BIOS 662 Fall 2018

Introduction to R

Based on a set of notes by

Wei Sun, Ph.D.

Department of Biostatistics
University of North Carolina at Chapel Hill

What is R?

A language and software environment for statistical computing and graphics.

- R is free!
- It is open-source and involves many developers.
- The R system is developing rapidly.
- Straightforward simple calculations and analysis.
- Allows low level control for some tasks.
- Extensive graphical abilities.
- Sometimes R is slow...

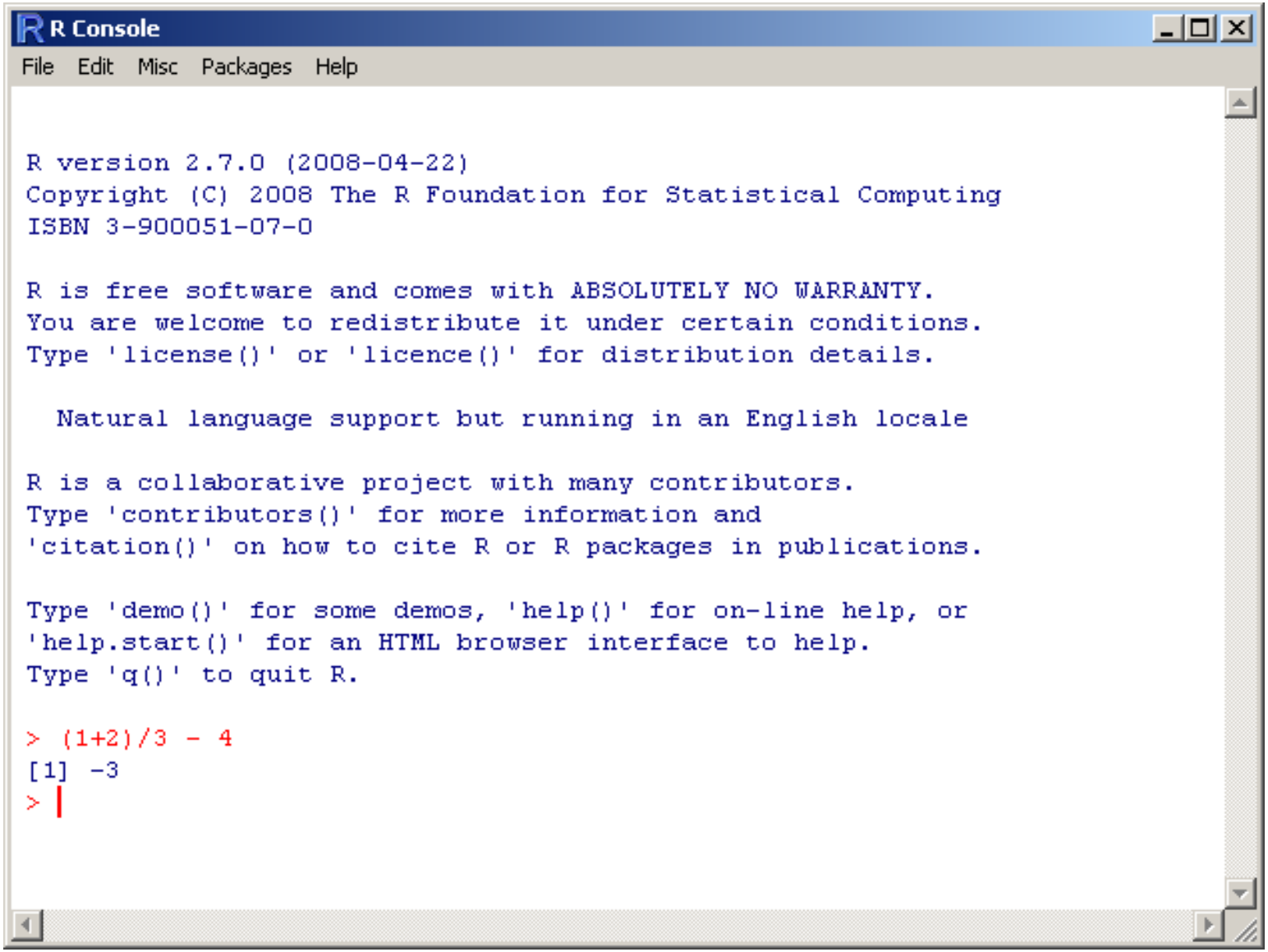


Figure 1: R graphical user interface (Windows)

