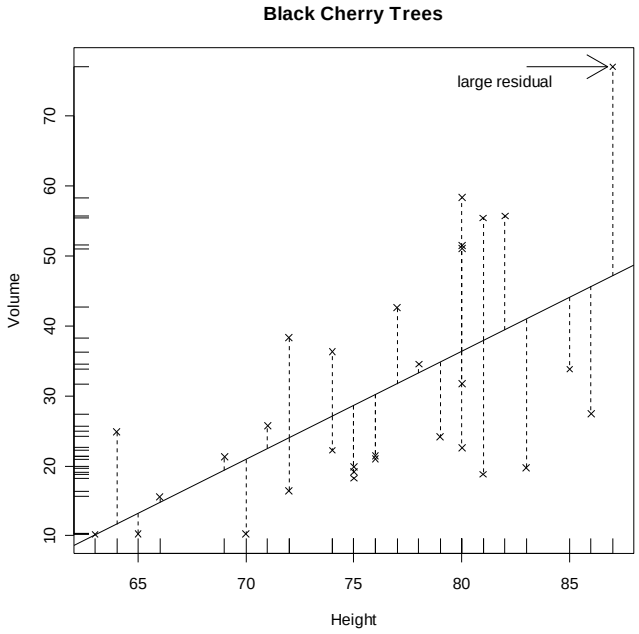
The R Guide

Version 2.5

W. J. Owen
Department of Mathematics and Computer Science
University of Richmond




wowen@richmond.edu.

Contact

Preface

R

R

Font Conventions

R

R **10 point Bold Courier New Font**

Acknowledgements

R

R

Contents

1. Overview and History

apropos(), c(), FALSE or F, help(), log(), ls(), matrix(), q(), rm(), TRUE or T

1.1 What is R?

R

R **free**

R

R

R R R R

R

volunteer R

http://www.r-project.org

R

R R

http://cran.us.r-project.org/

1.2 Starting and Quitting R

R

R

R R

Gui R

> R

R

R

```
R version 2.9.1 (2009-06-26)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

> R R R

\+

R q()

1.3 A Simple Example: the c() Function and the Assignment Operator

R c()

combines

2 5 1 6 5 5 4 1

R

```
> dieroll <- c(2,5,1,6,5,5,4,1)
> dieroll
[1] 2 5 1 6 5 5 4 1
>
```

- dieroll. R

DiEroLL

- **<-** **<** **-**
  dieroll *gets* **c(2,5,1,6,5,5,4,1)** R
  =
- **dieroll**
- **dieroll** [1]

R

↑ ↓
← →

1.4 The Workspace

R *workspace*
ls()

```
> ls()
[1] "dieroll"
```

dieroll

```
> newdieroll <- dieroll/2     # divide every element by two
> newdieroll
[1] 1.0 2.5 0.5 3.0 2.5 2.5 2.0 0.5
> ls()
[1] "dieroll"    "newdieroll"
```

- **newdieroll** **dieroll**
- # R
  #

rm()

```
> rm(newdieroll)     # this was a silly variable anyway
> ls()
[1] "dieroll"
```

R

Get in the habit of saving your work – it will probably help you in the future.

1.5 Getting Help

R `help()` `?`

If you have questions about any function in this manual, see the corresponding help file.

`log()` R

```
> help(log)
> ?log
```

Help Window

```
log                      package:base                      R Documentation

Logarithms and Exponentials

Description:

  `log' computes natural logarithms, `log10' computes common (i.e.,
  base 10) logarithms, and `log2' computes binary (i.e., base 2)
  logarithms. The general form `logb(x, base)' computes logarithms
  with base `base' (`log10' and `log2' are only special cases).
. . .          (skipped material)
Usage:

  log(x, base = exp(1))
  logb(x, base = exp(1))
  log10(x)
  log2(x)
. . .
Arguments:

       x: a numeric or complex vector.

    base: positive number.  The base with respect to which
          logarithms are computed.  Defaults to e=`exp(1)'.

Value:

  A vector of the same length as `x' containing the transformed
  values.  `log(0)' gives `-Inf' (when available).
. . .
```

log() R

x

base *e*

log10() log2()

```
> log(100)
[1] 4.60517
> log2(16)     # same as log(16,base=2) or just log(16,2)
[1] 4
> log(1000,base=10)     # same as log10(1000)
[1] 3
>
```

R log()

```
> log2(c(1,2,3,4))      # log base 2 of the vector (1,2,3,4)
[1] 0.000000 1.000000 1.584963 2.000000
>
```

R

apropos(). R

norm

```
> apropos("norm")
 [1] "dlnorm"    "dnorm"     "plnorm"         "pnorm"     "qlnorm"
 [6] "qnorm"     "qqnorm"    "qqnorm.default" "rlnorm"    "rnorm"
>
```

''

1.6 More on Functions in R

R

R

matrix()

Matrices

Description:

'matrix' creates a matrix from the given set of values.

Usage:

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE)
```

Arguments:

data: the data vector
nrow: the desired number of rows
ncol: the desired number of columns
byrow: logical. If `FALSE' the matrix is filled by columns, otherwise the matrix is filled by rows.
...

byrow TRUE FALSE T F

default

matrix()

```
> matrix()
```

NA

```
> a <- c(1,2,3,4,5,6,7,8)
> A <- matrix(a,nrow=2,ncol=4, byrow=FALSE)  # a is different from A
> A
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
>
```

byrow=FALSE

```
> A <- matrix(a,2,4)
```

1.7 Printing and Saving Your Work

R Console

Courier font **R Console**
R Console

1.8 Other Sources of Reference

R

R

- **The R program:**

  R

  An Introduction to R

- **Free Documentation:** R

  R for Beginners

  R

  R reference card

  R

- **Books:**

  Introductory Statistics with R

  R

  R

  Modern Applied Statistics with S

1.9 Exercises

R **mean** **median**

R **test**

info

Ident

1stR.txt

2. My Big Fat Greek[1] Calculator

+, -, *, /, ^, %*%,
abs(), as.matrix(), choose(), cos(), cumprod(), cumsum(), det(), diff(), dim(), eigen(), exp(), factorial(), gamma(), length(), pi, prod(), sin(), solve(), sort(), sqrt(), sum(), t(), tan().

