Contents

1 The R Programming Language

The R Programming Language is a statistical , data analysis , etc
R is a free software environment for statistical computing and graphics.

2 Writing R scripts

Editing your R script "R Editor".

- On the menu of the R console, click on file.
- Select open script or new script as appropriate.
- Navigate to your working directory and select your `.R` file
- A new dialogue box "`the R editor`" will open up.
- Input or select code you wish to compile.
- To compile this code, highlight it. Click the edit button on the menu.
- Select either "Run Line" or "Run Selection or All".
- Your code should now compile.
- To save your code, clink on "file" and then "`save as`".
- Save the file with the "`.R`" extension to your working directory.

3 Vector types

`R` operates on named data structures. The simplest such structure is the vector, which is a single entity consisting of an ordered collection of Numbers or characters.

- Numeric vectors
- Character vectors
- Logical vectors
- (also complex number vectors and colour vectors)

To create a vector, use the assignment operator and the concatenate function. For numeric vectors, the values are simply numbers.

```
># week8.r
>NumVec<-c(10.4,5.6,3.1,6.4)
```

Alternatively we can use the `assign()` command
For character vectors, the values are simply characters, specified with quotation marks.A logical vectors is a vector whose elements are TRUE, FALSE or NA

```
>CharVec<-c(‘‘blue", ‘‘green", ‘‘yellow")
>LogVec<-c(TRUE, FALSE)
```

4 Graphical data entry interface

`Data.entry()` is a useful command for inputting or editing data sets. Any changes are saved automatically (i.e. dont need to use the assignment operator). We can also used the `edit()` command, which calls the `R Editor`.

```
>data.entry(NumVec)
>NumVec <- edit(NumVec)
```

Another method of creating vectors is to use the following

```
numeric (length = n)
character (length = n)
logical (length = n)
```

These commands create empty vectors, of the appropriate kind, of length n. You can then use the graphical data entry interface to populate your data sets.

4.0.1 Accessing specified elements of a vector

The nth element of vector "Vec" can be accessed by specifying its index when calling "Vec".

```
>Vec[n]
```

A sequence of elements of vector "Vec" can be accessed by specifying its index when calling "Vec".

```
>Vec[l:u]
```

Omitting and deleting the nth element of vector "Vec"

```
>Vec[-n]
>Vec <- Vec[-n]
```

4.1 inputting data

Concatenation

4.2 using help

?mean

4.3 Packages

The capabilities of R are extended through user-submitted packages, which allow specialized statistical techniques, graphical devices, as well as and import/export capabilities to many external data formats.

5 Managing Precision

- `floor()` -
- `ceiling()` -
- `round()` -
- `as.integer()` -

1cm

```
> pi
[1] 3.141593
> floor(pi)
[1] 3
> ceiling(pi)
[1] 4
> round(pi,3)
[1] 3.142
> as.integer(pi)
[1] 3
```

6 Basic Operations

6.1 Complex numbers

6.2 Trigonometric functions

7 Matrices

7.0.1 exponentials, powers and logarithms

1cm

```
>x^y
>exp(x)
>log(x)
>log(y)
#determining the square root of x
>sqrt(x)
```

7.1 vectors

1cm

```
R handles vector objects quite easily and intuitively.

> x<-c(1,3,2,10,5)    #create a vector x with 5 components
> x
[1]  1  3  2 10  5
> y<-1:5              #create a vector of consecutive integers
> y
[1] 1 2 3 4 5
```

```
> y+2                  #scalar addition
[1] 3 4 5 6 7
> 2*y                  #scalar multiplication
[1]  2  4  6  8 10
> y^2                  #raise each component to the second power
[1]  1  4  9 16 25
> 2^y                  #raise 2 to the first through fifth power
[1]  2  4  8 16 32
> y                    #y itself has not been unchanged
[1] 1 2 3 4 5
> y<-y*2
> y                    #it is now changed
[1]  2  4  6  8 10
```

7.1.1 Misc

`seq()` and `rep()` are useful commands for constructing vectors with a certain pattern.

7.2 Matrices

A matrix refers to a numeric array of rows and columns.

One of the easiest ways to create a matrix is to combine vectors of equal length using cbind(), meaning "column bind". Alternatively one can use rbind(), meaning "row bind".