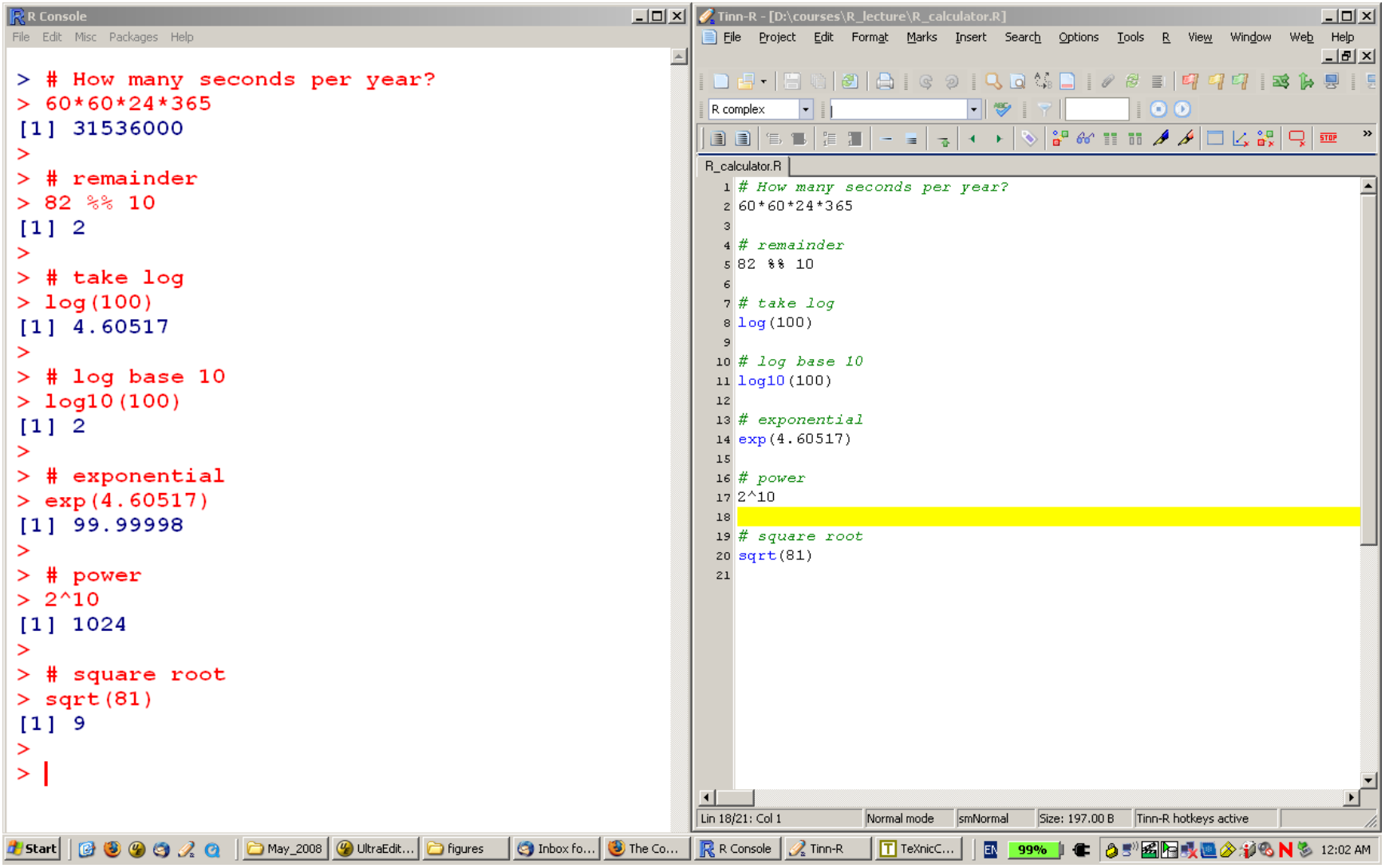


Figure 2: Use an appropriate editor, e.g., Tinn-R

Using R as a calculator

```
> # How many seconds in a year?
> 60*60*24*365
[1] 31536000
>
> # remainder
> 82 %% 10
[1] 2
>
> # natural log and log to base 10
> log(100)
[1] 4.60517
>
> log10(100)
[1] 2
>
> # exponential
> exp(4.60517)
[1] 99.99998
```

```
>
> # power
> 2^10
[1] 1024
>
> # square root
> sqrt(81)
[1] 9
```

Scalar Variables

```
> # Define a variable
> x = 123.45
> x
[1] 123.45
>
> # R language is case sensitive
> X
Error: object "X" not found
>
> # another way to define a variable
> z <- 66.55
> z + x
[1] 190
>
> # be careful
> w < - 8.9
Error: object "w" not found
```

Vectors

```
> # Define a vector
> v = c(1.2, 2.3, 3.4)
> v
[1] 1.2 2.3 3.4
> v*2
[1] 2.4 4.6 6.8
>
> # summation
> sum(v)
[1] 6.9
>
> # mean and standard deviation
> mean(v)
[1] 2.3
> sd(v)
[1] 1.1
```

```
> # function summary
> v = c(1.0, 3.0, -1.5, 0, 0.5)
> summary(v)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   -1.5     0.0     0.5     0.6     1.0     3.0
>
> # vector length
> length(v)
[1] 5
>
> # choose a subset
> 1:3
[1] 1 2 3
> v[1:3]
[1]  1.0  3.0 -1.5
>
> v1 = v[which(v>0)]
> v1
[1] 1.0 3.0 0.5
```

Matrices

```
> m1 = matrix(1:9, nrow=3, ncol=3)
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> m2 = matrix(1:9, nrow=3, ncol=3, byrow=TRUE)
> m2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
>
> m1 + m2
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    6   10   14
[3,]   10   14   18
```

```
>
> # matrix dimension
> dim(m1)
[1] 3 3
> dim(m2)
[1] 3 3
>
> # element-wise multiplication
> m1 * m2
     [,1] [,2] [,3]
[1,]    1    8   21
[2,]    8   25   48
[3,]   21   48   81
>
> # matrix multiplication
> m1 %*% m2
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126
```

```
>
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> # submatrix
> m1[2,2]
[1] 5
>
> m1[1:2,]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
>
> m1[c(1,3),2:3]
     [,1] [,2]
[1,]    4    7
[2,]    6    9
```

```
>
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> # function apply
> # apply(X, MARGIN, FUN, ...)
> # MARGIN =1 for rows, =2 for columns, =c(1,2) for rows and columns
> # FUN is the function to be applied
>
> apply(m1[1:2,], 2, sum)
[1]  3  9 15
>
> apply(m1[c(1,3),2:3], 1, mean)
[1] 5.5 7.5
>
>
>
```

```
> # diag(1,3) creates a 3x3 diagonal matrix with 1s on the diagonal
>
> m3 = diag(1,3) + matrix(c(0,1,2,0,0,1,0,0,0),nrow=3)
> m3
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    1    1    0
[3,]    2    1    1
>
> # matrix transpose
> t(m3)
     [,1] [,2] [,3]
[1,]    1    1    2
[2,]    0    1    1
[3,]    0    0    1
>
>
>
>
>
```

```
> # matrix inverse
> m4 = solve(m3)
> m4
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]   -1    1    0
[3,]   -1   -1    1
>
> m3 %*% m4
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
>
```

