THE UNIVERSITY OF CHICAGO | Center for Research Informatics

Introductory Statistics with R

Sabah Kadri, Ph.D.

Outline

- Brief introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

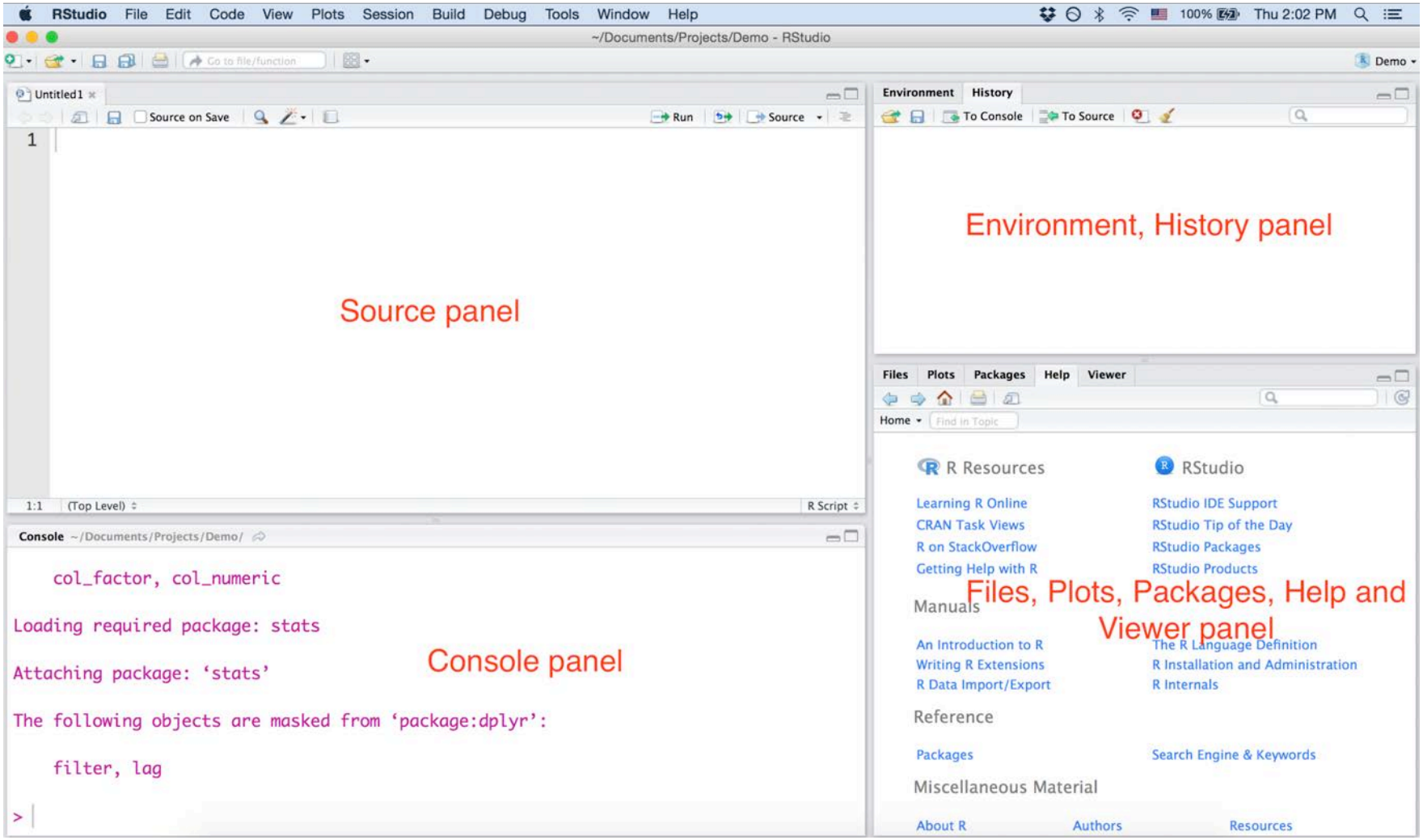
Outline

- Brief introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

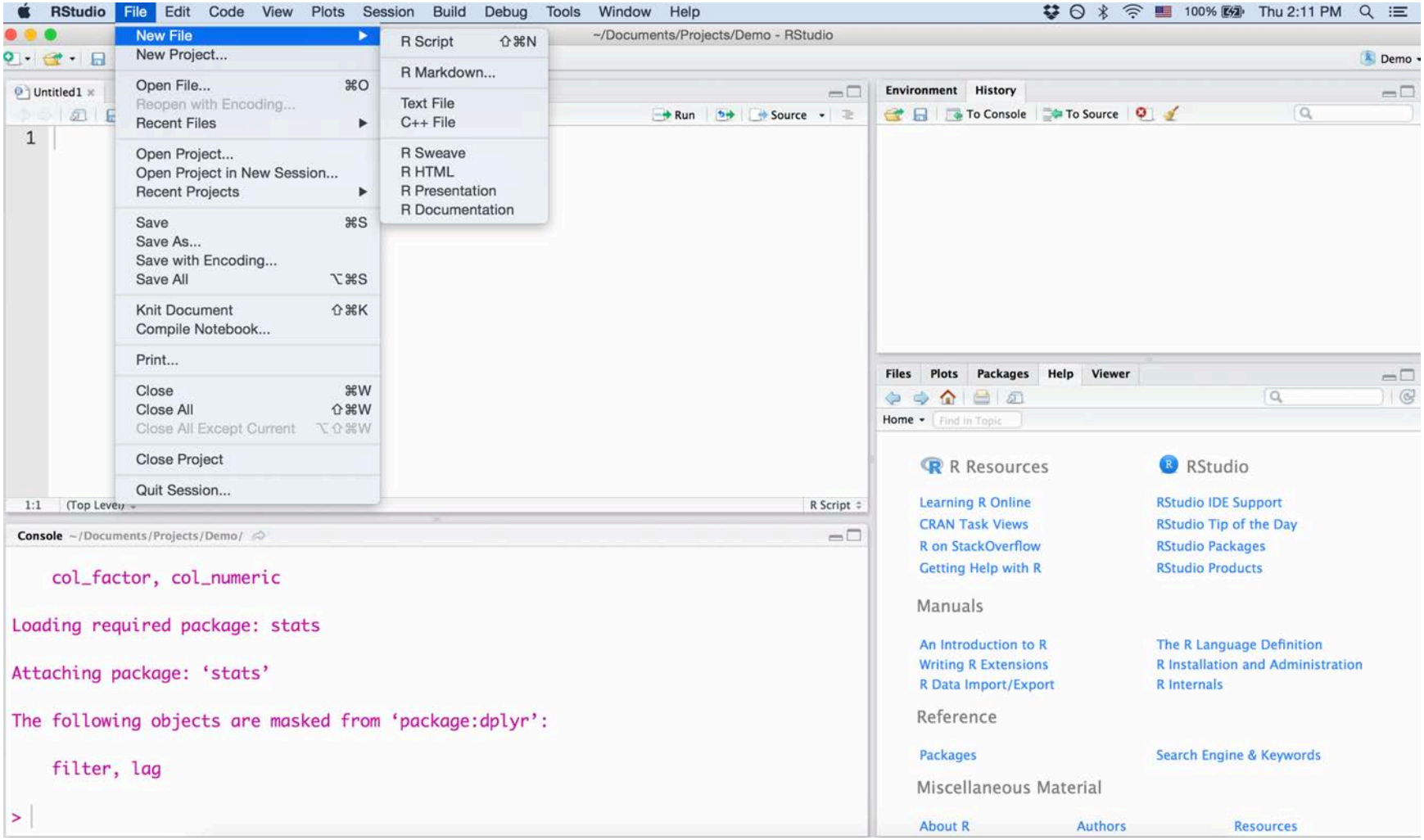
Installation of R and RStudio

- Install R on your computer
  - http://cran.us.r-project.org
