> plot(vario4b)      See Fig. 14.2

> #compare models to account for spatial correlations
> #quadratic polynomial model without spatial correlations
> dat1$row <- as.double(dat1$row); da .... [TRUNCATED]
> m2ns <- gls(yield ~ row + I(row^2) + col + I(col^2) + entryf,
data=dat1, method="REML")
> m2ns
Generalized least squares fit by REML
  Model: yield ~ row + I(row^2) + col + I(col^2) + entryf
  Data: dat1
  Log-restricted-likelihood: -76.5744

Coefficients:
(Intercept)         row    I(row^2)         col    I(col^2)
```

```
 0.9158800  0.71919791 -0.06175290 -0.19403210  0.01604484
```

The coefficients for entries that have little meaning here have been omitted.

```
Degrees of freedom: 128 total; 60 residual
Residual standard error: 0.4630163

> round(anova(m2ns, type= "marginal"), 3)
Denom. DF: 60
term         numDF F-value p-value
(Intercept)      1   3.755   0.057
row              1  38.942  <.0001
I(row^2)         1  23.907  <.0001
col              1  16.230  <.0001
I(col^2)         1  34.857  <.0001
entryf          63   1.031   0.453

> m2ns$sigma^2
[1] 0.2143841
```

MSE

The ANOVA above for the quadratic model without a spatial covariance structure displays F-values that are 1.4 to 2.0 times larger than those for the same model with a spatial covariance structure as shown in Table 5 of Chapter 14. The MSE is almost 10% smaller, but both are considerably smaller than for the RCBD analysis.

```
> lsmeans(m2ns, "entryf")
 entryf   lsmean         SE df lower.CL upper.CL
 1      2.411741 0.3424705 60 1.726698 3.096784
 2      2.668557 0.3602698 60 1.947910 3.389204
 3      2.291612 0.3406829 60 1.610145 2.973080
 4      2.765162 0.3378798 60 2.089302 3.441022
 5      2.082612 0.3525834 60 1.377341 2.787884
 6      2.792772 0.3399968 60 2.112677 3.472867

 59     3.025607 0.3522022 60 2.321098 3.730117
 60     1.723734 0.3327731 60 1.058088 2.389379
 61     2.327718 0.3429620 60 1.641692 3.013744
 62     2.453315 0.3546781 60 1.743853 3.162777
 63     3.099049 0.3418796 60 2.415188 3.782910
 64     2.668570 0.3480097 60 1.972447 3.364693

Confidence level used: 0.95
```

To save space, I've only shown the first and last six means for entries. These means do not match either the arithmetic means or the lsmeans obtained in Script 14.1 above because of the different parameterization. The standard errors are smaller than those for the RCBD analysis but they are 1% to 20% larger than those for the linear row-col model in Script 14.1.

```
> #quadratic polynomial model + exponential correlation
> mExp <- gls(yield ~ row + I(row^2) + col + I(col^2) + entryf,
data=dat1, corr=corExp .... [TRUNCATED]
> mExp
Generalized least squares fit by REML
```

```
  Model: yield ~ row + I(row^2) + col + I(col^2) + entryf
  Data: dat1
  Log-restricted-likelihood: -75.41009

Coefficients:
(Intercept)         row    I(row^2)         col    I(col^2)
 0.97678140  0.63859423 -0.05485748 -0.20862875 0.01697914
```

The coefficients for entries that have little meaning here have been omitted.

```
Correlation Structure: Exponential spatial correlation
 Formula: ~row + col
 Parameter estimate(s):
`         range            nugget
0.7314369618769   0.0000002714365
```

The estimated range and nugget are smaller than I anticipated, but not unreasonable considering the variabitity shown in the variograms.

```
Degrees of freedom: 128 total; 60 residual
Residual standard error: 0.4815282
> round(anova(mExp, type= "marginal"), 3)
Denom. DF: 60
term        numDF F-value p-value
(Intercept)     1   3.739   0.058
row             1  20.406  <.0001
I(row^2)        1  12.746   0.001
col             1   9.622   0.003
I(col^2)        1  19.706  <.0001
entryf         63   1.008   0.489

> mExp$sigma^2
[1] 0.2318695
```

MSE

The above ANOVA for the quadratic model plus the Exponential spatial covariance structure displays F-values that are about 3% larger for the corresponding row effects and 15% smaller for the column effects than those in the SAS model with a spatial covariance structure as shown in Table 5 of Chapter 14. The MSE is within 1% of the SAS value in that table.

```
> lsmeans(mExp, "entryf")
 entryf   lsmean        SE df lower.CL upper.CL
 1      2.192990 0.3464834 60 1.499920 2.886060
 2      2.715879 0.3617868 60 1.992198 3.439561
 3      2.463421 0.3418137 60 1.779692 3.147150
 4      2.644421 0.3401078 60 1.964104 3.324738
 5      1.947658 0.3689092 60 1.209729 2.685586
 6      2.777416 0.3452501 60 2.086813 3.468019

 59     2.918109 0.3506601 60 2.216684 3.619533
 60     1.771072 0.3391168 60 1.092738 2.449407
 61     2.411280 0.3507269 60 1.709722 3.112839
 62     2.578086 0.3470665 60 1.883850 3.272323
 63     3.136655 0.3452854 60 2.445981 3.827328
```

```
 64     2.534580 0.3496485 60 1.835179 3.233981

Confidence level used: 0.95
```

To save space, I've only shown the first and last six means for entries. These means do not match either the arithmetic means, the lsmeans obtained with the linear row-column model (Script 14.1), or the quadratic row-column model without a spatial covariance structure (above) because of the different parameterization. The standard errors are smaller than those for the RCBD analysis but similar to the quadratic row-column model without a spatial covariance structure.

```
> #compare AICs for nonspatial and spatial models
> sapply(list(m2ns,mExp), AIC)
[1] 291.1488 292.8202
```

The AIC for the quadratic row-column model with a spatial covariance structure is 1.7 units greater than the same model without the covariance structure.

```
> #mantel test of correlation of residuals: nonspatial model
> res.ns <- residuals(m2ns); dat1 <- add_column(dat1, res.ns)
> dist.ns <- dist(dat1$res.ns)
> mantel(dist1, dist.ns)
Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = dist1, ydis = dist.ns)

Mantel statistic r: 0.001534
      Significance: 0.485
```

The quadratic row-column model without a spatial covariance structure adequately removed the spatial correlation according to the Mantel test and its AIC values was less than the same model with the covariance structure. For these reasons and parsimony, one would generally prefer the simpler model that did not include the covariance structure.