2.1 Basic Math

R

R R

+,-,*,/,^.

```
> 2+3
[1] 5
> 3/2
[1] 1.5
> 2^3             # this also can be written as 2**3
[1] 8
> 4^2-3*2         # this is simply 16 - 6
[1] 10
> (56-14)/6 – 4*7*10/(5^2-5)  # this is more complicated
[1] -7
```

R

Name	Operation
sqrt()	
abs()	
sin() cos() tan()	?Trig
pi	π
exp() log()	
gamma()	
factorial()	
choose()	

```
> sqrt(2)
[1] 1.414214
> abs(2-4)
[1] 2
> cos(4*pi)
[1] 1
> log(0)          # not defined
[1] -Inf
> factorial(6)    # 6!
[1] 720
> choose(52,5)    # this is 52!/(47!*5!)
[1] 2598960
```

Σ Π

2.2 Vector Arithmetic

```
> x <- c(1,2,3,4)
> y <- c(5,6,7,8)
> x*y
[1]  5 12 21 32
> y/x
[1] 5.000000 3.000000 2.333333 2.000000
> y-x
[1] 4 4 4 4
> x^y
[1]     1    64  2187 65536
> cos(x*pi) + cos(y*pi)
[1] -2  2 -2  2
>
```

Name	Operation
length()	
sum()	
prod()	
cumsum(), cumprod()	
sort()	
diff()	

```
> s <- c(1,1,3,4,7,11)
> length(s)
[1] 6
> sum(s)       # 1+1+3+4+7+11
[1] 27
> prod(s)      # 1*1*3*4*7*11
[1] 924
> cumsum(s)
[1]  1  2  5  9 16 27
> diff(s)           # 1-1, 3-1, 4-3, 7-4, 11-7
[1] 0 2 1 3 4
> diff(s, lag = 2)   # 3-1, 4-1, 7-3, 11-4
[1] 2 3 4 7
```

2.3 Matrix Operations

R

matrix()

```
> a <- c(1,2,3,4,5,6,7,8,9,10)
> A <- matrix(a, nrow = 5, ncol = 2)   # fill in by column
> A
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

> B <- matrix(a, nrow = 5, ncol = 2, byrow = TRUE) # fill in by row
> B
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10

> C <- matrix(a, nrow = 2, ncol = 5, byrow = TRUE)
> C
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
>
```

R

Name	Operation
dim()	
as.matrix()	
%*%	
t()	
det()	
solve()	
eigen()	

A B C

```
> t(C)            # this is the same as A!!
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

> B%*%C
     [,1] [,2] [,3] [,4] [,5]
[1,]   13   16   19   22   25
[2,]   27   34   41   48   55
[3,]   41   52   63   74   85
[4,]   55   70   85  100  115
[5,]   69   88  107  126  145

> D <- C%*%B
> D
     [,1] [,2]
[1,]   95  110
[2,]  220  260

> det(D)
[1] 500

> solve(D)        # this is D^-1
      [,1]  [,2]
[1,]  0.52 -0.22
[2,] -0.44  0.19
>
```

2.4 Exercises

R

$$\left| - \right|$$

e^e

ln π $\sqrt{}$

$\boldsymbol{A}$ $\left(\right)$ $\boldsymbol{B}$ $\left(\right)$ $\boldsymbol{AB}$ $\boldsymbol{BA}$

dot product

3. Getting Data into R

$, :, attach(), attributes(), data(), data.frame(), edit(), file.choose(), fix(), read.table(), rep(), scan(), search(), seq()

c() R

```
> mykids <- c("Stephen", "Christopher")      # put text in quotes
> mykids
[1] "Stephen"     "Christopher"
```

R

3.1 Sequences

- **:**

```
> 1:9
[1] 1 2 3 4 5 6 7 8 9

> 1.5:10                    # you won't get to 10 here
[1] 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5

> c(1.5:10,10)              # we can attach it to the end this way
[1]  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.0

> prod(1:8)                 # same as factorial(8)
[1] 40320
```

- **seq()**

incremental value *length*

```
> seq(1,5)           # same as 1:5
[1] 1 2 3 4 5

> seq(1,5,by=.5)     # increment by 0.5
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
> seq(1,5,length=7)  # figure out the increment for this length
[1] 1.00000 1.66667 2.33333 3.00000 3.66667 4.33333 5.00000
```

- **rep()**

```
> rep(10,10)                  # repeat the value 10 ten times
 [1] 10 10 10 10 10 10 10 10 10 10

> rep(c("A","B","C","D"),2)   # repeat the string A,B,C,D twice
[1] "A" "B" "C" "D" "A" "B" "C" "D"

> matrix(rep(0,16),nrow=4)    # a 4x4 matrix of zeroes
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
>
```

3.2 Reading in Data: Single Vectors

c()

scan()

numerical

what =

```
> x <- scan()        # read what is typed into the variable x
```

R R

x

R

R

```
> passengers <- scan()
1: 2 4 0 1 1 2 3 1 0 0 3 2 1 2 1 0 2 1 1 2 0 0     # I hit return
23: 1 3 2 2 3 1 0 3     # I hit return again
31:                     # I hit return one last time
Read 30 items

> passengers            # print out the values
[1] 2 4 0 1 1 2 3 1 0 0 3 2 1 2 1 0 2 1 1 2 0 0 1 3 2 2 3 1 0 3
```

scan()

scan()

passengers.txt

```
> passengers <- scan("C:/passengers.txt")
Read 30 items
>
```

Notes:

- R
- _____
- **scan()**

  ```
  > dat <- scan("http://www...")
  ```

- _____

_____ **file.choose()**

3.3 Data Frames

3.3.1 Creating Data Frames

automobile type *driver seatbelt use*
R *data frame*

_____

_____.

edit() **fix()**

```
> new.data <- data.frame()    # creates an "empty" data frame
> new.data <- edit(new.data)  # request that changes made are
                              # written to data frame
```

OR

```
> new.data <- data.frame()    # creates an "empty" data frame
> fix(new.data)               # changes saved automatically
```

```
     var1 var2
```

```
> seatbelt <- c("Y","N","Y","Y","Y","Y","Y","Y","Y","Y",    # return
+ "N","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y",      # return
+ "Y","Y","N","Y","Y","Y","Y")
>
```

```
> car.dat <- data.frame(passengers,seatbelt)
```

```
> car.dat
   passengers seatbelt
1           2        Y
2           4        N
3           0        Y
4           1        Y
5           1        Y
6           2        Y
. . .
```

3.3.2 Datasets Included with R

R

```
> data()
```

trees

```
> data(trees)
```

trees
help(trees)

3.3.3 Accessing Portions of a Data Frame

$

trees

```
> trees
   Girth Height Volume
1    8.3     70   10.3
2    8.6     65   10.3
3    8.8     63   10.2
4   10.5     72   16.4
5   10.7     81   18.8
. . .
```

$

```
> trees$Height
 [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64 78
[22] 80 74 72 77 81 82 80 80 80 87
> sum(trees$Height)                # sum of just these values
[1] 2356
>
```

```
> trees[4,3]            # entry at forth row, third column
[1] 16.4

> trees[4,]             # get the whole forth row
  Girth Height Volume
4  10.5     72   16.4
>
```

$

R

search path **trees**

```
> attach(trees)
```

$

```
> Height
 [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64
[21] 78 80 74 72 77 81 82 80 80 80 87
```

search()

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

trees

R .GlobalEnv

package

detach()

attach() R

R attributes()

```
> attributes(trees)
$names
[1] "Girth"  "Height" "Volume"

$row.names
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11"
[12] "12" "13" "14" "15" "16" "17" "18" "19" "20" "21" "22"
[23] "23" "24" "25" "26" "27" "28" "29" "30" "31"

$class
[1] "data.frame"

>
```

trees

Height

```
> Height[Height > 75]      # pick off all heights greater than 75
 [1] 81 83 80 79 76 76 85 86 78 80 77 81 82 80 80 80 87
>
```

