Introduction to R

Gregg, Jay Sterling

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Introduction to

Jay Gregg

Course Description

This course introduces the statistical computing language R. The topics covered include: installing and opening R, installing and using a graphic user interface, accessing the R help and community, loading extension packages, basic calculations, basic programming (functions, logic, loops), working with matrices and data frames, computing summary statistics, and importing and exporting to excel and other graphical applications, and using and adapting existing code.

What is R?

- Statistical Programming Language
  - (built off of S)
  - wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, mapping, plotting, etc., etc…
- Computing Environment (flexible to do what you want; you can create new packages, etc.)
- **Free!** Source code is written in C and Fortran, and is freely available GNU license (Copyleft). Pre-compiled versions are also available.
- Community- active user support though the bulletin boards and posts. “How do I...?” questions are usually easily answered by a Google search.

Lesson 1

- Installing and opening R

Installing R

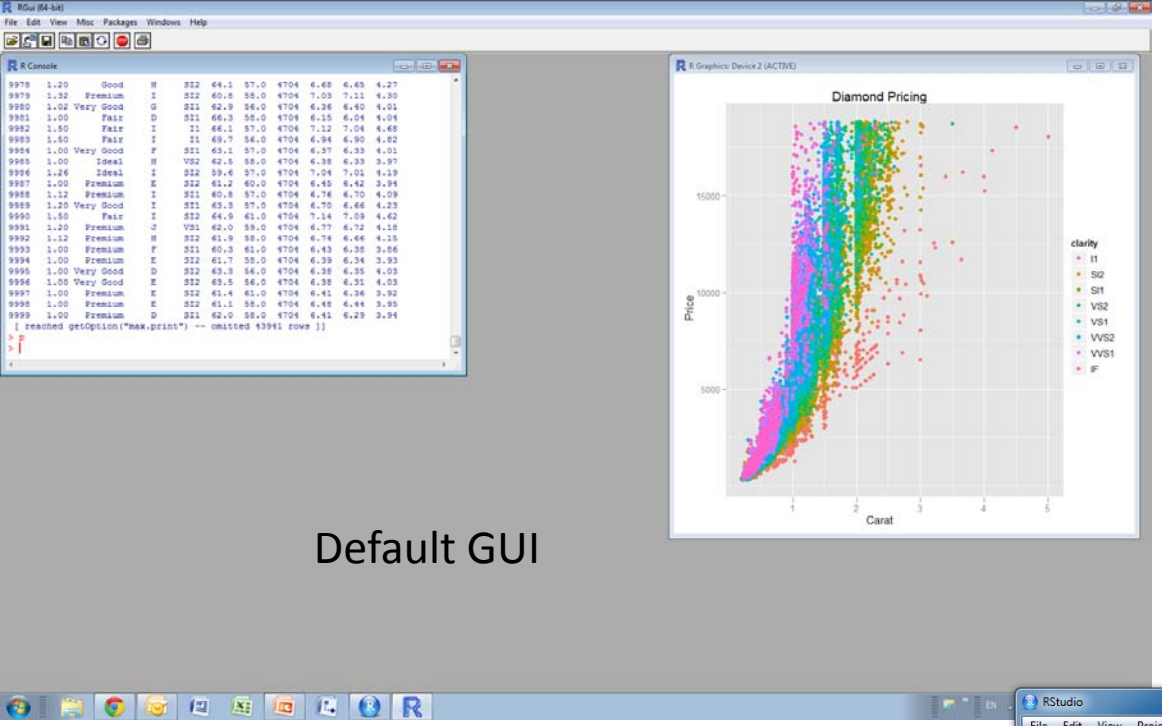
- Download

http://cran.r-project.org/bin/windows/base/

- Graphic User Interfaces

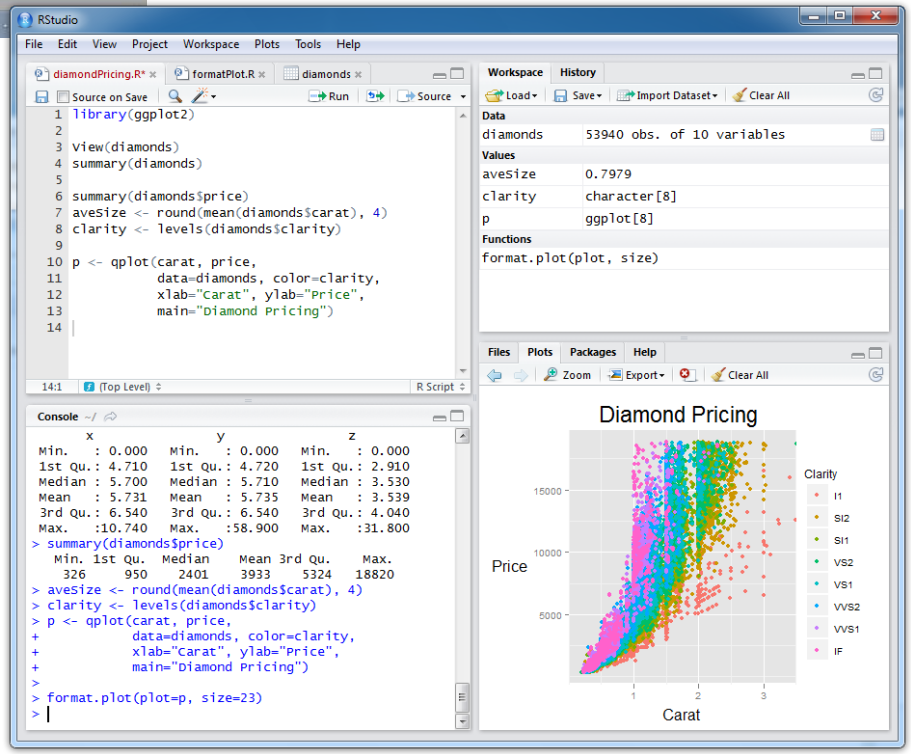
(one comes with the download, many others are available)

http://rstudio.org/



Default GUI

R Studio



Loading a package

- Users contribute to R by adding packages, which other users can then install.
- Loading a package
  - (Default GUI): Packages-> Load Packages -> (select mirror) -> select package name
  - (R Studio): Select Packages tab, select Install packages, type package name
- Update a package
  - (Default GUI): Packages-> Update Packages
  - (R Studio): Packages, Check for Updates