```
Upper quantiles of permutations (null model):
'  90%    95%  97.5%    99%
0.0338 0.0439 0.0564 0.0693
Permutation: free
Number of permutations: 999

> #mantel test of autocorrelation of residuals: Exp. spatial model
> resExp <- residuals(mExp); dat1 <- add_column(dat1, resExp)
> distExp <- dist(dat1$resExp)
> mantel(dist1, distExp)
Mantel statistic based on Pearson's product-moment correlation
Call:
mantel(xdis = dist1, ydis = distExp)

Mantel statistic r: -0.01381
      Significance: 0.691
```

The quadratic row-column model plus an exponential spatial covariance structure also adequately removed the spatial correlation according to the Mantel test.

```
Upper quantiles of permutations (null model):
`  90%    95%  97.5%    99%
0.0332 0.0429 0.0533 0.0609
Permutation: free
Number of permutations: 999

> #narrow angle variogram for quadratic nonspatial model residuals
> datNS <- tibble(dat1$col, dat1$row, dat1$res.ns)
> datNSgeo <- as.geodata(datNS)
> vario4ns <- variog4(datNSgeo, uvec=1:17, max.dist=17, option="bin",
direction=c(20, 40, 60, 80), unit.angle = "degrees", tolerance=10 )
variog: computing variogram for direction = 20 degrees (0.349 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 40 degrees (0.698 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 60 degrees (1.047 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 80 degrees (1.396 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing omnidirectional variogram

> plot(vario4ns)       See Fig. 14.3.

> plot(vario4ns$omnidirectional)        See Fig. 14.4.

> #narrow angle variogram for Exp. spatial model residuals
> datExp <- tibble(dat1$col, dat1$row, dat1$resExp)
> datEXPgeo <- as.geodata(datExp)

> vario4Exp <- variog4(datEXPgeo, uvec=1:17, max.dist=17,
option="bin", direction=c(20, 40, 60, 80), unit.angle = "degrees",
tolerance=10 )
variog: computing variogram for direction = 20 degrees (0.349 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 40 degrees (0.698 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 60 degrees (1.047 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing variogram for direction = 80 degrees (1.396 radians)
        tolerance angle = 10 degrees (0.175 radians)
variog: computing omnidirectional variogram

> plot(vario4Exp)           See Fig. 14.5.

> plot(vario4Exp$omnidirectional)       See Fig. 14.6.

> dev.off()
null device
          1
>
```

Literature Cited

Bates, Douglas, Martin Mächler, B.M. Bolker, , S.C. Walker. 2015. Fitting linear mixed-effect models using lme4. J. Stat. Software. 67: 1-48. doi: 10.18637/jss.v067.i01

Chang, Winston. 2017 Cookbook for R. www.cookbook-r.com_(accessed 13 Dec. 2017) *[2017 is year accessed]*.

Choe, Hye-Min, Mijeonng Kim, and Eun-Kyung Lee. 2017. EMSaov: An R package for the analysis of variance with the expected mean squares and its Shine application. The R Jour. 9:252-261.

Clark, M.. 2018 Mixed Models in R. CSCAR, ARC, Univ. of MI. https://m-clark.github.io/mixed-models-with-R/ (verified 21 Mar. 2018).

Faraway, J.J. 2015. Linear models with R. 2nd ed. CRC Press/Taylor and Francis Group. Boca Raton, FL.

Fox, J. and S. Weisberg. 2014. Time-Series Regression and Generalized Least Squares in R. An Appendix to An R Companion to Applied Regression. https://socialsciences.mcmaster.ca/jfox/Books/Companion/appendix.html (verified 21 Mar. 2018

Landram, R.G. and B. Alidaee. 1997. Computing orthogonal polynomials. Computers Oper. Res. 24:387-491.

Lemoine, N. R. 2012. R for Ecologists: Putting Together a Piecewise Regression. www.r-bloggers.com/r-for-ecologists-putting-together-a-piecewise-regression/ (verified 21 Mar. 2018).

Maier, R. 2015. Contrasts in R: A (sort of) complete guide to contrasts in R. http://rstudio-pubs-static.s3.amazonaws.com/65059_586f394d8eb84f84b1baaf56ffb6b47f.html (verified 22 Mar. 2018).

R Core Team. 2017. A language and environment for statistical computing. R Foundation for Statistical Computing , Vienna, Austria. https://www.R-project.org/ (verified 22 Mar. 2018).

Ryan, S.E. and L.S. Porth. 2007. A tutorial on the piecewise regression approach applied to the bedload transport data. Gen. Tech. Rep. RMRS-GTR-189. Fort Collins CO: USDA, Forest Service, Rocky Mountain Research Station. http://www.fs.fed.us/rm/pubs/rmrs_gtr189.pdf (verified 21 Mar 2018).

SSCC (Social Science Computing Cooperative). 2016. Mixed models: Testing significance of effects. https://www.ssc.wisc.edu/sscc/pubs/MM/MM_TestEffects.html (verified 22 Mar. 2018).

Teetor, Paul. 2013. R cookbook. O'Reilly Media, Inc. Sebastopol, CA

Weiss, Jack. 2007. Lecture 42(lab). https://www.unc.edu/courses/2007spring/enst/562/001/docs/lectures/lecture42.htm (verified 2 Mar 2018).

Chapter 13. Augmented Designs – Experimental Designs in which All Treatments are Not Replicated

Juan Burgueño, José Crossa, Francisco Rodriguez, and Kathleen M. Yeater

Example 1 (Chapter. 13). Completely randomized design with un-replicated treatments.

This example consists of two treatments (C1 and C2) replicated three times and six un-replicated treatments (G3-G8). The twelve lines of data were stored in a file named Ch13-Ex1dat.txt for these scripts. The SAS code for the five PROC GLIMMIX analyses, which the authors summarized in Tables 1-3 of Chapter 13, have been translated to R-scripts below. There are functions in R specifically for analyzing augmented design data, but the following R scripts use the more generic linear model analysis functions, lm() and lmer(), because they have been more completely vetted and allow more flexibility. Examine Example 1 of Chapter 3 in the text and this appendix for another illustration of an augmented design. R-script 13.1 emulates the SAS analysis of the treatment code only model.

Script 13.1 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(lsmeans)

#data management
#read in the data with read.csv function
dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
#add column names because they were not in data set
colnames(dat1) <- c("t", "d1", "d2", "y")
#attach as data frame
attach(dat1)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#conduct anova and lsmeans analysis
aov1 <- lm(y ~ t , data=dat1)
anova(aov1)
(lsm1 <- lsmeans(aov1, "t"))
contrast(lsm1, list("C1 v C2" = c(1, -1, 0,0,0,0,0,0),
                    "G3 v G4" = c(0,0,1,-1,0,0,0,0),
                    "C1 v G3" = c(1,0,-1,0,0,0,0,0)))
```

