Statistical Computing with R – MATH 6382[1,*]
Set 1 (Intro to R)

Tamer Oraby

UTRGV

tamer.oraby@utrgv.edu

[1] Partly based on "Statistical Computing with R" by Maria Rizzo, and "Introductory Statistics with R" by Peter Dalgaard, second edition. See also http://manuals.bioinformatics.ucr.edu/home/programming-in-r

[*] Last updated January 25, 2017

R

- R project at http://www.r-project.org/
- Comprehensive R Archive Network (CRAN) (Austria) http://cran.R-project.org/
- R is based on S
- Quick-R at http://www.statmethods.net/
- R-Tutorial at http://www.r-tutor.com/
- Wikispace http://rutrgv.wikispaces.com/

Different packages in R

- Packages or libraries (default is *base*)
- `library()` to know which packages are installed
- `install.packages("package name")` to install a new package
- `library(package name)` to use an installed package

R Environment

- You can use the console or save a script through file's menu
- Use the command prompt > to do calculations
  For example
  ```
  > exp(-2*pi)*log(2)
  ```
  whose output is
  ```
  [1] 0.001294413
  ```
  where `[1]` is a sign for line one
- ```
  > LETTERS[1:4]
  [1] "A" "B" "C" "D"
  ```
- ```
  > letters[1:4]
  [1] "a" "b" "c" "d"
  ```

R Environment

- A vector can be made using concatenation as in

```
> c(-2,-1,0,1,2)
[1] -2 -1  0  1  2
```

To assign the vector to `x` use the arrow sign `<-` (assignment operator) as in

```
> x<-c(-2,-1,0,1,2)
```

but without an output, but you can see what is `x` by typing

```
> x
```

giving

```
[1] -2 -1  0  1  2
> length(x)
[1] 5
```

- It could be also done through `> x<-seq(-2,2,1)` and is the same as `> x<- -2:2`

Some R functions

```
# write any comment after the hashtag
> sqrt(4)
[1] 2
> floor(4.1)
[1] 4
> ceiling(4.1)
[1] 5
> factorial(4)
[1] 24
> choose(4,2)
[1] 6
> sort(c(2,5,1))
[1] 1 2 5
> rank(c(2,5,1))
[1] 2 3 1
```

Some R functions

```
> tail(c(10,20,30),1)
[1] 30
> tail(c(10,20,30),2)
[1] 20 30
> head(c(10,20,30),1)
[1] 10
> rev(c(10,20,30))
[1] 30 20 10
> x<-c(10,20,30,40,50)
> x[-4]
[1] 10 20 30 50
> x[-c(3,5)]
[1] 10 20 40
```

Some R Math functions

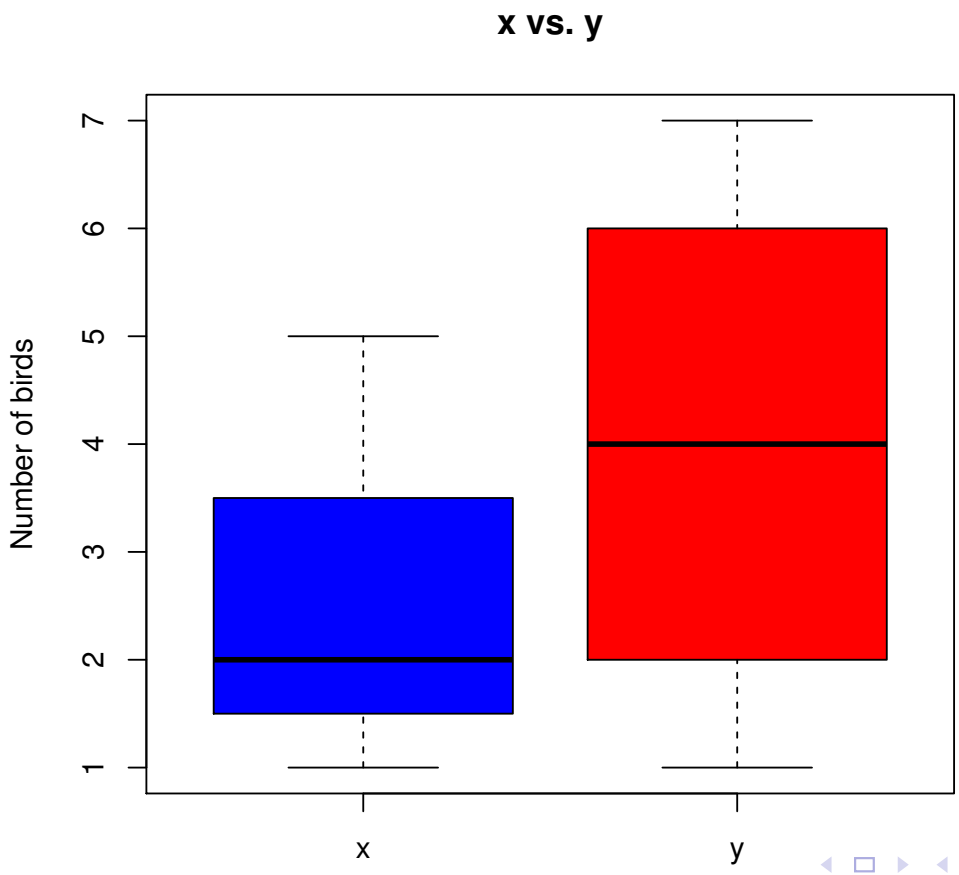
```
> optim()
[1]
> uniroot()
[1]
> integrate()
[1]
> deriv()
[1]
```