3.3.4 Reading in Datasets

read.table()

R
header = T
R
scan() **file.choose()**

```
> smith <- read.table(file.choose(), header=T)
```

read.csv()

3.3.5 Data files in formats other than ASCII text

R
R **R Data Import/Export**
R

3.4 Exercises

R

```
1 2 3 1 2 3 1 2 3
10.00000 10.04545 10.09091 10.13636 10.18182 10.22727 10.27273
10.31818 10.36364 10.40909 10.45455 10.50000
"1"   "2"   "3"   "banana"  "1"   "2"   "3"   "banana"
```

scan()
blahblah

schedule
coursenumber
coursedays
grade

stackloss R **Water.Temp**
Acid.Conc. **tempacid**

4. Introduction to Graphics

R

4.1 The Graphics Window

R *device*

R

- ___ ___

- *Recording* *History*

x11()

quartz()

4.2 Two Basic Graphing Functions

R

*4.2.1 The **`plot()`** Function*

R **plot()**

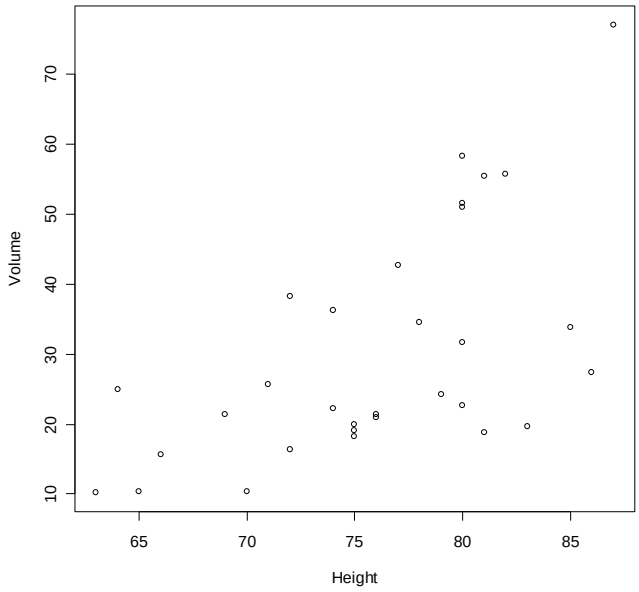
plot()

index

trees R

Height **Volume**

```
> plot(Height, Volume)     # object trees is in the search path
```



plot()

?par

*4.2.2 The **`curve()`** Function*

curve()

curve() **plot()**

```
curve(expr, from, to, add = FALSE, ...)
```

Arguments:
expr: an expression written as a function of 'x'

from, to: the range over which the function will be plotted.

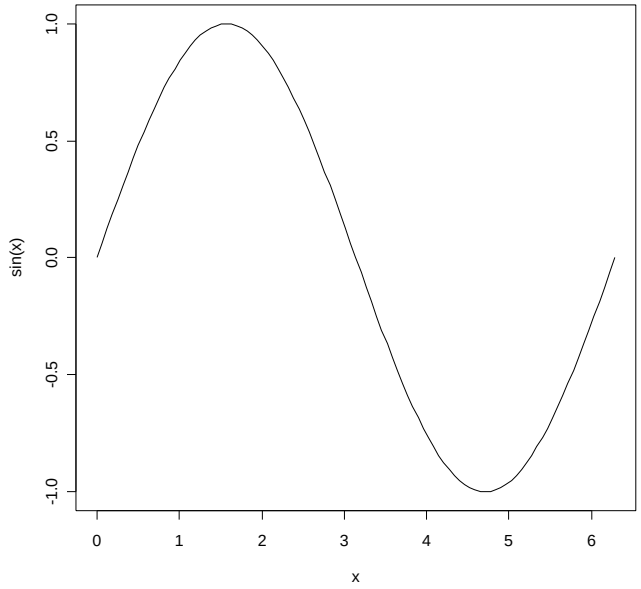
add: logical; if 'TRUE' add to already existing plot.

expr **'x'**

add **TRUE**

curve() π

```
> curve(sin(x), from = 0, to = 2*pi)
```



4.3 Graph Embellishments

add

Function	**Operation**
abline()	
arrows()	
lines()	
points()	
rug()	
segments()	**lines()**
text()	
title()	

4.4 Changing Graphics Parameters

```
par()
```

```
> par(mfrow = c(2, 2))  # gives a 2 x 2 layout of plots
> par(lend = 1)         # gives "butt" line end caps for line plots[2]
> par(bg = "cornsilk")  # plots drawn with this colored background
> par(xlog = TRUE)      # always plot x axis on a logarithmic scale
```

```
par()
```

```
par()
```

```
> oldpar <- par(no.readonly = TRUE)

      ... then, make your changes in par() ...

> par(oldpar)     # default (original) parameter settings restored
```

4.5 Exercises

R

```
> demo(graphics)
> demo(persp)     # for 3-d plots
> demo(image)
```

5. Summarizing Data

5.1 Numerical Summaries

R

numerical data

Name	Operation
mean()	
median()	
fivenum()	
summary()	
min(), max()	
quantile()	
var(), sd()	
cov(), cor()	

counts
categorical

table()

mtcars R

```
> data(mtcars)       # load in dataset
> attach(mtcars)     # add mtcars to search path
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
. . .
```

. mpg disp wt gear carb, cyl

```
> mean(hp)
[1] 146.6875
> var(mpg)
[1] 36.3241
> quantile(qsec, probs = c(.20, .80))     # 20th and 80th percentiles
   20%    80%
16.734 19.332
> cor(wt,mpg)           # not surprising that this is negative
[1] -0.8676594
```

```
> table(cyl)
cyl
 4  6  8
11  7 14
```

relative frequencies

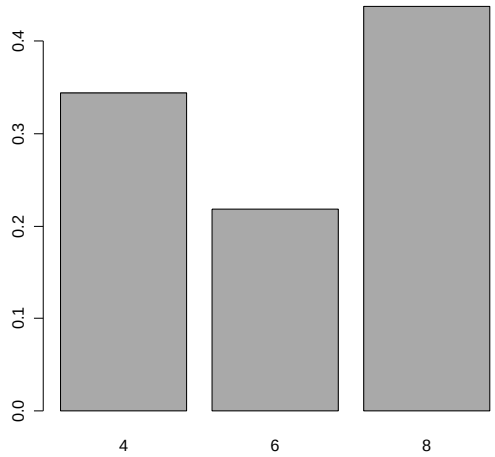
```
> table(cyl)/length(cyl)         # note: length(cyl) = 32
cyl
      4       6       8
0.34375 0.21875 0.43750
```

5.2 Graphical Summaries

- **barplot():**

barplot()
table()

```
> barplot(table(cyl)/length(cyl))   # use relative frequencies on
                                    # the y-axis
```



?barplot

- **hist():**

hist(x,breaks="Sturges",prob=FALSE,main=paste("Histogram of" ,xname))

Arguments:

x: a vector of values for which the histogram is desired.

breaks: one of:

*** a character string (in double quotes) naming an algorithm to compute the number of cells**

The default for 'breaks' is '"Sturges"': Other names for which algorithms are supplied are '"Scott"' and '"FD"'

