Charles Fleming

Short Tutorial in R

October 2013

Contents

Chapter 1

Introduction

Upon the divestiture of AT&T in 1984, the source code for the S language became proprietary, whereas the C language remained in the public domain. The GCC compiler which was developed by Richard Stallman was licensed under the General Public License (GPL), and it paved the way for computer programmers to compile programs which were written in the C language for free. Soon afterwards a compiler was developed to compile FORTRAN programs. The availability of the GCC compiler and the GPL has lead to the worldwide development of Linux and eventually to the development of R . Because the owners of the source code for the S language refused to port S to the Linux platform, the R language was developed to imitate the syntax of S, but in such a way as to be independent of the source code of S for copyright reasons. A large number of FORTRAN programs which were in the public domain were incorporated into R , and because of the vigorous beta testing and development, R has supplanted the S language. In fact, John Chambers who led the development of S at AT&T has since become a core developer of R Because of the robust quality of R , because of its versatility, and by virtue of being licensed under the GPL, the popularity of R has grown immensely.

Because R is governed by the General Public License, the collection of uncompiled programs which constitute the source code of R must be made freely available to anyone who might want to study the logic of a certain procedure, and to allow someone to correct an error or to modify the code, in order to improve its efficiency. If an improvement to the program is deemed a good one by the core developers of R , then it will be implemented into the official version.

There are many libraries or what are nicknamed *add-on* packages which are collections of procedures which serve a specialized purpose. Some of the standard libraries are invoked automatically when R is initiated. Others, because there are so many of them, are not included in the installation of R and must be invoked manually. For example, the library *foreign* is a set of routines which will allow someone to bring a SAS or SPSS set of data into an R session

(Fleming, 2004) [1].

A set of data which was obtained from an experiment on examining the effects of levels of nitrogen and the structure of an habitat on the number of species of arthropods over a four month period on seven locations is based on a four parameter fixed effects linear model with the response variable, y_{ijkl}, being the number of species of arthropods:

$$y_i = \mu + \alpha_i + \beta_j + \gamma_k + \delta_l + \epsilon_{ijkl} \text{ where } \epsilon_{ijkl} \sim N(0, \sigma^2) \tag{1.1}$$

The definitions of the factors are given in Table 1.1.

Table 1.1: Definitions of Factors

Effect	Definition	Given Variable Name	Level	Meaning
y	Number of Species	Richness		
α	Fertilization Treatment	Fert	0 L H	None Low High
β	Habitat Structure	Thatch	0 Th	Thatch Removed Thatch Present
γ	Month		1 2 3 4	17 June 27 June 12 July 12 August
δ	Block	Block	1 2 3 4 5 6 7	

The analysis of this set of data will serve as an example for applying various commands of R to a set of data.

A model is deemed to be a good model if the underlying theory makes sense, if there appears to be a conspicuous pattern in a plot of the data, if we can reject the hypothesis that the parameters of the model are zero, and if the assumptions of the model like the assumption that the residuals are indistinguishable from white noise are valid.

[1] See page 19

Various procedures provided by R will be used to answer the last three questions. Presumably, the theory that fertilizer, thatch, and time affect the size and vitality of a population of arthropods is based on expert knowledge. Ultimately the question is whether or not the set of experimental data supports the theory.

It seems that no matter how often some commands or methods are used, some of them are not easy to remember. The flexibility and the multitude of procedures which are available in R together with the propensity of researchers to write their own customized computer programs, sometimes makes an R program rather difficult to understand when someone else is trying to learn the programmer's logic. As a result, care should be exercised in documenting an R program not only for the benefit of someone else, but also for the programmer himself who might refer to the program later but has since forgotten the rationale his own programming methods.

Commands which appear enclosed in a box may be executed by highlighting and pasting them in an R session.

Chapter 2

Preparing the Data for Analysis

2.1 Creating an ASCII File from a Spreadsheet for R

There are several ways to bring the contents of a set of data into R . The simplest way is to use a set of data which has been stored in the ASCII format. In a Microsoft system, another name for an ASCII formatted file is a *.txt* file.

The fields of the data of the ASCII file need to be delimited by a punctuation mark like a comma, “," as shown in Table 2.1 or a vertical bar, “|" as shown in Table 2.2.

Table 2.1: Comma Delimited ASCII File

Richness,Fert,Thatch,Block
25,H,Th,1
31,H,O,1
22,O,Th,1
22,L,O,1
19,O,O,1

If the statistical computing package, *SAS*, is available on a computer, then it is possible to bring the contents of a SAS data file directly into R by means of the library *foreign*. [1]

There is not a simple way to bring a Microsoft Excel data file directly into R . Instead, each page of a Microsoft Excel data file is saved as a “.txt" or “.csv" file and a delimiter used as illustrated in Tables 2.1 and 2.2.

[1] See page 19 of (Fleming, 2004)

Table 2.2: Vertical Bar Delimited ASCII File

Richness|Fert|Thatch|Block
25|H|Th|1
31|H|O|1
22|O|Th|1
22|L|0|1
19|O|O|1

2.2 Bringing Data into R

The command to use to bring the contents of an ASCII data file into R is:

```
read.table()
```

For example,

```
data_17jun<-read.table(file="SpeciesRichnessNitrogen_17jun.txt",header=TRUE,sep="|")
```

The <− symbol is used in R to emphasize that R is written according to object oriented programming. What is called an *object* is written to the left of <−. A function is known as a procedure. The symbol <− is translated into English as *is assigned to*, that is, the content of the data found in the file
`SpeciesRichnessNitrogen_17jun.txt`
is assigned to the object `data_17jun`.