Resources

Wikibook that covers and reviews much of the basics in R

http://en.wikibooks.org/wiki/R_Programming

The R project homepage

http://www.r-project.org/

The R help forum (usually comes up on google)

https://stat.ethz.ch/mailman/listinfo/r-help

Exercise 1

- download and install R
- download and install R studio
- make sure the “datasets” package is installed
  - install another package, just for fun; e.g. ggplot2
- update packages

Lesson 2

- Basic R calculations

Very Basic R: Calculator

- `+ - / * ^ ()`

```
((2*33)/6)^2-1
```

- Exercise 2.1: what is the answer? Use R

- `<-` stores a variable

```
scale <- 10
equation <- ((2*33)/6)^2-1
my.lucky.number <- equation/scale
```

- Exercise 2.2: what is my.lucky.number? How do you see it?
- R respects order of operations

```
operations <- 6-4*3^2
```

- Exercise 2.3: what is the answer? How can you add parenthesis to make the answer 36? Hint: press up arrow to edit previous entries.

More very basic stuff

- Comments

```
# R will ignore anything after a pound sign
2-1*4 #I can also add comments to the end of the line
```

- Strings can also be stored as a variable; need to use “ “

```
# saving a string variable
name<- "Mrs. Robinson"
```

- Help- use the ?function. It will open a browser window which describes the function with an example

```
?sqrt
```

Exercise 2.4: Copy and paste the sqrt example into R

Sequences

- Sequences, done with the ":", or with seq command

```
s <- 1:10
t <- 10:1
u <- s*t
v <- seq(0, 20, by=pi)
w <- seq(0, 20, length.out=12)
alpha <- letters[1:26]
```

- Sort; to reorganize the numbers in increasing or decreasing order.

```
sort(u)          #increasing order
sort(u, TRUE)    #decreasing order
```

combining and pasting

- Combine: important function used to combine elements

```
c(1:10, 2*(30:50))
c("apples","bananas","carrots")
```