rbind / cbind

```
> # Generate a sequence
> # seq(from, to, by)
> # below could use just x = seq(1, 7, 3)
> x = seq(1, 7, by=3)
> x
[1] 1 4 7
>
> # Replicate a vector x
> # rep(x, times)
> y = rep(1, 3)
> y
[1] 1 1 1
>
> # row-wise bind
> rbind(x,y)
  [,1] [,2] [,3]
x    1    4    7
y    1    1    1
>
```

```
> # column-wise bind
> cbind(x,y)
     x y
[1,] 1 1
[2,] 4 1
[3,] 7 1
```

Types of variables

```
> v = 1:5
> v
[1] 1 2 3 4 5
> mode(v)
[1] "numeric"
>
> a = "Hello, World :)"
> a
[1] "Hello, World :)"
> mode(a)
[1] "character"
>
> b = v==2
> b
[1] FALSE  TRUE FALSE FALSE FALSE
> mode(b)
[1] "logical"
>
>
```

```
> # factor
>
> treatments = c("placebo", "100mg", "200mg")
>
> # sample() draws a sample with or without replacement
> csamp = sample(treatments, 6, replace=TRUE)
>
> csamp
[1] "200mg"   "placebo" "placebo" "100mg"   "placebo" "200mg"
>
> # as.factor() forces its argument to be an object of class factor
> as.factor(csamp)
[1] 200mg   placebo placebo 100mg   placebo 200mg
Levels: 100mg 200mg placebo
>
> table(csamp)
csamp
  100mg   200mg placebo
      1       2       3
```

```
> # list
> p = c("regulation of apoptosis", "response to tumor")
> g = list(gene="Tp53", process=p, expression=c(1.2,9.1))
> g
$gene
[1] "Tp53"

$process
[1] "regulation of apoptosis" "response to tumor"

$expression
[1]  1.2 9.1

> g[[1]]
[1] "Tp53"
> g[1]
$gene
[1] "Tp53"
> g$gene
[1] "Tp53"
```

```
>
> # [ can select more than one element
> # whereas [[ and $ can select just one
>
> g[1:2]
$gene
[1] "Tp53"

$process
[1] "regulation of apoptosis" "response to tumor"
>
> g[[1:2]]
Error in g[[1:2]] : subscript out of bounds
>
```

Data Frames

A data frame is a list with class "data.frame". Usually, its columns are vectors of the same length. A numerical or logical vector is included as is, and a character vector is coerced to be of type factor.

```
> sym = c("BRCA1", "BRCA2", "RAS1", "APC", "Tp53")
>
> # rnorm(n, mean, sd);  defaults: mean=0, sd=1
> ep1 = round(rnorm(5,0,1),2)
> ep2 = round(rnorm(5,0,1),2)
>
> dat = data.frame(sym=sym, e1=ep1, e2=ep2)
> dat
    sym    e1    e2
1 BRCA1 -0.59 -1.02
2 BRCA2 -1.07  1.20
3  RAS1 -1.73 -0.59
4   APC -1.40  0.63
5  Tp53 -1.76 -0.06
```

```
>
> mode(dat)
[1] "list"
> dim(dat)
[1] 5 3
> names(dat)
[1] "sym" "e1"  "e2"
>
> dat1 = cbind(sym, ep1, ep2)
> dat1[1:2,]
     sym     ep1     ep2
[1,] "BRCA1" "-0.59" "-1.02"
[2,] "BRCA2" "-1.07" "1.2"
> dat1 = as.data.frame(dat1)
> dat1[1:2,]
    sym   ep1   ep2
1 BRCA1 -0.59 -1.02
2 BRCA2 -1.07   1.2
```

R functions, datasets, and packages

- "All R functions and datasets are stored in packages. Only when a package is loaded are its contents available."
- By default, some standard packages (e.g., base, stats) are included in the binary distribution of R and they are loaded into the R environment automatically when one opens the R interface.
- Some recommended packages are included in the binary R distribution, but are not loaded automatically.
- Contributed packages need to be installed before one can load and use them.

