

22S:166  
Lab session 2  
Elementary data analysis in R

Sep. 11, 2009

## 1 Setting up

Use NX Client to get onto the Linux network. Use `ssh` to log into one of the machines in the 346 lab.

Choose a subdirectory to use for this R session. Go to the “Datasets” section of the course web page. Read the file called `BaP.info`, and download the data file called `BaP.txt` into the subdirectory you wish to use. Then call up R from that subdirectory.

## 2 Using on-line help in R

You can get help on any R function by typing `help ( <command name> )`. For example, to get help on the `library` function, enter

```
> help(library)
```

To use “prettier” help in separate window, enter

```
> help.start()
```

After a while, a web window will come up. You can click on choices for help documents in that window. If you type `help ( <command name> )` in the command window later in the R session, the results will appear in the web window.

Another function that is useful in learning R and getting help is `apropos`. It finds all functions whose *names* contain the character string given as the argument. Only packages that have been loaded into memory are searched. For example,

```
> apropos("rank")
```

Yet another useful function is `help.search`. It looks for any function in any installed package that mentions the search term in its help. The name of the package containing the function is in parentheses. For example,

```
> help.search("rank")
```

Help files with alias or concept or title matching rank using regular expression matching:

```
rank(base)           Sample Ranks
SignRank(stats)     Distribution of the Wilcoxon Signed Rank
                    Statistic
```

## 3 Using built-in R datasets

Use the `search` function to determine which R packages are loaded automatically when you bring R up.

```
> search()
[1] ".GlobalEnv"      "package:stats"    "package:graphics"
[4] "package:grDevices" "package:utils"    "package:datasets"
[7] "package:methods" "Autoloads"        "package:base"
```

Notice that the `datasets`, `stats`, and `graphics` packages are listed. This means you can access any of the built-in R datasets and any functions in the `stats` and `graphics` packages without having to load the packages yourself. To get names and descriptions of the datasets, enter

```
> help(package=datasets)
```

To display the Orange dataset (it is a data frame), just enter

```
> Orange
```

To get descriptive information on the dataset named Orange, enter

```
> help(Orange, package=datasets)
```

Notice the example code at the end of the help. Copy the line that begins `coplot` into the command window and execute it. This is an example of a very powerful plotting function in the `graphics` package. Type in the necessary command to get help on this function.

## 4 Reading in an external file

To read the BaP data into a data frame called `BaP`, enter

```
BaP <- read.table("BaP.txt", header = T)
```

If the data file was in a different subdirectory, we would have to enter its full path name.

We can also read it in straight from its Internet location:

```
BaP <- read.table("http://www.stat.uiowa.edu/ftp/kcowles/datasets/BaP.txt", header = T)
```

To get a scatterplot with the indoor measurements on the X axis and the outdoor measurements on the Y axis, enter:

```
plot(BaP$indoor, BaP$outdoor)
```

If we want to refer to the variables `indoor` and `outdoor` without referencing the dataframe, we need to `attach` the dataframe.

```
attach(BaP)
```

Note that this makes `BaP` the second item in the search list.

```
search()
```

Now we could just enter `plot(indoor, outdoor)`.

When we are done using the data frame, we should `detach` it to free up memory.

```
detach(BaP)
```

Use `search()` to verify that the `BaP` data frame has been detached.

## 5 Using the library that accompanies the assigned readings

Verzani wrote an add-on R library to accompany his book *Simple R*. In the online version of the book, he refers to this library as the `Simple` library. However, the name has been changed and it is now called the `UsingR` library. I have installed it under R on our Linux system. This library does not load automatically when you call up R. We have to load it using the `library` function. We will load this library and use some of the datasets in it to learn some data analysis functions in R.

```
> .libPaths( c(.libPaths(), '/group/statsoft/Rlibs64'))
> library(UsingR) # load library
> help(package="UsingR") # get information on all functions and datasets in library
> help(corn) # get information on one particular dataset in library
```

```
> corn # display dataframe
  New Standard
1  110      102
2  103       86
3   95       88
4   94       75
5   87       89
6  119      102
7  102      105
8   93       88
9   87       83
10  98       89
11 105      100
12 117      110
```

## 6 Elementary data analysis in R

We want to determine whether the mean yield of New corn is larger than that of Standard corn.

Begin with exploratory analysis of data – both numeric and graphical.

```
> summary(corn)
      New      Standard
Min.   : 87.00  Min.   : 75.00
1st Qu.: 93.75  1st Qu.: 87.50
Median :100.00  Median : 89.00
Mean   :100.83  Mean   : 93.08
3rd Qu.:106.25  3rd Qu.:102.00
Max.   :119.00  Max.   :110.00

> boxplot(corn) # side-by-side boxplots of all variables in dataset

> attach(corn) # make the individual variables available by name

> hist( New, probability = T) # make a probability histogram
> lines( density( New ), col = "red" )

> hist( Standard, probability = T)
> lines( density( Standard ), col = "red" )

# Further assessment of whether samples might come from normal populations

> qqnorm( Standard ); qqline( Standard, col = 2)
> qqnorm( New ); qqline( New, col = 2)
```