Output 13.1 jb

```
> #set options & activate additional packages needed
> options(scipen=7)
> library(lsmeans)

> #data management
> #read in the data with read.csv function
> dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
> #add column names because they were not in data set
> colnames(dat1) <- c("t", "d1", "d2", "y")
> #attach as data frame
> attach(dat1)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #conduct anova and lsmeans analysis
> aov1 <- lm(y ~ t , data=dat1)
> anova(aov1)
Analysis of Variance Table

Response: y
Term       Df  Sum Sq Mean Sq F value    Pr(>F)
t           7 262.333  37.476  16.061 0.008773 **
Residuals   4   9.333   2.333
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The line for "t" in the above ANOVA table matches the first line of the left section of Table 1 in Chapter 13 (P1-f Model). The lsmeans and SEs below match the BLUE and Standard Error columns for the P1-f analysis in Table 2 of Chapter 13.

```
> (lsm1 <- lsmeans(aov1, "t"))
 t    lsmean        SE df lower.CL upper.CL
 C1 22.33333 0.8819171  4 19.88474 24.78193
 C2 27.66667 0.8819171  4 25.21807 30.11526
 G3 28.00000 1.5275252  4 23.75891 32.24109
 G4 32.00000 1.5275252  4 27.75891 36.24109
 G5 27.00000 1.5275252  4 22.75891 31.24109
 G6 38.00000 1.5275252  4 33.75891 42.24109
 G7 35.00000 1.5275252  4 30.75891 39.24109
 G8 28.00000 1.5275252  4 23.75891 32.24109

Confidence level used: 0.95

> contrast(lsm1, list("C1 v C2" = c(1, -1, 0,0,0,0,0,0),
+                     "G3 v G4" = c(0,0,1,-1,0,0,0,0),
+                     "C1 v G3" = c(1,0,-1,0,0,0,0,0)))
 contrast  estimate       SE df t.ratio p.value
 C1 v C2  -5.333333 1.247219  4  -4.276  0.0129
 G3 v G4  -4.000000 2.160247  4  -1.852  0.1377
 C1 v G3  -5.666667 1.763834  4  -3.213  0.0325
```

The above contrast estimates, SEs, df, and p.values are the same as those for the Fixed Models (upper) section of Table 3 in Chapter 13.

==========

R-script 13.2 emulates the SAS analysis of the model including d1, the coding which identifies Check Treatments 1 and 2 plus the Unreplicated Treatments (3).

Script 13.2 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(lsmeans)

#data management
#read in the data with read.csv function
dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
#add column names because they were not in data set
colnames(dat1) <- c("t", "d1", "d2", "y")
#attach as data frame
attach(dat1)
#set up factor (class) variable from numeric variable
dat1$d1f <- factor(dat1$d1)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#conduct anova and lsmeans analysis
contrasts(dat1$d1f) <- contr.sum
contrasts(dat1$t) <- contr.sum
aov1 <- lm(y ~ d1f + t %in% d1f, data=dat1)
anova(aov1)
(lsm1 <- lsmeans(aov1, specs= c("d1f", "t")))
contrast(lsm1, list("C1 v C2" = c(1, 0, 0, 0, -1, 0, rep(0,18)),
      "G3 v G4" = c(rep(0 ,8), 1, 0, 0, -1, rep(0,12)),
      "C1 v G3" = c(1, rep(0 ,7), -1, 0, rep(0,14)) ,
      "Mean: C1+C2" = c(.5, 0, 0.5, rep(0, 13)),
      "Mean: G3-G8" = c(rep(0, 4), rep(c(0, 1/6), 6))))
```

Output 13.2 jb

```
> #set options & activate additional packages needed
> options(scipen=7)
> library(lsmeans)

> #data management
```

```
> #read in the data with read.csv function
> dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
> #add column names because they were not in data set
> colnames(dat1) <- c("t", "d1", "d2", "y")
> #attach as data frame
> attach(dat1)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #conduct anova and lsmeans analysis
> contrasts(dat1$d1f) <- contr.sum
> contrasts(dat1$t) <- contr.sum

> aov1 <- lm(y ~ d1f + t %in% d1f, data=dat1)
> anova(aov1)
Analysis of Variance Table

Response: y
Term      Df  Sum Sq Mean Sq F value   Pr(>F)
d1f        2 163.000  81.500 34.9286 0.002933 **
d1f:t      5  99.333  19.867  8.5143 0.029548 *
Residuals  4   9.333   2.333
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The lines for "d1f" and "d1f:t" in the above ANOVA table match the lines of the second and third rows of the left section of Table 1 in Chapter 13 (P2-f Model). The lsmeans and SEs below match the BLUE and Standard Error columns for the P2-f analysis in Table 2 of Chapter 13, except that the missing cells of the augmented design are made more apparent by the NAs.