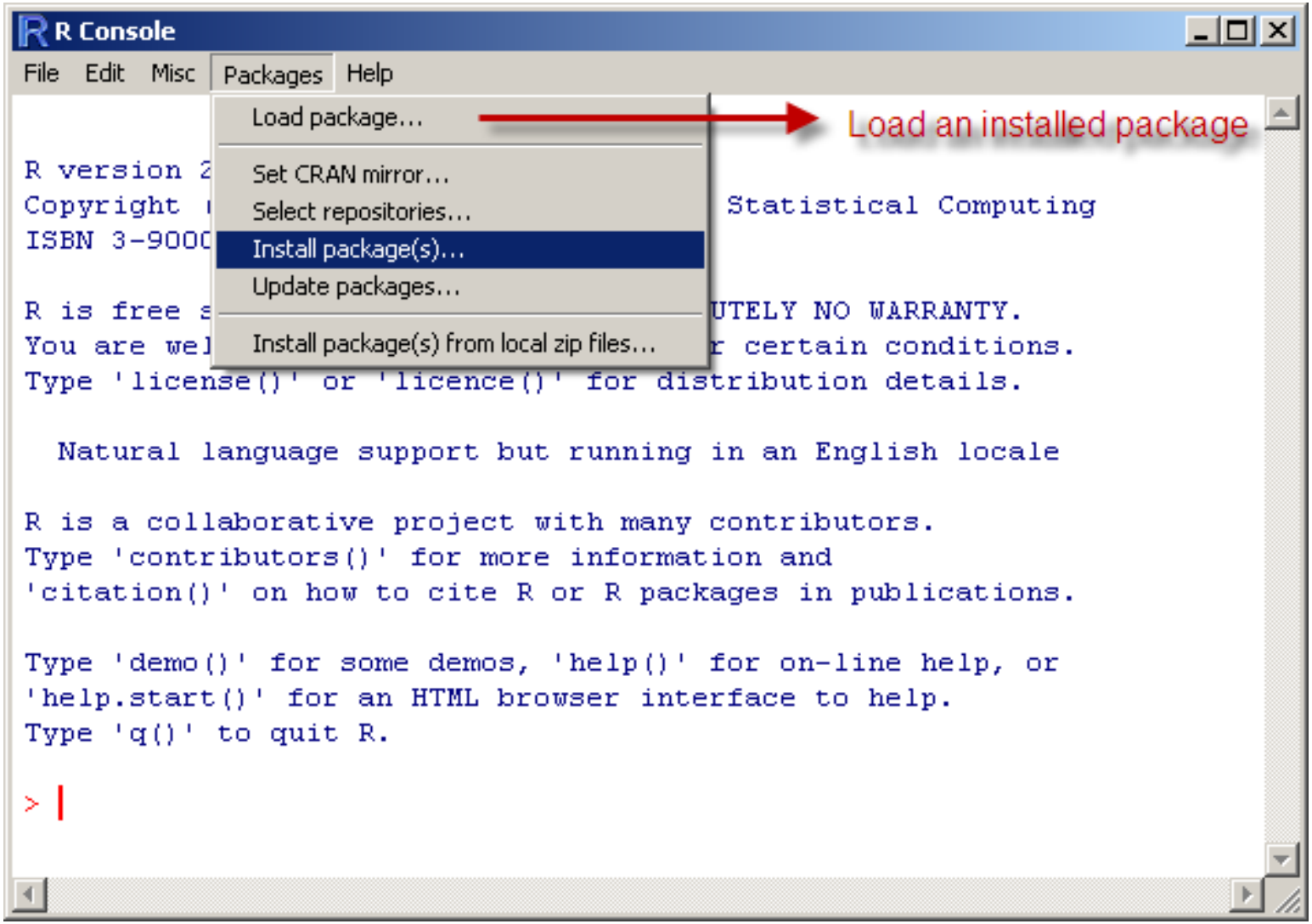


Figure 3: Load/install R packages

Help

- How does one know which function to use?
- Suppose one is looking for something related to the uniform distribution
  - help(package="stats")
  - help.search("uniform")
  - google it
- How to use a function?
  - ?runif
  - help(runif)