- Install RStudio IDE (an R Integrated Development Environment tool)
  - http://www.rstudio.com/products/rstudio/download

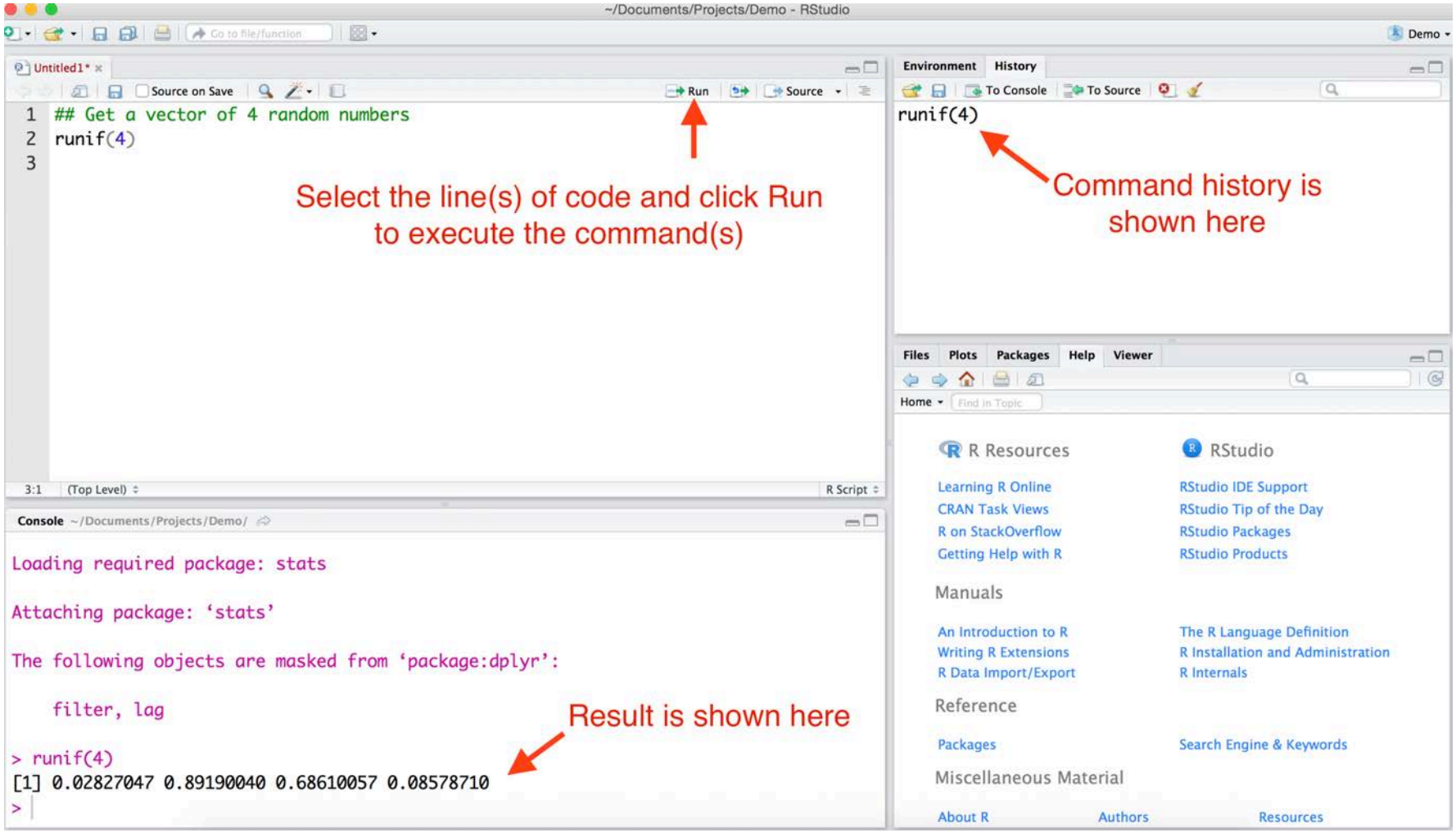
RStudio layout



Create new R script file in RStudio



Run R script file in RStudio



Getting help in R

- documentation on “mean”
  - `?mean`
  - `help(mean)`
- returns a character vector giving the names of all objects in the search list matching
  - `apropos('mean')`