Some R Stat functions

```
x<-c(2,5,2,1)
mean(x)
median(x)
var(x) # variance
sd(x) # standard deviation
summary(x) # summary descriptive measures
fivenum(x) # five number summary
quantile(x) # five number summary as quantiles
quantile(x,c(.05,.95)) # Percentiles:  5%, and 95%
IQR(x) # Interquartile range
boxplot(x,main="Boxplot of the variable
x",xlab="Variable x") # boxplot
boxplot(x,main="Boxplot of the variable
x",xlab="Variable x",horizontal=TRUE) # horizontal
boxplot
```

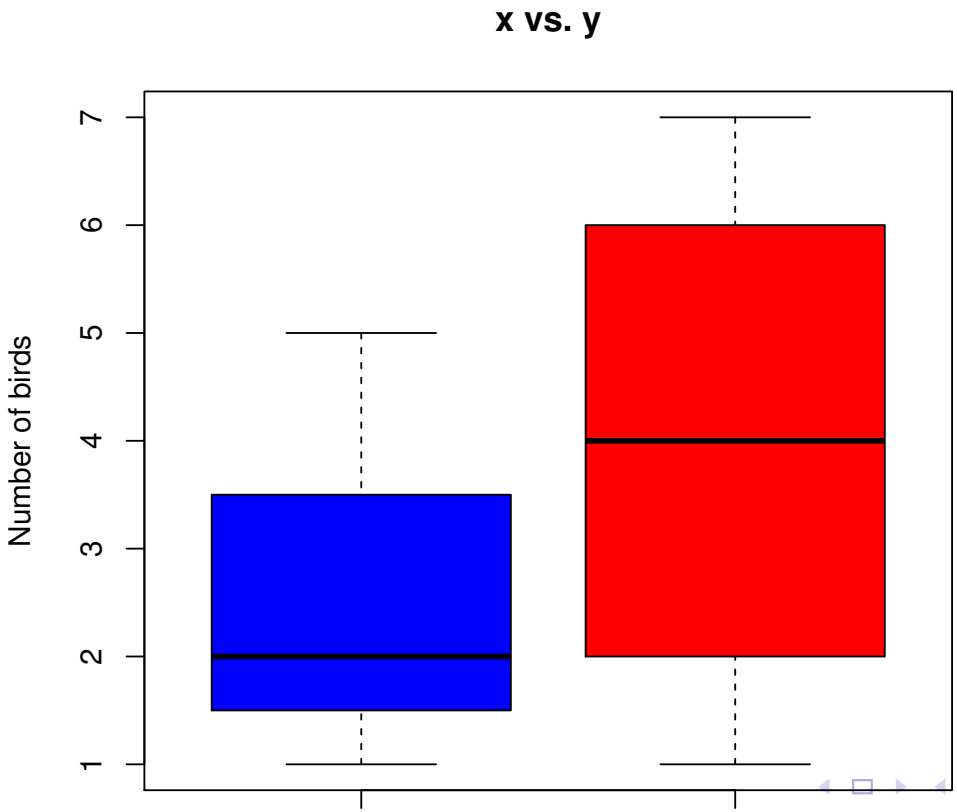
Some R Stat functions

```
data<-list(x=c(2,5,2,1),y=c(7,1,2,2,6,6))
boxplot(data,ylab="Number of birds",main="x vs.
y",col=c("blue","red"))
```