2.3 Manipulating Objects

There are different kinds of objects. There are numeric objects; there are character objects; there are matrix objects. In other words, an object has attributes which define its properties for the R programming language.

We will create the numeric object called a vector as follows:

```
x<-c(1,2,3,4)
```

The contents of x, can be shown by executing

```
x
```

A character object can be created as follows:

```
y<-c("a","b","c","d")
```

The quotation marks designate the letters as character values. Even numbers can be designated as characters as seen in:

```
z<-c("1","2","3","4")
```

To view the attributes of an object, the `str()` command is used:

```
str(x)
```

which produces:

```
num [1:4] 1 2 3 4
```

and `str(z)` which produces:

```
chr [1:4] "1" "2" "3" "4"
```

We can force a conversion of a numeric object to a character object by means of:

```
xx<-as.character(x)
str(xx)
```

Likewise, a character object may be converted to a numeric object by:

```
zz<-as.numeric(z)
str(zz)
```

A class of objects which is useful for producing an ANOVA table is: `factor()`

Consider the vector, `w<-c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4)`. It is a numeric object, but it is not suitable for use in producing a One-way ANOVA if it meant to be blocking variable with four levels. In order to create an object which can represent something like a block, the `factor()` is used. For example,

```
ww<-factor(w)
str(ww)
```

`ww` is now a factor with four levels.

To view the structure of an imported set of data, like `data_17jun`, the `str()` is used again:

```
str(data_17jun)
```

to produce the following listing of its contents:

```
'data.frame':   42 obs. of  14 variables:
 $ Richness: int  25 31 22 22 19 20 23 24 26 25 ...
 $ Fert    : Factor w/ 3 levels "H","L","O": 1 1 3 2 3 2 1 2 2 1 ...
```

```
 $ Fert.1  : int  3 3 1 2 1 2 3 2 2 3 ...
 $ Thatch  : Factor w/ 2 levels "O","Th": 2 1 2 1 1 2 2 1 2 1 ...
 $ Thatch.1: int  1 0 1 0 0 1 1 0 1 0 ...
 $ Block   : int  1 1 1 1 1 1 2 2 2 2 ...
 $ Sample  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X       : logi  NA NA NA NA NA NA ...
 $ No.Fert : int  22 19 21 16 22 15 17 20 14 15 ...
 $ Low     : int  22 20 24 26 20 18 21 26 17 21 ...
 $ High    : int  25 31 23 25 24 20 26 24 21 20 ...
 $ X.1     : logi  NA NA NA NA NA NA ...
 $ Thatch.2: int  25 22 20 23 26 16 22 24 20 21 ...
 $ OThatch : int  31 22 19 24 25 21 15 20 18 20 ...
```

We are informed that the object `data_17jun` is a data frame. A data frame is class of objects for storing a set of data in the R system. There are 14 variables, that is, there are 14 columns and 42 observations, or 42 rows. One can think of a data frame as an array of data 42 rows by 14 columns. The variable `Richness` occupies the first column in the array; `Fert` occupies the second column and so on. It is indicated that `Fert` and `Thatch` are already factors. But `Block` is not a factor; therefore, it will have to be converted later into a factor. Furthermore, there is no variable for `month` which will we eventually have to be created for creating an analysis of variance table.

Not all fields are needed for analyzing the data. The ones which we want are: `Richness`,`Fert`,`Thatch`, and `Block`. The others can be ignored. To produce a set of data with only the needed four fields, we will make a sub-set of it by selecting only those columns which correspond to the four fields.

Keeping in mind that `data_17jun` is like an array with 42 rows and 14 columns, we will use the conventions of matrix algebra to chose the columns which we need.

Suppose we enter: `data_17jun[1,]`, we will get the first row of `data_17jun`:

```
   Richness Fert Fert.1 Thatch Thatch.1 Block Sample  X
1        25    H      3     Th        1     1      1 NA
  No.Fert Low High X.1 Thatch.2 OThatch
       22  22   25  NA      25       31
```

The four fields of interest lie in the 1^{st}, 2^{nd}, 4^{th}, and 6^{th} columns. To extract the data from only those four columns we execute:

`data_17jun[,c(1,2,4,6)]` to produce:

```
Richness Fert Thatch Block
    25    H     Th     1
    31    H      O     1
    22    O     Th     1
    22    L      O     1
    19    O      O     1
```

If for some reason, we are only interested in rows 1, 2, 3, 4, and 5, then we would execute:

```
data_17jun[c(1,2,3,4,5),]
```

. Note that the notation `data_17jun[i,j]` identifies the (i, j) element of the array `data_17jun`.