These results are equivocal as to whether the normality assumption holds. A t-test might be appropriate, but a nonparametric test probably is safer. We will do both and see whether they agree.

```
> t.test( New, Standard, alternative = "greater" )
```

Welch Two Sample t-test

```
data: New and Standard
t = 1.8061, df = 21.996, p-value = 0.04231
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.3815088      Inf
sample estimates:
mean of x mean of y
100.83333  93.08333
```

```
# Note that the confidence interval is one-sided, to match the requested one-sided alternative
# hypothesis.
```

```
> help(wilcox.test) # get help on nonparametric test for equality of centers
```

```
> wilcox.test(New, Standard, alternative = "greater", conf.int = T)
```

Wilcoxon rank sum test with continuity correction

```
data: New and Standard
W = 99, p-value = 0.06264
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
-3.221176e-05      Inf
sample estimates:
difference in location
      7.00006
```

Warning messages:

```
1: In wilcox.test.default(New, Standard, alternative = "greater", conf.int = T) :
  cannot compute exact p-value with ties
2: In wilcox.test.default(New, Standard, alternative = "greater", conf.int = T) :
  cannot compute exact confidence intervals with ties
```

# Results are equivocal. Since there are tied values in the data, p-value isn't exact. However, # the evidenc is not compelling of a difference in average yields between New and Standard corn.

```
> detach(corn)
```

Let's also investigate estimation and testing for proportions. Researchers wanted to study the effectiveness of wrist guards in protecting in-line skaters from wrist injury. They interviewed 161 people who came into hospital emergency rooms with injuries form in-line skating. The table summaries relevant findings.

Wrist guard	Wrist injury	
	Y	N
Y	6	47
N	45	63

Let's get a point estimate and confidence interval for the population proportion who will sustain wrist injuries in the population of in-line skaters who are injured while wearing a wrist guard.

```
> binom.test( 6,53)
```

Exact binomial test

```
data: 6 and 53
number of successes = 6, number of trials = 53, p-value = 5.805e-09
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.04269639 0.23028992
sample estimates:
probability of success
```

0.1132075

NOw in the population of in-line skaters who are injured while not wearing a wrist guard.

```
> binom.test( 45, 108)
```

Exact binomial test

```
data: 45 and 108
number of successes = 45, number of trials = 108, p-value = 0.1014
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.3225425 0.5154792
sample estimates:
probability of success
      0.4166667
```

Now let's test for the equality of the two population proportions, and get a confidence interval for the difference.

```
> wearers <- c(53,108)
> wristinjury <- c(6,45)
>
> proptest(wristinjury, wearers)
Error: could not find function "proptest"
> prop.test(wristinjury, wearers)
```

2-sample test for equality of proportions with continuity correction

```
data: wristinjury out of wearers
X-squared = 13.7577, df = 1, p-value = 0.0002080
alternative hypothesis: two.sided
95 percent confidence interval:
-0.4437038 -0.1632144
sample estimates:
 prop 1  prop 2
0.1132075 0.4166667
```

If we want a one-sided test and a one-sided confidence interval (for the alternative hypothesis that the proportion of wrist injuries is smaller in the population of wrist-guard wearers), we would do the following:

```
> prop.test(wristinjury, wearers, alternative="less")
```

2-sample test for equality of proportions with continuity correction

```
data: wristinjury out of wearers
```

```
X-squared = 13.7577, df = 1, p-value = 0.0001040
alternative hypothesis: less
95 percent confidence interval:
-1.000000 -0.183501
sample estimates:
  prop 1   prop 2
0.1132075 0.4166667
```

## 7 Writing R programs

The safest way to write an R program in the Unix or Linux environment, is to open a terminal window (separate from the one in which you are using R) and from there, use a text editor to write the program code and save it as a plain text file with the extension `.R`. You may wish to develop the code interactively in R and copy the lines into the program file in the text editor. You can then run the program in R using the `source` function, for example, if the program was named “`dostuff.R`,”

```
> source("dostuff.R")
```

## 8 Writing R functions

Similarly, to write an R function that contains more than a few lines of code, write the code in a text editor outside of R and save it in a text file. Then you can use the `source` function in R to read in the function and assign it to an R object.

We will write a function that does the following:

1. Accepts a vector as an argument
2. If the vector is numeric and has at least 20 distinct values, returns the quantiles of the values
3. Otherwise returns a tabulation of the values

Use your text editor to create a plain text file named “`process.R`” that contains the following code.

```
function(x)
{
  if ( is.character(x) | is.factor(x)
      | (is.numeric(x) & length( unique(x) ) < 20 ))
      table(x)
  else
      quantile(x)
}
```

Now read it into an R function by entering the following in R:

```
> process <- source("process.R")$value
```

If R reports syntax errors, correct them in the text file; then try `source` again.

Test your function by creating different types of vectors and using them as arguments. For example,

```
myvect <- c(rep("a",5), rep("w",2), "n", "q", "v")
process(myvect)
```

```
myvect <- rnorm(10)
process(myvect)
```

```
myvect <- rnorm(500)
process(myvect)
```