*** a single number giving the number of cells for the histogram**

prob: logical; if FALSE, the histogram graphic is a representation of frequencies, the 'counts' component of the result; if TRUE, _relative_ frequencies ("probabilities"), component 'density', are plotted.

main: the title for the histogram

breaks

R

$\lceil \quad n + \rceil$

n $\lceil \rceil$

$bin\ width$

iqr

$\cdot iqr \cdot n^{-/}$

s

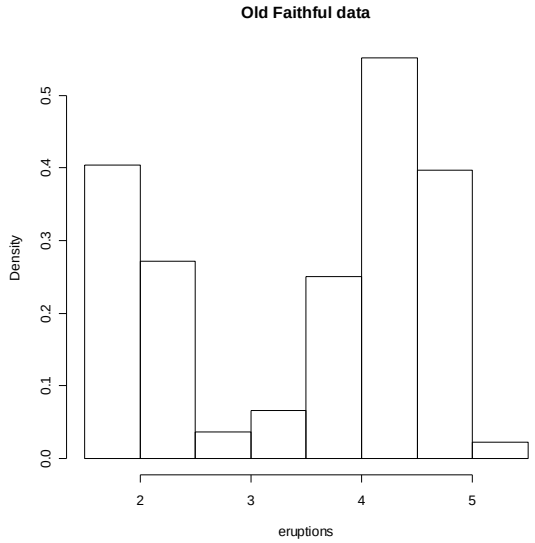
$\cdot s \cdot n^{-/}$

faithful R

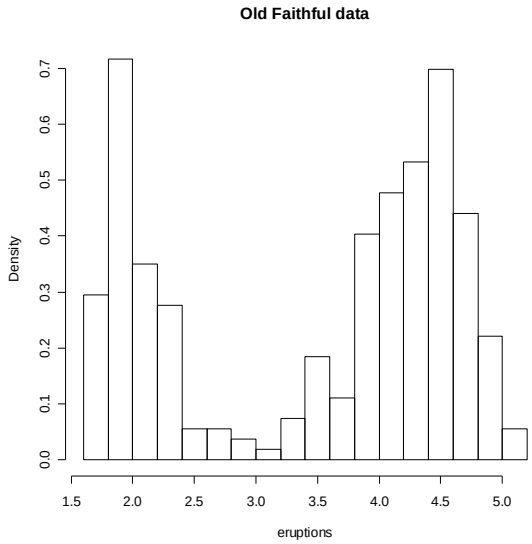
eruptions

waiting

```
> data(faithful)
> attach(faithful)
> hist(eruptions, main = "Old Faithful data", prob = T)
```




```
> hist(eruptions, main = "Old Faithful data", prob = T, breaks=18)
```



- **stem()**

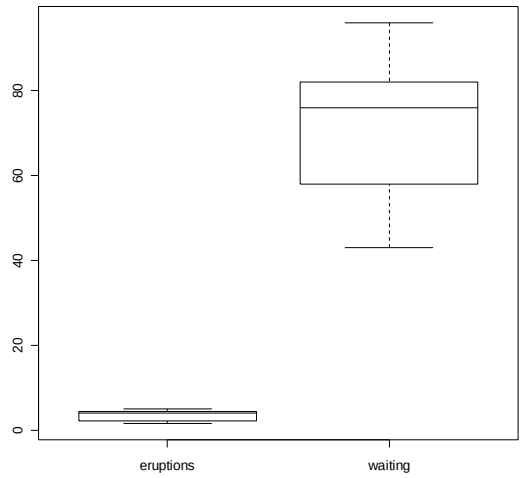
R Console **scale**

```
> stem(waiting)

  The decimal point is 1 digit(s) to the right of the |

  4 | 3
  4 | 55566666777788899999
  5 | 00000111111222223333333444444444
  5 | 555555666677788889999999
  6 | 00000022223334444
  6 | 555667899
  7 | 00001111123333333444444
  7 | 55555555666666666777777777778888888888888889999999999
  8 | 00000000111111111111222222222223333333333333334444444444
  8 | 55555566666677888888999
  9 | 00000012334
  9 | 6
```

- **boxplot()**

on the same graph

```
> boxplot(faithful)     # same as boxplot(eruptions, waiting)
```



?boxplot

- **ecdf():**

$\frac{}{n}$ x

x

```
> plot(ecdf(x))
```

- **qqnorm() and qqline()**

q q

```
> qqnorm(x)    # creates the NPP for values stored in x
> qqline(x)    # adds a reference line for the NPP
```

Name	**Operation**
pairs()	
persp()	
pie()	
qqplot()	
ts.plot()	

5.3 Exercises

stackloss R

stack.loss

stack.loss

6. Probability, Distributions, and Simulation

6.1 Distribution Functions in R

R

R

Distribution	R name	Additional arguments	Argument defaults
	beta	shape1, shape2	
	binom	size, prob	
	chisq	df (degrees of freedom)	
	unif	min, max	min = 0, max = 1
	exp	rate	rate = 1
F	f	df1, df2	
	gamma	shape, rate (or scale = 1/rate)	scale = 1
	geom	prob	
	hyper	m, n, k (sample size)	
	nbinom	size, prob	
	norm	mean, sd	mean = 0, sd = 1
	pois	lambda	
t	t	df	
	weibull	shape, scale	scale = 1

R **d** **p**
q **r**
rname

```
> drname(x, ...)  # the pdf/pmf at x (possibly a vector)
> prname(q, ...)  # the CDF at q (possibly a vector)
> qrname(p, ...)  # the p^th (possibly a vector) percentile/quantile
> rrname(n, ...)  # simulate n observations from this distribution
```

Note: since probability density/mass functions can be parameterized differently, R's definitions are given in the Appendix R

```
> x <- rnorm(100)          # simulate 100 standard normal RVs, put in x
> w <- rexp(1000,rate=.1)     # simulate 1000 from Exp(θ = 10)
> dbinom(3,size=10,prob=.25)  # P(X=3) for X ~ Bin(n=10, p=.25)
> dpois(0:2, lambda=4)        # P(X=0), P(X=1), P(X=2) for X ~ Poisson
> pbinom(3,size=10,prob=.25)  # P(X ≤ 3) in the above distribution
> pnorm(12,mean=10,sd=2)      # P(X ≤ 12) for X~N(mu = 10, sigma = 2)
> qnorm(.75,mean=10,sd=2)     # 3rd quartile of N(mu = 10,sigma = 2)
> qchisq(.10,df=8)            # 10th percentile of χ²(8)
> qt(.95,df=20)               # 95th percentile of t(20)
```

6.2 A Simulation Application: Monte Carlo Integration

$$I \quad \int_a^b g\ x\ dx$$

$g\ x$

I

n

n $a\ b$

$$\widehat{I} = b-a\ \frac{}{n}\sum_{i=}^{n} g \quad {}_i$$

n

$$I_n \rightarrow\ b-a\ \mathbf{E}[g\ \mathbf{X}\ ]$$

$$[g \quad] = \int_a^b g\ x\ \frac{}{b-a}dx = \left(\frac{}{b-a}\right)I$$

I_n I n

e.g.