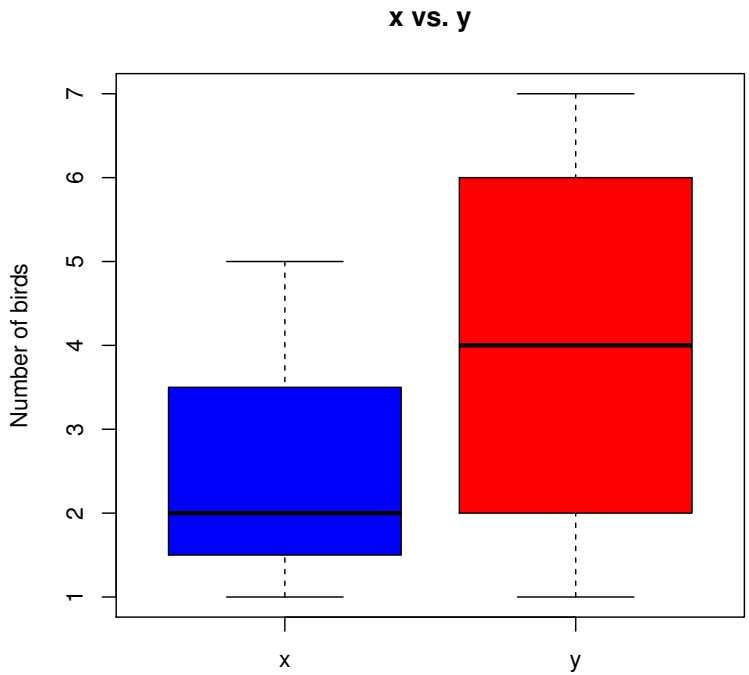
Some R Stat functions

```
data<-c(2,5,2,1,7,1,2,2,6,6)
label<-c(1,1,1,1,2,2,2,2,2,2)
boxplot(data ~ label,ylab="Number of birds",main="x
vs.  y",col=c("blue","red"))
```



Some R Stat functions

```
x=c(2,5,2,1);y=c(7,1,2,2,6,6)
data<-c(x,y)
label<-c(rep(1,length(x)),rep(2,length(y)))
boxplot(data ~ label,ylab="Number of birds",main="x
vs.  y",col=c("blue","red"))
```



Some R probability functions

- Selecting a random sample

```
x<-c(2,5,2,1,4)
sample(x,2,rep=F) # F or FALSE is the default
sample(x,2,rep=T) # with replacement is set TRUE
```

- Selecting a weighted random sample

```
sample(x,2,prob=c(.2,.3,.1,.05,.35))
```

Some R matrix and array functions

`> rep(0,5)` is the same as `numeric(5)` and `integer(5)`

```
[1] 0 0 0 0 0
> matrix(0,3,5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
```

Some R matrix and array functions

```
> diag(1,3,3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> diag(c(1,3,3))
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    3    0
[3,]    0    0    3
```

But `diag(matrix)` gives a vector of the diagonal elements of that matrix.

Some R matrix and array functions

```
> matrix(c(3,6,4,9,1,3), nrow=2, ncol=3)
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> A<-matrix(c(3,6,4,9,1,3), 2, 3,byrow=T)
> A
     [,1] [,2] [,3]
[1,]    3    6    4
[2,]    9    1    3
> dim(A)
[1] 2 3
```

Some R matrix and array functions

```
> A<- matrix(c(3,6,4,9,1,3), 2, 3)
> A
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> t(A)
     [,1] [,2]
[1,]    3    6
[2,]    4    9
[3,]    1    3
> A[1,2]
[1] 4
> A[,2]
[1] 4 9
> A[1,]
[1] 3 4 1
```

Some R matrix and array functions

```
> A<- matrix(c(3,6,4,9,1,3), 2, 3)
> A
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> B<- matrix(c(2,4,7,4,2,1), 2, 3)
> B
     [,1] [,2] [,3]
[1,]    2    7    2
[2,]    4    4    1
> A*B
     [,1] [,2] [,3]
[1,]    6   28    2
[2,]   24   36    3
```

Some R matrix and array functions

```
> A %*% t(B)
     [,1] [,2]
[1,]   36   29
[2,]   81   63
> A ^ 2 # Same as A*A (point-wise power)
     [,1]  [,2]  [,3]
[1,]    9    16     1
[2,]   36    81     9
```

Some R matrix and array functions

- To solve $Ax = b$ for x

```
> A<- matrix(c(3,6,4,9), 2, 2)
> b<-c(3,7)
> solve(A,b)
[1] -0.3333333 1.0000000
```

- To find inverse matrix of A

```
> solve(A)
     [,1]      [,2]
[1,]    3 -1.333333
[2,]   -2  1.000000
```