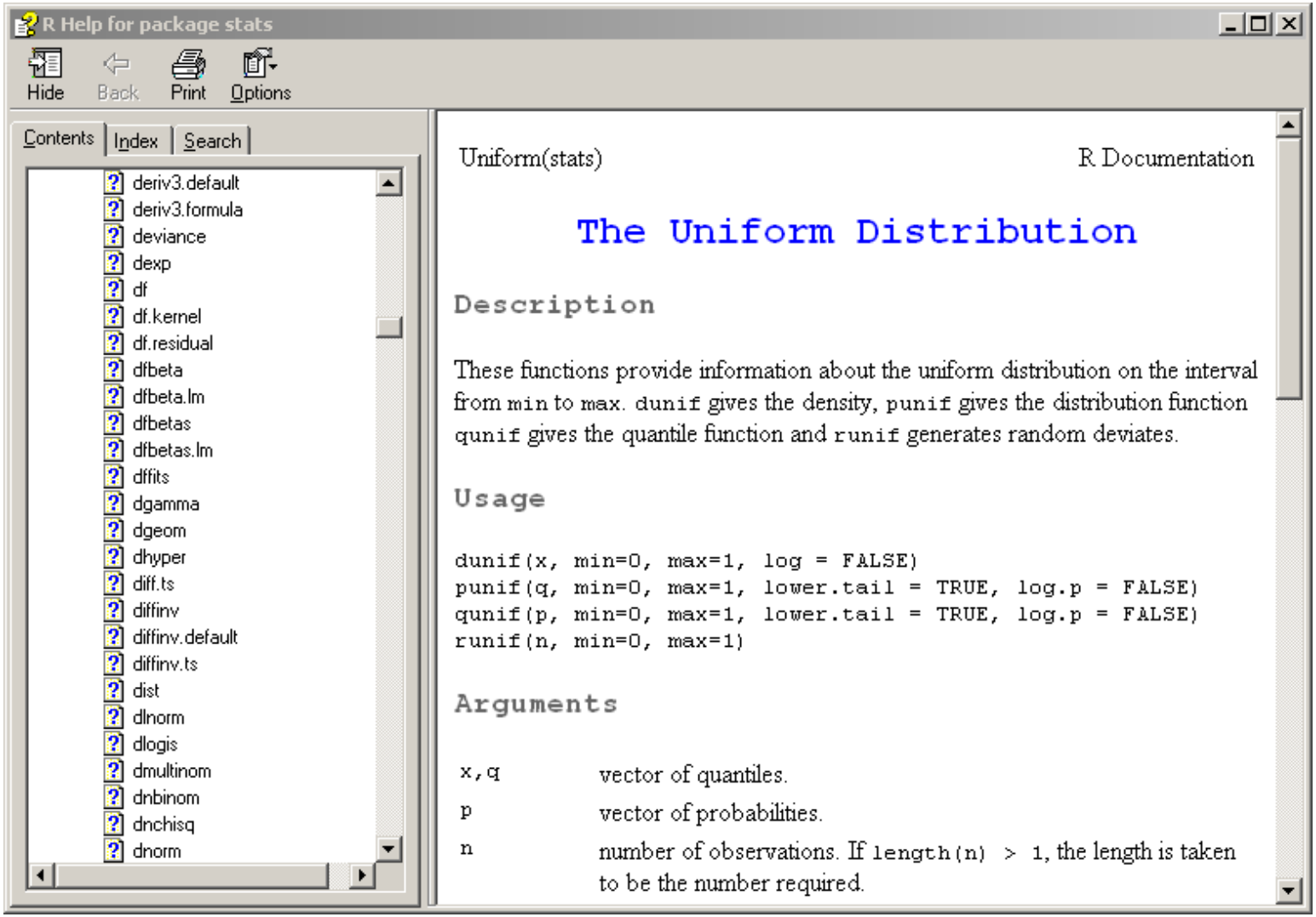


Figure 4: Help file for function “runif”

Loops and conditional execution

```
> x = runif(100)
> x[1:5]
[1] 0.1996935 0.2580256 0.2381857 0.4106282 0.8495470
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01639 0.32640 0.53370 0.53420 0.76960 0.99880
>
> sum(x[x>0.5 & x<0.8])
[1] 18.87469
>
> y = 0
> for(i in 1:length(x)){
+   if(x[i]>0.5 & x[i]<0.8){y = y + x[i]}
+ }
> y
[1] 18.87469
```

Writing one's own functions

```
> hi <- function(yourname, myname="David"){
+   str1 = paste("Hello, ", yourname, ", This is ", myname, sep="")
+   str1
+ }
> hi("World")
[1] "Hello, World, This is David"
> hi <- function(yourname){
+   str1 = paste("Hello,", yourname, sep=" ");substr(str1,1,8);
+ }
> hi("World")
[1] "Hello, W"
```

- Parameters, default values of the parameters
- Output is the last variable evaluated
- Different commands separated by a semi-colon, or by starting a new line

R vs. other programming languages, software packages

- S/Splus
  - R regarded as an implementation of the S language, which forms the basis of S-Plus.
  - Syntax of R is almost the same as that of S (or S-Plus).
- SAS
  - SAS procedures by default give just copies of the output, while R functions usually have all the intermediate results stored
  - One has better control in R, and R is open source
  - R has no warranty
- perl/python
  - perl has more powerful text processing facilities than R
  - There are few statistical functions in perl

R vs. c/c++

- In c/c++, need to take care of many low level things, such as memory allocation. There are some c libraries available, so one doesn't always have to start from scratch. But still, it is not always a pleasant experience spending a whole day debugging c code.
- R is interactive. This means one can track the results at each step. Gives better control, fewer bugs. R has rich libraries. Most of the time one just needs to install a library and call a function. For large scale analysis, R is slow.
- c/c++: Spend one day on programming, take 10 minutes to run the program.
- R: Spend 10 minutes writing the program. Wait 1 day for the results.

Reading data from files

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           encoding = "unknown")
```

Two versions of an input file

test_dat.txt

```
BRCA1 0.65 0.65
BRCA1 -1.41 -1.41
RAS1 -0.64 -0.64
APC -0.28 -0.28
Tp53 -0.27 -0.27
```

test_dat_with_header.txt

```
gene sym    e1  e2
BRCA1   0.65    0.65
BRCA1   -1.41   -1.41
RAS1    -0.64   -0.64
APC'    -0.28   -0.28
Tp53    -0.27   -0.27
```

Reading data from files

