R Primer for
Introduction to Mathematical Statistics
8th Edition
Joseph W. McKean

This brief introduction to R suffices for the R discussed in Hogg, McKean, and Craig's (2018) *Introduction to Mathematical Statistics, 8th Ed.*.

The package R can be dowloaded at CRAN, (https://cran.r-project.org/). It is freeware and there are versions for most platforms including Windows, Mac, and Linux. To install R simply follow the directions at CRAN. Installation should only take a few minutes. For more information on R, there are free downloadable manuals on its use at the CRAN website. There are many reference texts that the reader can consult, including the books by Venables and Ripley (2002), Verzani (2014), and Crawley (2007).

Once R is installed, in Windows, click on the R icon to begin an R session. The R prompt is a >. To exit R, type `q()`, which results in the query `Save workspace image? [y/n/c]:`. Upon typing `y`, the workspace will be saved for the next session. R has a built-in help (documentation) system. For example, to obtain help on the `mean` function, simply type `help(mean)`. To exit help, type `q`. We would recommend using R while working through the sections in this primer.

1 Basics

The commands of R work on numerical data, character strings, or logical types. To separate commands on the same line, use semicolons. Also, anything to the right of the symbol # is disregarded by R; i.e., to the right of # can be used for comments. Here are some arithmetic calculations:

```
> 8+6 - 7*2

[1] 0

> (150/3) + 7^2 -1 ; sqrt(50) - 50^(1/2)

[1] 98

[1] 0

> (4/3)*pi*5^3    # The volume of a sphere with radius 5

[1] 523.5988

> 2*pi*5          # The circumference of a sphere with radius 5

[1] 31.41593
```

Results can be saved for later calculation by the **assignment** function, `<-` or equivalently `=`. Names can be a mixture of letters, numbers, or symbols. For example:

```
> r <- 10 ; Vol <- (4/3)*pi*r^3 ; Vol

[1] 4188.79

> r = 100 ; circum = 2*pi*r ; circum

[1] 628.3185
```

Variables in R include scalars, vectors, or matrices. In the last example the variables `r` and `Vol` are scalars. Scalars can be combined into vectors with the `c` function. Further, arithmetic functions on vectors are performed componentwise. For instance, here are two ways to compute the volumes of spheres with radii $5, 6, \ldots, 9$.

```
> r <- c(5,6,7,8,9) ; Vol <- (4/3)*pi*r^3 ; Vol

[1]  523.5988  904.7787 1436.7550 2144.6606 3053.6281

> r <- 5:9 ; Vol <- (4/3)*pi*r^3 ; Vol

[1]  523.5988  904.7787 1436.7550 2144.6606 3053.6281
```

Components of a vector are referred to by using brackets. For example, the 5th component of the vector `vec` is `vec[5]`. Matrices can be formed from vectors using the commands `rbind` (combine rows) and `cbind` (combine columns) on vectors. To illustrate let $\mathbf{A}$ and $\mathbf{B}$ be the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}.$$

Then $\mathbf{AB}$, $\mathbf{A}^{-1}$, and $\mathbf{B}'\mathbf{A}$ are computed by

```
> c1 <- c(1,3) ; c2 <- c(4,2); a <- cbind(c1,c2)
> r1 <- c(1,3,5,7); r2 <- c(2,4,6,8); b <- rbind(r1,r2)
> a%*%b; solve(a) ; t(b)%*%a

     [,1] [,2] [,3] [,4]
[1,]    9   19   29   39
[2,]    7   17   27   37
```

```
   [,1] [,2]
c1 -0.2  0.4
c2  0.3 -0.1

     c1 c2
[1,]  7  8
[2,] 15 20
[3,] 23 32
[4,] 31 44
```

Brackets are also used to refer to elements of matrices. Let `amat` be a 4×4 matrix. Then the $(2, 3)$ element is `amat[2,3]` and the upper right corner 2×2 submatrix is `amat[1:2,3:4]`. This last item is an example of subsetting of a matrix. Subsetting is easy in R. For example, the following commands obtain the negative, positive, and elements of 0 for a vector `x`:

```
> x = c(-2,0,3,4,-7,-8,11,0)
> xn = x[x<0]; xn

[1] -2 -7 -8

> xp = x[x>0]; xp

[1]  3  4 11

> x0 = x[x==0]; x0

[1] 0 0
```