```
> (lsm1 <- lsmeans(aov1, specs= c("d1f", "t")))
 d1f t    lsmean        SE df lower.CL upper.CL
 1   C1 22.33333 0.8819171  4 19.88474 24.78193
 2   C1       NA        NA NA       NA       NA
 3   C1       NA        NA NA       NA       NA
 1   C2       NA        NA NA       NA       NA
 2   C2 27.66667 0.8819171  4 25.21807 30.11526
 3   C2       NA        NA NA       NA       NA
 1   G3       NA        NA NA       NA       NA
 2   G3       NA        NA NA       NA       NA
 3   G3 28.00000 1.5275252  4 23.75891 32.24109
 1   G4       NA        NA NA       NA       NA
 2   G4       NA        NA NA       NA       NA
 3   G4 32.00000 1.5275252  4 27.75891 36.24109
 1   G5       NA        NA NA       NA       NA
 2   G5       NA        NA NA       NA       NA
 3   G5 27.00000 1.5275252  4 22.75891 31.24109
 1   G6       NA        NA NA       NA       NA
 2   G6       NA        NA NA       NA       NA
 3   G6 38.00000 1.5275252  4 33.75891 42.24109
 1   G7       NA        NA NA       NA       NA
 2   G7       NA        NA NA       NA       NA
 3   G7 35.00000 1.5275252  4 30.75891 39.24109
```

```
 1    G8        NA         NA NA        NA        NA
 2    G8        NA         NA NA        NA        NA
 3    G8 28.00000 1.5275252  4 23.75891 32.24109

Confidence level used: 0.95

> contrast(lsm1, list("C1 v C2" = c(1, 0, 0, 0, -1, 0, rep(0,18)),
+                     "G3 v G4" = c(rep(0 ,8), 1, 0, 0, -1,
rep(0,12)),
+                     "C1 v G3" = c(1, rep(0  .... [TRUNCATED]
 contrast     estimate       SE df  t.ratio p.value
 C1 v C2     -5.333333 1.247219   4  -4.276  0.0129
 G3 v G4     -4.000000 2.160247   4  -1.852  0.1377
 C1 v G3     -5.666667 1.763834   4  -3.213  0.0325
 Mean: C1+C2 25.000000 0.6236096  4  40.089  <.0001
 Mean: G3-G8 31.333333 0.6236096  4  50.245  <.0001
```

The first three contrast estimates, SEs, df, and p.values above are the same as those for the Fixed Models (upper) section of Table 3 in Chapter 13. Note that there are 3 levels for d1f (d1) and 8 for t, so the reference grid has 24 cells; thus, there must be 24 coefficients for each contrast; and that the rep() function simplifies entering them. The last two contrasts give the means etc. for the checks and unreplicated treatments and those match the ones given in Table 2 of Chapter 13 for the P2-f model.

==========

R-script 13.3 emulates the SAS analysis of the model including d2, the coding which identifies Check Treatments (1) versus the Unreplicated Treatments (2).

Script 13.3 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(lsmeans)


#data management
#read in the data with read.csv function
dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
#add column names because they were not in data set
colnames(dat1) <- c("t", "d1", "d2", "y")
#attach as data frame
attach(dat1)
#set up factor (class) variables from numeric variables
dat1$d2f <- factor(dat1$d2)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#conduct anova and lsmeans analysis
```

```
contrasts(dat1$d2f) <- contr.sum
contrasts(dat1$t) <- contr.sum
aov1 <- lm(y ~ d2f + t %in% d2f, data=dat1)
anova(aov1)
(lsm1 <- lsmeans(aov1, specs= c("d2f", "t")))
contrast(lsm1, list("C1 v C2" = c(1, 0, -1, rep(0, 13)),
      "G3 v G4" = c(0, 0, 0, 0, 0, 1, 0, -1, rep(0, 8)),
      "C1 v G3" = c(1, 0, 0, 0, 0, -1, rep(0, 10)) ,
      "Mean: C1+C2" = c(.5, 0, 0.5, rep(0, 13)),
      "Mean: G3-G8" = c(rep(0, 4), rep(c(0, 1/6), 6))))
#anova by slice of d2
by(dat1, dat1$d2f, function(df) anova(lm(y ~ t, data=df)))
```

Output 13.3 jb

```
> #set options & activate additional packages needed
> options(scipen=7)
> library(lsmeans)

> #data management
> #read in the data with read.csv function
> dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
> #add column names because they were not in data set
> colnames(dat1) <- c("t", "d1", "d2", "y")
> #attach as data frame
> attach(dat1)
> #set up factor (class) variables from numeric variables
> dat1$d2f <- factor(dat1$d2)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #conduct anova and lsmeans analysis
> contrasts(dat1$d2f) <- contr.sum
> contrasts(dat1$t) <- contr.sum

> aov1 <- lm(y ~ d2f + t %in% d2f, data=dat1)

> anova(aov1)
Analysis of Variance Table
Response: y
Term      Df  Sum Sq Mean Sq F value   Pr(>F)
d2f        1 120.333 120.333  51.571 0.001991 **
d2f:t      6 142.000  23.667  10.143 0.020989 *
Residuals  4   9.333   2.333
---------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The lines for "d2f" and "d2f:t" in the above ANOVA table match the lines of the fourth and fifth rows of the left section of Table 1 in Chapter 13 (P3-f Model). The lsmeans and SEs below match the BLUE and Standard Error columns for the P3-f analysis in Table 2 of Chapter 13, except that the missing cells of the augmented design are made more apparent by the NAs.

```
> (lsm1 <- lsmeans(aov1, specs= c("d2f", "t")))
 d2f t    lsmean        SE df lower.CL upper.CL
 1   C1 22.33333 0.8819171  4 19.88474 24.78193
 2   C1       NA        NA NA       NA       NA
 1   C2 27.66667 0.8819171  4 25.21807 30.11526
 2   C2       NA        NA NA       NA       NA
 1   G3       NA        NA NA       NA       NA
 2   G3 28.00000 1.5275252  4 23.75891 32.24109
 1   G4       NA        NA NA       NA       NA
 2   G4 32.00000 1.5275252  4 27.75891 36.24109
 1   G5       NA        NA NA       NA       NA
 2   G5 27.00000 1.5275252  4 22.75891 31.24109
 1   G6       NA        NA NA       NA       NA
 2   G6 38.00000 1.5275252  4 33.75891 42.24109
 1   G7       NA        NA NA       NA       NA
 2   G7 35.00000 1.5275252  4 30.75891 39.24109
 1   G8       NA        NA NA       NA       NA
 2   G8 28.00000 1.5275252  4 23.75891 32.24109

Confidence level used: 0.95

> contrast(lsm1, list("C1 v C2" = c(1, 0, -1, rep(0, 13)),
+        "G3 v G4" = c(0, 0, 0, 0, 0, 1, 0, -1, rep(0, 8)),
+        "C1 v G3" = c(1, 0, 0, 0, 0, -1, r .... [TRUNCATED]

 contrast     estimate       SE  df t.ratio p.value
 C1 v C2     -5.333333 1.247219   4  -4.276  0.0129
 G3 v G4     -4.000000 2.160247   4  -1.852  0.1377
 C1 v G3     -5.666667 1.763834   4  -3.213  0.0325
 Mean: C1+C2 25.000000 0.6236096  4  40.089  <.0001
 Mean: G3-G8 31.333333 0.6236096  4  50.245  <.0001
```

The first three contrast estimates, SEs, df, and p.values above are the same as those for the Fixed Models (upper) section of Table 3 in Chapter 13. Note that there are 2 levels for d2f (d2) and 8 for t, so the reference grid has 16 cells; thus, there must be 16 coefficients for each contrast; and that the rep() function simplifies entering them. The last two contrasts give the means etc. for the checks and unreplicated treatments and those match the ones given in Table 2 of Chapter 13 for the P3-f model.