7.2.1 Matrices Inversion

7.2.2 Matrices Multiplication

7.3 Data frame

A Data frame is

Descriptive Statistics

8 Basic Statistics

1cm

```
> X=c(1,4,5,7,8,9,5,8,9)
> mean(X);median(X)       #mean and median of vector
[1] 6.222222
[2] 7
> sd(X)                   #standard deviation of Vector
[1] 2.682246
> length(X)               #sample size of vector
[1] 9
> sum(X)
[1] 56
> X^2
[1]  1 16 25 49 64 81 25 64 81
> rev(X)
[1] 9 8 5 9 8 7 5 4 1
> sort(X)                 #items in ascending order
[1] 1 4 5 5 7 8 8 9 9
> X[1:5]
[1] 1 4 5 7 8
```

9 Summary Statistics

The R command `summary()` returns a summary statistics for a simple dataset. The R command `fivenum()` returns a summary statistics for a simple dataset, but without the mean. Also, the quartiles are computed a different way.

1cm

```
> summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.43   19.20   20.09   22.80   33.90
>
> fivenum(mtcars$mpg)
[1] 10.40 15.35 19.20 22.80 33.90
```

10 Bivariate Data

1cm

```
> Y=mtcars$mpg
> X=mtcars$wt
>
> cor(X,Y)          #Correlation
[1] -0.8676594
>
> cov(X,Y)          #Covariance
[1] -5.116685
```

MTCARSboxplot.png

Figure 1: Boxplot

11 Histograms

Histograms can be created using the `hist()` command. To create a histogram of the car weights from the Cars93 data set 1cm

```
hist(mtcars$mpg, main="Histogram of MPG (Data: MTCARS) ")
```

`R` automatically chooses the number and width of the bars. We can change this by specifying the location of the break points. 1cm

```
hist(Cars93$Weight, breaks=c(1500, 2050, 2300, 2350, 2400,
2500, 3000, 3500, 3570, 4000, 4500), xlab="Weight",
main="Histogram of Weight")
```

12 Boxplot

Boxplots can be used to identify outliers.

By default, the `boxplot()` command sets the orientation as vertical. By adding the argument `horizontal=TRUE`, the orientation can be changed to horizontal. 1cm

```
boxplot(mtcars$mpg, horizontal=TRUE, xlab="Miles Per Gallon",
main="Boxplot of MPG")
```

Advanced R code

13 Data frame

A Data frame is

13.1 Merging Data frames

14 Functions

Syntax to define functions
1cm

```
myfct <- function(arg1, arg2, ...) { function_body }
```

The value returned by a function is the value of the function body, which is usually an unassigned final expression, e.g.: return()
Syntax to call functions 1cm

```
myfct(arg1=..., arg2=...)
```

15 Time and Date

It is useful . The length of time a program takes is interesting.
1cm

```
date() # returns the current system date and time
```

16 The Apply family

Sometimes want to apply a function to each element of a vector/data frame/list/array.
Four members: lapply, sapply, tapply, apply
lapply: takes any structure and gives a list of results (hence the 'l')
sapply: like lapply, but tries to simplify the result to a vector or matrix if possible (hence the 's')
apply: only used for arrays/matrices
tapply: allows you to create tables (hence the 't') of values from subgroups defined by one or more factors.

Data Visualization

17 Plots

This section is an introduction for producing simple graphs with the R Programming Language.

- Line Charts
- Bar Charts
- Histograms
- Pie Charts
- Dotcharts

17.0.1 Comparison of variances

Even though it is possible in R to perform the two-sample t test without the assumption that the variances are the same, you may still be interested in testing that assumption, and R provides the var.test function for that purpose, implementing an F test on the ratio of the group variances. It is called the same way as `t.test`:.

```
> var.test(expend~stature)
```

