Install and Use Software R on a Windows PC

Mai Zhou
University of Kentucky

R is a statistical software with a Gnu copyright (like Linux). It is a freely available "clone" of Splus–a popular, award winning commercial software. Almost everything you learn with R can be applied to Splus (and vise versa). Splus is one of the two softwares [the other is SAS] authorized by the Center for Drug Evaluation and Research of FDA. The R version this note refers to is called rw1040, it runs under Win95 and all later versions, (Win98, Win2000, WinNT4.0, WinMe and WinXP). Newer version of R may be available when you read this (R 1.5.0 will be available in April 29, 2002). All the procedures should still work.

1 Detailed steps of install R on your Windows PC

This section is for people who want to install R on their home PCs. R is already installed on the PCs in B&E MicroLab, and on all the PCs in MS network (those in the Statistics/Math Department, for example). If you are using R at B&E Lab or MS domain, skip to section 2. R is also available on the 'new' t-machines in the MS network.

1. Download file from
   http://cran.us.r-project.org/bin/windows/base/
2. You need just one file: `R-2.x.x-win32.exe` After downloading, find and click the file `R-2.x.x-win32.exe` and follow the instructions.
3. If all goes well you will find R in Start - All programs menu
4. Quit R by click [File:Exit] OR simply type `q()` inside R. If you want to save what you did in R, you should click yes (or type y) to the dialog window asking if you want to [Save workspace image?].

In the basic statistical analysis course, the base package R is enough. There are also over 200 add-on packages to R. `http://cran.us.r-project.org/bin/windows/contrib/***.zip` If you are taking the *bootstrap* course then you need `boot.zip` and may be `bootstrap.zip`.

Another possibility is that if you can get a CDROM contain R, then you install R from the CD.

Now suppose you have successfully installed R on your Windows PC.

2 First Tutorial of R

The following are the R command for you to reproduce. (I omitted the R output) (you can omit my comments–those after the `#` sign). Notice R command is case sensitive (like Unix, unlike DOS).

```
> # this is R prompt, meaning R is ready to take your input
> data1 <- c(21, 25, 25, 18, 44, 20, 25, 15, 19, 20, 30)
> # The above generates a dataset called data1 inside R.
> ls()  #[el-s, not one-s] see what is there (do you see data1?)
> summary(data1)   # give the info on the dataset data1
> stem(data1)      # give a stem and leaf plot of data1
> hist(data1)      # plot a histogram
> hist(data1, 5)
> hist(data1, c(2.5,17.5,32.5,47.5,62.5,77.5))
> data2 <- rnorm(100) # generate 100 normal random numbers and put in data2
> hist(data2)
> data()           # Now look at built-in data sets, see what is available
> data(sunspots)   # attach the data set sunspots (this takes memory space)
> sunspots         # display the data on screen
> plot(sunspots)
> # Put your name and ID on the picture by
> text(locator(), "your name and ID")
> # and then use mouse to click the appropriate spot on the graph.
> rm(data1)        # delete data1
> demo(graphics)   # see some demo. Hit many returns until you are done.
> q()              # quit R.
```

A common mistake for new R users is to create a data set with a name that has already been used by R. For example, `c, t`, etc. At first this do not seem to give an error message but will cause bad confusion later. So use longer and unique names, and before creating the data or function, make sure there is no such object already taking this name:

```
> mydata
Error: Object "mydata" not found
>
```

This means you are safe to use the name `mydata`.

A neat feature of R that can save a lot of typing is the command line editing and recall. Use arrow keys to navigate in the command history list, much like Unix k-shell.

2.1 Printing and Saving Graph to a File

There is a pull-down menu, and printing icon for both the graphics window and console window. To print, simply click File:Print, or click on the icon. Try not to resize your graphics window.

You can save your file by use the pull-down menu. For example File:Save as:Postscript.

You can also copy the graph to Excel, or Word and print from there. Suppose you made some nice graph in R. Make the R graph window the active one, and click File:copy-to-clip-board:as bitmap. Open MS Excel, click Edit:Paste Special (Paste may also work). Now your graph is in Excel. You can resize it, print preview it, and print it.

You should print both the console (commands) and graphs (if any) when doing homework.

2.2 Memory

R has its own memory management. Use `gc()` to see how much memory is available. (`gc` stands for Garbage Collection). Most likely you do not need to worry about memory problems since R version 1.2.0.

2.3 Color Plots and Math Symbols in the Plots

R can produce color plots. Use `plot(x,col="red")` etc. to get a color plot. Use `colors()` to see over 600 names of different available colors.