- To find determinant of A

```
> det(A)
[1] 3
```

Some R matrix and array functions

- To find eigen-values and vectors of a matrix

```
> A<- matrix(c(3,6,4,9), 2, 2)
> eigen(A)
$values
[1] 11.7445626  0.2554374
$vectors
           [,1]       [,2]
[1,] -0.4159736 -0.8245648
[2,] -0.9093767  0.5657675
```

- To find eigen-values of the matrix

```
> eigen(A)$values
```

- To find eigen-vectors of the matrix

```
> eigen(A)$vectors
```

Some R matrix and array functions

```
> A<- matrix(c(3,6,4,9,1,3), 2, 3)
> A
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> B<- matrix(c(2,4,7,4,2,1), 2, 3)
> B
     [,1] [,2] [,3]
[1,]    2    7    2
[2,]    4    4    1
> rbind(A,B)
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
[3,]    2    7    2
[4,]    4    4    1
```

Some R matrix and array functions

```
> A<- matrix(c(3,6,4,9,1,3), 2, 3)
> A
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> B<- matrix(c(2,4,7,4,2,1), 2, 3)
> B
     [,1] [,2] [,3]
[1,]    2    7    2
[2,]    4    4    1
> cbind(A,B)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    4    1    2    7    2
[2,]    6    9    3    4    4    1
```

Some R matrix and array functions

```
> A<- matrix(c(3,6,4,9,1,3), 2, 3)
> A
     [,1] [,2] [,3]
[1,]    3    4    1
[2,]    6    9    3
> rowSums(A)
[1] 8 18
> rowMeans(A)
[1] 2.666667 6.000000
```

Try

```
> colSums(A)
> colMeans(A)
```

Some R matrix and array functions

```
> A <- array(1:12, c(2, 3, 2))
> A
, , 1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
, , 2
     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
> A[„1]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Data frames in R

```
> A<- matrix(c(3,6,4,9,1,3), 3, 2)
> is.matrix(A)
[1] TRUE
> tab<-as.data.frame(A)
> tab
 V1 V2
1  3  9
2  6  1
3  4  3
> is.data.frame(tab)
[1] TRUE
```

Data frames in R

```
> colnames(tab)<-c("height","weight")
> tab
 height   weight
1      3        9
2      6        1
3      4        3
> height
Error:  object 'height' not found
> tab$height
[1] 3 6 4
> tab[[1]]
[1] 3 6 4
> tab[,1]
[1] 3 6 4
```

Data frames in R

```
> attach(tab)
> height
[1] 3 6 4
> dim(height)
NULL
```

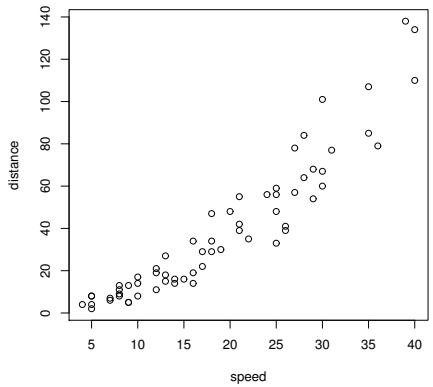
Importing data from Microsoft Excel in R

- To automatically import comma separated value (csv) file

```
> speeddata<-read.table("F:/UTRGV/Spring
2015/EX623.csv",sep=",",header=TRUE)
```

- Or choose the file from an open window

```
> speeddata<-read.table(file.choose(),
sep=",",header=TRUE)
> names(speeddata)
[1] "speed" "distance"
> plot(speeddata$speed,speeddata$distance)
```



Importing and exporting data between Microsoft Excel and R

```
install.packages("xlsx")
library(xlsx) # needed every time you do the following
```

- To import Excel File (xlsx) file
  ```
  read.xlsx("myfile.xlsx", sheetName = "Sheet1")
  ```
- To export to Excel File (xlsx) file
  ```
  write.xlsx(dataframename,"myfile.xlsx")
  ```

Writing your own function in R

```
nameoffunction<-function( inputs ) {expression
return(output) }
```

Example: A function to find moving average of length two

```
movave<-function(x) {
d<-length(x)
y<-rbind(x[1:d-1],x[2:d])
y<-colMeans(y)
return(y)}
> movave(c(1,2,3,4))
[1] 1.5 2.5 3.5
```

Conditional statements in R

- Conditional statement
  `if(condition) {Command} else {Alternative Command}`
- Conditional assignment
  `z<-ifelse(condition,value of z if condition is true, value of z if false)`

- Comparison Operators
  equal: ==
  not equal: !=
  greater/less than: > <
  greater/less than or equal: >= <=

Conditional statements in R

- Logical Operators
  and: &
  or: |
  not: !

Conditional statements in R

Examples:

- Conditional statement