Installing and load non-base packages

Installs ‘package’

```
install.packages(‘package’)
```

Loads the functions and data in ‘package’

```
library(package)
```

Bioconductor provides tools for analysis of high throughput genomic data

```
source("http://bioconductor.org/biocLite.R")
biocLite(package) e.g. DESeq, limma
```

Getting and setting your working directory

Get the current working directory:

```
getwd()
```

Change the working directory:

```
setwd("<absolute path>")
   eg. setwd("/Users/john/Documents")
       setwd("C:/MyDocuments/")
setwd("<relative path>")
   eg.   setwd("../MyFiles/")
```

Vectors

One dimensional collection of character strings, numbers, logical values etc.

```
> my_vector = c(1,2,5,10,13)      OR
> my_vector <- c(1,2,5,10,13)
```

```
> my_vector
[1]  1  2  5 10 13
> c(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
```

Note: *= and <- can be used for assignment interchangeably*

Data Frames

- A dataframe is a list containing multiple named vectors of same length – similar to a spreadsheet

```
> name <- c("Mike Jones","John Smith","Sarah Adler")
> Gender <- c("M","M","F")
> Age <- c(47,38,52)
> Height <- c(5.9,5.7,5.4)
> patient_info = data.frame (name, gender, age, height)
> patient_info
          name gender age height
1  Mike Jones      M  47    5.9
2  John Smith      M  38    5.7
3 Sarah Adler      F  52    5.4
```

Data Frames

- An alternative way to declare the dataframe

```
> patient_info <- data.frame(
   name=c("Mike Jones","John Smith","Sarah Adler"),
   gender=c("M","M","F"),
   age=c(47,38,52),
   height=c(5.9,5.7,5.4)
)
> patient_info
          name gender age height
1   Mike Jones      M  47    5.9
2   John Smith      M  38    5.7
3 Sarah Adler      F  52    5.4
```

Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

Summary statistics for a single vector

- Find the number of elements in the vector

```
> my_vector = c(1,2,5,10,13)
> length(my_vector)
[1] 5
```

- Sum up the elements of the vector

```
> sum(my_vector)
[1] 31
```

Descriptive statistics

```
> my_vector = c(1,2,5,10,13)
```

- Other descriptive statistics

```
> mean(my_vector)
[1] 6.2
```
Average

```
> median(my_vector)
[1] 5
```
Median

```
> min(my_vector)
[1] 1
```
Minimum

```
> max(my_vector)
[1] 13
```
Maximum

```
> var(my_vector)
[1] 26.7
```
Variance

```
> sd(my_vector)
[1] 5.167204
```
Standard Deviation

Random sampling

```
> big_vector <- 1:10000
> x=sample(big_vector, 100)
> x
  [1] 1965 2420 6545 7272 2260 2333 9275 5366 3469 1010 7403 8748 7519 7345 1883  691 1087
 [18] 4905 5870 6923 1470 7666  809  711 4910 1857  520 2019 3776  598 8498 1396 3338 2725
 [35] 1141 5401  697 5636 6475 6252 2488 1166 7438 3931 1695 7584 8373 2223 9434 3674 3256
 [52]  266 7606 1384 8551 6984 5816 2849 2397 2789 2392 8776 5995 7534 3135  709 5888 1542
 [69] 8460  148 8814 9953 5900 5404 6877 2715 9892 8598  882 9115 2987 7404 3557 2634 3918
 [86] 9910 9372 6326 3609 6843 5082 7323 4340 9258 9979 1996 4010 9737 4941  731
```

Challenge 1: For x, find the

(i) minimum value

(ii) median value

(iii) the 50th percentile value

Challenge 1

Challenge 1: For x, find the

(i) minimum value

(ii) median value

(iii) the 50th percentile value

```
> min(x)
[1] 148
> median(x)
[1] 4622.5
> quantile(x, 0.5)
   50%
4622.5
```

Quantiles

```
> quantile(x)
      0%      25%      50%      75%     100%
 148.00 2172.00 4622.50 7403.25 9979.00

> quantile(x,0.5)
   50%
4622.5

> quantile(x,0.3)
   30%
2413.1
```

50th percentile

30th percentile

Operation on a vector

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    148    2172    4622    4768    7403    9979
```

**** Take a subset of the data ****

```
> x[1:10]
 [1] 1965 2420 6545 7272 2260 2333 9275 5366 3469 1010
```

**** Scalar operation on the subset ****