- Paste: merges elements

```
paste(letters[1:5], 1:5, sep="")
```

Exercise 2.5: Create a vector called v that contains the powers of 3 from 3^0 to 3^5, the codes 1a through 5e, and then the strings, “hello” and “world”. Sort it.

answer

```
v <- c(3^(0:5),
  paste(1:5,letters[1:5],sep=""),
  "hello", "world")
v
```

```
 [1] "1"     "3"     "9"     "27"    "81"    "243"   "1a"    "2b"    "3c"
[10] "4d"    "5e"    "hello" "world"
```

```
sort(v)
```

```
 [1] "1"     "1a"    "243"   "27"    "2b"    "3"     "3c"    "4d"    "5e"
[10] "81"    "9"     "hello" "world"
```

Lesson 3

- Functions, Logic, Loops

Functions

- A function is a command that takes arguments, and then does operations
- e.g. `sqrt(), round(), abs(), plot()` etc.
- Many functions pre-exist within R, others are added with library(package)
  - these functions can be viewed by typing just the name and no parentheses or arguments
- Defining a user function

```
my.function <- function(x, y)
    {
    (x+y)^2
    }
```

- Exercise 3.1 Create a function that divides the absolute difference between x and y by z.

  Calculate it with 2.4, 5.3, and 2. What happens if you calculate it with 2.4, 5.3 and 0?

answer

```
my.function <- function(x, y, z)
  {
  abs(x-y)/z
  }
```

```
> my.function(2.4,5.3,2)
[1] 1.45
> my.function(2.4,5.3,0)
[1] Inf
>
```

Logic

- is

```
is.double(), is.logical(), is.vector(), is.matrix(), is.data.frame()...
```

Exercise 3.1: What are the answers to these?

```
x <- 5
is.logical(x)
is.logical(is.logical(x))
```

- If
- if (cond1=true) { cmd1 } else { cmd2 }

```
y <- 7
if (x==y) { print(1) } else { print(2) }
```

ifelse(test, true value, false value)

```
ifelse(x<5 | x>8, x, 0)
ifelse(x>=5 & x<=8, x, 1)
```

Exercise 3.2: Adapt the function from Exercise 2.5 to return 0 if z = 0.

Exercise 2.5 Create a function that divides the absolute difference between x and y by z. Calculate it with 2.4, 5.3, and 2. What happens if you calculate it with 2.4, 5.3 and 0?

answer

```
my.function <- function(x, y, z)
  {
  ifelse(z==0, 0, abs(x-y)/z)
  }
```

```
> my.function(2.4,5.3,2)
[1] 1.45
> my.function(2.4,5.3,0)
[1] 0
>
```

Loops

- for: used for running a command through a set of variables

for(variable in sequence) { statements }

```
for(i in 1:20) {
   print(i*2)
   print(letters[i])
   }
```

- while: runs until the condition is false

while(condition) {statements}

```
z <- 0
   while(z < 15){
   z <- z + 2
   print(z)
   }
```

- Exercise 3.3: Write a loop that calculates:

$$\sum_{i=1}^{10000}\left(\frac{1}{i}-\frac{1}{i+1}\right)$$

answer

```
x <-0
for(i in 1:10000){
   a <- (1/i)-(1/(i+1))
   x <- x + a
}
```

or

```
x <- 0
i <- 1
while(i <= 10000){
   a <- (1/i)-(1/(i+1))
   x <- x + a
   i <- i + 1
}

> x
[1]      0.9999
```

Lesson 4

- matrices, indexes, data frames
- applying simple summary statistics

Matrices

- How to create a matrix (not all switches are necessary all the time):

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

```
aa <- matrix(data=1:12, nrow=3, ncol=4, byrow=FALSE, dimnames=NULL)
bb <- matrix(1:12, 3, 4)
cc <- matrix(1:12, 3, 4, byrow=TRUE)
dd <- matrix(nrow=3, ncol=4)
ee <- matrix()
ff <- matrix(nrow=0, ncol=4)

gg <- bb+cc #addition of matrices
hh <- bb*cc #multiplication of the individual elements
ii <- t(cc)  #matrix transpose
jj <- bb %*% ii #matrix multiplication
dim(jj)
diag(jj)
kk <- rbind(bb, cc) #bind by rows
ll <- cbind(bb, cc) #bind by columns
```