$$\int^{\pi} \quad x \quad -x \quad dx$$

```
> u <- runif(1000000, min=0, max=pi/2)
> pi/2*mean(4*sin(2*u)*exp(-u^2))   # a=0, b=pi/2, so b–a=pi/2
[1] 2.189178
```

```
> u <- runif(1000000, min=0, max=pi/2)  # generate a new uniform vector
> pi/2*mean(4*sin(2*u)*exp(-u^2))   # a=0, b=pi/2, so b–a=pi/2
[1] 2.191414
```

R **integrate()** **?integrate**

6.3 Graphing Distributions

6.3.1 Discrete Distributions

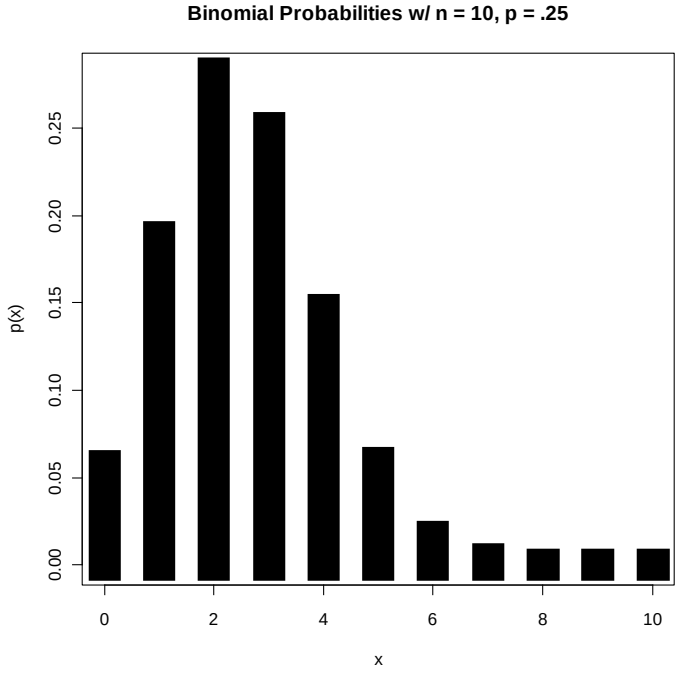
plot()

R

type **lwd**

n *p*

```
   > x <- 0:10
   > y <- dbinom(x, size=10, prob=.25)    # evaluate probabilities
   > plot(x, y, type = "h", lwd = 30, main = "Binomial Probabilities w/ n
= 10, p = .25", ylab = "p(x)", lend ="square")   # not a hard return here!
```



x

x

y **"h"**

lwd = 30

x **y**

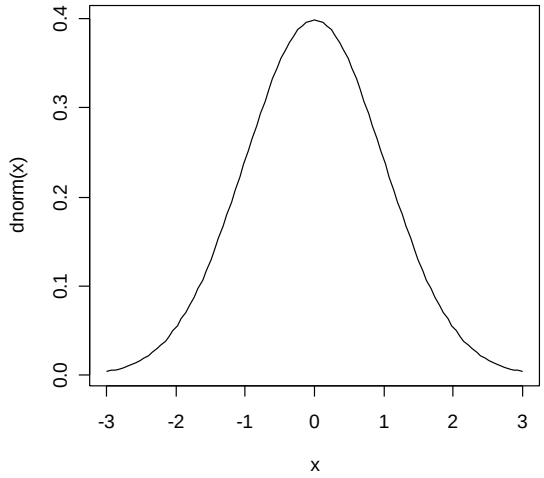
```
   > plot(0:10, dbinom(x, size=10, prob=.25), type = "h", lwd = 30)
```

6.3.2 Continuous Distributions

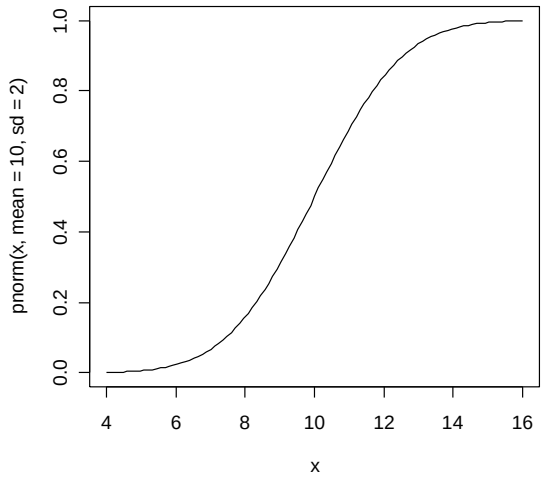
curve()

R

```
> curve(dnorm(x), from = -3, to = 3)  # the standard normal curve
```




```
> curve(pnorm(x, mean=10, sd=2), from = 4, to = 16)  # a normal CDF
```

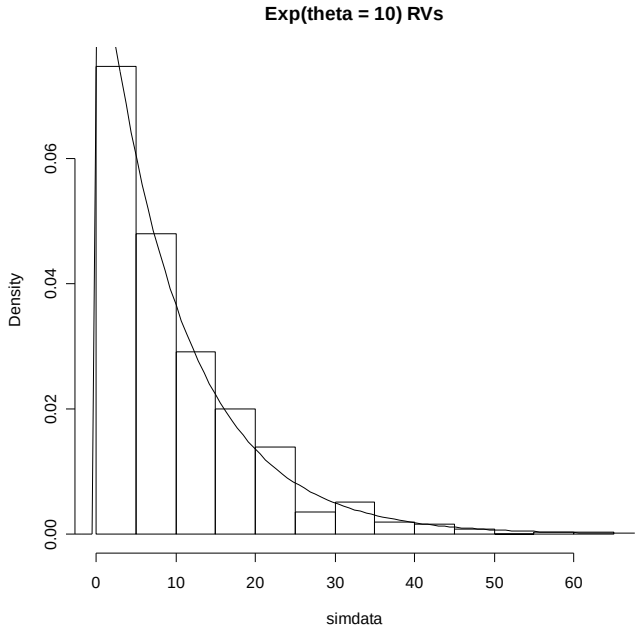


qnorm()

curve()

from **to** R

```
> simdata <- rexp(1000, rate=.1)
> hist(simdata, prob = T, breaks = "FD", main="Exp(theta = 10) RVs")
> curve(dexp(x, rate=.1), add = T)     # overlay the density curve
```



6.4 Random Sampling

R

resampling methods

R **sample()**

sample(x, size, replace = FALSE, prob = NULL)

Arguments:

x: Either a (numeric, complex, character or logical) vector of more than one element from which to choose, or a positive integer.

size: non-negative integer giving the number of items to choose.

replace: Should sampling be with or without replacement?