We will save the sub-sets of data for each month by:

```
d17jun<-data_17jun[,c(1,2,3,4)]
d27jun<-data_27jun[,c(1,2,3,4)]
d12jul<-data_12jul[,c(1,2,3,4)]
d12aug<-data_12aug[,c(1,2,3,4)]
```

Next, we will concatenate the four sets of data to create one consolidated set of data by stacking so to speak one set of data on top of another by means of the `rbind()` command which concatenates by row.

```
data0<-rbind(data17jun,data27jun,data12jul,data12aug)
```

Fortunately, the number of columns in each of the four sets of data are the same and they have the same column names. If the structure of the sets are not compatible, then the `rbind()` will fail.

The object, `data0`, contains all the desired data from the original species Excel file except for `month`, although it is implicit in the name of each data file. We will create a vector called `month` which will be concatenated to `data0` which is now almost complete.

We note that each of the four sets of data which were extracted from the Excel file have the same number of rows, that is, 42. The experimental set of data is balanced. We can verify that by measuring the length of the any column of a set of data like the first column by executing:

```
length(d17jun[,1])
length(d27jun[,1])
length(d12jul[,1])
length(d12aug[,1])
```

To construct a vector which contains the number of a month repetitively 42 times, we will use the command, `rep()`:

```
rep(1,42)
```

To complete the process of creating a variable month, we will execute:

```
month1<-rep(1,length(d17jun[,1]))
month2<-rep(2,length(d27jun[,1]))
month3<-rep(3,length(d12jul[,1]))
month4<-rep(4,length(d12aug[,1]))
```

then concatenate them

```
month<-t(cbind(t(month1),t(month2),t(month3),t(month4)))
```

Finally, we concatenate `month` by columns to our main set of data, `data0` to get:

```
data<-cbind(data0,month)
str(data)
```

The `str()` command verifies that the set of data, `data`, has all the necessary information we need to construct an analysis of variance table.

Chapter 3

Statistical Analysis of the Data

3.1 Linear Model

The nature of the original set of species data as contained in the Microsoft Excel data file, suggests the following four factor fixed effects linear model:

$$y_i = \mu + \alpha_i + \beta_j + \gamma_k + \delta_l + \epsilon_{ijkl} \text{ where } \epsilon_{ijkl} \sim N(0, \sigma^2) \tag{3.1}$$

which was formulated in the introduction as equation (1.1).

According to the model, we are dealing with a five dimensional set of data. As such, it is impossible to make a picture of the data at once. Instead, we will look at two dimensional slices of the data, in order to discover whether there exists any conspicuous relationships between the variables or trends in the data.

To begin with, we will look at each variable individually as if there are not other factors present. The presence of other factors will muddy any patterns which might appear, but in the gross context, we might see some obvious trends.

Parenthetically, the R commands to produce Figure 3.3 are:

```
fig0<-paste("/home/georgetown_plot3.ps",sep="")
postscript(horizontal=FALSE, file=fig0)
par(cex.lab=1.5, cex.main=1.5,cex=1.25)
plot(factmonth,Richness,
main="Box Plots of Richness by factmonth",xlab="Month",ylab="Richness")
dev.off()
```

There appears in Figure 3.1 at an increase in Richness due to a high level of fertilizer over no fertilizer. We, therefore, should expect to reject the hypothesis that the richness of is the same across levels of fertilizer in the ANOVA table.