Indices and Subscripting

- matrix[row, column]

```
aa <- matrix(data=1:12, nrow=3, ncol=4)
aa[1, 3]
aa[2:3, 1:2]
aa[2:3, 2:1]
aa[3, ] <- show this entire
aa[-3,] <- omit this row
aa[1, 3] <- 0  #replace an element
colnames(aa)<-c("Amy", "Beth", "Cathy",
  "Debbie")
rownames(aa)<-paste("Trial",1:3)
aa<-as.data.frame(aa)  #convert to data frame
aa$Cathy #now we can index by column name!
```

Simple Statistics

```
aa1 <- matrix(data=1:12, nrow=3, ncol=4)
aa1[1, 3] <- 0  #replace an element
colnames(aa1)<-c("Amy", "Beth", "Cathy",
  "Debbie")
rownames(aa1)<-paste("Trial",1:3)

mean(aa)
mean(aa1)
sum(aa)
sum(aa1)
var(as.vector(aa1[1,]))
summary(aa)
```

Data Frames

- Data frames give more flexibility to elements and structure than a matrix. In a matrix, all elements have to be the same type (all numbers, all strings). In a data frame, all columns have to be the same type.

```
CO2 #built in data frame on CO2 uptake in plants

> is.data.frame(CO2)
[1] TRUE
> is.matrix(CO2)
[1] FALSE
> nrow(CO2)
[1] 84
> ncol(CO2)
[1] 5
> dim(CO2)
[1] 84  5
Summary(CO2)
```

Exercise 4

- Create a matrix from the CO2 data frame that contains all the nonchilled uptake results inthe first column, and chilled Uptake results in the second column.

Name the matrix columns, and find the mean of each.

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2
7	Qn1	Quebec	nonchilled	1000	39.7
8	Qn2	Quebec	nonchilled	95	13.6
9	Qn2	Quebec	nonchilled	175	27.3
10	Qn2	Quebec	nonchilled	250	37.1
11	Qn2	Quebec	nonchilled	350	41.8
12	Qn2	Quebec	nonchilled	500	40.6
13	Qn2	Quebec	nonchilled	675	41.4
14	Qn2	Quebec	nonchilled	1000	44.3
15	Qn3	Quebec	nonchilled	95	16.2
16	Qn3	Quebec	nonchilled	175	32.4
17	Qn3	Quebec	nonchilled	250	40.3
18	Qn3	Quebec	nonchilled	350	42.1
19	Qn3	Quebec	nonchilled	500	42.9
20	Qn3	Quebec	nonchilled	675	43.9
21	Qn3	Quebec	nonchilled	1000	45.5
22	Qc1	Quebec	chilled	95	14.2
23	Qc1	Quebec	chilled	175	24.1
24	Qc1	Quebec	chilled	250	30.3
25	Qc1	Quebec	chilled	350	34.6
26	Qc1	Quebec	chilled	500	32.5
27	Qc1	Quebec	chilled	675	35.4
28	Qc1	Quebec	chilled	1000	38.7
29	Qc2	Quebec	chilled	95	9.3
30	Qc2	Quebec	chilled	175	27.3
31	Qc2	Quebec	chilled	250	35.0
32	Qc2	Quebec	chilled	350	38.8
33	Qc2	Quebec	chilled	500	38.6
34	Qc2	Quebec	chilled	675	37.5
35	Qc2	Quebec	chilled	1000	42.4
36	Qc3	Quebec	chilled	95	15.1
37	Qc3	Quebec	chilled	175	21.0
38	Qc3	Quebec	chilled	250	38.1
39	Qc3	Quebec	chilled	350	34.0
40	Qc3	Quebec	chilled	500	38.9
41	Qc3	Quebec	chilled	675	39.6
42	Qc3	Quebec	chilled	1000	41.4
43	Mn1	Mississippi	nonchilled	95	10.6
44	Mn1	Mississippi	nonchilled	175	19.2
45	Mn1	Mississippi	nonchilled	250	26.2
46	Mn1	Mississippi	nonchilled	350	30.0
47	Mn1	Mississippi	nonchilled	500	30.9
48	Mn1	Mississippi	nonchilled	675	32.4
49	Mn1	Mississippi	nonchilled	1000	35.5
50	Mn2	Mississippi	nonchilled	95	12.0
51	Mn2	Mississippi	nonchilled	175	22.0
52	Mn2	Mississippi	nonchilled	250	30.6
53	Mn2	Mississippi	nonchilled	350	31.8
54	Mn2	Mississippi	nonchilled	500	32.4
55	Mn2	Mississippi	nonchilled	675	31.1
56	Mn2	Mississippi	nonchilled	1000	31.5
57	Mn3	Mississippi	nonchilled	95	11.3
58	Mn3	Mississippi	nonchilled	175	19.4
59	Mn3	Mississippi	nonchilled	250	25.8
60	Mn3	Mississippi	nonchilled	350	27.9
61	Mn3	Mississippi	nonchilled	500	28.5
62	Mn3	Mississippi	nonchilled	675	28.1
63	Mn3	Mississippi	nonchilled	1000	27.8
64	Mc1	Mississippi	chilled	95	10.5
65	Mc1	Mississippi	chilled	175	14.9
66	Mc1	Mississippi	chilled	250	18.1
67	Mc1	Mississippi	chilled	350	18.9
68	Mc1	Mississippi	chilled	500	19.5
69	Mc1	Mississippi	chilled	675	22.2
70	Mc1	Mississippi	chilled	1000	21.9
71	Mc2	Mississippi	chilled	95	7.7
72	Mc2	Mississippi	chilled	175	11.4
73	Mc2	Mississippi	chilled	250	12.3
74	Mc2	Mississippi	chilled	350	13.0
75	Mc2	Mississippi	chilled	500	12.5
76	Mc2	Mississippi	chilled	675	13.7
77	Mc2	Mississippi	chilled	1000	14.4
78	Mc3	Mississippi	chilled	95	10.6
79	Mc3	Mississippi	chilled	175	18.0
80	Mc3	Mississippi	chilled	250	17.9
81	Mc3	Mississippi	chilled	350	17.9
82	Mc3	Mississippi	chilled	500	17.9
83	Mc3	Mississippi	chilled	675	18.9
84	Mc3	Mississippi	chilled	1000	19.9