```
> getwd()
[1] "C:/Documents and Settings/David/My Documents"
>
> setwd("G:/Z_CSCC/BIOS662_2011/From Wei Sun/R_lecture/R_lecture/dat")
> list.files()
[1] "test_dat.txt"              "test_dat_with_header.txt"
>
> dat1 = read.table("test_dat.txt")
> dim(dat1)
[1] 5 3
> dat1[1:2,]
     V1    V2    V3
1 BRCA1  0.65  0.65
2 BRCA1 -1.41 -1.41
>
>
>
>
```

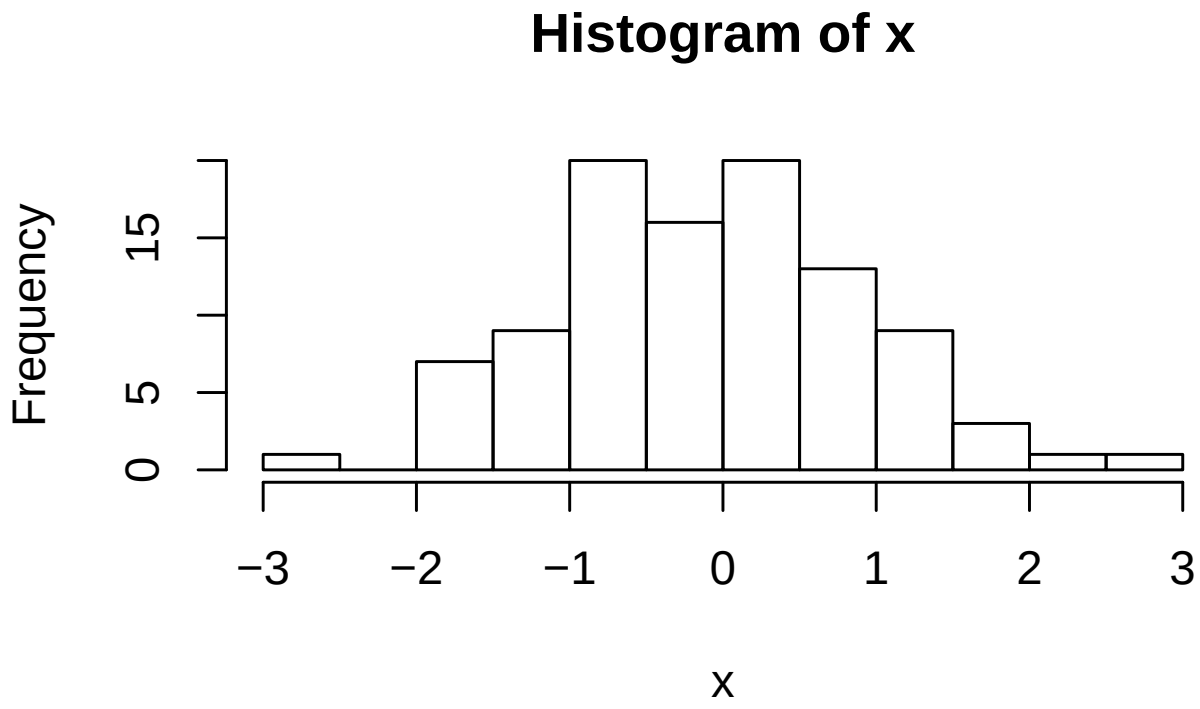
```
> dat2 = read.table("test_dat_with_header.txt",
+  header=TRUE)
Error in scan(file, what, nmax, sep, dec, quote, ...
  line 3 did not have 4 elements
In addition: Warning message:
In read.table("test_dat_with_header.txt", header = TRUE) :
  incomplete final line found by readTableHeader ...
>
> dat2 = read.table("test_dat_with_header.txt",
+  header=TRUE, sep="\t", quote = "")
>
> dim(dat2)
[1] 5 3
> dat2[1:2,]
  gene.sym    e1    e2
1    BRCA1  0.65  0.65
2    BRCA1 -1.41 -1.41
```

Writing data to files

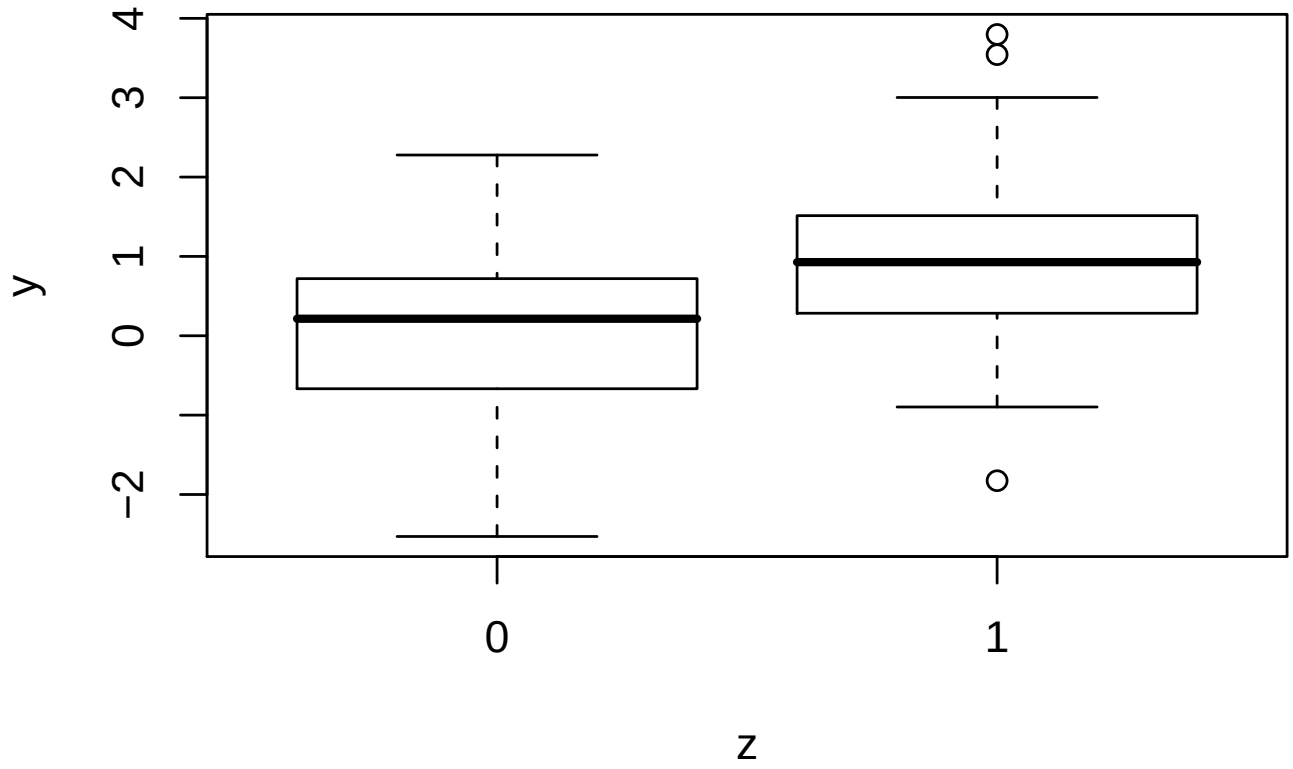
```
# default
write.table(dat2, file = "d2.txt", append = FALSE, quote = TRUE,
            sep = " ", row.names = TRUE, col.names = TRUE)