```
> #anova by slice of d2
> by(dat1, dat1$d2f, function(df) anova(lm(y ~ t, data=df)))
dat1$d2f: 1
Analysis of Variance Table

Response: y
Term       Df Sum Sq Mean Sq F value  Pr(>F)
t           1 42.667  42.667  18.286 0.01289 *
Residuals   4  9.333   2.333
---------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The line for "t" in the above ANOVA table matches the sixth row ("t(d2=1") of the left section of Table 1 in Chapter 13 (P3-f Model).

```
dat1$d2f: 2
Analysis of Variance Table

Response: y
Term      Df Sum Sq Mean Sq F value Pr(>F)
t          5 99.333  19.867
Residuals  0  0.000

Warning messages:

1: contrasts dropped from factor t due to missing levels
2: contrasts dropped from factor t due to missing levels
3: In anova.lm(lm(y ~ t, data = df)) :
  ANOVA F-tests on an essentially perfect fit are unreliable
```

The line for "t" in the above ANOVA table matches the seventh row ("t(d2=2") of the left section of Table 1 in Chapter 13 (P3-f Model) for the df. Because there was only one rep of each of the test entries (G3-G8) no error could be computed due to perfect fit. However, the error from the previous ANOVA on the checks could be used and the F-ratio and P(>F) calculated with keyboard entries (in red) as shown below and they equal those in Table 1 of Chapter 13.

```
> 19.867/2.333
[1] 8.515645
> 1-pf(8.5156, 5, 4)
[1] 0.02953962
>
==========================================================
```

R-script 13.4 emulates the SAS likelihood ratio test of the random effect, d1:t, and the treatment BLUPs.

Script 13.4 jb

```
-------------------------------------------------------------------------------------------------
#set options & activate additional packages needed
options(scipen=7)
library(lme4)
library(lmerTest)
library(lsmeans)
library(RLRsim)
library(merTools)
stdlmer <- lme4::lmer
fullaov <- lmerTest::anova

#data management
#read in the data with read.csv function
dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
#add column names because they were not in data set
colnames(dat1) <- c("t", "d1", "d2", "y")
#attach as data frame
attach(dat1)
#set up factor (class) variables from numeric variables
```

```
dat1$d1f <- factor(dat1$d1)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#conduct anova with d1f:t as a random effect, merMod object
contrasts(dat1$d1f) <- contr.sum
contrasts(dat1$t) <- contr.sum
aov0 <- stdlmer(y ~ d1f + (1|d1f:t), data=dat1)
summary(aov0)
#likelihood ratio test of the random term
exactRLRT(aov0)
lsmeansLT(aov0)
ext1 <- REextract(aov0)
se2 <- ext1[,4]
#redo anova to get merModlmerTest object
aov1 <- lmer(y ~ d1f + (1|d1f:t), data=dat1)
fullaov(aov1)
re1 <- ranef(aov1, condVar=T)
predv1 <- predict(aov1)
dat2 <- data.frame(t, predv1)
#detach(dat1)
stat1 <- aggregate(dat2$predv1, by=list(dat2$t), FUN=mean)
#se1 <- attr(re1[[1]], which="postVar")
stat2 <- data.frame(stat1, se2)
colnames(stat2) <- c("t", "BLUP", "SE")
stat2
#linear combinations blups
mat1 <-    c(1, -1, 0, 0, 0, 0, 0, 0,
             0, 0, 1, -1, 0, 0, 0, 0,
             1, 0, -1, 0, 0, 0, 0, 0)
lcb <- matrix(mat1, nrow = 3, ncol = 8, byrow = T)
rownames(lcb) <- c("C1 v C2", "G3 v G4", "C1 v G3")
(blups <- lcb %*% stat2$BLUP)
```

Output 13.4 jb

```
> #set options & activate additional packages needed
> options(scipen=7)
> library(lme4)
> library(lmerTest)
> library(lsmeans)
> library(RLRsim)
> library(merTools)
> stdlmer <- lme4::lmer
> fullaov <- lmerTest::anova
```

I renamed two functions to distinguish which package they came from

```
> #data management
> #read in the data with read.csv function
> dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
> #add column names because they were not in data set
> colnames(dat1) <- c("t", "d1", "d2", "y")
> #attach as data frame
> attach(dat1)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #conduct anova with d1f:t as a random effect
> contrasts(dat1$d1f) <- contr.sum
> contrasts(dat1$t) <- contr.sum

> aov0 <- stdlmer(y ~ d1f + (1|d1f:t), data=dat1)

> summary(aov0)
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ d1f + (1 | d1f:t)
   Data: dat1

REML criterion at convergence: 50.1

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.09109 -0.27552 -0.08348  0.33959  1.09109

Random effects:
 Groups   Name        Variance Std.Dev.
 d1f:t    (Intercept) 17.533   4.187
 Residual              2.333   1.528
Number of obs: 12, groups:  d1f:t, 8
```

These variance components match those in the covariance section for the P2-r model in Table 1 of Chapter 13. Note that the Std. Dev. is simply the square root of the variance, not the standard error of the estimated variance.

```
> #likelihood ratio test of the random term
> exactRLRT(aov0)
        simulated finite sample distribution of RLRT.
        (p-value based on 10000 simulated values)

data:
RLRT = 4.0851, p-value = 0.0293
```