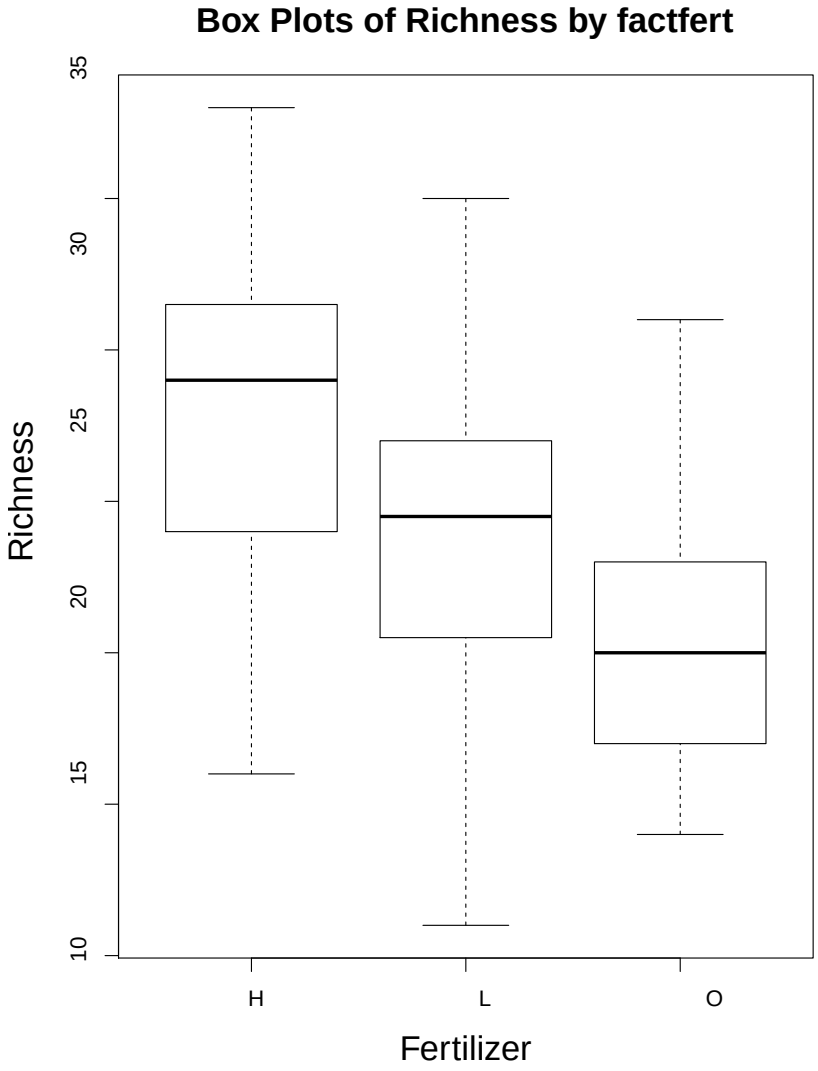

Figure 3.1: Box Plots of Richness by Levels of Fertilizer

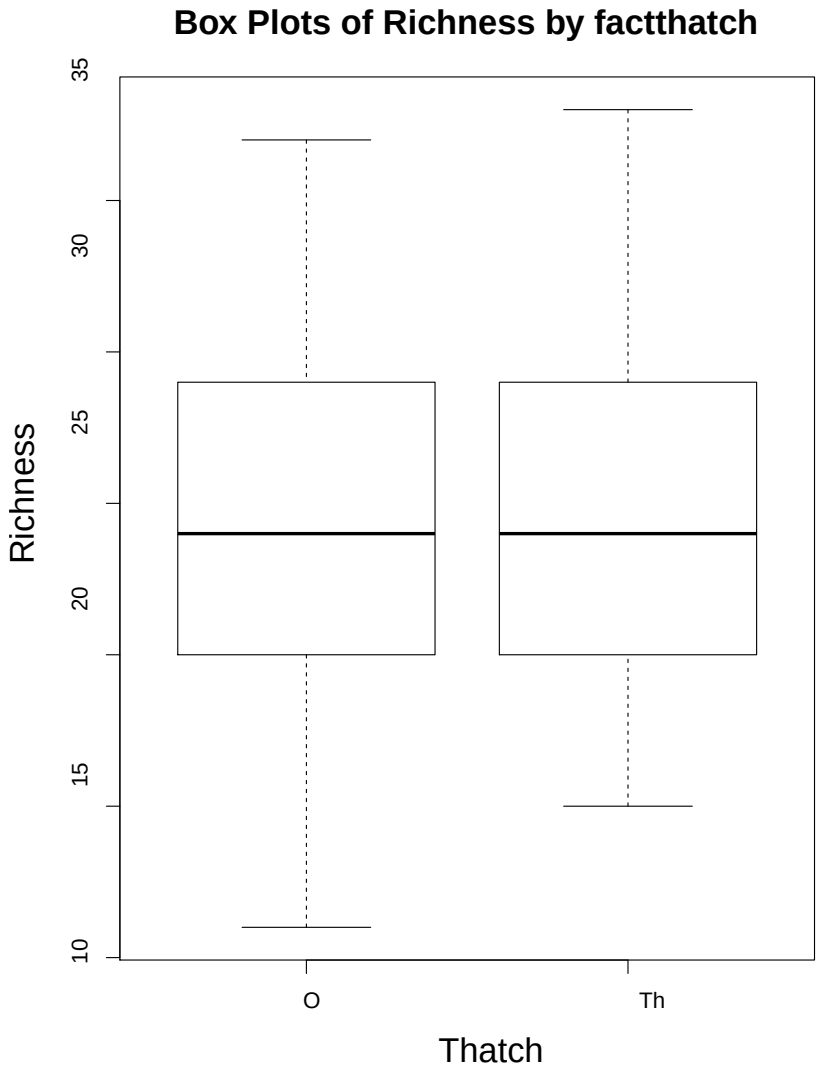

Figure 3.2: Box Plots of Richness by Levels of Thatching

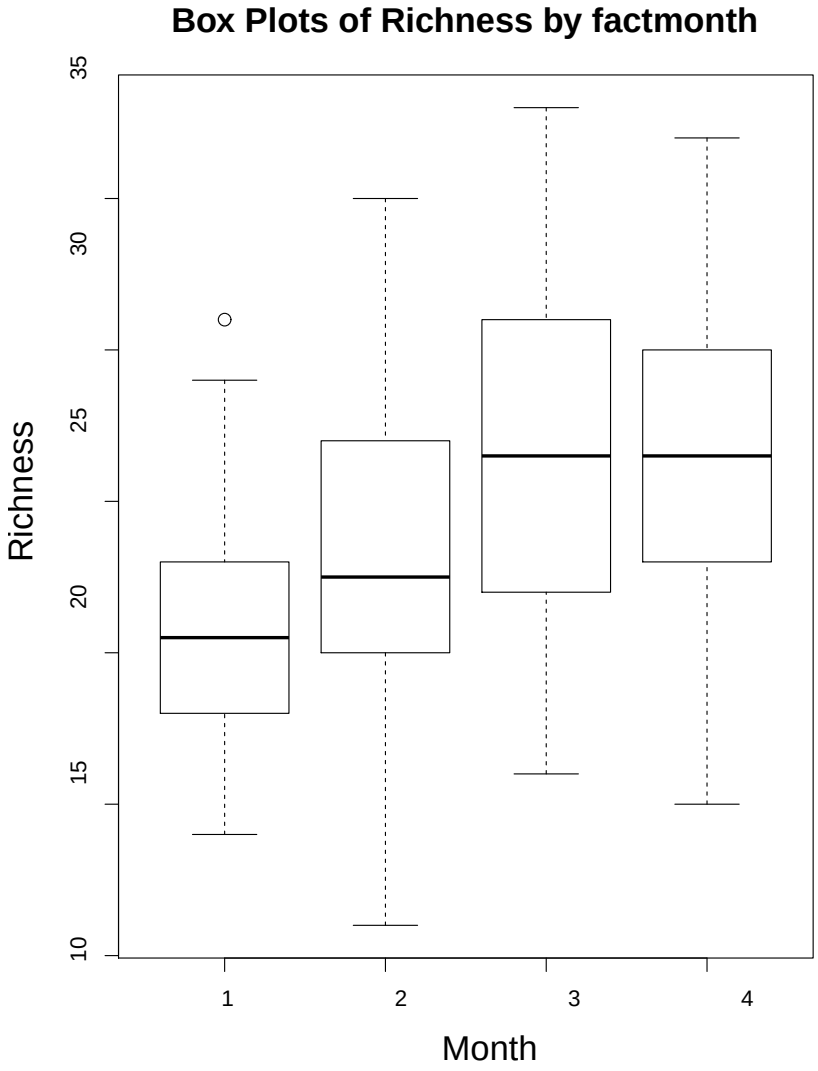

Figure 3.3: Box Plots of Richness by Levels of Month

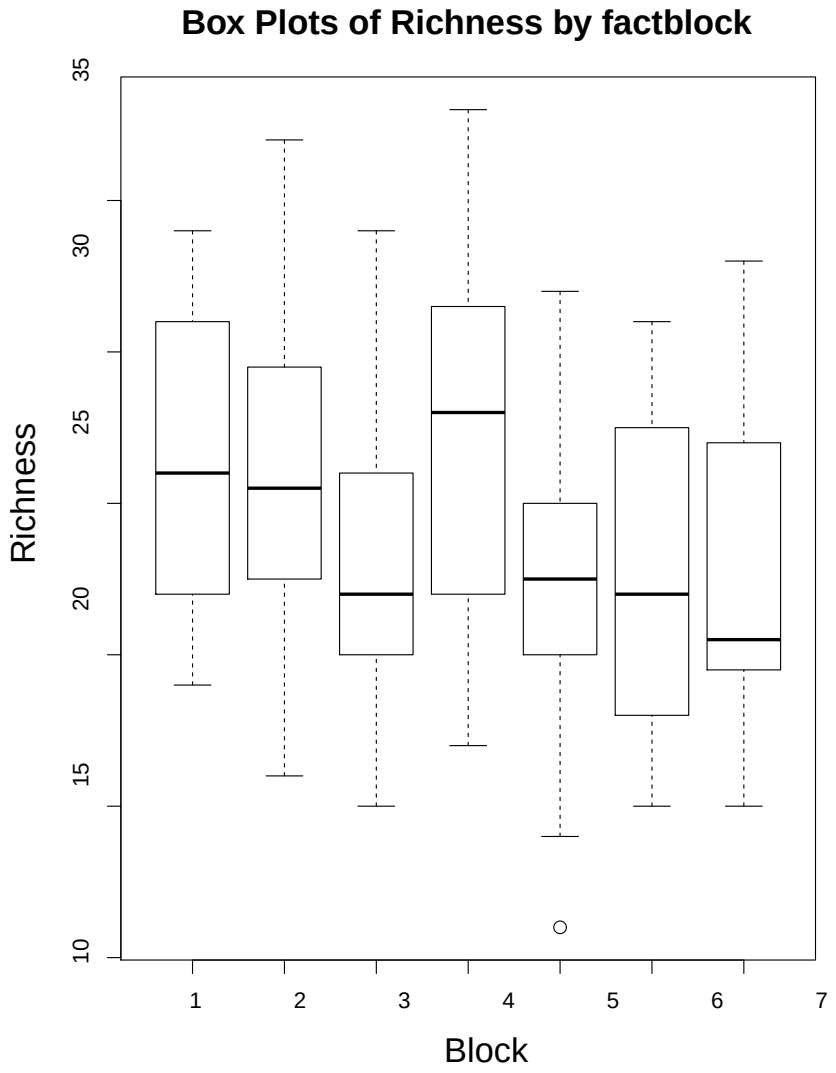

Figure 3.4: Box Plots of Richness by Levels of Blocks

Figure 3.2 suggests that removing the thatch does not affect the richness in species of the area.

Perhaps month has an affect on richness as suggested in Figure 3.3. We might see a corresponding significant effect in the ANOVA table.

A blocking factor is used to eliminate an effect. Presumably, `block` accounts for seven geographic areas. Whether there is a difference in richness between locations evidently is not a matter of concern whereas within a block the factors of fertilizer, thatching, and month are being examined to assess whether they can explain the response.