```
> x[1:10]/100
 [1] 24.20 65.45 72.72 22.60 23.33 92.75 53.66 34.69 10.10
```

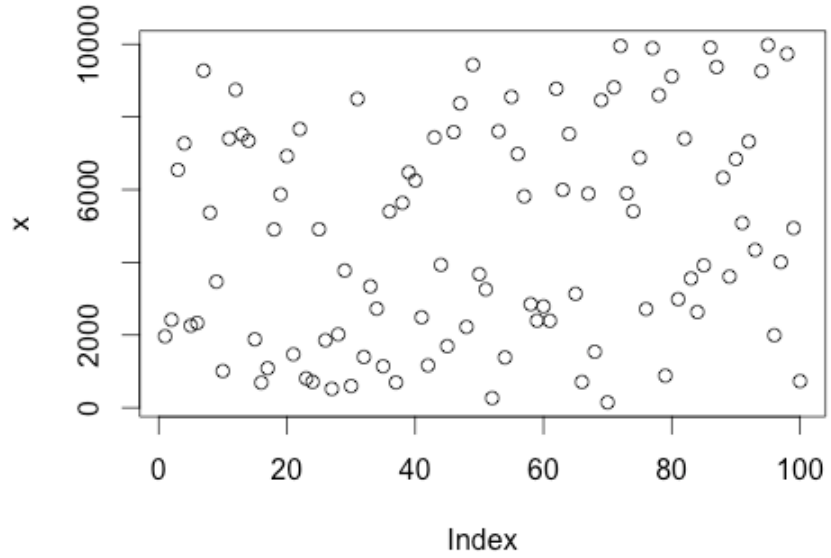
**** Scalar operation on the subset ****

```
> x[1:10] + x[11:20]
 [1]  9368 11168 14064 14617  4143  3024 10362 10271  9339  7933
```

Plotting a vector/distribution

```
> big_vector <- (1:10000)
> X = sample(big_vector,100)
> x
  [1] 1965 2420 6545 7272 2260 2333 9275 5366 3469 1010 7403 8748 7519 7345 1883  691 1087
 [18] 4905 5870 6923 1470 7666  809  711 4910 1857  520 2019 3776  598 8498 1396 3338 2725
 [35] 1141 5401  697 5636 6475 6252 2488 1166 7438 3931 1695 7584 8373 2223 9434 3674 3256
 [52]  266 7606 1384 8551 6984 5816 2849 2397 2789 2392 8776 5995 7534 3135  709 5888 1542
 [69] 8460  148 8814 9953 5900 5404 6877 2715 9892 8598  882 9115 2987 7404 3557 2634 3918
 [86] 9910 9372 6326 3609 6843 5082 7323 4340 9258 9979 1996 4010 9737 4941  731
```

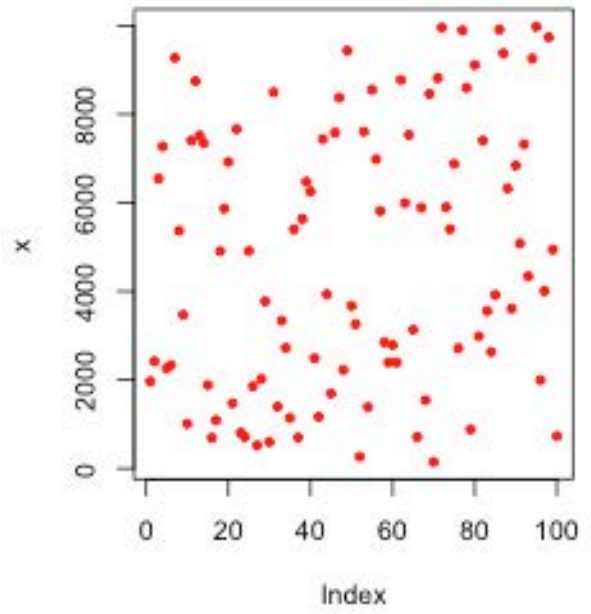
```
> plot(x)
```



Properties of plot()

```
> par(mfrow=c(1,3))
> plot(x, pch=20, col="red")
```

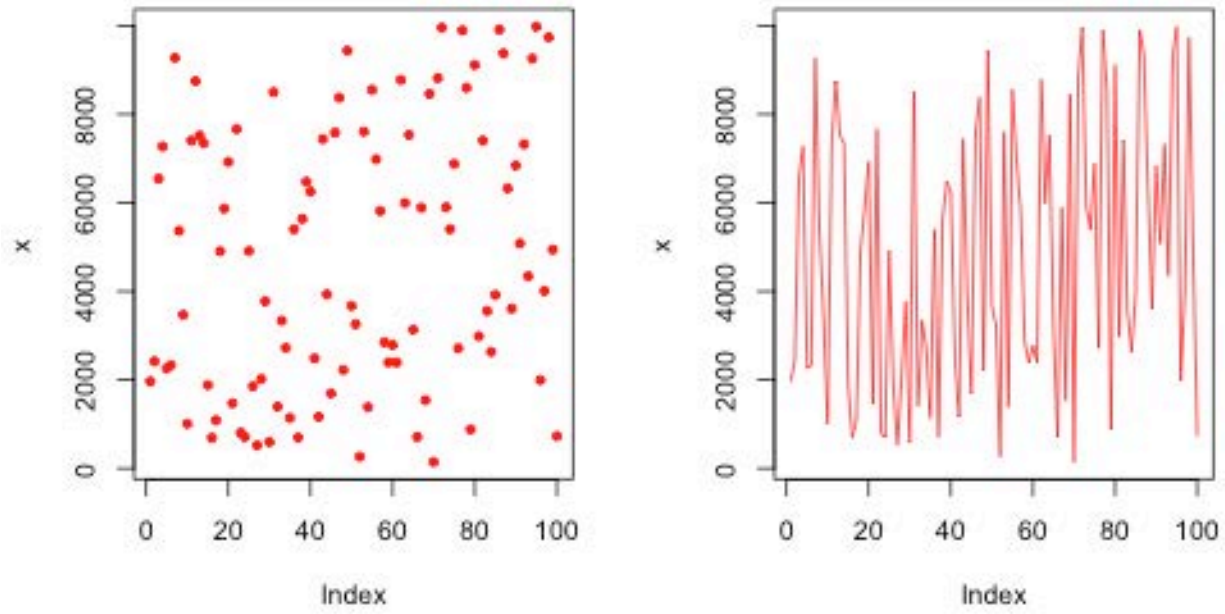
par(mfrow=c(rows,columns))



Properties of plot()