-
-

```
> code here
```

17.1 Charts

1cm

```
# Define 2 vectors cars <- c(1, 3, 6, 4, 9) trucks <- c(2, 5, 4,
5, 12)

# Calculate range from 0 to max value of cars and trucks g_range
<- range(0, cars, trucks)

# Graph autos using y axis that ranges from 0 to max # value in
cars or trucks vector.  Turn off axes and # annotations (axis
labels) so we can specify them ourself plot(cars, type="o",
col="blue", ylim=g_range,
   axes=FALSE, ann=FALSE)

# Make x axis using Mon-Fri labels axis(1, at=1:5,
lab=c("Mon","Tue","Wed","Thu","Fri"))
```

```
# Make y axis with horizontal labels that display ticks at # every
4 marks. 4*0:g_range[2] is equivalent to c(0,4,8,12). axis(2,
las=1, at=4*0:g_range[2])

# Create box around plot box()

# Graph trucks with red dashed line and square points
lines(trucks, type="o", pch=22, lty=2, col="red")

# Create a title with a red, bold/italic font title(main="Autos",
col.main="red", font.main=4)

# Label the x and y axes with dark green text title(xlab="Days",
col.lab=rgb(0,0.5,0)) title(ylab="Total", col.lab=rgb(0,0.5,0))

# Create a legend at (1, g_range[2]) that is slightly smaller #
(cex) and uses the same line colors and points used by # the
actual plots legend(1, g_range[2], c("cars","trucks"), cex=0.8,
   col=c("blue","red"), pch=21:22, lty=1:2);
```

17.2 Bar charts

1cm

```
# Define the cars vector with 7 values
cars <- c(1, 3, 6, 4, 9, 5, 7)
# Graph cars
barplot(cars)
```

17.3 Boxplots

17.4 Setting graphical parameters

17.5 Miscellaneous

The following code can be used to make variations of the plots.
1cm

```
# Make an empty chart
plot(1, 1, xlim=c(1,5.5), ylim=c(0,7), type="n", ann=FALSE)

# Plot digits 0-4 with increasing size and color
text(1:5, rep(6,5), labels=c(0:4), cex=1:5, col=1:5)

# Plot symbols 0-4 with increasing size and color
points(1:5, rep(5,5), cex=1:5, col=1:5, pch=0:4)
text((1:5)+0.4, rep(5,5), cex=0.6, (0:4))
```

```
# Plot symbols 5-9 with labels
points(1:5, rep(4,5), cex=2, pch=(5:9))
text((1:5)+0.4, rep(4,5), cex=0.6, (5:9))

# Plot symbols 10-14 with labels
points(1:5, rep(3,5), cex=2, pch=(10:14))
text((1:5)+0.4, rep(3,5), cex=0.6, (10:14))

# Plot symbols 15-19 with labels
points(1:5, rep(2,5), cex=2, pch=(15:19))
text((1:5)+0.4, rep(2,5), cex=0.6, (15:19))

# Plot symbols 20-25 with labels
points((1:6)*0.8+0.2, rep(1,6), cex=2, pch=(20:25))
text((1:6)*0.8+0.5, rep(1,6), cex=0.6, (20:25))
```

17.6 Lattice Graphs

17.7 setting up

Execute the following command: 1cm

```
library(lattice)
```

For information on lattice, type: 1cm

```
help(package = lattice)
```

The examples in this section are generally drawn from the R documentation and Murrell (2006).

Murrell gives three reasons for using Lattice Graphics:

They usually look better. They can be extended in powerful ways. The resulting output can be annotated, edited, and saved

17.8 3 Dimensional Graphs

How to do a 3-d graph

Statistical Analysis using R

18 Confidence Intervals

18.1 Confidence Intervals for Large Samples

18.2 Confidence Intervals for Small Samples

19 Linear Models

The Slope and Intercept 1cm

20 ANOVA

20.1 Subsetting datasets by rows

Suppose we wish to divide a data frame into two different section. The simplest approach we can take is to create two new data sets, each assigned data from the relevant rows of the original data set.

Suppose our dataset "Info" has the dimensions of 200 rows and 4 columns. We wish to separate "Info" into two subsets , with the first and second 100 rows respectively. (We call these new subsets "Info.1" and "Info.2".)

```
Info.1 = Info[1:100,] #assigning "info" rows 1 to 100
Info.2 = Info[101:200,] #assigning "info" rows 101 to 200
```

More useful commands such as rbind() and cbind() can be used to manipulate vectors.

Part 2 Strategies for Data project

- Exploratory Data Analysis

  The first part of your report should contain some descriptive statistics and summary values. Also include some tests for normality.

- Regression You should have a data set with multiple columns, suitable for regression analysis. Familiarize yourself with the data, and decide which variable is the dependent variable.

  Also determine the independent variables that you will use as part of your analysis.