For R vectors `x` and `y` of the same length, the plot of `y` versus `x` is obtained by the command `plot(y ~ x)`. The following segment of R code obtains plots found in Figure 1 of the volume and circumference of the sphere versus the radius for a sequence of radii from 0 to 8 in steps of 0.1. The first plot is a simple plot; the second plot adds some labeling and a title; the third plot draws a curve of the relationship; and the fourth plot show the relationship between the circumference of the circle versus the radius.

```
> par(mfrow=c(2,2))      # This sets up a 2 by 2 page of plots
> r <- seq(0,8,.1);  Vol <- (4/3)*pi*r^3 ; plot(Vol ~ r)     # Plot 1
> plot(Vol ~ r,xlab="Radius",ylab="Volume")                  # Plot 2
> title("Volume of Sphere versus its radius")
> plot(Vol ~ r,pch=" ",xlab="Radius",ylab="Volume")
```

```
> lines(Vol ~ r)                                                    # Plot 3
> circum <- 2*pi*r
> plot(circum ~ r,pch=" ",xlab="Radius",ylab="Circumference")
> lines(circum ~ r); title("Circumference vs Radius")               # Plot 4
```

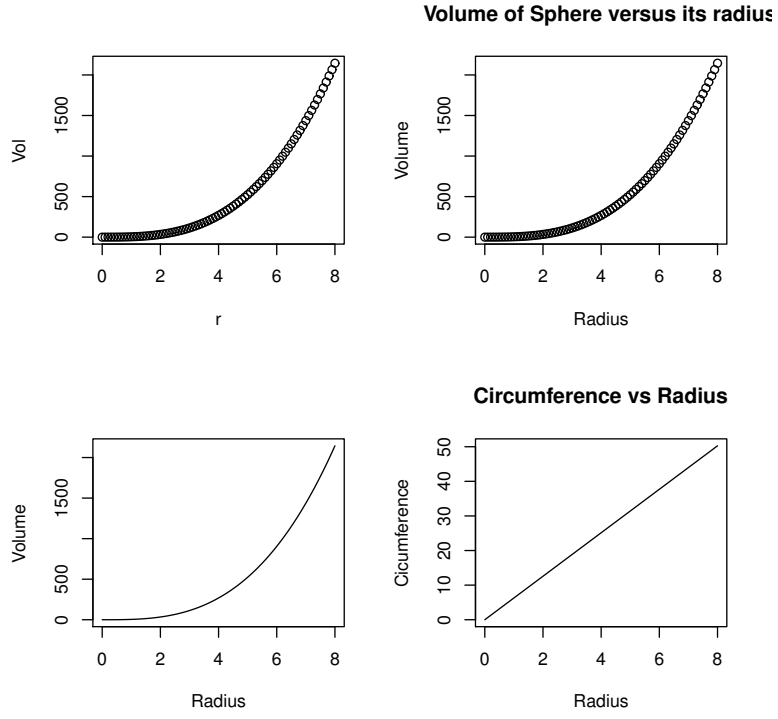

Figure 1: Spherical Plots discussed in Text.

2 Probability Distributions

For many distributions, R has functions which obtain probabilities, compute quantiles, and generate random variates. Here are two common examples. Let X be a random variable with a $N(\mu, \sigma^2)$ distribution. In R, let `mu` and `sig` denote the mean and standard deviation of X, respectively. Then the R commands and meanings are:

`pnorm(x,mu,sig)`	$P(X \leq x)$.
`qnorm(p,mu,sig)`	$P(X \leq q) = p$.
`dnorm(x,mu,sig)`	$f(x)$, where f is the pdf of X.
`rnorm(n,mu,sig)`	n variates generated from distribution of X.