```
> x<-2;y<-3
> if(x==y) {z<-x+2} else {z<-x-2}
> z
[1] 0
```

- Conditional assignment

```
z<-ifelse(x==y,x+2, x-2)
> z
[1] 0
```

Loops in R

- For loop

```
> for(x in sequence){statements}
```

Example:

```
> x<-c(1,2,3)
> for(i in 1:length(x)) {z[i]<-x[i]+2}
> z
[1] 3 4 5
```

- While loop

```
> while(condition){statements}
```

Example:

```
> x<-0
> z<-c()
> while(x<=4) {z<-c(z,x+2); x<-x+1}
> z
[1] 2 3 4 5 6
```

Logical functions in R

```
> x<-c(4,1,-1)
> (x<0)
[1] FALSE FALSE TRUE
> x<0
[1] FALSE FALSE TRUE
> any(x<0)
[1] TRUE
> all(x<0)
[1] FALSE
```

Apply Loops in R

- apply loop

```
> apply(matrix or array,1 for row or 2 for
column or c(1,2) for both,Function, Possible
inputs)
```

Example:

```
> A<-matrix(1:4,2,2)
> apply(A,1,mean)
[1] 2 3
> apply(A,2,mean)
[1] 1.5 3.5
```

Apply Loops in R

- tapply loop (nonuniform lists)

```
tapply(vector, factor, function)
```

Example:

```
> data<-c(2,5,2,1,7,1,2,2,6,6)
> label<-c(1,1,1,1,2,2,2,2,2,2)
> tapply(data,label,mean)
   1    2
2.5  4.0
```

Replicate Loops in R

- Replicate loop when there is random generation

  `replicate(number of times, expr={steps})`

  Example:

```
> y<-2
> x<-replicate(3, expr={y+runif(0,1)})
> x
[1]   2.110178   2.328782   2.823872
```

More probability in R

For any distribution (*dist*) like `binom, geom, nbinom, hyper, pois, unif, exp, gamma, beta, norm, lnorm, t, chisq, f, cauchy, weibull`

`d`*dist* computes density function of *dist*
`p`*dist* computes cumulative distribution function (cdf) of *dist*
`q`*dist* computes inverse cdf of *dist*
`r`*dist* generates random instances from *dist*

Use `help(pdist)` to find arguments for that function

Example:

```
> rnorm(3, mean = 0, sd = 1)
[1] 1.6362274 -1.7700555 0.4951518
```

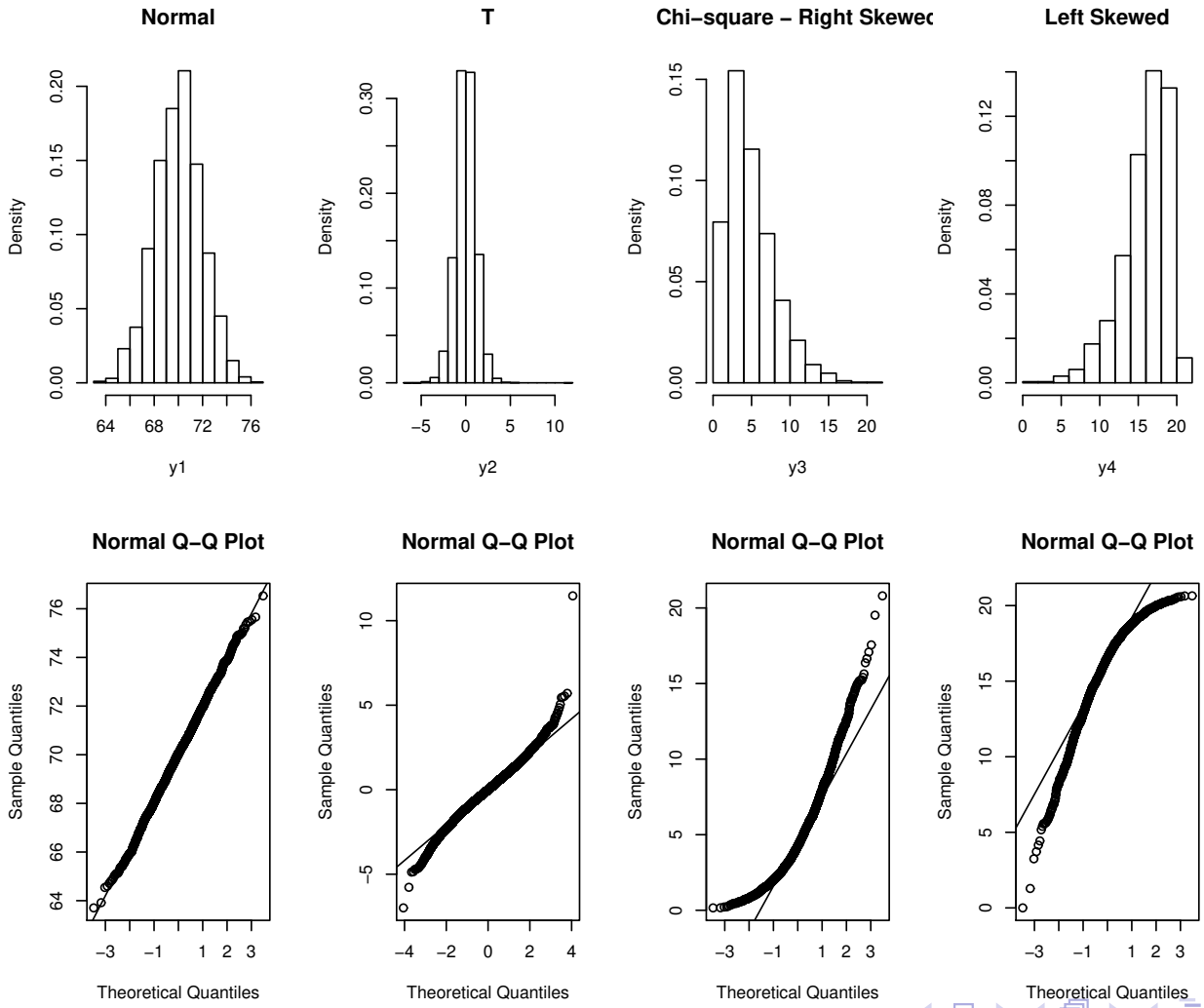
More probability in R

Example:

```
y1 <- rnorm(2000,mean=70,sd=2)
y2 <- rt(20000,df=10)
y3 <- rchisq(2000,df=5)
y4 <- max(y3)-y3
par(mfrow=c(2,4))
hist(y1, freq=FALSE, main= "Normal")
hist(y2, freq=FALSE, main= "T")
hist(y3, freq=FALSE, main= "Chi-square - Right
Skewed")
hist(y4, freq=FALSE, main= "Left Skewed")
qqnorm(y1);qqline(y1)
qqnorm(y2);qqline(y2)
qqnorm(y3);qqline(y3)
qqnorm(y4);qqline(y4)
```

More probability in R

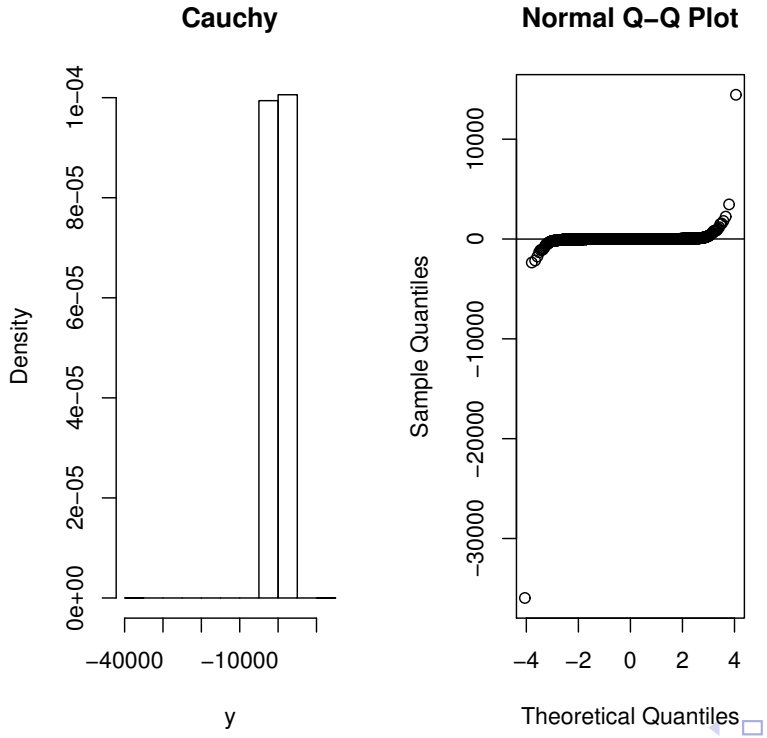
Example (cont'd):



More probability in R

Example: Try Cauchy distribution

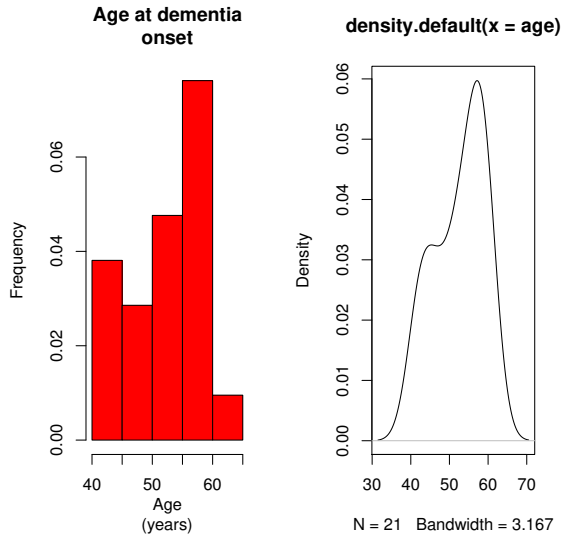
```
y<-rcauchy(20000, location = 0, scale = 1)
par(mfrow=c(1,2))
hist(y, freq=FALSE, main= "Cauchy")
qqnorm(y);qqline(y)
```



More Statistics in R

Example