prob: An optional vector of probability weights for obtaining the elements of the vector being sampled.

```
> sample(1:100, 1)              # choose a number between 1 and 100
[1] 34

> sample(1:6, 10, replace = T)   # toss a fair die 10 times
 [1] 1 3 6 4 5 2 2 5 4 5

> sample(1:6, 10, T, c(.6,.2,.1,.05,.03,.02))   # not a fair die!!
 [1] 1 1 2 1 4 1 3 1 2 1

> urn <- c(rep("red",8),rep("blue",4),rep("yellow",3))
> sample(urn, 6, replace = F)    # draw 6 balls from this urn
[1] "yellow" "red"    "blue"   "yellow" "red"    "red"
```

6.5 Exercises

n p

α θ

t

λ x

$\int x \, dx$

n

μ σ

$_n x$ x

sample()

---

integrate(f = function(x) exp(x^3), lower = 0, upper = 2)

7. Statistical Methods

R

7.1 One and Two-sample t-tests

t.test()

t

Usage:

```
t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)
```

Arguments:

x, y: numeric vectors of data values. If y is not given, a one sample test is performed.

alternative: a character string specifying the alternative hypothesis, must be one of `"two.sided"' (default), `"greater"' or `"less"'. You can specify just the initial letter.

mu: a number indicating the true value of the mean (or difference in means if you are performing a two sample test). Default is 0.

paired: a logical indicating if you want the paired t-test (default is the independent samples test if both x and y are given).

var.equal: (for the independent samples test) a logical variable indicating whether to treat the two variances as being equal. If `TRUE', then the pooled variance is used to estimate the variance. If 'FALSE' (default), then the Welsh suggestion for degrees of freedom is used.

conf.level: confidence level (default is 95%) of the interval estimate for the mean appropriate to the specified alternative hypothesis.

t.test()

confidence bound

trees

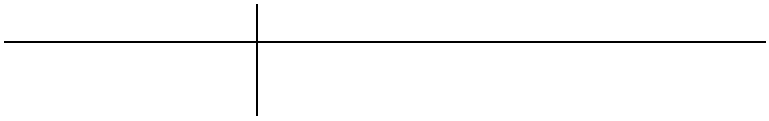
height

```
> data(trees)
> t.test(trees$Height, mu = 70)
```

```
        One Sample t-test

data:  trees$Height
t = 5.2429, df = 30, p-value = 1.173e-05
alternative hypothesis: true mean is not equal to 70
95 percent confidence interval:
 73.6628 78.3372
sample estimates:
mean of x
       76
```

R

```
> drug <- c(15, 10, 13, 7, 9, 8, 21, 9, 14, 8)
> plac <- c(15, 14, 12, 8, 14, 7, 16, 10, 15, 12)
> t.test(drug, plac, alternative = "less", var.equal = T)

        Two Sample t-test

data:  drug and plac
t = -0.5331, df = 18, p-value = 0.3002
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
     -Inf 2.027436
sample estimates:
mean of x mean of y
     11.4      12.3
```



R R

```
> add <- c(24.6, 18.9, 27.3, 25.2, 22.0, 30.9)
> noadd <- c(23.8, 17.7, 26.6, 25.1, 21.6, 29.6)
> t.test(add, noadd, paired=T, alt = "greater")

        Paired t-test

data:  add and noadd
t = 3.9994, df = 5, p-value = 0.005165
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.3721225       Inf
sample estimates:
mean of the differences
                   0.75
```

7.2 Analysis of Variance (ANOVA)

aov()

```
> aov(x ~ a)              # one-way ANOVA model
> aov(x ~ a + b)          # two-way ANOVA with no interaction
> aov(x ~ a + b + a:b)    # two-way ANOVA with interaction
> aov(x ~ a*b)            # exactly the same as the above
```

x *all*

a **b** R *numerical*

categorical
factor variables

7.2.1 Factor Variables

Type	*A*	*B*	*C*
Strength lb/in^2			

R

```
> Str <- c(3225,3320,3165,3145,3220,3410,3320,3370,3545,3600,3580,3485)
> Type <- c(rep("A",4), rep("B",4), rep("C",4))
```

Type **Str**

R **Type** *character* R

factor()

```
> Type <- factor(Type)     # consider these alternative expressions[7]
> Type
 [1] A A A A B B B B C C C C
Levels: A B C
>
```

Type **Levels**

```
> levels(Type)
[1] "A" "B" "C"
```

Str

```
> tapply(Str,Type,mean)
      A       B       C
3213.75 3330.00 3552.50
```

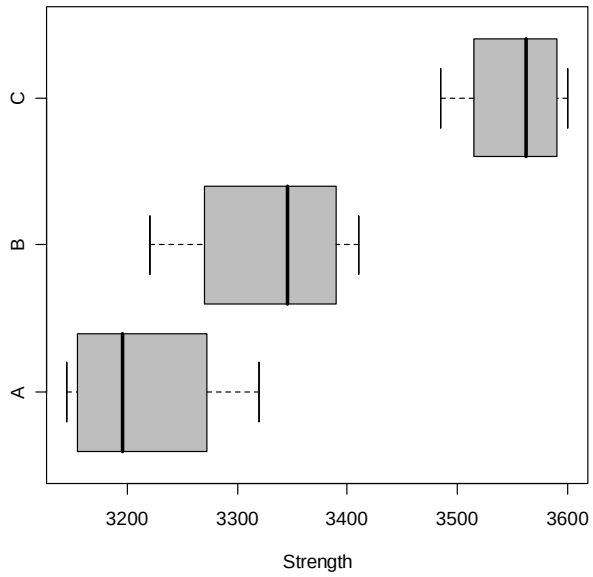
tapply() *table* *applied*

```
> tapply(Str,Type,var)
       A        B        C
6172.917 6733.333 2541.667
```

boxplot()

```
> boxplot(Str ~ Type, horizontal = T, xlab="Strength", col = "gray")
```

> Type <- factor(rep(LETTERS[1:3], each = 4)
> Type <- gl(3, 4, lab = LETTERS[1:3])



7.2.2 The ANOVA Table

aov()

```
> anova.fit <- aov(Str ~ Type)
```

anova.fit *linear model object* R **summary()**

```
> summary(anova.fit)
            Df Sum Sq Mean Sq F value    Pr(>F)
Type         2 237029  118515  23.016 0.0002893 ***
Residuals    9  46344    5149
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

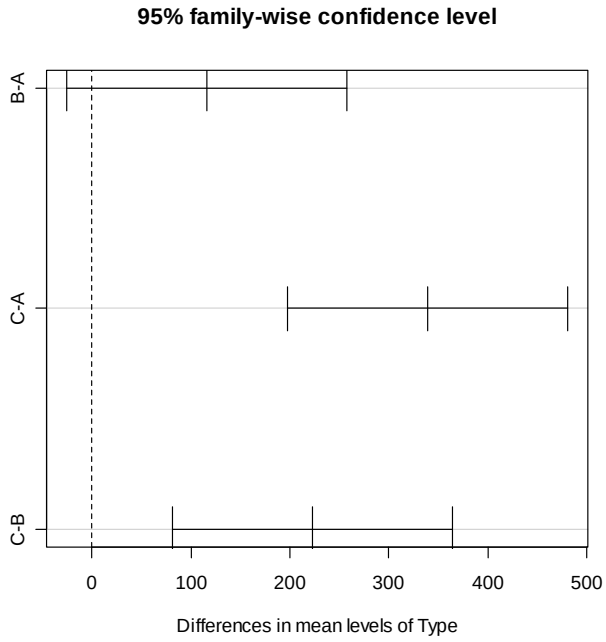
7.2.3 Multiple Comparisons

---

qtukey()