As a numerical illustration, suppose the height of a male is normally distributed with mean 70 inches and standard deviation 4 inches.

```
> 1-pnorm(72,70,4)      # Prob. man exceeds 6 foot in ht.

[1] 0.3085375

> qnorm(.90,70,4)       # The upper 10th percentile in ht.

[1] 75.12621

> dnorm(72,70,4)        # value of density at 72

[1] 0.08801633

> rnorm(6,70,4)        # sample of size 6 on X

[1] 71.23986 66.66026 65.73312 65.14238 70.54796 64.48749
```

For the next figure, 2, we generate 100 variates, histogram the sample, and overlay the plot of the density of X on the histogram. Note the `pr=T` argument in the histogram. This scales the histogram to have area 1.

```
> x = rnorm(100,70,4);  x=sort(x)
> hist(x,pr=T,main="Histogram of Sample")
> y = dnorm(x,70,4)
> lines(y~x)
```

For a discrete random variable the pdf is the probability mass function (pmf). For suppose X is binomial with 100 trials and 0.6 as the probability of success.

```
> pbinom(55,100,.6)      #  Probability of at most 55 successes

[1] 0.1789016

> dbinom(55,100,.6)      #  Probability of exactly 55 successes
```

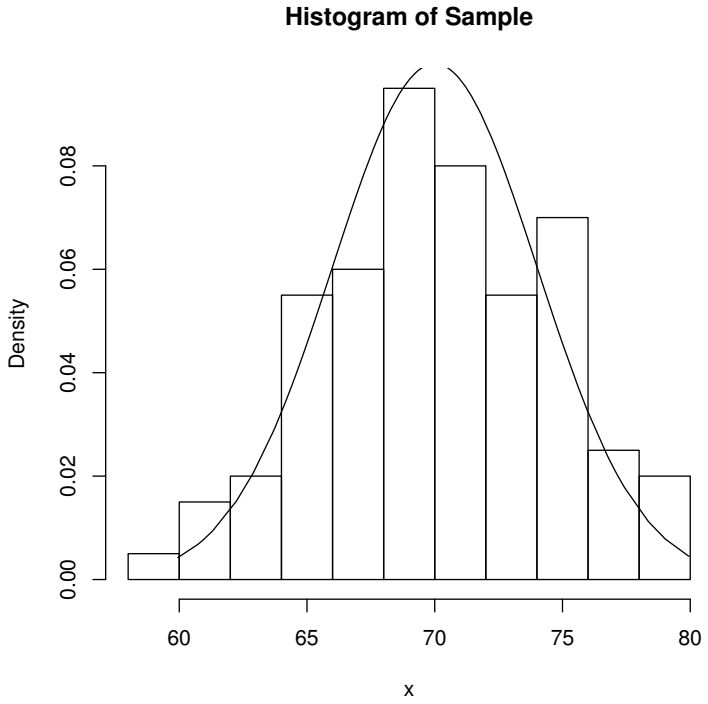

Figure 2: Histogram of a Random Sample from a $N(70, 4^2)$ distribution overlaid with the pdf of this normal.

```
[1] 0.04781118
```

Most other well known distributions are in core R. For example, here is the probability that a χ^2 random variable with 30 degrees of freedom exceeds 2 standard deviations form its mean, along with a Γ-distribution confirmation.

```
> mu=30; sig=sqrt(2*mu); 1-pchisq(mu+2*sig,30)

[1] 0.03471794

> 1-pgamma(mu+2*sig,15,1/2)

[1] 0.03471794
```

One other sampling command, is the `sample` command which returns a random sample from a vector. It can either be sampling with replacement (`replace=T`) or sampling without replacement (`replace=F`). Here are samples of size 12 from the first 20 positive integers.

```
> vec = 1:20
> sample(vec,12,replace=T)

 [1] 13 11  2 11 15 18 13  3 15 18  4  8

> sample(vec,12,replace=F)

 [1]  5 10 19 16  4  8 17  2 12 15  3 13
```