nonchilled

chilled

nonchilled

chilled

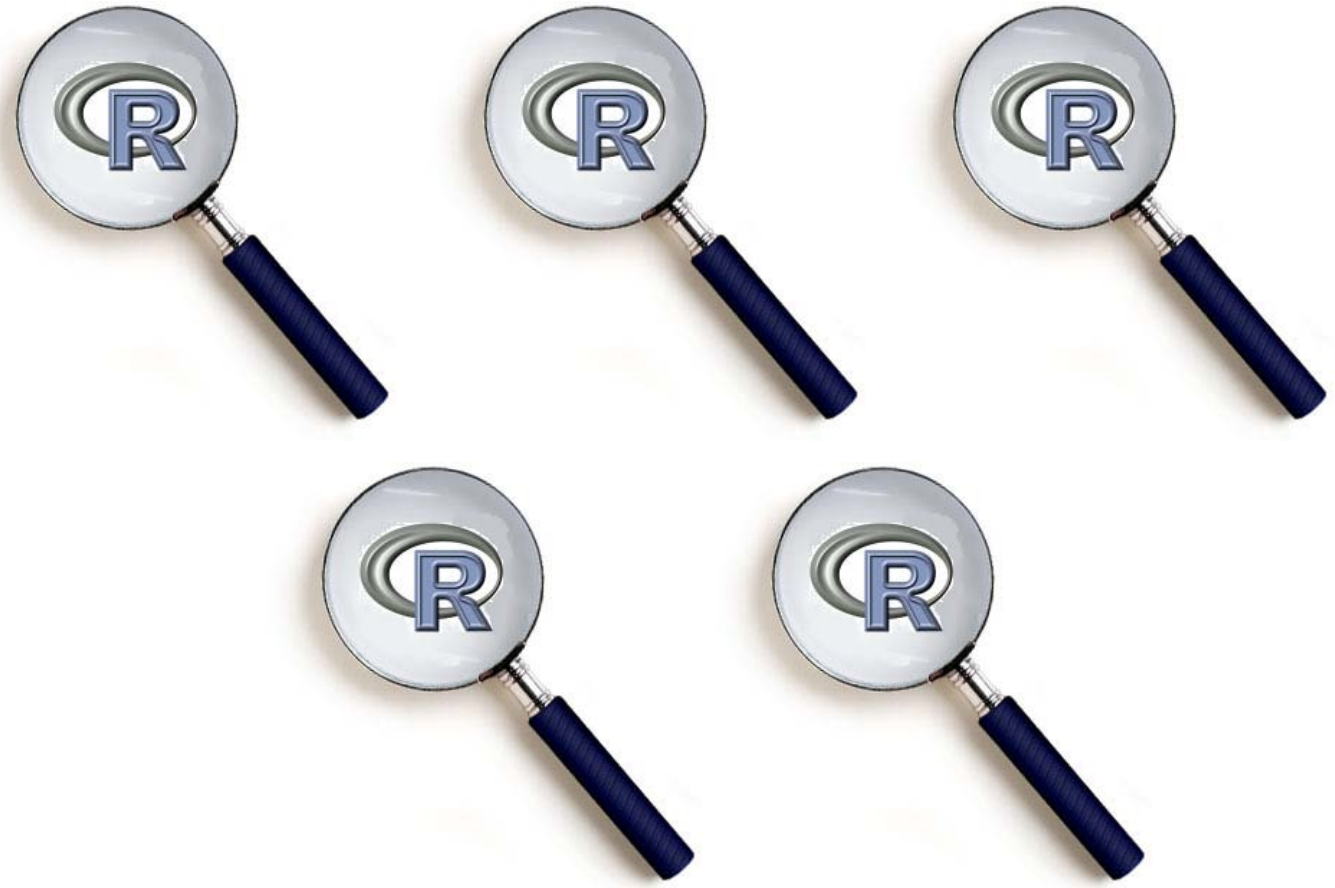
answer

```
chilledCO2 <- matrix(data =
  c(CO2$uptake[c(1:21,43:63,22:42,64:8
  4), ncol=2)
colnames(chilledCO2) <-
  c("nonchilled", "chilled")


> mean(chilledCO2[,1])
[1] 30.64286
> mean(chilledCO2[,2])
[1] 23.78333
```