To make a picture like the ones in Figures 3.1, 3.2, 3.3, and 3.4, the variables, `Fert`, `Thatch`, `month`, `Block` must be objects which are classified as factors in R . Consequently, we will convert them into objects which have the factor attribute by executing the following:

```
factfert<-factor(data$Fert)
factthatch<-factor(data$Thatch)
factmonth<-factor(data$month)
factblock<-factor(data$Block)
```

As noted before, one way to extract a column of data is to use matrix notation as in: `data[,1]` Another way to do the same thing is to take advantage of the syntax peculiar to a data frame namely: `data$Richness` which will extract the same information as was done by `data[,1]`. The convenience of the previous method is that we do not have to know the particular column in the data frame where the Richness measurements are stored. Either method has its advantages.

The `plot(x,y)` command which was used to create Figure 3.1 and the other figures follows the convention that the x position corresponds to the x-axis and the y position corresponds to the y-axis.

It can be seen in the parenthetical comment above that a plot can be embellished with a title by means of the `main` option and labels to the axes can be add by the `xlab` and `ylab` options.

3.2 Analysis of Variance Table

An analysis of variance table can be produced by means of the `anova(lm())` command. In R a model such as the one given by equation 3.1 is written as:

```
Richness~ factfert+factthatch+factmonth+factblock
```

The `lm()` command is the command for producing least squares estimates of the parameters of a fixed effects linear model. For example,

```
lm(Richness~ factfert+factthatch+factmonth+factblock)
```

will produce useful information like `fitted.values` and `residuals`. Before we create an analysis of variance table, we will save the output of the `lm()` to an object.

```
lm.georgetown<-lm(Richness~ factfert+factthatch+factmonth+factblock)
```

```
str(lm.georgetown)
```
shows that `lm.georgetown` has a complex structure. In `lm.georgetown`, there is sufficient information to construct an analysis of variance table. It must be remembered that for a factorial design, the variables of the data must have the factor as an attribute.

To create an analysis of variance table, it is sufficient to execute:

```
anova(lm.georgetown)
```
or equivalently:

```
anova(lm(Richness~ factfert+factthatch+factmonth+factblock))
```