3 R Functions

The syntax for R functions is the same as the syntax in R. This easily allows for the development of packages, a collection of R functions, for specific tasks. The schematic for an R function is

```
name-function <- function(arguments){
         ...  body of function  ...
}
```

Example 3.1. Consider a process where a measurement is taken over time. At each time n, $n = 1, 2, \ldots$, the measurement x_n is observed but only the sample mean $\overline{x}_n = (1/n)\sum_{i=1}^{n} x_i$ of the measurements at time n is recorded and the point $(n, \overline{x}_n)$ is added to the running plot of sample means. How is this possible? There is a simple update formula for the sample mean which is easily derived and which is given by

$$\overline{x}_{n+1} = \frac{n}{n+1}\overline{x}_n + \frac{1}{n+1}x_{n+1}; \tag{1}$$

hence, the sample mean for the sequence $x_1, \ldots, x_{n+1}$ can be expressed as a linear combination of the sample mean at time n and the $(n+1)st$ measurement. The following R function codes this update formula:

```
mnupdate <- function(n,xbarn,xnp1){
#   Input: n is sample size; xbarn is mean of sample of size n;
#          xnp1 is (n+1) (new) observation
#   Output: mean of sample of size (n+1)
       mnupdate <- (n/(n+1))*xbarn + xnp1/(n+1)
       return(mnupdate)
}
```

To run this function we first source it in R. If the function is in the file `mnupdate.R` in the current directory then the source command is `source("mnupdate.R")`. It can also be copied and pasted into the current R session. Here is an execution of it with a verification for accuracy.

```
> source("mnupdate.R")
> x = c(3,5,12,4); n=4; xbarn = mean(x);
> x; xbarn          #Old sample and its mean

[1]  3  5 12  4

[1] 6

> #
> #
> xp1 = 30       # New observation
> c(x,xp1);  mnupdate(n,xbarn,xp1)    #  New sample and Mean of new sample

[1]  3  5 12  4 30

[1] 10.8
```

■

Suppose in the last example, a running update of the sample standard deviation is also needed. Recall that the sample variance of the observations $x_1, \ldots, x_n$ is given by $s_n^2 = \sum_{i=1}^n (x_i - \overline{x}_n)^2/(n-1)$ and the sample deviation is $s_n = \sqrt{s_n^2}$. To derive the update formula for s_n^2, first consider the identity:

$$s_n^2 = \frac{1}{n(n-1)} \sum_{i<j} (x_i - x_j)^2 = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2. \tag{2}$$

To derive this identity, add in and subtract out $\overline{x}_n$ in the far right sum and then simplify. The update formula for the sample variance is given by

$$s_{n+1}^2 = \frac{n-1}{n} s_n^2 + \frac{1}{n+1} (x_{n+1} - \overline{x}_n)^2. \tag{3}$$

To derive this formula, start with (2) for $n+1$, write it as

$$n(n+1)s_{n+1}^2 = \sum_{i<j} (x_i - x_j)^2 = \sum_{i<j<n+1} (x_i - x_j)^2 + \sum_{i<n+1} (x_i - x_{n+1})^2,$$

and then, again, add in and subtract out $\overline{x}_n$ in the far right sum and simplify. The following R function computes the update for the standard deviation:

```
sdupdate <- function(n,xbarn,snd,xnp1){
#   Input: n is sample size; xbarn is mean of sample of size n;
#          snd is standard deviation of sample of size n;
#          xnp1 is (n+1) (new) observation
#   Output: standard deviation of sample of size (n+1)
       sdupdate <- ((n-1)/n)*(snd^2) + (1/(n+1))*(xnp1 - xbarn)^2
       sdupdate <- sqrt(sdupdate)
       return(sdupdate)
}
```

Here is a continuation of the last R run with a verification of the update formula:

```
> source("sdupdate.R")
> #Old sample and its sd and mean.
> x = c(3,5,12,4);  x; n=4; n;  sdn=sd(x) ; sdn; xbarn=mean(x); xbarn

[1]  3  5 12  4

[1] 4

[1] 4.082483

[1] 6

> #
> #
> xp1 = 30; c(x,xp1)      #New sample

[1]  3  5 12  4 30

> sdupdate(n,xbarn,sdn,xp1)    # New samples sd

[1] 11.30044

> #
> #
> sd( c(x,xp1))      #  Confirmation

[1] 11.30044
```