```
> par(mfrow=c(1,3))
> plot(x,pch=20,col="red")
> plot(x,col="red",type="l")
```

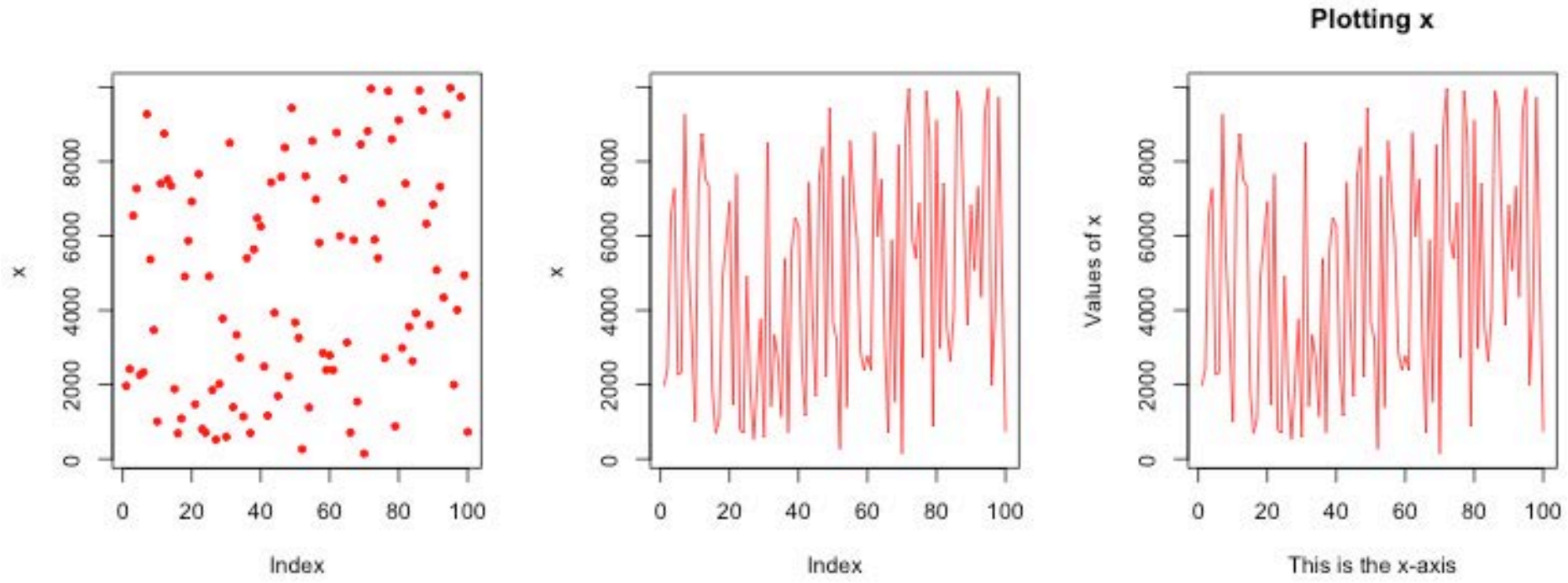
par(mfrow=c(rows,columns))



Properties of plot()

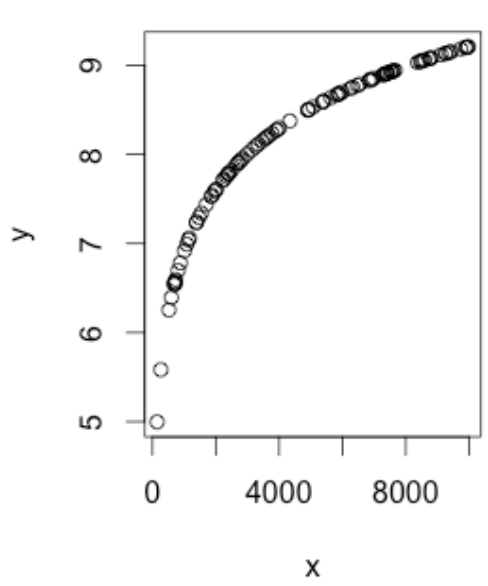
```
> par(mfrow=c(1,3))
> plot(x,pch=20,col="red")
> plot(x,col="red",type="l")
> plot(x,col="red",type="l",main="Plotting x",xlab="This is the x-axis",ylab="Values of x")
```

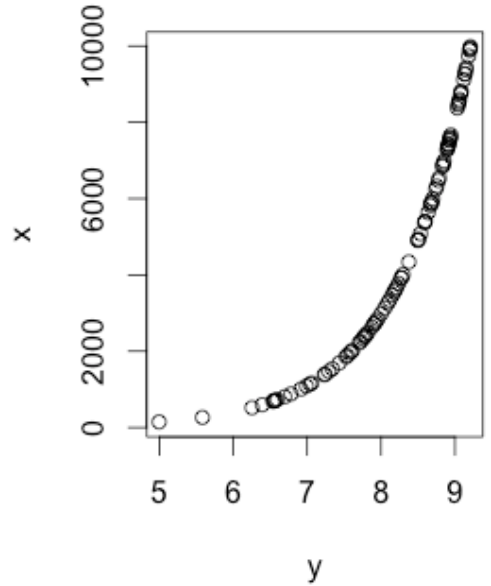
par(mfrow=c(rows,columns))



Plotting two vectors/distributions against each other