In either case, the following is produced:

```
> anova(lm.georgetown)
Analysis of Variance Table

Response: Richness
            Df  Sum Sq Mean Sq F value    Pr(>F)
    factfert     2 1620.33  810.17 70.3243 < 2.2e-16 ***
    factthatch   1   24.38   24.38  2.1163    0.1478
    factblock    6  685.24  114.21  9.9134 3.000e-09 ***
    factmonth    3  917.79  305.93 26.5553 6.467e-14 ***
    Residuals  155 1785.67   11.52
    ---
    Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the factor for Thatch has a p-value of .1478. We may assume that it does not make a significant contribution in explaining the response variable, `Richness`.

As was expected from inspecting Figures 3.1 and 3.3, fertilizer and month are significant factors in explaining `Richness`. Somewhat surprising is that `Block` is an important factor even though according to the box plots given in Figure 3.4, there does not appear to be a conspicuous difference between them. We need to keep in mind that a plot of the data like Figure 3.4 is only a two dimensional slice of a five dimensional set of data.

In order to asses the validity of the assumption of the model that $\epsilon_{ijkl} \sim N(0, \sigma^2)$, that is, the assumption that the residuals resemble white noise. The plot of residuals versus predicted values shown in Figure 3.5 shows a random pattern; therefore, we may consider that the assumption of the ϵ_{ijkl}'s is valid.

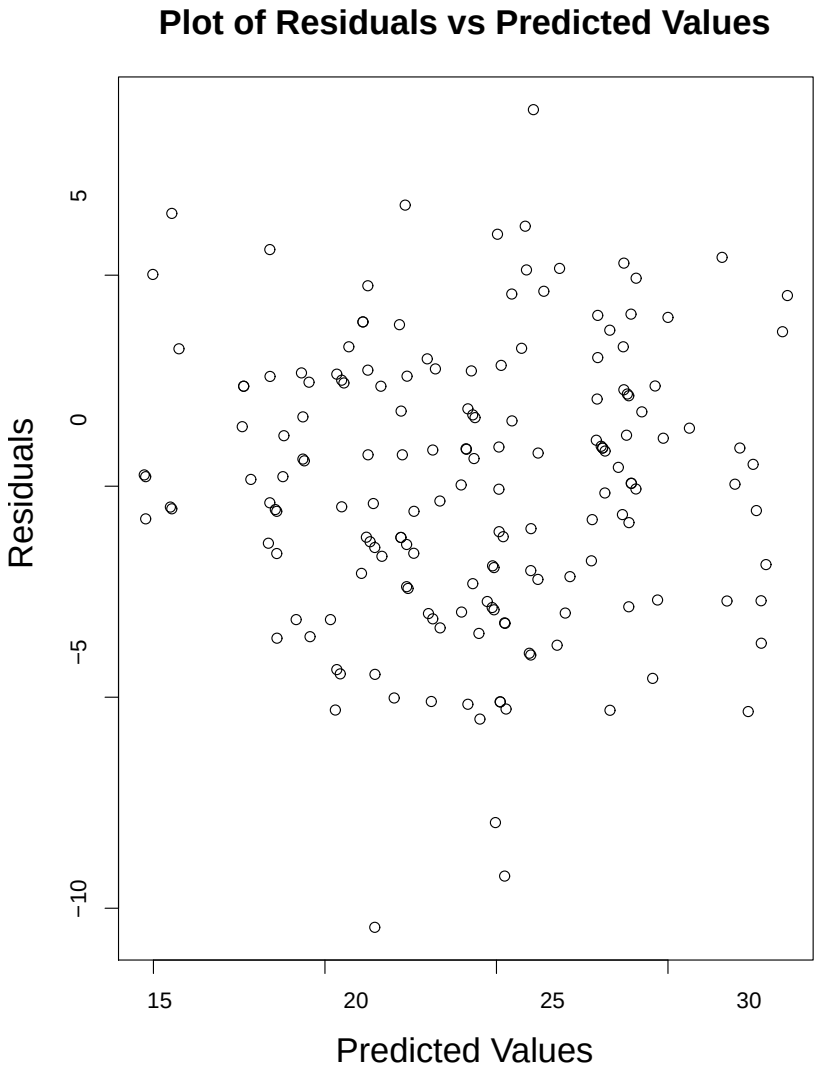

Figure 3.5: Diagnostic Plot for the Assumption of Normality of the Residuals

3.3 Advanced Techniques

Confidence intervals, of course, are of paramount importance in making statistical inferences. Because the current set of data lies in a five dimensional space, five dimensional confidence regions are impossible to draw. For the same reason when a series of box plots were made to examine the data in two dimensional slices, two dimensional confidence intervals are constructed for `Richness` according to `month` and `Thatch`. Though the effects of `Block` are confounded in the confidence intervals, the confidence interval provide a useful portrayal of the effects of `Fert`, `Thatch`, and `month` on `Richness`.

The R code which produced Figure 3.6 is given in the Appendix.

3.4 Useful but Simple Procedures

There are some useful elementary commands listed in Table 3.1.

Table 3.1: Useful Commands

Procedure	R Command
Histogram	hist()
Box Plot	boxplot()
Help	? as in ?hist
Fitted Line	abline(lsfit(x,y))
Vertical Line	abline(v=5) for example
Horizontal Line	abline(h=0) for example
Descriptive Statistics	summary(x)
Mean	mean(x)
Standard Deviation	sd(x)
Comment Symbol	#
Comment out a section of code	if(0){ }

Notation for arithmetic operations used in R is given on page 7 of (Fleming, 2004)

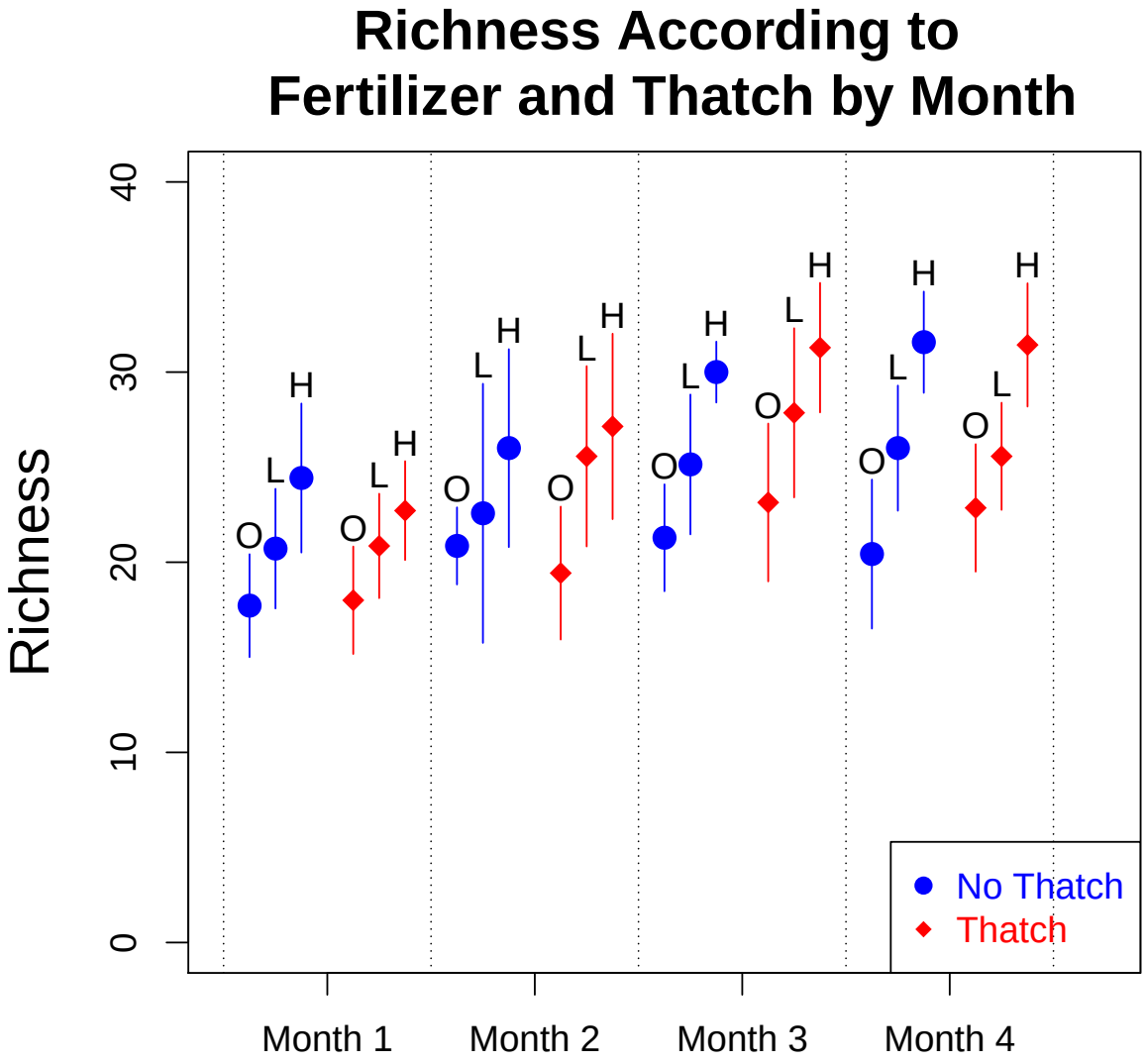

Figure 3.6: 95% Confidence Intervals of Richness by Fert, Thatch, and month.

Appendix A

Code Used to Produce Figure 3.6