- Correlation Analysis Compute the Pearson correlation for the dependent variable with the respective independent variables. As part of your report, mention the confidence interval for the correlation estimate Choose the independent variables with the highest correlation as your candidate variables. For these independent variables, perform a series of simple linear regression procedures.

  ```
  lm(y~x1)
  lm(y~x2)
  ```

  Comment on the slope and intercept estimates and their respective p-values. Also comment on the coefficient of determination (multiple R squared). Remember to write the regression equations. Perform a series of multiple linear regressions, using pairs of candidate independent variables.

  ```
  lm(y~x1 +x2)
  lm(y~x2 +x3)
  ```

  Again, comment on the slope and intercept estimates, and their respective p-values. In this instance, compare each of the models using the coefficient of determinations. Which model explains the data best?

20.2 Analysis of residuals

Perform an analysis of regression residuals (you can pick the best regression model from last section). Are the residuals normally distributed? Histogram / Boxplot / QQ plot / Shapiro Wilk Test Also you can plot the residuals to check that there is constant variance.

```
y=rnorm(10)
x=rnorm(10)
fit1=lm(y~x)
res.fit1 = resid(fit1)
plot(res.fit1)
```

Probability Distributions

21 Generating a set of random numbers

1cm

```
rnorm(10)
```

22 The Poisson Distribution

23 The Binomial Distribution

24 Using probability distributions for simulations

25 Probability Distributions

25.1 Generate random numbers

Graphical methods

26 Scatterplots

27 Adding titles, lines, points to plots

```
library(MASS)
# Colour points and choose plotting symbols according to a levels of a factor
plot(Cars93$Weight, Cars93$EngineSize, col=as.numeric(Cars93$Type),
pch=as.numeric(Cars93$Type))

# Adds x and y axes labels and a title.
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")
# Add lines to the plot.
lines(x=c(min(Cars93$Weight), max(Cars93$Weight)), y=c(min(Cars93$EngineSize),
max(Cars93$EngineSize)), lwd=4, lty=3, col="green")
abline(h=3, lty=2)
abline(v=1999, lty=4)
# Add points to the plot.
```

Programming

28 Writing Functions

A simple function can be constructed as follows:

```
function_name <- function(arg1, arg2, ...){
commands
output
}
```

You decide on the name of the function. The function command shows R that you are writing a function. Inside the parenthesis you outline the input objects required and decide what to call them. The commands occur inside the .

The name of whatever output you want goes at the end of the function. Comments lines (usually a description of what the function does is placed at the beginning) are denoted by "#".

```
sf1 <- function(x){
x^2
}
```

This function is called sf1. It has one argument, called x. Whatever value is inputted for x will be squared and the result outputted to the screen. This function must be loaded into `R` and can then be called. We can call the function using:

```
sf1(x = 3)
#sf1(3)
[1] 9
To store the result into a variable x.sq
x.sq <- sf1(x = 3)
x.sq <- sf1(3)
> x.sq
[1] 9
```

Example

```
sf2 <- function(a1, a2, a3){
x <- sqrt(a1^2 + a2^2 + a3^2)
return(x)
}
```

This function is called sf2 with 3 arguments. The values inputted for a1, a2, a3 will be squared, summed and the square root of the sum calculated and stored in x. (There will be no output to the screen as in the last example.) The return command specifies what the function returns, here the value of x. We will not be able to view the result of the function unless we store it.

```
sf2(a1=2, a2=3, a3=4)
sf2(2, 3, 4) # Can't see result.
res <- sf2(a1=2, a2=3, a3=4)
res <- sf2(2, 3, 4) # Need to use this.
res
[1] 5.385165
```

We can also give some/all arguments default values.

```
mypower <- function(x, pow=2){
x^pow
}
```

If a value for the argument pow is not specified in the function call, a value of 2 is used.

```
mypower(4)
[1] 16
```

If a value for "pow" is specified, that value is used.

```
mypower(4, 3)
[1] 64
mypower(pow=5, x=2)
[1] 32
```

```
> code here
```

```
> code here
```

28.0.1 slide234

The TS are ¡equation here¿ The p-values for both of these tests are 0 and so there is enough evidence to reject H_0 and conclude that both 0 and 1 are not 0, i.e. there is a significant linear relationship between x and y. Also given are the R^2 and R^2 adjusted values. Here $R^2 = SSR/SST = 0.8813$ and so 88.13% of the variation in y is being explained by x. The final line gives the result of using the ANOVA table to assess the model t.