```
age<-c(60, 58, 52, 58 ,59 ,58 ,51 ,61, 54 ,59 ,55
,53, 44, 46 ,47 ,42 ,56 ,57 ,49 ,41 ,43)
par(mfrow=c(1,2)) # try par(new=TRUE)
hist(age,breaks=5,freq=FALSE,main="Age at dementia
onset",xlab="Age
(years)",ylab="Frequency",col="red")
plot(density(age))
```



More Statistics in R

Empirical Cumulative Distribution Function

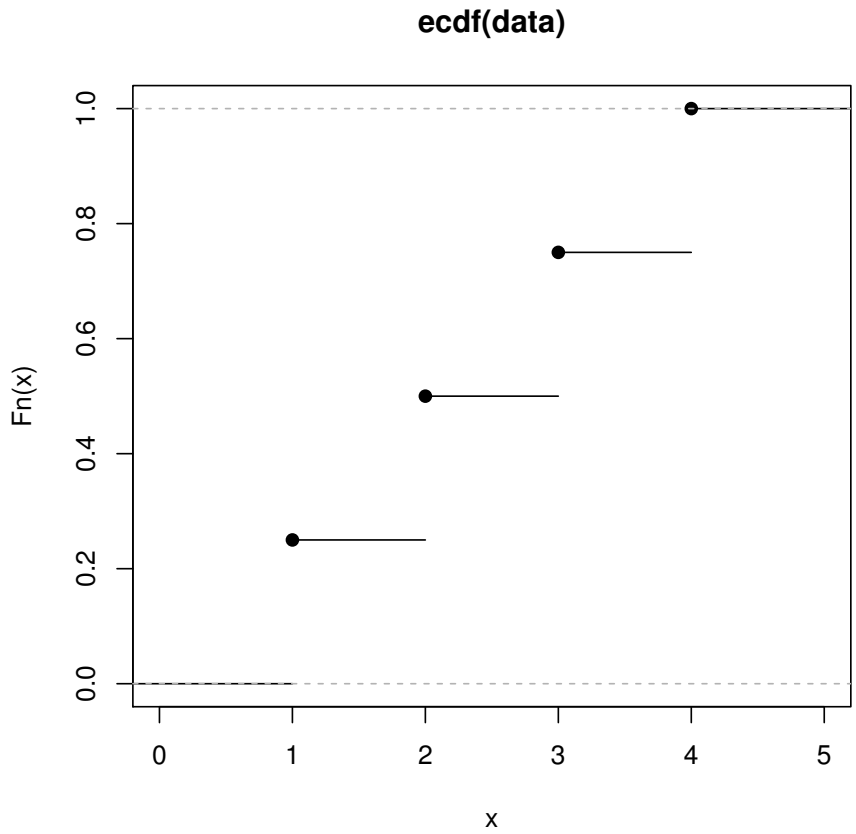
The empirical cumulative distribution function $\hat{F}_n(x)$ (ecdf) is an estimate of the cdf $F(x) = P(X \leq x)$ based on collected sample $x_1, x_2, \ldots, x_n$ which when ordered in ascending order to $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ then

$$\hat{F}_n(x) = \begin{cases} 0 & \text{if } x < x_{(1)}, \\ \dfrac{k}{n} & \text{if } x_{(k)} \leq x < x_{(k+1)}; \text{ for } k = 1, 2, \ldots, n-1, \\ 1 & \text{if } x_{(n)} \leq x. \end{cases}$$

More Statistics in R

Empirical Cumulative Distribution Function