You can use `point(x,col="white")` (or background color) to erase the points.

Other functions that can take the `col=` option include `lines` etc.

Another feature of R over Splus is that R have TEX-like ability to produce math symbols and Greek letters in the plot. A simple example:

```
> plot(rnorm(100),type="n")
> text(20, 0, expression(theta(mu)), col="blue")
> text(40, 0, expression(theta^{"2+x"}), col="blue")
```

3 Second Tutorial, more features

R is a software that you can learn well by doing, so here are some more examples:

To see a demonstration do	`demo()` or `demo(graphics)`
To delete a vector or data set named x, do	`rm(x)`
To see what is there, do	`ls()`
To randomly select 9 integers from 20 to 40 WITH NO REPEATS do	`sample(20:40, 9, replace=FALSE)`
To randomly allocate 18 subjects (labeled from 1 to 18) into 3 groups, do	`sample(1:3, 18, replace=TRUE)`
To randomly allocate 18 subjects (labeled from 1 to 18) into 3 groups, and require 6 in each group, do	`sample(rep(1:3, 6), 18, replace=FALSE)`
To find more about the function sample, do	`> ?sample`

(assume you have help pages installed)

To find the (sample) mean of the sample stored in data1 do	`mean(data1)`
To find the sample standard deviation of data1 do	`sd(data1)`
To find the sample variance of data1 do	`var(data1)`
To find the (sample) median of data1 do	`median(data1)`
To find the sample range of data1 do	`range(data1)`
To generate a box-whisker plot of data1 do	`boxplot(data1)`
To compute the 5 factorial, 5! , do	`prod(1:5)`
The number of ways to choose 5 out of 20, do	`choose(20, 5)`

The `sd, choose` are for R, not S(plus) 3.4 but one can easily add those to S(plus).

To generate 100 standard normal random numbers,	`rnorm(100)`
To convert the continuous data into discrete,	`table(cut(rnorm(100), 8))`

3.1 Input data set

To execute several pre-typed commands stored in an ASCII file mycode, do `source("mycode")` or better use mouse to click File:sourceRcode. for example the file mycode may look like this (2 lines only):

```
data2 <- c(20,25,33,45,67)
data2 <- c(data2,77,98,32)
```

(this creates a vector `data2` with 8 numbers, `c` is for combine), or it can contain several dozen lines of commands.

For pre-existing larger data sets, it is better to have the data in an ASCII file outside R and then read the data into R by:

```
> data3 <- read.table("C:/STAT/test.dat", header=TURE)
```

For larger data sets (over 100 MB) you may also want to use `scan()` for reading data. Use `write()` to write a data set.

If your data is in the MS Excel file, xxx.xls, then first save it in the CSV format, and then inside R use `read.csv()` similar to `read.table()`. See help page for other formats.

3.2 Use R as probability tables

To find the probability of $X \leq 5$ for a binomial N=25, p=0.3 random variable, do

```
> pbinom(5, 25, 0.3)
```

To find the probability of X=5 for a binomial N=25, p=0.3 random variable, do

```
> dbinom(5, 25, 0.3)
```

To find the probability of $5 \leq X$ for a binomial N=25, p=0.3 distribution, do

```
> 1-pbinom(4,25,0.3)      #NOTICE the 4 not 5!
```

To find the probability of $5 \leq X \leq 10$ for the same binomial distribution, do

```
> 1-( 1-pbinom(10, 25, 0.3)+pbinom(4, 25, 0.3) ) # or simply
> pbinom(10, 25, 0.3) - pbinorm(4, 25, 0.3)      # or
> sum( dbinom(5:10, 25, 0.3) )
```

To generate a binomial probabilities table for N=25, p=0.3, do

```
> dbinom(0:25, 25, 0.3)
```

Due to lack of spacing, the output could be hard to read. If you want a better looking printout, you could do

```
> print(dbinom(0:25, 25, 0.3), print.gap=2)
```

Or an even better printout, do

```
> print( cbind( 0:25, dbinom(0:25,25,0.3) ), print.gap=3)
==================================================
```

To find the probability of a STANDARD normal random variable, Z, less then 1 (say), do `> pnorm(1)`

When the normal random variable is NOT standard, then you need to specify mean and sd. For example to find the same probability but for a normal random variable, X, with mean -2, sd 3, do

```
> pnorm(1, mean=-2, sd=3)
```

Continue with the example, if I want the probability of X between 2 and 3 do

```
> pnorm(3, mean=-2, sd=3) - pnorm(2, mean=-2, sd=3)
```

There is a function called `dnorm()` but we do not need it here.