Testing a model with one random term is a problem in R because R uses different functions for the analysis of fixed models and mixed models; thus, a simulation method to do the test of the random effect is recommended (SSCC, 2016). The resulting p-value is somehat larger than the 0.0216 value reported in the far right column of Table 1 of Chapter 13. Despite the difference, the conclusion is the same in this case: the d1f:t term is significantly greater than zero at the 5% level.

```
> ext1 <- REextract(aov0)
> se2 <- ext1[,4]
Fixed effects:
term        Estimate Std. Error t value
(Intercept)  27.1111     2.1064  12.871
d1f1         -4.7778     3.2467  -1.472
d1f2          0.5556     3.2467   0.171
Correlation of Fixed Effects:
Eff. (Intr)   d1f1
d1f1  0.244
d1f2  0.244 -0.737

> lsmeansLT(aov0)
Least Squares Means table:
eff.   d1f   Est.  StdErr.  DF      t LowCI UpCI p-value
d1f  1 1.0  22.33     4.28  4.2  5.22  10.7 34.0   0.006**
d1f  2 2.0  27.67     4.28  4.2  6.47  16.0 39.3   0.002**
d1f  3 3.0  31.33     1.82  5.0 17.22  26.7 36.0 <2e-16***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The mean and standard error of the unreplicated varieties match those in Table 2 of Chapter 13 for the P2-r model.

```
> ext1 <- REextract(aov0)
> #redo anova to get merModlmerTest object
> aov1 <- lmer(y ~ d1f + (1|d1f:t), data=dat1)

> fullaov(aov1)
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom
term Sum Sq Mean Sq NumDF DenDF F.value Pr(>F)
d1f  9.3197  4.6599     2 4.309  1.9971 0.2434
```

The Mean Sq, NumDf, and F.value equal those in the Type III Tests of Fixed Effects in Table 1 of Chapter 13 for the P2-r model. R does not have the containment option for calculating the DenDF so this value and the resuling Pr(>F) are slightly different (4.3 v 4.0 and 0.24 vs 0.25).

```
> re1 <- ranef(aov1, condVar=T)
> predv1 <- predict(aov1)
> dat2 <- data.frame(t, predv1)
> #detach(dat1)
> stat1 <- aggregate(dat2$predv1, by=list(dat2$t), FUN=mean)
> #se1 <- attr(re1[[1]], which="postVar")
> stat2 <- data.frame(stat1, se2)
> colnames(stat2) <- c("t", "BLUP", "SE")

> stat2
No.  t     BLUP        SE
1   C1 22.33333 0.8629838
2   C2 27.66667 0.8629838
3   G3 28.39150 1.4350206
4   G4 31.92170 1.4350206
5   G5 27.50895 1.4350206
```

```
6    G6 37.21700 1.4350206
7    G7 34.56935 1.4350206
8    G8 28.39150 1.4350206
```

The point estimates of the variety means (BLUPs) are identical to those in Table 2 in Chapter 13 for the P2-r model. The standard errors (SEs) are within 0.02 of those in Table 2. Note that the R community often refers to BLUPs as Conditional Modes.

```
> #linear combinations blups
> mat1 <-       c(1, -1, 0, 0, 0, 0, 0, 0,
+                 0, 0, 1, -1, 0, 0, 0, 0,
+                 1, 0, -1, 0, 0, 0, 0, 0)

> lcb <- matrix(mat1, nrow = 3, ncol = 8, byrow = T)

> rownames(lcb) <- c("C1 v C2", "G3 v G4", "C1 v G3")

> (blups <- lcb %*% stat2$BLUP)
Contrast     [,1]
C1 v C2 -5.333333
G3 v G4 -3.530201
C1 v G3 -6.058166
```

The point estimates for the contrasts among variety means (BLUPs) are identical to those in Table 3 in Chapter 13 for the P2-r model. I was not able to find a function to determine the standard errors of these contrasts in R; however, the standard approach for the difference in two means produces estimates that are reasonably close. That is, standard error = square root(variance A + variance B) using the standard errors from the previous table gives 1.220, 2.029, and 1.675 compared to 1.247, 2.029, and 1.698 in Table 3 of Chapter 13.

==========

R-script 13.2 emulates the SAS analysis of the model with d1, the coding which identifies Check Treatments 1 and 2 plus the Unreplicated Treatments (3).

Script 13.5 jb

```
#set options & activate additional packages needed
options(scipen=7)
library(lme4)
library(lmerTest)
library(lsmeans)
library(RLRsim)
library(merTools)
stdlmer <- lme4::lmer
fullaov <- lmerTest::anova

#data management
#read in the data with read.csv function
dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
#add column names because they were not in data set
colnames(dat1) <- c("t", "d1", "d2", "y")
#attach as data frame
```

```
attach(dat1)
#make factor (class) variables from numeric variables
dat1$d2f <- factor(dat1$d2)
#make contrasts for t nested in d2 and add to dat1
#requires a complete set of orthogonal contrasts to get
#correct variance components
ct1 <- c(rep(1/3, 3), rep(-1/3, 3), rep(0, 6))
ct2 <- c(rep(0, 6), 1, -1, rep(0, 4))
ct3 <- c(rep(0, 6), 1/2, 1/2, rep(-1/2, 4))
ct4 <- c(rep(0, 8), 1, rep(-1/3, 3))
ct5 <- c(rep(0, 9), 1, rep(-1/2, 2))
ct6 <- c(rep(0, 10), 1, -1)
dat1 <- data.frame(dat1, ct1, ct2, ct3, ct4, ct5,ct6)
#check first & last 6 rows of data with head & tail functions
head(dat1)
tail(dat1)
#check stats for each variable
summary(dat1)