28.0.2 slide235

In SLR, the ANOVA table tests ¡EQN¿The TS is the F value and the critical value and p-values are found in the F tables with (p - 1) and (n - p) degrees of freedom.

This output gives the p-value = 0, therefore there is enough evidence to reject H0 and conclude that there is a signicant linear relationship between y and x. The full ANOVA table can be accessed using :

¡TABLE HERE¿

28.0.3 slide236

Once the model has been tted, must then check the residuals. The residuals should be independent and normally distributed with mean of 0 and constant variance. A Q-Q plot checks the assumption of normality (can also use a histogram as in MINITAB) while a, plot of the residuals versus fitted values gives an indication as to whether the assumption of constant variance holds.

¡HISTOGRAM¿

28.0.4 slidename

```
> xbar <- 83
> sigma <- 12
> n <- 5
> sem <- sigma/sqrt(n)
> sem
[1] 5.366563
> xbar + sem * qnorm(0.025)
[1] 72.48173
> xbar + sem * qnorm(0.975)
[1] 93.51827
```

28.0.5 Testing the slope (II)

You can compute a t test for that hypothesis simply by dividing the estimate by its standard error

$$t = \frac{\hat{\beta}}{S.E.(\hat{\beta})} \tag{1}$$

which follows a t distribution on n - 2 degrees of freedom if the true β is zero.

- The standard χ^2 test in chisq.test works with data in matrix form, like fisher.test does.
- For a 2 by 2 table, the test is exactly equivalent to prop.test.

```
> chisq.test(lewitt.machin)
```

28.0.6 Chi-squared Test

A chi^2 test is carried out on tabular data containing counts, e.g. the number of animals that died, the number of days of rain, the number of stocks that grew in value, etc.

Usually have two qualitative variables, each with a number of levels, and want to determine if there is a relationship between the two variables, e.g. hair colour and eye colour, social status and crime rates, house price and house size, gender and left/right handedness.

The data are presented in a contingency table: right-handed left-handed TOTAL

	right-handed	left-handed	TOTAL
Male	43	9	52
Female	44	4	48
TOTAL	87	13	100

The hypothesis to be tested is $H0$:There is no relationship between gender and left/right-handedness $H1$:There is a relationship between gender and left/right-handedness The values that we collect from our sample are called the observed (O) frequencies (counts). Now need to calculate the expected (E) frequencies, i.e. the values we would expect to see in the table, if H0 was true.

28.0.7 Two Sample Tests

All of the previous hypothesis tests and confidence intervals can be extended to the two-sample case.

The same assumptions apply, i.e. data are normally distributed in each population and we may want to test if the mean in one population is the same as the mean in the other population, etc.

Normality can be checked using histograms, boxplots and Q-Q plots as before. The Anderson-Darling test can be used on each group of data also.

28.0.8 Implementation

This can be carried out in R by hand:

```
>obs.vals <- matrix(c(43,9,44,4), nrow=2, byrow=T)
>row.tots <- apply(obs.vals, 1, sum)
>col.tots <- apply(obs.vals, 2, sum)
>exp.vals <- row.tots%o%col.tots/sum(obs.vals)
>TS <- sum((obs.vals-exp.vals)^2/exp.vals)
>TS
>[1] 1.777415
```

R Graphics

29 E

nhancing your scatter plots

29.1 Adding lines

Previously we have used scatter plots to plot bivariate data. They were constructed using the plot() command. Recall that we can use the arguments `xlim` and `ylim` to control the vertical and horizontal range of the plots, by specifying a two element vector (min and max) for each.

Using the `abline()` command, we can add lines to our scatter plots. We specify the argument according to the type of line required. A demonstration of three types of line is provided below. Additionally we change the colour of the added lines, by specifying a colour in the `col` argument. We can also change the line type to one of four possible types, using the `lty` argument.