****

[added Feb 99] For hyper-geometric distribution, just use `dhyper()` and `phyper()`. Example:

```
-------------
| f11 |     | 19    If this is the 2x2 table and we are referring
-------------       the table entry inside by f11
|     |     | 11
-------------
  14    16
```

The probability of observing an entry of f11=6 in such table, assume equal proportion, is

```
> dhyper(6, 14, 16, 19)
```

The function `phyper()` give the cumulative probability.

For chi-square distribution we have, similarly, for cumulative probability up to 3.84

```
> pchisq(3.84, df=1, ncp=0)
```

There is also a `> dchisq(1, df=2)` function but we do not need it in this course.

4 Some exercise problems and more hints

If you have trouble doing the problem, look for solutions in the two tutorials before. All the functions you need are there or right after the problems here.

Problem 1: Suppose the entire population of USA is evenly split (YES/NO) on one issue. And we are using simple random sampling to conduct a poll (binomial model). Use R to find out the following probability:

(a). The chance of getting 6 or more YES when poll 10 person.

(b). The chance of getting 60 or more YES when poll 100 person.

(c). The chance of getting 600 or more YES when poll 1000 person.

(d). The chance of getting 1200 or more YES when poll 2000 person.

(e). The chance of getting at least 300 but no more then 600 YES when poll 1500 person. (include 300 and 600)

So, if your poll resulted 1200 YES based on 2000 random interviews and a politician believed in "50-50 evenly split" try to dismiss your poll by saying the poll "cannot be trusted because the randomness and the fact you only interviewed a tiny fraction of the whole population, anything can happen". How do you counter? [You should be able to do all the calculations needed with `pbinom()` .]

Problem 2: Simulate the experiment of rolling one fair die 500 times and plot result.

```
> result1 <- sample(1:6, 500, replace=TURE)
> hist(result1, breaks=0:6, probability=TRUE)
```

Problem 3: Simulate the experiment of rolling another fair die 500 times.

```
> result2 <- sample(1:6, 500, replace=TRUE)
```

Now add the two experiments together, this is the experiment of roll two dice and count the total dots (Monopoly), also 500 times.

```
> result3 <- result1 + result2
> hist(result3, breaks=1:12)
```

To get a small normal probability table use R code

`> pnorm(seq(-3.5, 3.5, 0.5))` compare with the output in the book.

Problem 4: Find the probability that X is between 3.085 and 4.226 where X is a normal distributed random variable with mu=2, sd=4 (sd is also called sigma)

Problem 5: First use R function `pnorm()` and by trial and error, i.e. try different values for g so that you get $P(X < g) = 0.07$ where X is a normal distributed random variable with mu=30, sigma=4.

Secondly, try to make use of a new R function called `qnorm()`. See hint below.

==

To find what value g that satisfy $P(X < g) = 0.3$ where X is normal with mu=3, sigma=2 do

```
> qnorm(0.3, 3, 2)
```

To find what value g that satisfy $P(X > g) = 0.3$ where X is normal with mu=3, sigma=2 do

```
> qnorm(0.7, 3, 2)
```

because $P(X > g) = 0.3$ means $P(X < g) = 0.7$ for X normal ("=" do not matter here).

==

Assume the sample is contained in x, a vector of length n. The following is helpful in computing the 95% student t confidence interval for mu

```
> n <- length(x)
> mean(x) + qt(0.975, df=n-1)*sd(x)/sqrt(n)   #this is upper confidence limit
> mean(x) - qt(0.975, df=n-1)*sd(x)/sqrt(n)   #this is lower confidence limit
```

The following is to computing the 93% Z confidence Interval for mu (assume sigma known)

```
> mean(x) + qnorm(0.965)*sigma/sqrt(n)
> mean(x) - qnorm(0.965)*sigma/sqrt(n)
```

==

The sample success rate is phat= # of successes/n The following is to compute the 93% confidence interval for p. (population proportion)

```
> phat <- 536/1308          ## i.e. observed 536 successes in 1308 trials
> phat + qnorm(0.965)*sqrt(phat*(1-phat))/sqrt(1308)
> phat - qnorm(0.965)*sqrt(phat*(1-phat))/sqrt(1308)
```

==

Problem 6: Sears claims "in over half American homes people rely on Kenmore appliances". Suppose in 1200 randomly selected households, 589 have Kenmore appliances. The hypothesis to be tested are Ho: p=0.5 Ha: $p < 0.5$

(a) Compute P-value.

(b) If we use a significance level of 0.03, what is your conclusion?