Lesson 5

- Importing and Exporting to Excel



Setup- things to do beforehand

- It is easiest if data are saved as a .csv file from Excel. Make sure the setting is correct in Excel with the decimal separator “.”

- In R, if you are going to do a lot of importing and exporting of data, plots, etc., it is best to set up a working directory. Create a work folder somewhere convenient, and tell R where to find it.

```
setwd("C:\\Users\\jsgr\\Desktop\\R Project")
```

importing and exporting a csv

- Reading in a csv

```
my.data<-read.csv("data.csv", header=TRUE)
```

- can also use read.csv2 for “,” decimal separators, or read.delim for tab, etc...

```
?read.table
```

- exporting:

```
write.csv(my.data, file="data.csv", quote = FALSE,
   eol = "\n", na = "")
```

exporting a graphic

- You can export a plot to pdf, png, jpeg, bmp or tiff by

```
pdf("filename.pdf")
png("filename.png")
jpeg("filename.jpg")
bmp("filename.bmp")
tiff("filename.tiff")
```

prior to the plot command, and then after the plot command:

```
dev.off()
```

- You can also create a multi-page pdf of plots

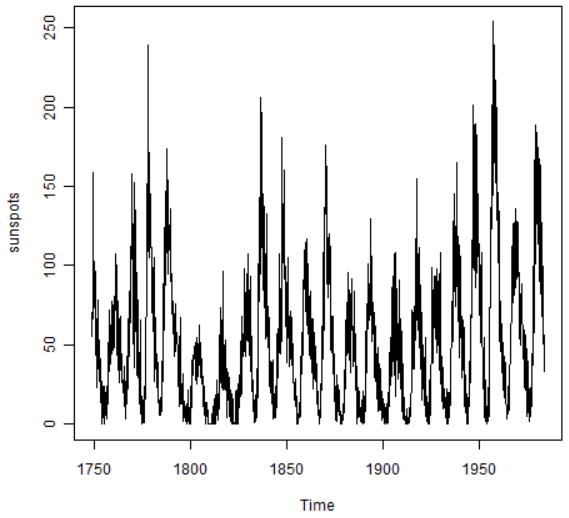
```
pdf("Graphs.pdf", onefile = TRUE, paper="a4r",
   width=11.1, height=7.5)
plot(Titanic)
plot(women)
dev.off()
```