The line types are follows

- `lty =1` Normal full line (default)
- `lty =2` Dashed line
- `lty =3` Dotted line
- `lty =4` Dash-dot line

```
x=rnorm(10)
y=rnorm(10)
plot(x,y)
plot(x,y,xlim=c(-4,4),ylim=c(-4,4))
abline(v =0 , lty =2 )    # add a vertical dotted line (here the y-axis) to the plot
abline(h=0  ,lty =3)    # add a horizontal dotted line (here the x-axis) to the plot
abline(a=0,b=1,col="green") # add a line to your plot with intercept "a" and slope "b"
```

29.2 Changing your plot character

To change the plot character (the symbol for each covariate, we supply an additional argument to the plot() function. This argument is formulated as pch=n where n is some number. Additionally we change the colour of the characters, by specifying a colour in the col argument.

```
plot(x,y,pch=15,col="red") #Square plot symbols
plot(x,y,pch=16,col="green") #Orb plot symbols
plot(x,y,pch=17,col="mauve") #Triangular plot symbols
plot(x,y,pch=36 ,col="amber") #Dollar sign plot symbols
```

Recall that we can add new variates to an existing scatterplot using the points() function. Remember to set the vertical and horizontal limits accordingly.

```
y1 = rnorm(10); y2 = rnorm(10)
plot(x,y1, pch=8,col="purple" ,xlim=c(-5,5),ylim=c(-5,5))
points(x,y2,pch=12,col="green")
```

29.3 Adding the regression model line

The `abline()` function can be used to add a regression model line by supplying as an argument the `coef()` values for intercept and slope estimates .These estimates can be inputted directly by using both functions in conjunction.

```
Fit1 =lm(y1~x);  coef(Fit1)
abline(coef(Fit1))
```

29.4 Adding a title

It is good practice to label your scatterplots properly. You can specify the following argument

- main="Scatterplot Example", This provides the plot with a title
- sub="Subtitle", This adds a subtitle
- xlab="X variable ", This command labels the x axis
- ylab="y variable ", This command labels the y-axis

We can also add text to each margin, using the `mtext()` command. We simply require the number of the side. (1 = bottom, 2=left,3=top,4=right). We can change the colour using the col argument.

```
plot(x,y,main="Scatterplot Example",   sub="subtitle",    xlab="X variable ", ylab="y variable ")
mtext("Enhanced Scatterplot", side=4,col="red ")
```

Alternatively , we can also use the command title() to add a title to an existing scatterplot.

```
title(main="Scatterplot Example)
```

30 Combining plots

It is possible to combine two plots. We used the graphical parameters command `par()` to create an array. Often we just require two plots side by side or above and below. We simply specify the numbers of rows and columns of this array using the `mfrow` argument, passed as a vector.

```
par(mfrow=c(1,2))
plot(x,y1) # draw first plot
plot(x,y2) # draw second plot
par(mfrow=c(1,1)) # reset to default setting.
```

31 Plot of single vectors

If only one vector is specified i.e. `plot(x)`, the plot created will simply be a scatter-plot of the values of x against their indices.

$plot(x)$ Suppose we wish to examine a trend that these points represent. We can connect each covariate using a line.

$plot(x, type = "l")$ If we wish to have both lines and points, we would input the following code. This is quite useful if we wish to see how a trend develops over time. $plot(x, type = "b")$

32 Exercise

The following are measurements (in mm) of a critical dimension on a sample of twelve engine crankshafts:

```
224.120  224.001  224.017  223.982  223.989  223.961
223.960  224.089  223.987  223.976  223.902  223.980
```

(a) Calculate the mean and standard deviation for these data. (b) The process mean is supposed to be ? = 224mm. Is this the case? Give reasons for your answer. (c) Construct a 99% confidence interval for these data and interpret. (d) Check that the normality assumption is valid using 2 suitable plots.

```
> x<-c(224.120,224.001,224.017,223.982 ,223.989 ,223.961,
+ 223.960 ,224.089 ,223.987 ,223.976 , 223.902 ,223.980)
>
> mean(x)
[1] 223.997
>
> sd(x)
[1] 0.05785405
>
> t.test(x,mu=224,conf.level=0.99)

        One Sample t-test

data:  x
t = -0.1796, df = 11, p-value = 0.8607
alternative hypothesis: true mean is not equal to 224
99 percent confidence interval:
 223.9451 224.0489
sample estimates:
mean of x
  223.997
```

33 Exercise 2

The height of 12 Americans and 10 Japanese was measured. Test for a difference in the heights of both populations.

```
Americans
174.68    169.87      165.07     165.95  204.99  177.61
170.11    170.71      181.52  167.68  158.62  182.90
Japanese
158.76   168.85   159.64   180.02   164.24
161.91   163.99   152.71   157.32   147.20
```