# Wei prefers these parameters
write.table(dat, file = "d2.txt", append = FALSE, quote = FALSE,
            sep = "\t", row.names = FALSE, col.names = TRUE)


save(dat2,  file = "d2.RData")

load("d2.RData")
```

Graphics

- high-level plotting functions
  - scatter plot: `plot(x,y)`
  - histogram: `hist(x)`
  - boxplot: `boxplot(x)`
- low-level plotting commands
  - add points: `points(x, y)`
  - add lines: `lines(x, y)`, `abline(a, b)`
  - add text: `text(x, y, labels)`
  - add legend: `legend(x, y, legend)`

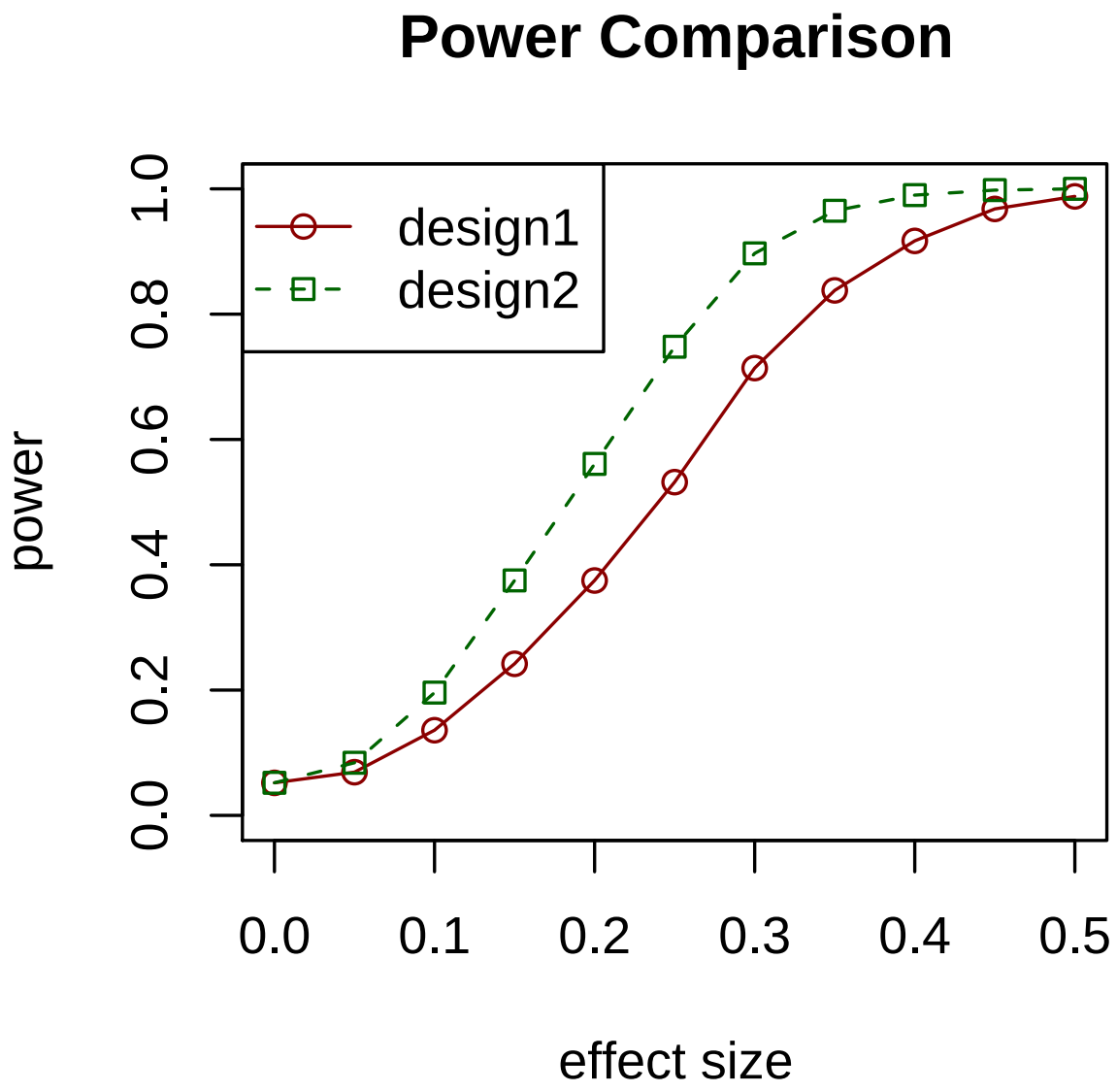


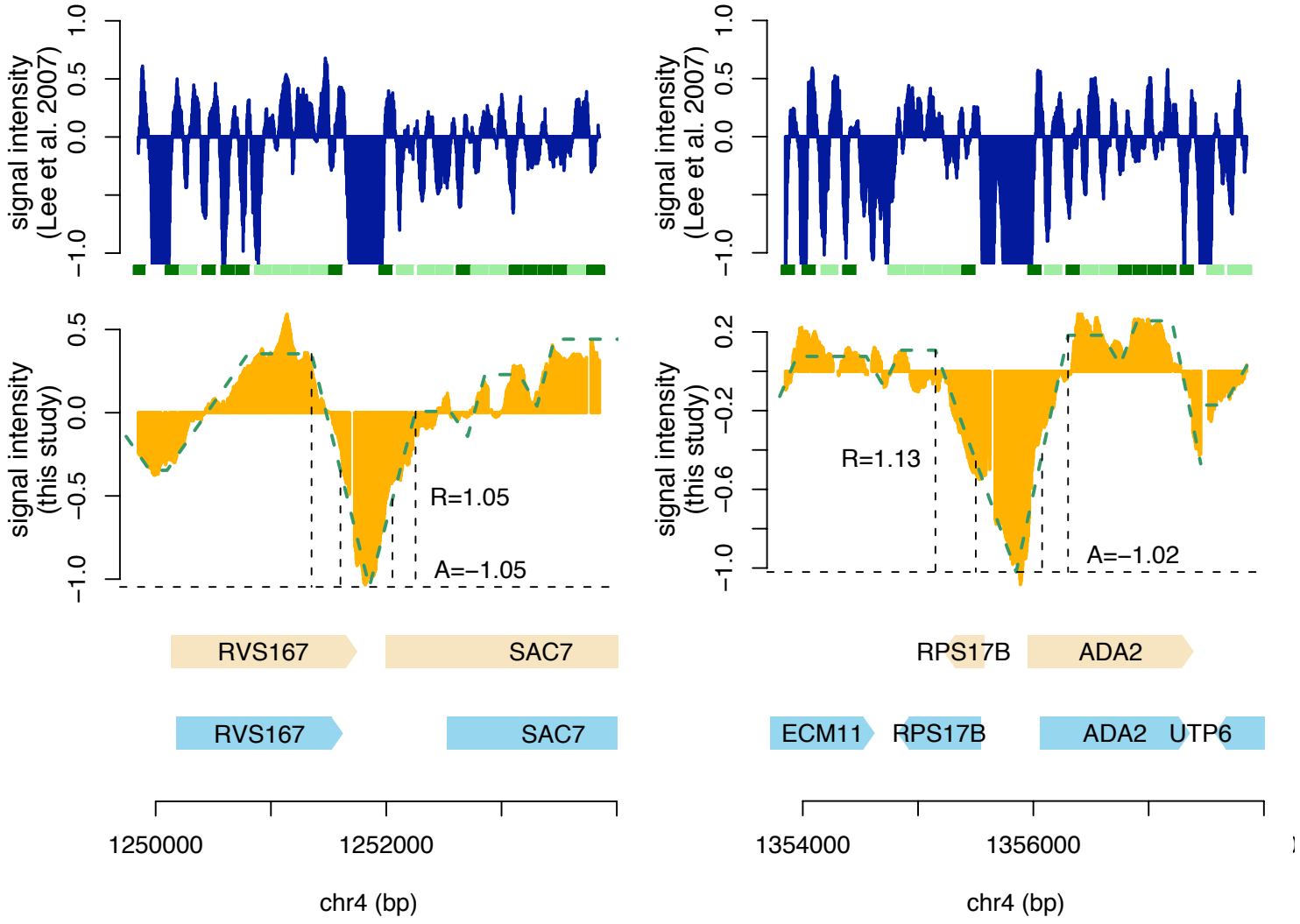