==

One sample t-test: First put your data in a vector called data6 (a possible example: `> data6<-c(33.9, 52.4,48.6, 53.5, 43.8)` remember quiz?)

To test (actually just to get P-value) Ho: mu=46.5 Ha: mu< 46.5 (less than) based on the data in data6, you do

```
> t.test(data6, alternative="less", mu=46.5)
```

The other possible alternatives are: "greater" and "two.sided" In addition to P-value this also computes a 95% t-confidence interval.

==

The function `t.test` can also do two-sample t-tests. Suppose you have two (not paired) samples stored in vectors called xbefore and xafter then do

```
> t.test(x=xbefore, y=xafter, alternative="less", mu=0, paired=FALSE)
```

If the data are actually paired, then change `paired=FALSE` to `paired=TRUE`.

==

One sample proportion test: Suppose your sample has 600 success in 1000 trials. To test if the success probability is equal to 0.5 (actually just to get P-value) Ho: p=0.5 Ha: p not= 0.5 (not equal) you do

```
> prop.test(600, n=1000, p=0.5, alternative="two.sided")
```

The other possible alternatives are: "less" and "greater" In addition to P-value this also computes a 95% confidence interval.

==

For further info on the use of `t.test( )` and `prop.test( )` please see the online help page that comes with R. (`> ?t.test`)

==

If you are going to use the add-on package(s) in the library, then do (for complete list of available packages for library, see below)

```
> library()        # to see what packages are available. Assume you have ctest.
> library(ctest)   # load the library ctest (which include binom.test)
> library(help=ctest) # see what new functions ctest packeage include
> binom.test(600, n=1000, p=0.5, alternative="two.sided")
```

If you cannot find the package you need, then you need to install it first.

`binom.test` is similar to `prop.test`, the difference is that `prop.test` uses approximate computation, `binom.test` uses exact computation but `prop.test` can be used to test several samples. It also gave an (approximate) confidence interval.

Day one:

1. R (or Splus) is a vector language. Every operation is best to be vectorized.

```
 +, -, *, /, ^, ... etc.  Use subscript [ ] to extract part of a vector.
```

2. R does its job by way of functions. There are over 3000 functions in R.

`exp(x), log(x), sqrt(x), q(), c(x,y), ...` etc. vector version.

3. R is easily extendible. [day two: write your own functions]

4. R is good for graphics. modify your graph until you are satisfied.

5. R is not very well suited for HUGE data sets: eg. more than 1 gig [but may improve in the future]

6. Some packages in R is the same as in Splus. [eg. survival]

7. In R you can pick and chose your random number generator.

Day two:

R supports a function `de()` which is a spreadsheet for data-editing. For people familiar with MS Excel/Lotus 123 this is easy to edit data.

Simulation and bootstrapping using R. the function boot(), jackknife() from bootstrap package. An example with exponential tilting for test hypothesis.

Use of boot(), bcanon() on bio-equivalent example.

Use of exp.tilt() bootstrap() on testing 2 sample problem.

Survival analysis in R, package survival5.

Empirical likelihood with hazard and right censored data.

Write your own functions: examples (avoid `for` loops)

You can edit you own function outside R and then source it into R,

or You can edit your function junk inside R by

```
> junk <- edit(mean)
```

here mean is an existing function that you want to modify to make a new function called junk. Other editor is also possible. If you want to edit `junk` and still call it `junk`, do

```
> fix(junk)
```

If your function is really short, you could even do without any editor, just say

```
> junk <- function(x) { x/(x+5) }
```

inside R.

Here is another example (that generates fake data from given sample size, mean and standard deviation):

```
fakedata <- function(size, xbar, sdd) {
            if(sdd<=0) stop("sdd must > 0")
            if(!is.numeric(xbar)) stop("xbar must be a real number")
            fake1 <- rnorm(size)
            fake2 <- fake1 - mean(fake1)
            fake2 * (sdd/sd(fake2)) + xbar
}
```

This function comes in handy when you want to do a t-test but only are given the sample size, sample mean and standard deviation (not the entire data).

```
> mydata <- fakedata(50,11.8,0.6)
> t.test(mydata, mu=12, alternative="less")
```

Some graphics in addition to `demo(graphics)`:

Suppose `x, resid` are two vectors of same length. We can plot them `plot(x,y)` . If we see some outliers, we would like to find out which point is it. Use `identify(x,y, n=5)` and mouse to click on the points, it will plot the index of the points. If you do not say `n=5` then stop the process by use menu.

```
> x <- rnorm(200)
> hist(x, probability=TRUE)
> lines(density(x))
```

For further references, read "An Introduction to R" that is also available for download at R site.