```
> Y = log(x)
> y
  [1] 7.583248 7.791523 8.786457 8.891787 7.723120 7.754910 9.135078 8.587838 8.151622
 [10] 6.917706 8.909641 9.076580 8.925188 8.901775 7.540622 6.538140 6.991177 8.498010
 [19] 8.677610 8.842604 7.293018 8.944550 6.695799 6.566672 8.499029 7.526718 6.253829
 [28] 7.610358 8.236421 6.393591 9.047586 7.241366 8.113127 7.910224 7.039660 8.594339
 [37] 6.546785 8.636930 8.775704 8.740657 7.819234 7.061334 8.914357 8.276649 7.435438
 [46] 8.933796 9.032768 7.706613 9.152075 8.209036 8.088255 5.583496 8.936693 7.232733
 [55] 9.053804 8.851377 8.668368 7.954723 7.781973 7.933438 7.779885 9.079776 8.698681
 [64] 8.927181 8.050384 6.563856 8.680672 7.340836 9.043104 4.997212 9.084097 9.205629
 [73] 8.682708 8.594895 8.835938 7.906547 9.199482 9.059285 6.782192 9.117677 8.002025
 [82] 8.909776 8.176673 7.876259 8.273337 9.201300 9.145482 8.752423 8.191186 8.830982
 [91] 8.533460 8.898775 8.375630 9.133243 9.208238 7.598900 8.296547 9.183688 8.505323
[100] 6.594413
> par(mfrow=c(1,2))
> plot(x,y)
> plot(y,x)
```



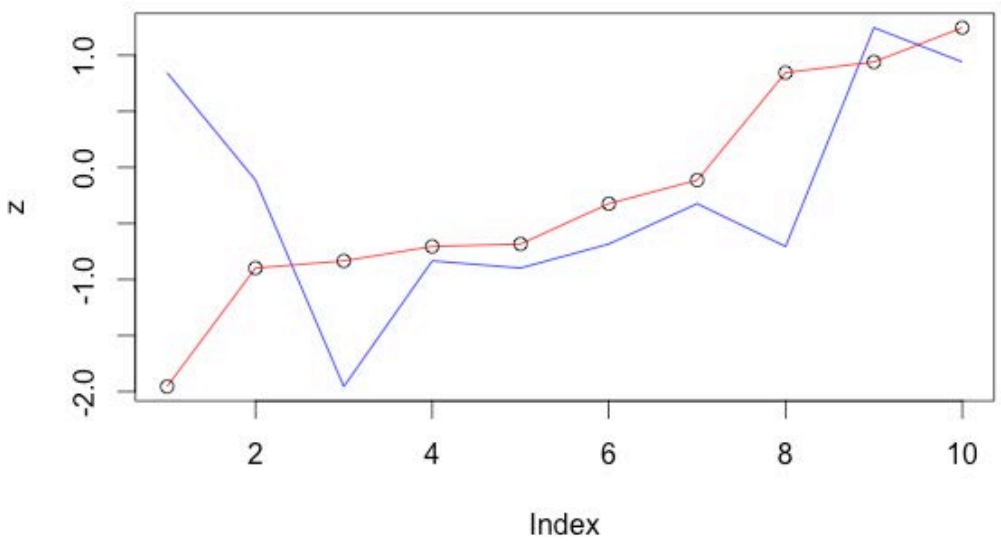


More on correlations later

Plotting on the same plot

```
> par(mfrow=c(1,1))
> Z = rnorm(10)
> z
 [1]  0.8451165 -0.1134582 -1.9549015 -0.8338934 -0.8977647 -0.6833283
-0.3236706 -0.7055102
 [9]  1.2461166  0.9404709
> plot(z,type="l",col="blue")
> sorted_z=sort(z)
> sorted_z
 [1] -1.9549015 -0.8977647 -0.8338934 -0.7055102 -0.6833283 -0.3236706
-0.1134582  0.8451165
 [9]  0.9404709  1.2461166
> lines(sorted_z,col="red")
> points(sorted_z,col="black")
```

sort(z) sorts the vector in ascending order



Barplots

```
> barplot(z)
```

Challenge 2: Plot barplots for z and sorted_z one below the other

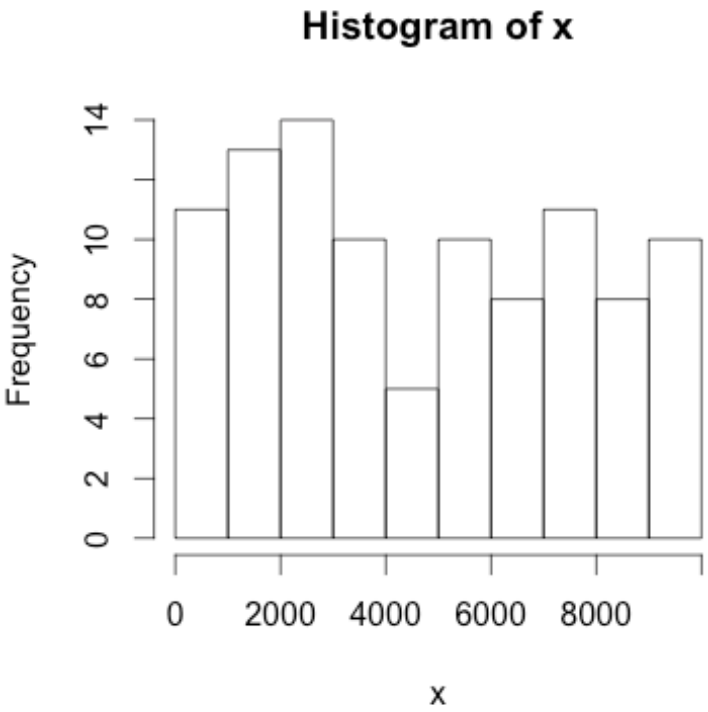
Challenge 2

Challenge 2: Plot barplots for z and sorted_z one below the other

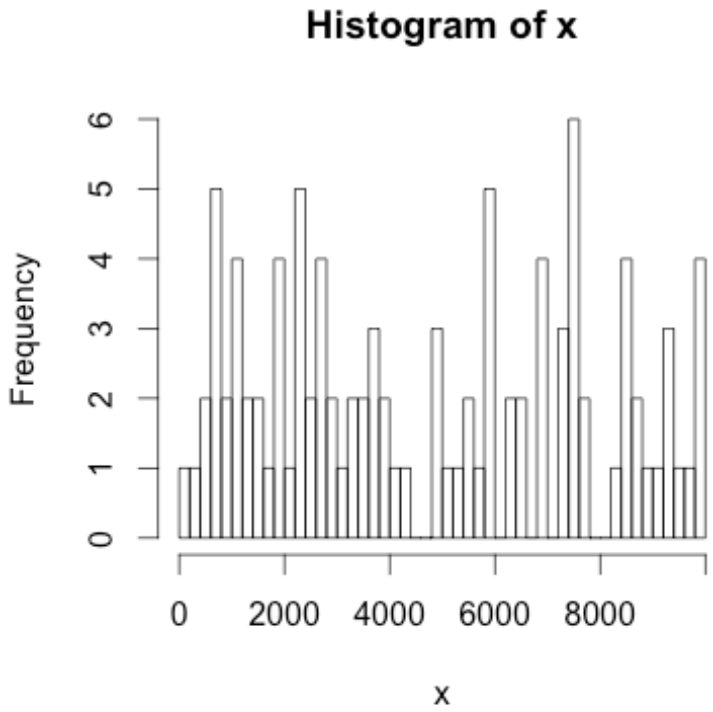