```
> x = rnorm(100)
> hist(x)
```




```
> y = c(rnorm(100, 0, 1),rnorm(100, 1, 1))
> z = rep(c(0,1),each=100)
> par(mar=c(5,4,1,1))
> boxplot(y~z, xlab="z", ylab="y")
```

```
> dat = read.table("power.txt", header=TRUE)
>
> dat[1:2,]
  effect design1 design2
1   0.00   0.052   0.052
2   0.05   0.069   0.084
>
> windows(width=4,height=4)
> plot(c(0, 0.5), c(0,1), type="n", ylab="power",
+   xlab="effect size", main="Power Comparison")
>
> lines(dat$effect,  dat$design1, col="darkred", lty=1)
> points(dat$effect, dat$design1, col="darkred", pch=21)
>
> lines(dat$effect,  dat$design2, col="darkgreen", lty=2)
> points(dat$effect, dat$design2, col="darkgreen", pch=22)
>
> legend("topleft", legend=c("design1", "design2"),
+   pch=c(21, 22), lty=c(1,2),
+   col=c("darkred", "darkgreen"))
```


Power Comparison
design1
design2
power
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.1
0.2
0.3
0.4
0.5
effect size


signal intensity
(Lee et al. 2007)
1.0
0.5
0.0
−1.0
signal intensity
(this study)
0.5
0.0
−0.5
−1.0
R=1.05
A=−1.05
RVS167
SAC7
RVS167
SAC7
1250000
1252000
chr4 (bp)
signal intensity
(Lee et al. 2007)
1.0
0.5
0.0
−1.0
signal intensity
(this study)
0.2
−0.2
−0.6
−1.0
R=1.13
A=−1.02
RPS17B
ADA2
ECM11
RPS17B
ADA2
UTP6
1354000
1356000
chr4 (bp)