34 Exercise 3

A large group of students each took two exams. The marks obtained in both exams by a sample of eight students is given below

```
Student 1 2 3 4 5 6 7 8
Exam 1 57 76 47 39 62 56 49 81
Exam 2 67 81 62 49 57 61 59 71
```

Test the hypothesis that in the group as a whole the mean mark gained did not vary according to the exam against the hypothesis that the mean mark in the second exam was higher

```
>
> Ex1<-c(57,76,47,39,62,56,49,81)
> Ex2<-c(67,81,62,49,57,61,59,71)
> t.test(Ex1-Ex2)

        One Sample t-test

data:  Ex1 - Ex2
t = -1.6733, df = 7, p-value = 0.1382
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -12.065666   2.065666
sample estimates:
mean of x
       -5
```

35 Exercise 4

A poll on social issues interviewed 1025 people randomly selected from the United States. 450 of people said that they do not get enough time to themselves. A report claims that over 41% of the population are not satisfied with personal time. Is this the case?

```
> prop.test(450,1025,p=0.40,alternative="greater")

        1-sample proportions test with continuity correction

data:  450 out of 1025, null probability 0.4
X-squared = 6.3425, df = 1, p-value = 0.005894
alternative hypothesis: true p is greater than 0.4
95 percent confidence interval:
 0.413238 1.000000
```

```
sample estimates:
        p
0.4390244
```

Exercise 23b: A company wants to investigate the proportion of males and females promoted in the last year. 45 out of 400 female candidates were promoted, while 520 out of 3270 male candidates were promoted. Is there evidence of sexism in the company?

```
> x.vec=c(45,520)
> n.vec=c(400,3270)
>  prop.test(x.vec,n.vec)

 2-sample test for equality of proportions with continuity correction

data:  x.vec out of n.vec
X-squared = 5.5702, df = 1, p-value = 0.01827
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.08133043 -0.01171238
sample estimates:
   prop 1    prop 2
0.1125000 0.1590214
```

?

36 Exercise

Generate a histogram for data set 'scores', with an accompanying box-and-whisker plot. The colour of the histogram's bar should be yellow. The orientation for the boxplot should be horizontal.

```
scores <-c(23,19,22,22,19,20,25,26,26,19,24,23,17,21,28,26)

par(mfrow=c(2,1))  # two rows , one column

hist(scores,main="Distribution of scores",xlab="scores",col="yellow")

boxplot(scores ,horizontal=TRUE)

par(mfrow =c(1,1))  #reset
```

37 The R Programming Language

The R Programming Language is a statistical , data analysis , etc

R is a free software environment for statistical computing and graphics.

38 Writing R scripts

Editing your R script "R Editor".

- On the menu of the R console, click on file.
- Select open script or new script as appropriate.
- Navigate to your working directory and select your `.R` file
- A new dialogue box "`the R editor`" will open up.
- Input or select code you wish to compile.
- To compile this code, highlight it. Click the edit button on the menu.
- Select either "Run Line" or "Run Selection or All".
- Your code should now compile.
- To save your code, clink on "file" and then "`save as`".
- Save the file with the "`.R`" extension to your working directory.

39 Vector types

`R` operates on named data structures. The simplest such structure is the vector, which is a single entity consisting of an ordered collection of Numbers or characters.

- Numeric vectors
- Character vectors
- Logical vectors
- (also complex number vectors and colour vectors)

To create a vector, use the assignment operator and the concatenate function. For numeric vectors, the values are simply numbers.

```
># week8.r
>NumVec<-c(10.4,5.6,3.1,6.4)
```

Alternatively we can use the `assign()` command

For character vectors, the values are simply characters, specified with quotation marks.A logical vectors is a vector whose elements are TRUE, FALSE or NA

```
>CharVec<-c(‘‘blue", ‘‘green", ‘‘yellow")
>LogVec<-c(TRUE, FALSE)
```

40 Graphical data entry interface

`Data.entry()` is a useful command for inputting or editing data sets. Any changes are saved automatically (i.e. dont need to use the assignment operator). We can also used the `edit()` command, which calls the `R Editor`.

```
>data.entry(NumVec)
>NumVec <- edit(NumVec)
```

Another method of creating vectors is to use the following

```
numeric (length = n)
character (length = n)
logical (length = n)
```

These commands create empty vectors, of the appropriate kind, of length n. You can then use the graphical data entry interface to populate your data sets.