4 Loops

Occasionally in the text, we use a loop in an R program to compute a result. Usually it is a simple `for` loop of the form

```
for(i in 1:n){
    ... R code often as a function of i ...
    #  For the n-iterations of the loop, i runs through
    #  the values i=1, i=2, ... , i=n.
}
```

For example the following code segment produces a table of squares, cubes, square-roots, and cube-roots for the integers from 1 to n.

```
#   set n at some value
tab <- c()           #  Initialize the table
for(i in 1:n){
     tab <- rbind(tab,c(i,i^2,i^3,i^(1/2),i^(1/3)))
}
tab
```

The following R function computes the sample variance of a vector of observations using the identity (2). It of course has a nested loop to compute the nested double sum.

```
svar2 <- function(x){
    n <- length(x)
    svar2 <- 0
    for(i in 1:(n-1)){
          for(j in (i+1):n){
               svar2 <- svar2 + (x[i]-x[j])^2
          }
    }
    svar2 <- (1/(n*(n-1)))*svar2
    return(svar2)
}
```

5 Input and Output

Many texts on R, including the references cited above, have information on input and output (I/O) in R. We only discuss several ways which are useful for the R discussion in our text. For output, we discuss two commands. The

first writes an array (matrix) to a text file. Suppose `amat` is a matrix with p columns. Then the command `write(t(amat),ncol=p,file="amatrix.dat")` writes the matrix `amat` to the text file `amatrix.dat` in the current directory. Simply put the "Path" before the file as `file="Path/amatrix.dat"` to send it to another directory. The second way writes out variables to an R object file called an "rda" file. The variables can include scalars, vectors, matrices, and strings. For example the next line of code writes to an rda file the scalars `avar` and `bscale` and the matrix `amat` along with an information string.

```
info <- "This file contains the variable ....."
save(avar,bscale,amat,info,file="try.rda")
```

Again, a path can direct the rda to the desired directory. The command `load("try.rda")` will load these variables (names and values) into the current session. Most of the data sets discussed in the text are in rda files.

For input, we have already discussed the `c` and `load` functions. The `c` function is tedious, though, and a much easier way is to use the `scan` function. For example, the following lines of code assign the vector $(1, 2, 3)$ to `x`:

```
x <- scan()
1   2
  3
```

The separator between values is white space and the empty line after the data signals the end of `x`'s values. Note that this allows data to be copied and pasted into R. A matrix can also be scanned similarly by using the `read.table` function; for example, the following command inputs the above matrix $\boldsymbol{A}$ with column header "c1" and "c2":

```
a <- read.table(header = TRUE, text = "
      c1 c2
      1   4
      3   2
      ")
```

Notice that copy and paste is also easily used with this command. If the matrix $\boldsymbol{A}$ is in the file `amat.dat` with no header, it can be read in as

```
a <- matrix(scan("amat.dat"),ncol=2,byrow=T)
```

References

Crawley, M.J. (2007), *The R Book*, London: John Wiley & Sons, Ltd.

Hettmansperger, T. P. and McKean, J. W. (2011), *Robust Nonparametric Statistical Methods, 2nd Edition*, Boca Raton, FL: Chapman-Hall.

Hogg, R.V., McKean, J.W. and Craig, A.T. (2018), *Introduction to Mathematical Statistics, 8th Edition*, Pearson: Boston.

Kloke, J. and McKean, J. W. (2014), *Nonparametric statistical methods using R*, Boca Raton, FL: Chapman-Hall.

Pinheiro, J.C. and Bates, D.M. (2000), *Mixed–Effects Models in S and S–Plus*, New York: Springer.

Verzani, J. (2014), *Using R for Introductory Statistics, 2nd Edition*, Boca Raton, FL: Chapman-Hall.