```
> par(mfrow=c(2,1))
> barplot(z)
> barplot(sorted_z)
```

Graphical display of distributions: Histograms

```
> par(mfrow=c(1,1))
> hist(x)
```

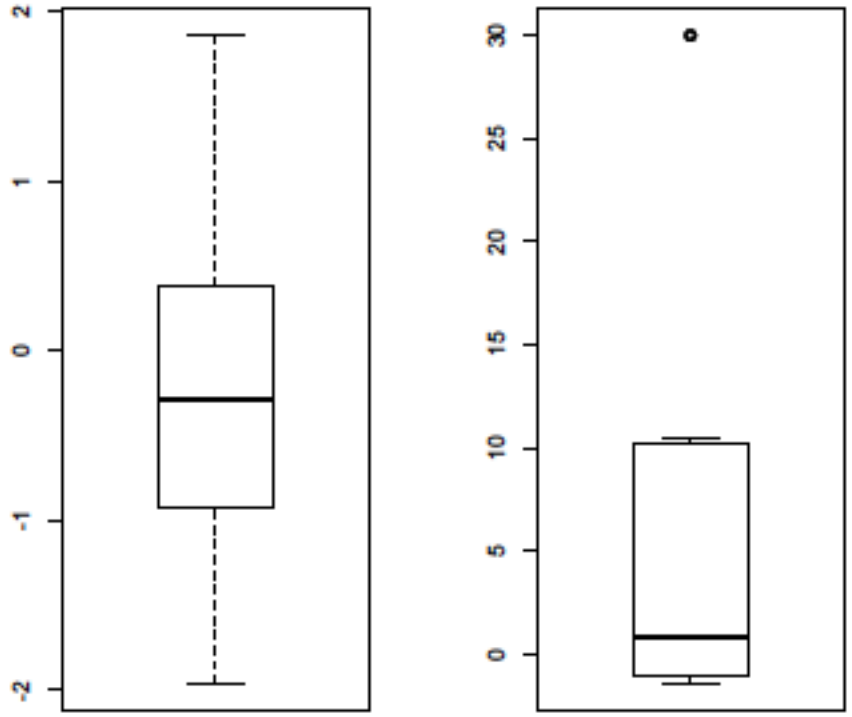



```
> hist(x,breaks=50)
```



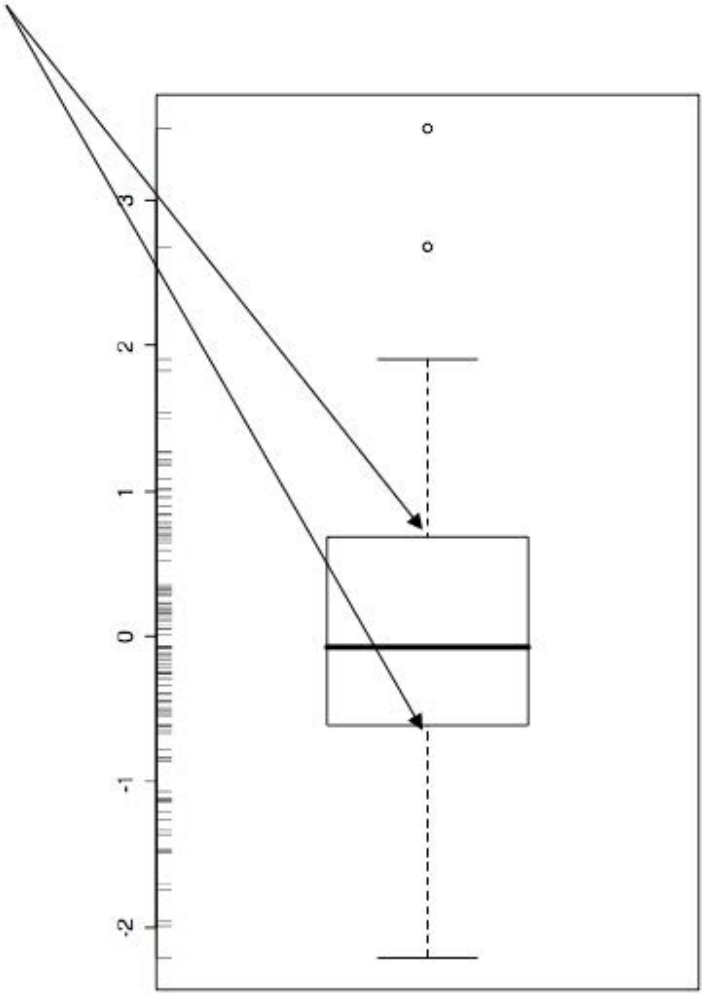
Boxplots

```
> dat <- rnorm(10)
> dat2 <- c(dat, 10, 10.5, 30 )
> par( mfrow=c(1,2) )
> boxplot(dat)
> boxplot(dat2)
```



Boxplots

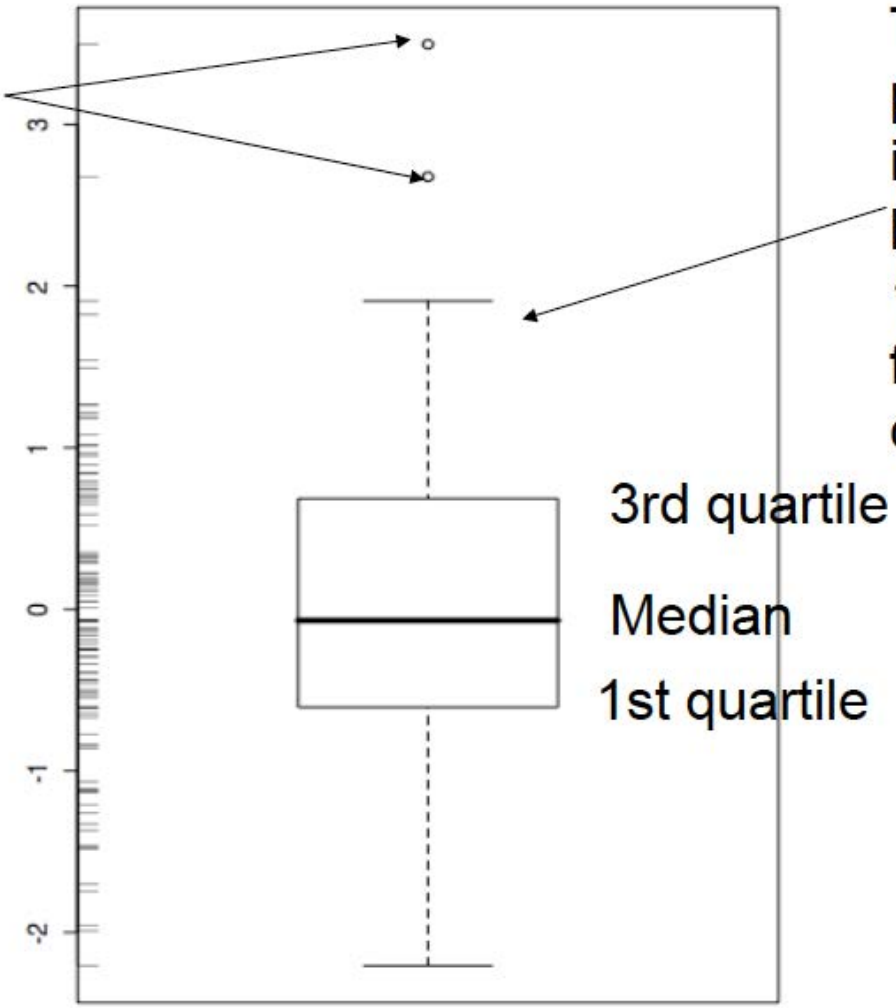
Interquartile range (IQR)



3rd quartile
Median