40.0.1 Accessing specified elements of a vector

The nth element of vector "Vec" can be accessed by specifying its index when calling "Vec".

```
>Vec[n]
```

A sequence of elements of vector "Vec" can be accessed by specifying its index when calling "Vec".

```
>Vec[l:u]
```

Omitting and deleting the nth element of vector "Vec"

```
>Vec[-n]
>Vec <- Vec[-n]
```

41 Reading data

41.1 inputting data

Concatenation

41.2 using help

?mean

41.3 Adding comments

41.4 Packages

The capabilities of R are extended through user-submitted packages, which allow specialized statistical techniques, graphical devices, as well as and import/export capabilities to many external data formats.

42 Managing Precision

- floor() -
- ceiling() -
- round() -
- as.integer() -

```
> pi
[1] 3.141593
> floor(pi)
[1] 3
> ceiling(pi)
[1] 4
> round(pi,3)
[1] 3.142
> as.integer(pi)
[1] 3
```

43 Basic Operations

43.1 Complex numbers

43.2 Trigonometric functions

44 Matrices

44.0.1 exponentials, powers and logarithms

```
>x^y
>exp(x)
>log(x)
>log(y)
#determining the square root of x
>sqrt(x)
```

44.1 vectors

```
R handles vector objects quite easily and intuitively.

> x<-c(1,3,2,10,5)    #create a vector x with 5 components
> x
[1]  1  3  2 10  5
> y<-1:5              #create a vector of consecutive integers
> y
[1] 1 2 3 4 5
> y+2                 #scalar addition
[1] 3 4 5 6 7
> 2*y                 #scalar multiplication
[1]  2  4  6  8 10
> y^2                 #raise each component to the second power
[1]  1  4  9 16 25
> 2^y                 #raise 2 to the first through fifth power
[1]  2  4  8 16 32
> y                   #y itself has not been unchanged
[1] 1 2 3 4 5
> y<-y*2
> y                   #it is now changed
[1]  2  4  6  8 10
```

44.1.1 Misc

`seq()` and `rep()` are useful commands for constructing vectors with a certain pattern.

44.2 Matrices

A matrix refers to a numeric array of rows and columns.

One of the easiest ways to create a matrix is to combine vectors of equal length using cbind(), meaning "column bind". Alternatively one can use rbind(), meaning "row bind".

44.2.1 Matrices Inversion

44.2.2 Matrices Multiplication

44.3 Data frame

A Data frame is

Descriptive Statistics

45 Basic Statistics

1cm

```
> X=c(1,4,5,7,8,9,5,8,9)
> mean(X);median(X)       #mean and median of vector
[1] 6.222222
[2] 7
> sd(X)                   #standard deviation of Vector
[1] 2.682246
> length(X)               #sample size of vector
[1] 9
> sum(X)
[1] 56
> X^2
[1]  1 16 25 49 64 81 25 64 81
> rev(X)
[1] 9 8 5 9 8 7 5 4 1
> sort(X)                 #items in ascending order
[1] 1 4 5 5 7 8 8 9 9
> X[1:5]
[1] 1 4 5 7 8
```

46 Summary Statistics

The R command `summary()` returns a summary statistics for a simple dataset. The R command `fivenum()` returns a summary statistics for a simple dataset, but without the mean. Also, the quartiles are computed a different way.

1cm

```
> summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.43   19.20   20.09   22.80   33.90
>
> fivenum(mtcars$mpg)
[1] 10.40 15.35 19.20 22.80 33.90
```

47 Bivariate Data

1cm

```
> Y=mtcars$mpg
> X=mtcars$wt
>
> cor(X,Y)          #Correlation
[1] -0.8676594
```

```
>
> cov(X,Y)          #Covariance
[1] -5.116685
```

48 Histograms

Histograms can be created using the `hist()` command. To create a histogram of the car weights from the Cars93 data set 1cm

```
hist(mtcars$mpg, main="Histogram of MPG (Data: MTCARS) ")
```

`R` automatically chooses the number and width of the bars. We can change this by specifying the location of the break points.

```
hist(Cars93$Weight, breaks=c(1500, 2050, 2300, 2350, 2400,
2500, 3000, 3500, 3570, 4000, 4500), xlab="Weight",
main="Histogram of Weight")
```