```
# The delta, gamma, epsilon and gp are used for spacing the
# confidence intervals in the plot.

delta<-rep(0,length(data[,1]))
delta[data[,2]=="H"]<-.75
delta[data[,2]=="L"]<-.5
delta[data[,2]=="O"]<-.25
gamma<-rep(0,length(data[,1]))
gamma[data[,3]=="O"]<-0
gamma[data[,3]=="Th"]<-1
epsilon<-data[,4]
gp<-2*month-1+delta+gamma


# The function conf is used to construct the confidence
# interval (a,b).
# It will draw the vertical lines for the confidence with the
# lines() command.
# It will annotate the lines and give the lines colors.
# Note the use of the deparse(substitute()) combination.
# This will take the value of fert supplied by the user and
# pass it into the program in such as way that it can be
# automatically printed in the plot.
# See R Programming Tips cited in the bibliography
# for more information about useful but arcance
```

```
# deparse, parse, substitution, and expression commands.

conf<-function(fert,thatch,mon,...){
ind<-data$Fert==fert & data$month==mon & data$Thatch==thatch
x<-data[ind,1]  ##subset Richness=x
a<-mean(x)-sd(x)/sqrt(length(x))*qt(.975,length(x)-1)
b<-mean(x)+sd(x)/sqrt(length(x))*qt(.975,length(x)-1)
cv<-round(sd(x)/mean(x),3)*100
str.A<-paste("CV= ",cv," Thatch= ",thatch, " Fert= ",fert)
index0<-1:length(data[,1])
index<-index0[ind]
first<-deparse(substitute(fert))
str.B<-paste(fert,sep="")
text(gp[index[1]],b+1,str.B)
if(thatch=="Th"){
lines(c(gp[index[1]],gp[index[1]]),c(a,b),lwd=1,col="red")
points(gp[index[1]],mean(x),col="red",pch=18,cex=1.35)
}
if(thatch=="O"){
lines(c(gp[index[1]],gp[index[1]]),c(a,b),lwd=1,col="blue")
points(gp[index[1]],mean(x),col="blue",pch=19,cex=1.35)
}
}


# This set of commands will make the graph.
# Ignore the fig0 and postscript and dev.off() commands
# when producing the plot for display on the monitor.
# Use them, however, to make a paper
# copy of the plot.

fig0<-paste("/home/georgetown1.pdf",sep="")
pdf(file=fig0)

par(cex.lab=1.5, cex.main=1.5,cex=1.25)
plot(c(1,9.5),c(0,40),type="n",
    main="Richness According to \n Fertilizer and Thatch by Month",
    xlab="",ylab="Richness",xaxt="n")

ffert<-c("H","L","O")
```

```
fthatch<-c("Th","O")
for(j in 1:4){
for(i in 1:3){
for(k in 1:2){
conf(ffert[i],fthatch[k],j)
}
}
}

abline(v=1,lty=3)
abline(v=3,lty=3)
abline(v=5,lty=3)
abline(v=7,lty=3)
abline(v=9,lty=3)

axis(1,at=c(2,4,6,8),
    labels=c("Month 1","Month 2","Month 3","Month 4"),las=1)
legend("bottomright", legend = c("No Thatch","Thatch"),
    pch=c(19,18),text.col = c("blue","red"), col=c("blue","red"))

dev.off()
```

Bibliography

John M. Chambers. *Programming with Data A Guide to the S Language*. Springer-Verlag, New York, New York, 1998.

Peter Dalgaard. *Introductory Statistics with R*. Springer-Verlag, New York, New York, 2002.

Charles Fleming. *R Notes*. Unpublished, Washington, D.C., 2004.

Charles Fleming. *Simple Graphs in R*. Unpublished, Washington, D.C., 2005.

Charles Fleming. *R Programming Tips*. Unpublished, Washington, D.C., 2013.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer-Verlag, New York, New York, 1999.