```
> TukeyHSD(anova.fit)  # the default is 95% CIs for differences
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Str ~ Type)

$Type
      diff       lwr      upr     p adj
B-A 116.25 -25.41926 257.9193 0.1085202
C-A 338.75 197.08074 480.4193 0.0002399
C-B 222.50  80.83074 364.1693 0.0044914

>
```

```
> plot(TukeyHSD(anova.fit))
```



7.3 Linear Regression

R

lm()

---

nls()

```
> lm(y ~ x)                   # simple linear regression (SLR) model
> lm(y ~ x1 + x2)             # a regression plane
> lm(y ~ x1 + x2 + x3)        # linear model with three regressors
> lm(y ~ x – 1)               # SLR w/ an intercept of zero
> lm(y ~ x + I(x^2))             # quadratic regression model
> lm(y ~ x + I(x^2) + I(x^3))    # cubic model
```

y ~ x

y **x**

x1 x2 x3

I()

R

lm()

cars **speed**

dist

speed **dist**

cars

```
> fit <- lm(dist ~ speed)
```

fit *linear model object*

```
> attributes(fit)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

fit

fit$residuals *residual plot*

```
> fit

Call:
lm(formula = dist ~ speed)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

summary()

```
> summary(fit)

Call:
lm(formula = dist ~ speed)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-Squared: 0.6511,     Adjusted R-squared: 0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.490e-12
```

Residual standard error σ

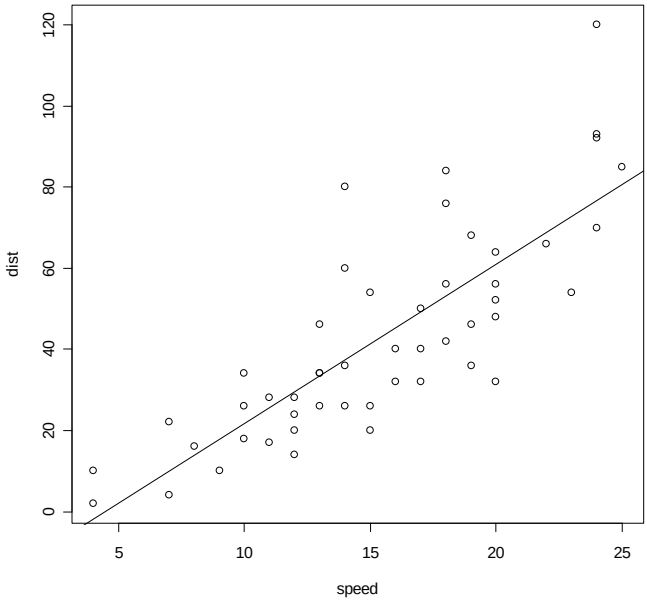
anova(fit)

```
> anova(fit)
Analysis of Variance Table

Response: dist
          Df  Sum Sq Mean Sq F value    Pr(>F)
speed      1 21185.5 21185.5  89.567 1.490e-12 ***
Residuals 48 11353.5   236.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

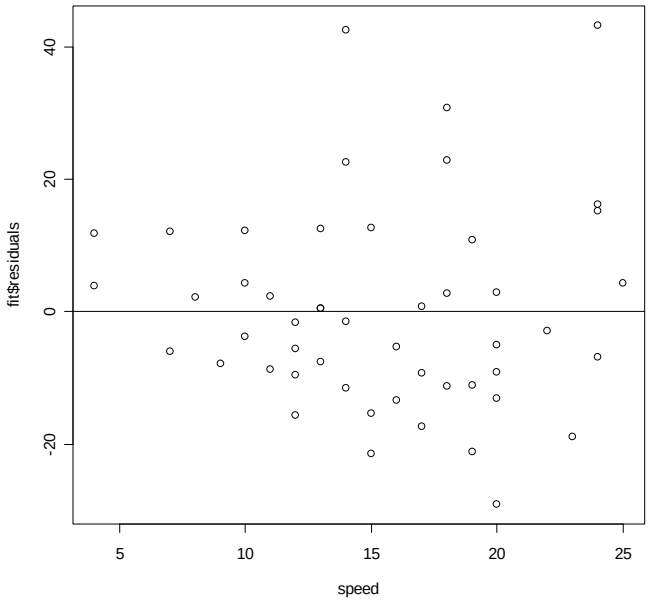
confint()

```
> confint(fit)       # the default is a 95% confidence level
                2.5 %    97.5 %
(Intercept) -31.167850 -3.990340
speed         3.096964  4.767853
```

```
> plot(speed, dist)        # first plot the data
> abline(fit)              # could also type abline(-17.579, 3.932)
```




```
> plot(speed, fit$residuals)
> abline(h=0)                    # add a horizontal reference line at 0
```



predict.lm()

```
> predict.lm(fit, newdata=data.frame(speed=c(15,22)),int="conf")
    fit      lwr      upr
1 41.40704 37.02115 45.79292
2 68.93390 61.89630 75.97149

> predict.lm(fit,newdata=data.frame(speed=c(15,22)),int="predict")
    fit      lwr       upr
1 41.40704 10.17482  72.63925
2 68.93390 37.22044 100.64735
```

newdata

7.4 Chi-square Tests

R

chisq.test()

?chisq.test

```
chisq.test(x, y = NULL, correct = TRUE, p = rep(1/length(x),length(x)))
```

Arguments:

x: a vector, table, or matrix.

y: a vector; ignored if 'x' is a matrix.

correct: a logical indicating whether to apply continuity correction when computing the test statistic.

p: a vector of probabilities of the same length of 'x'.

7.4.1 Goodness of Fit

x

table() **p**

p

Die face	
Frequency	

```
> counts <- c(43, 49, 56, 45, 66, 41)
> probs <- rep(1/6, 6)
> chisq.test(counts, p = probs)

        Chi-squared test for given probabilities

data:  counts
X-squared = 8.96, df = 5, p-value = 0.1107

>
```

7.4.2 Contingency Tables

R

table()

	Male	*Female*
Normal		
Color-blind		

```
> color.blind <- matrix(c(442, 514, 38, 6), nrow=2, byrow=T)
> color.blind
     [,1] [,2]
[1,]  442  514
[2,]   38    6
>
```

```
> dimnames(color.blind) <- list(c("normal","c-b"),c("Male","Female"))
> color.blind
       Male Female
normal  442    514
c-b      38      6
```

color.blind

```
> chisq.test(color.blind, correct=F)   # no correction for this one

        Pearson's Chi-squared test

data:  color.blind
X-squared = 27.1387, df = 1, p-value = 1.894e-07
```

chisq.test()

```
> out <- chisq.test(color.blind, correct=F)
> attributes(out)
$names
[1] "statistic" "parameter" "p.value"   "method"    "data.name"
[6] "observed"  "expected"  "residuals"

$class
[1] "htest"
```

```
> out$expected             # these are the expected counts
         Male Female
normal 458.88 497.12
c-b     21.12  22.88
```