#statistical analyses
#model to get SEs of BLUP means from merMod object
aov0 <- stdlmer(y ~ d2f + (1|d2f:t),   data=dat1)
ext1 <- REextract(aov0)
se2 <- ext1[,4]
#redo anova to get merModlmerTest object
aov1 <- lmer(y ~ d2f + (1|d2f:t), data=dat1)
fullaov(aov1)
lsmeansLT(aov1)
re1 <- ranef(aov1, condVar=T)
predv1 <- predict(aov1)
dat2 <- data.frame(t, predv1)
#detach(dat1)
stat1 <- aggregate(dat2$predv1, by=list(dat2$t), FUN=mean)
#se1 <- attr(re1[[1]], which="postVar")
stat2 <- data.frame(stat1, se2)
colnames(stat2) <- c("t", "BLUP", "SE")
stat2
#linear combinations blups
mat1 <-    c(1, -1, 0, 0, 0, 0, 0, 0,
            0, 0, 1, -1, 0, 0, 0, 0,
            1, 0, -1, 0, 0, 0, 0, 0)
lcb <- matrix(mat1, nrow = 3, ncol = 8, byrow = T)
rownames(lcb) <- c("C1 v C2", "G3 v G4", "C1 v G3")
(blups <- lcb %*% stat2$BLUP)
#estmate covariance for checks (ct1) and others (ct2-ct6)
aov2 <- stdlmer(y ~ d2f + I((1|ct1)) + I((1|ct2+ct3+ct4+ct5+ct6)),
data=dat1)
summary(aov2)
m1<- stdlmer(y ~ d2f + I((1|ct1)),   data=dat1)
m2<- stdlmer(y ~ d2f + I((1|ct2+ct3+ct4+ct5+ct6)),   data=dat1)
#likelihood ratio test (LRT) of covariance term for checks
exactRLRT(m=m1, mA=aov2, m0=m2)
```

```
#LRT of covariance term for unreplicated entries
exactRLRT(m=m2, mA=aov2, m0=m1)
```

Output 13.5 jb

```
> #set options & activate additional packages needed
> options(scipen=7)
> library(lme4)
> library(lmerTest)
> library(lsmeans)
> library(RLRsim)
> library(merTools)
> stdlmer <- lme4::lmer
> fullaov <- lmerTest::anova
```

I renamed two functions to distinguish which package they came from

```
> #data management
> #read in the data with read.csv function
> dat1 <- read.csv("Ch13-Ex1dat.txt", header=F)
> #add column names because they were not in data set
> colnames(dat1) <- c("t", "d1", "d2", "y")
> #attach as data frame
> attach(dat1)
> #make factor (class) variables from numeric variables
> dat1$d2f <- factor(dat1$d2)

> #make contrasts for t nested in d2 and add to dat1
#requires a complete set of orthogonal contrasts to get
#correct variance components
> ct1 <- c(rep(1/3, 3), rep(-1/3, 3), rep(0, 6))
> ct2 <- c(rep(0, 6), 1, -1, rep(0, 4))
> ct3 <- c(rep(0, 6), 1/2, 1/2, rep(-1/2, 4))
> ct4 <- c(rep(0, 8), 1, rep(-1/3, 3))
> ct5 <- c(rep(0, 9), 1, rep(-1/2, 2))
> ct6 <- c(rep(0, 10), 1, -1)
> dat1 <- data.frame(dat1, ct1, ct2, ct3, ct4, ct5,ct6)
```

Checks on input data from head(), tail(), and summary() omitted

```
> #statistical analyses
> #model to get SEs of BLUP means from merMod object
> aov0 <- stdlmer(y ~ d2f + (1|d2f:t),   data=dat1)
> ext1 <- REextract(aov0)
> se2 <- ext1[,4]
> #redo anova to get merModlmerTest object
> aov1 <- lmer(y ~ d2f + (1|d2f:t), data=dat1)

> fullaov(aov1)
Analysis of Variance Table of type III  with  Satterthwaite
approximation for degrees of freedom
term  Sum Sq Mean Sq NumDF   DenDF  F.value  Pr(>F)
```

```
d2f    7.8706  7.8706     1  5.3956    3.359   0.122
```

Many of these results only approximate those for the Fixed Effects in Table 1 of Chapter 13 for the P3-r model. However, the final outcome, Pr(>F), is very close to the one in Table 1 (0.122 versus 0.1213).

```
> lsmeansLT(aov1)
Least Squares Means table:
Eff.  d2f    Est.    SE   DF     t  L.CI Up.CI  p-value
d2f 1 1.0   25.00  2.96  5.1  8.45  17.5  32.5  0.0003***
d2f 2 2.0   31.33  1.78  6.1 17.57  27.0  35.7  <2e-16***
---------------------------------------------------------
Sig. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated means of the fixed effects equal those in Table 2 of Chapter 13 for the P3-r model. But the standard errors are somewhat different (2.96 versus 2.67 and 1.78 versus 1.82).

```
> re1 <- ranef(aov1, condVar=T)
> predv1 <- predict(aov1)
> dat2 <- data.frame(t, predv1)
> #detach(dat1)
> stat1 <- aggregate(dat2$predv1, by=list(dat2$t), FUN=mean)
> #se1 <- attr(re1[[1]], which="postVar")
> stat2 <- data.frame(stat1, se2)
> colnames(stat2) <- c("t", "BLUP", "SE")
```

```
> stat2
   t      BLUP        SE
1 C1 22.45220 0.8638514
2 C2 27.54780 0.8638514
3 G3 28.40928 1.4336901
4 G4 31.91814 1.4336901
5 G5 27.53206 1.4336901
6 G6 37.18145 1.4336901
7 G7 34.54980 1.4336901
8 G8 28.40928 1.4336901
```

The point estimates of the variety means (BLUPs) well match those in Table 2 in Chapter 13 for the P3-r model. The standard errors (SEs) are within 0.02 of those in Table 2. Note that the R community often refers to BLUPs as Conditional Modes.

```
> #linear combinations blups
> mat1 <-       c(1, -1, 0, 0, 0, 0, 0, 0,
+                 0, 0, 1, -1, 0, 0, 0, 0,
+                 1, 0, -1, 0, 0, 0, 0, 0)

> lcb <- matrix(mat1, nrow = 3, ncol = 8, byrow = T)
> rownames(lcb) <- c("C1 v C2", "G3 v G4", "C1 v G3")
> (blups <- lcb %*% stat2$BLUP)
             [,1]
C1 v C2 -5.095593
G3 v G4 -3.508869
C1 v G3 -5.957072
```