Exercise 5

- Use the dataset sunspots in R; print it to the screen
- convert sunspots to a matrix, then a data frame keeping the same data structure
- name the columns and rows
- output the data as a csv file to your working folder
- output a plot of the original dataset as a png

answer

```
sunspots
sunspots.mtx <- matrix(data=sunspots, ncol=12,
  byrow=TRUE)
sunspots.df <- as.data.frame(sunspots.mtx)
colnames(sunspots.df) <-  c("Jan", "Feb",
  "Mar", "Apr", "May", "Jun", "Jul", "Aug",
  "Sep", "Oct", "Nov", "Dec")
rownames(sunspots.df) <- 1749:1983
write.csv(sunspots.df, file="sunspots.csv",
  quote = FALSE, eol = "\r")
png("sunspots.png")
plot(sunspots)
dev.off()
```



Lesson 6

- Reading and adapting existing code

Adapting Code

- It is often easiest to find existing examples of code at the end of the help files, or online, and adapt these to fit your needs

- Always work in a text editor or graphic user interface, so you can experiment and make adjustments quickly

Lattice Plot Example: Example of what can be done in R

```
#See if you can follow what is happening in this example

library(lattice)

#create a colorset

colorSet <- matrix(ncol=1, nrow=0)

for(a in 1:240)
    {
    #the rgb function takes a value from 0 to 255 and converts it to hexidecimal color codes
    newColor <- rgb(round(15+(a/240)*240), round(120-(a/240)*120), round(255-(a/240)*240),
    maxColorValue=255)
    colorSet <- rbind(colorSet,newColor)
    }

#Wire Frame Plot

x.scale <- 1:12
y.scale <- 1959:1997
g <- expand.grid(x = x.scale, y = y.scale)
g$z <- co2
title <- expression("Mauna Loa CO"[2]*" Concentration")
wireframe(z ~ x * y, g, screen = list(z = 15, x = -75), drape = TRUE, aspect = c(2.5,1),
    colorkey = FALSE, scales=list(arrows=FALSE, col="#000000"), xlab=list(label="Month", font=2),
    ylab=list(label="Year",font=2), zlab=list(label="ppm", font=2), zlim=c(310,370),
    main=list(label=title, cex=2), cuts=length(colorSet), col.regions=colorSet)
```

Mauna Loa CO_2 Concentration

Exercise 6

- adapt the given code to make a lattice plot for the sunspots time series.
- If you are feeling ambitious, try to adjust the colors and orientation of the figure

answer:

```
library(lattice)

#create a colorset

colorSet <- matrix(ncol=1, nrow=0)

for(a in 1:240)
    {
    #the rgb function takes a value from 0 to 255 and converts it to hexidecimal color codes
    newColor <- rgb(round(135+(a/240)*120), round(15+(a/240)*240), round(255-(a/240)*240),
    maxColorValue=255)
    colorSet <- rbind(colorSet,newColor)
    }

#Wire Frame Plot

x.scale <- 1:12
y.scale <- 1749:1983
g <- expand.grid(x = x.scale, y = y.scale)
g$z <- sunspots
title <- expression("Sunspots")
wireframe(z ~ x * y, g, screen = list(z = 130, x = -50), drape = TRUE, aspect = c(2.5,1),
    colorkey = FALSE, scales=list(arrows=FALSE, col="#000000"), xlab=list(label="Month", font=2),
    ylab=list(label="Year",font=2), zlab=list(label="mean monthly sunspots", rot=90, font=2),
    zlim=c(0,260), main=list(label=title, cex=2), cuts=length(colorSet), col.regions=colorSet)
```

Sunspots

Final thoughts

- It is unlikely that anybody knows all of R; new packages are always being added, we use what we need to solve a specific problem
- Every command/ function you know is another tool, and you are only restricted by the number of tools you know.
- Use it or lose it
- Recycle and adapt code is that is already out there
- R is quite powerful, but it depends on the user knowing what (s)he is doing with it
- There is almost always more than one way to do something
  - And there is almost always a more efficient and elegant way to do what you have just coded
- NAs are a headache

Enjoy!