7.5 Other Tests

R

- **prop.test()**

p

binom.test()

- **var.test()**

- **cor.test()**

- **wilcox.test()**

t.test()

- **kruskal.test()**

lm()

- **friedman.test()**

- **ks.test()**

- **shapiro.test()**

8. Advanced Topics

R

8.1 Scripts

source()
don't

>

```
x1 <- rnorm(500)     # Simulate 500 standard normals
x2 <- rnorm(500)     #               ""
x3 <- rnorm(500)     #               ""
y1 <- x1 + x2
y2 <- x2 + x3
r <- cor(y1,y2)
```

corsim.R

```
> source("C:/corsim.R")
> r
[1] 0.5085203
>
```

r **x1 x2 x3 y1 y2**

8.2 Control Flow

R

?Control

```
if
if else
for
while
repeat
break
next
```

Operator	Meaning
==	
!=	
<, <=	
>, >=	
&	
|	

x

```
> x <- rnorm(1)
> if(x > 2) print("This value is more than the 97.72 percentile")
```

x z

for

{}

```
> n <- 50
> for(i in 1:100)  {
+   x[i] <- mean(rexp(n, rate = .5))
+   z[i] <- (x[i] – 2)/sqrt(2/n)
+ }
>
```

π

π n

s s/n

4*s/n π

```
> eps <- 1; s <- 0; n <- 0    # initialize values
> while(eps > .001) {
+   n <- n + 1
+   x <- runif(1,-1,1)
+   y <- runif(1,-1,1)
+   if(x^2 + y^2 < 1) s <- s + 1
+   pihat <- 4*s/n
+   eps = abs(pihat - pi)
+ }
>

> pihat        # this is our estimate
[1] 3.141343
> n            # this is how many steps it took
[1] 1132
```

8.3 Writing Functions

R

```
fname <- function(arg1, arg2, ...)  { R code }
```

fname arg1, arg2, ...

R

if

```
> f1 <- function(a, b) {
+ # This function returns the maximum of two scalars or the
+ # statement that they are equal.
+ if(is.numeric(c(a,b))) {
+   if(a < b) return(b)
+   if(a > b) return(a)
+   else print("The values are equal")    # could also use cat()
+ }
+ else print("Character inputs not allowed.")
+ }
>
```

f1

a b

is.numeric() TRUE

FALSE

```
> f1(4,7)
[1] 7

> f1(pi,exp(1))
[1] 3.141593

> f1(0,exp(log(0)))
[1] "The values are equal"

> f1("Stephen","Christopher")
[1] "Character inputs not allowed."
```

f1

fix() **edit()**

$$P = A \cdot \frac{r/1200}{1-(1+r/1200)^{-12y}}$$

A **r**
y **P**
R

```
> mortgage <- function(A = 100000, r = 6, y = 30)  {
+ P <- A*r/1200/(1-(1+r/1200)^(-12*y))
+ return(round(P, digits = 2))
+ }
```

round()

```
> mortgage()
[1] 599.55
> mortgage(200000,5.5)     # use default 30 year loan value
[1] 1135.58
> mortage(y = 15)          # D'oh!  Bad spelling...
Error: couldn't find function "mortage"
> mortgage(y = 15)
[1] 843.86
```

8.4 Numerical Methods

R

- **optimize()**

- **uniroot()**

n

θ

$$f\ x \quad \theta x^{\theta-} \quad -x^{\theta} \quad x \quad \theta$$

log likelihood

$$l\ \theta \quad n \quad \theta + \theta - \quad \sum_{i=}^{n} \quad x_i \ - \sum_{i=}^{n} x_i^{\theta}$$

$$l'\ \theta \quad \frac{n}{\theta} + \sum_{i=}^{n} \quad x_i \ - \sum_{i=}^{n} x_i^{\theta} \quad x_i$$

θ _____ θ θ *maximizes* $l\ \theta$ $x \quad x \quad x_n$ *root* $l'\ \theta$

θ

```
> data <- rweibull(20, shape=4, scale=1)
> data
 [1] 0.6151766 0.9417053 0.8244651 1.3818235 0.9866513 0.6121007 1.1391467
 [8] 0.8711759 1.3590739 0.4826331 1.0755436 1.0318024 1.1728524 0.8062276
[15] 1.3630116 1.2094519 0.7547847 1.1178163 0.6184907 0.9310806
```

```
> l <- function(theta, x)
+ {
+ return(length(x)*log(theta) + (theta-1)*sum(log(x)) - sum(x^theta))
+ }
```

θ

```
> optimize(l, interval=c(0,20), x = data, max = T)
$maximum
[1] 3.874208            ← value that achieves the maximum

$objective
[1] -1.834197           ← value of the function at the maximum
```

θ

l′ θ
uniroot()

8.5 Exercises

- *n* μ
  σ
- *m*
-
-

n *m* μ σ

dl *l′* θ
uniroot()

Appendix: Well-known probability density/mass functions in R

_____ $p\ x$

- **Binomial** `size` n `prob` p

$$p\ x\ = \binom{n}{x} p^{x}\ \ - p\ ^{n-x}\ \ x = \qquad\qquad n$$

- **Geometric** `prob` p

$$p\ x\ = p\ \ - p\ ^{x}\ \ x =$$

- **Hypergeometric** m n k

$$p\ x\ = \frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}}\ \ x \qquad \_\_\_\_$$

- **Negative binomial** `size` n `prob` p

$$p\ x\ = \binom{x+n-}{n-} p^{n}\ \ - p\ ^{x}\ \ x =$$

- **Poisson** `lambda` λ

$$p\ x\ = \frac{\lambda^{x}}{x} e^{-\lambda}\ \ x =$$

_____ $f\ x$

- **Beta** `shape1` a `shape2` b

$$f\ x\ = \frac{\Gamma\ a+b}{\Gamma\ a\ \Gamma\ b} x^{a-}\ \ - x\ ^{b-}\ \ \leq x \leq$$

- **Continuous uniform**

$$f\ x\ = \frac{\qquad\qquad}{-}\ \ \leq x \leq$$

- **Exponential** `rate` λ

$$f\ x\ = \lambda \qquad -\lambda x\ \ x \geq$$

- **Gamma** `shape` a `scale` s

$$f\ x\ = \frac{}{\Gamma\ a\ s^{a}} x^{a-} \qquad -x\ \ s\ \ x \geq$$

- **Normal** `mean` μ `sd` σ

$$f\ x\ = \frac{}{\sigma\sqrt{\ \pi}} \qquad \left[-\frac{x-\mu}{\sigma} \right] -\infty < x < \infty$$

- **Weibull** `shape` a `scale` b

$$f\ x\ = \frac{a}{b}\left(\frac{x}{b}\right)^{a-} \qquad \left[-\left(\frac{x}{b}\right)^{a} \right] x \geq$$

References

The New S Language

Zeit. Wahr. ver. Geb **57**

Monte Carlo Statistical Methods

Biometrika **66**

J. Amer. Statist. Assoc **21**

Index