The point estimates for the contrasts among variety means (BLUPs) approximate those in Table 3 in Chapter 13 for the P3-r model. I was not able to find a function to determine the standard errors of these contrasts in R; however, the standard approach for the difference in two means produces estimates that are reasonably close for the first two contrasts. That is, standard error = square root(variance A + variance B) using the standard errors from the previous table gives 1.220 and 2.029 compared to 1.247 and 2.029 in Table 3 of Chapter 13. I was not able to reproduce the standard error for the third contrast in R.

```
> #estmate covariance for checks (ct1) and others (ct2-ct6)
> aov2 <- stdlmer(y ~ d2f + I((1|ct1)) + I((1|ct2+ct3+ct4+ct5+ct6)),
data=dat1)

> summary(aov2)
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ d2f + I((1 | ct1)) + I((1 | ct2 + ct3 + ct4 + ct5 + ct6))
   Data: dat1

REML criterion at convergence: 54.1

Scaled residuals:
`    Min      1Q  Median      3Q     Max
-0.9956 -0.3186 -0.1025  0.3634  0.9956

Random effects:
 Groups                       Name        Variance Std.Dev.
 ct2 + ct3 + ct4 + ct5 + ct6 (Intercept) 17.533    4.187
 ct1                          (Intercept) 13.444    3.667
 Residual                                  2.333    1.528
Number of obs: 12, groups:  ct2 + ct3 + ct4 + ct5 + ct6, 7; ct1, 3
```

Using a complete set of orthogonal contrasts makes it possible to get the variance components reported in the right half of Table 1 in Chapter 13 for the P3-r model. See below for a test of these terms.

```
Fixed effects:
`            Estimate Std. Error t value
(Intercept)   25.000      4.964   5.036
d2f2           6.333      6.434   0.984

Correlation of Fixed Effects:
     (Intr)
d2f2 -0.772

> m1<- stdlmer(y ~ d2f + I((1|ct1)),   data=dat1)

> m2<- stdlmer(y ~ d2f + I((1|ct2+ct3+ct4+ct5+ct6)),   data=dat1)

> #likelihood ratio test (LRT) of covariance term for checks
> exactRLRT(m=m1, mA=aov2, m0=m2)

        simulated finite sample distribution of RLRT.
```

```
        (p-value based on 10000 simulated values)

data:
RLRT = 4.5664, p-value = 0.0084
```

It might be possible to piece together a likelihood ratio test for including a random term for the check varieties; however, it is easier and more appropriate to use the exactRLRT function simulation in R (SSCC, 2016). The resulting p-value is less than the one reported in the far right column of Table 1 of Chapter 13 for the P3-r model (0.0084 versus 0.0163), but both conclude this variance is significantly greater than zero at the 5% level.

```
> #LRT of covariance term for unreplicated entries
> exactRLRT(m=m2, mA=aov2, m0=m1)

        simulated finite sample distribution of RLRT.

        (p-value based on 10000 simulated values)

data:
RLRT = 4.0851, p-value = 0.0264
```

As in the case above for the term for the checks, I've used the exactRLRT function simulation in R (SSCC, 2016). The resulting p-value is larger than the one reported in the far right column of Table 1 of Chapter 13 for the P3-r model (0.0264 versus 0.0216), but both conclude this variance is significantly greater than zero at the 5% level.

Table 1

Table 1. Treatment randomization by location and block : equivalent of Chapter 2, Table 6†

Loc	Block	Plot									
		1	2	3	4	5	6	7	8	9	10
		---------- Treatment number ----------									
Central	1	6	9	8	10	7	1	4	5	2	3
Central	2	8	2	9	3	4	6	1	5	10	7
Central	3	1	9	5	7	10	2	3	8	4	6
East	1	10	7	8	4	6	1	5	2	9	3
East	2	9	2	4	10	5	1	6	8	3	7
East	3	6	3	9	2	10	5	8	7	4	1
West	1	7	4	2	1	3	6	5	10	9	8
West	2	6	3	8	4	10	2	1	5	9	7
West	3	8	4	10	6	7	2	9	5	1	3

† Text copied from R output (sketch list) and formatted in Word.

Table 2. Descriptive statistics for pH by location, mulch, and calcium treatment. These values match those in Table 7 of Chapter 2, except for five cases discussed in the Output 2.2 section.

	Loc	Mulch	Ca_Trt	N	Mean	StdDev	Min	Max
1	Central	no	control	3	5.4	0.1	5.3	5.6
2	Central	no	G1X	3	5.5	0.4	5.2	5.9
3	Central	no	G2X	3	5.7	0.2	5.5	5.9
4	Central	no	L1X	3	6.0	0.4	5.5	6.3
5	Central	no	L2X	3	6.1	0.1	6.0	6.1
6	Central	yes	control	3	5.7	0.3	5.5	6.0
7	Central	yes	G1X	3	5.9	0.2	5.7	6.1
8	Central	yes	G2X	3	5.5	0.4	5.1	5.9
9	Central	yes	L1X	3	6.0	0.2	5.7	6.1
10	Central	yes	L2X	3	6.1	0.3	5.8	6.3
11	East	no	control	3	4.0	0.2	3.8	4.2
12	East	no	G1X	3	4.0	0.1	3.9	4.1
13	East	no	G2X	3	4.0	0.2	3.9	4.2
14	East	no	L1X	3	4.2	0.3	3.9	4.4
15	East	no	L2X	3	4.4	0.3	4.1	4.8
16	East	yes	control	3	4.1	0.2	4.0	4.3
17	East	yes	G1X	3	4.0	0.2	3.9	4.2
18	East	yes	G2X	3	3.9	0.2	3.8	4.1
19	East	yes	L1X	3	4.3	0.3	4.1	4.7
20	East	yes	L2X	3	4.3	0.4	4.1	4.8
21	West	no	control	3	6.6	0.6	5.9	7.1
22	West	no	G1X	3	6.6	0.4	6.2	6.9
23	West	no	G2X	3	6.5	0.8	5.7	7.1
24	West	no	L1X	3	6.9	0.3	6.6	7.2
25	West	no	L2X	3	6.9	0.4	6.5	7.3
26	West	yes	control	3	6.8	0.3	6.5	7.0
27	West	yes	G1X	3	6.7	0.3	6.4	7.1
28	West	yes	G2X	3	6.5	0.7	5.8	7.2
29	West	yes	L1X	3	6.9	0.3	6.6	7.2
30	West	yes	L2X	3	7.0	0.2	6.8	7.3

Table 3. Spatial covariance structures preprogrammed in SAS PROC MIXED and R Package-nlme.

Structure	SAS	R
Spherical	Yes	Yes
Exponential	Yes	Yes
Gaussian	Yes	Yes
Linear	Yes	Yes
Linear log	Yes	No
Power	Yes	No
Matérn	Yes	No
Rational quadratic	No	Yes