```
> data<-c(1,2,3,4)
> Fn<-ecdf(data)
> plot(Fn)
> Fn(1.5)
[1] 0.25
> summary.stepfun(Fn)
Step function with
continuity 'f'= 0 , 4
knots at
[1] 1 2 3 4
 and 5 plateau levels
(y) at
[1] 0.00 0.25 0.50
0.75 1.00
```



More Statistics in R

Kolmogorov–Smirnov Test

Kolmogorov–Smirnov Test examines whether X has the same distribution as Y; that is

$$H_o : F_X \equiv F_Y \text{ vs. } H_a : F_X \not\equiv F_Y \text{ (default, or } < \text{ or } >)$$

Example: To test if two samples have the same distribution

```
> x<-rnorm(100)
> y<-rexp(50,1)
> ks.test(x,y)
 Two-sample Kolmogorov-Smirnov test
data:  x and y
D = 0.5, p-value = 5.285e-08
alternative hypothesis:  two-sided
```

More Statistics in R

Kolmogorov–Smirnov Test

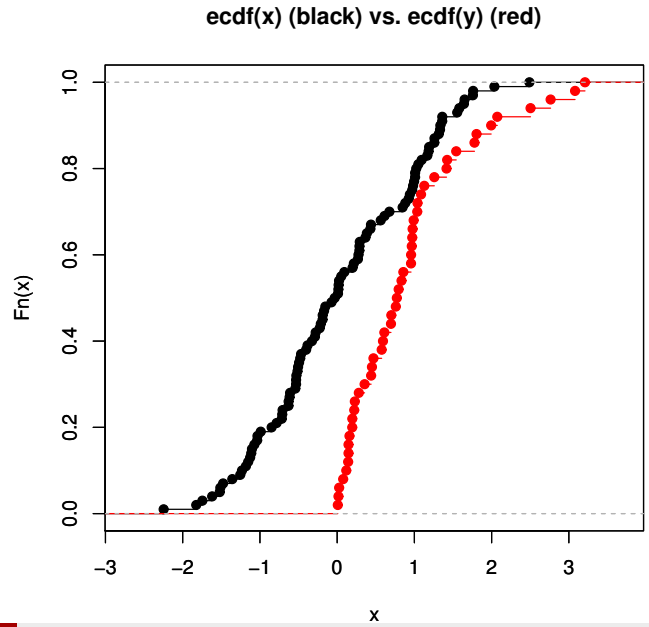
Example: To test if the sample y stochastically dominates the sample x (that is, $F_X > F_Y$)

```
> x<-rnorm(100)
> y<-rexp(50,1)
> ks.test(x,y,alternative="greater")
 Two-sample Kolmogorov-Smirnov test
data:  x and y
D^+ = 0.5, p-value = 5.778e-08
alternative hypothesis:  the CDF of x lies above
that of y
```

More Statistics in R

Kolmogorov–Smirnov Test

```
> plot(ecdf(x),main="",xlim=c(min(c(x,y))-.5
+,max(c(x,y))+.5))
> par(new=T)
> plot(ecdf(y),col="red"
+,xlim=c(min(c(x,y))-.5,max(c(x,y))+.5)
+,main="ecdf(x) (black) vs. ecdf(y) (red)")
```



More Statistics in R

Kolmogorov–Smirnov Test

Example: To test if the sample x is normally distributed with mean 0 and standard deviation 1.

```
> x<-rnorm(100)
> ks.test(x,"pnorm",0,1)
 One-sample Kolmogorov-Smirnov test
data:  x
D = 0.1028, p-value = 0.2416
alternative hypothesis:  two-sided
```

Compare to Shapiro–Wilk test for normality

```
> shapiro.test(x)
 Shapiro-Wilk normality test
data:  x
W = 0.9829, p-value = 0.221
```

Advanced statistics in R

The following code will do Exercise 6.23 in the textbook "Introduction to Regression Modeling" by Abraham and Ledolter.

```
speeddata<-read.table("F:/UTRGV/Fall
2016/EX623.csv",sep=",",header=TRUE)
attach(speeddata)
fitmodel<-lm(distance ~ speed) # lm Linear model
summary(fitmodel)
anova(fitmodel)
plot(speed,distance)
abline(fitmodel)
```

More statistics in R

Please find more than 20 lessons online at Wikispace http://rutrgv.wikispaces.com/

End of Set 1