1st quartile

Boxplots

More extreme data - outliers



The data point that is highest but within 1.5 IQR from the center

3rd quartile

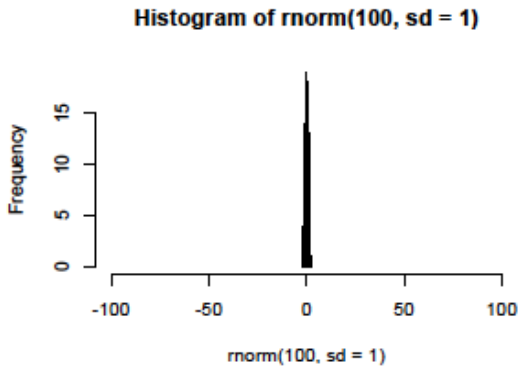
Median

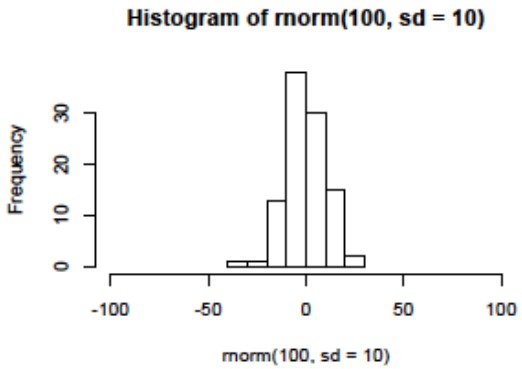
1st quartile

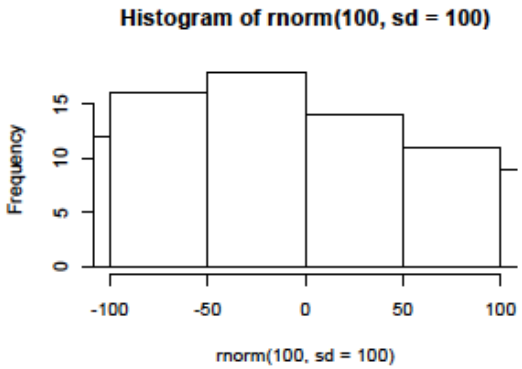
Any data observation which lies more than 1.5*IQR lower than the first quartile is considered an outlier.

Variance, Standard deviation & data spread

What is the difference between the following distributions?

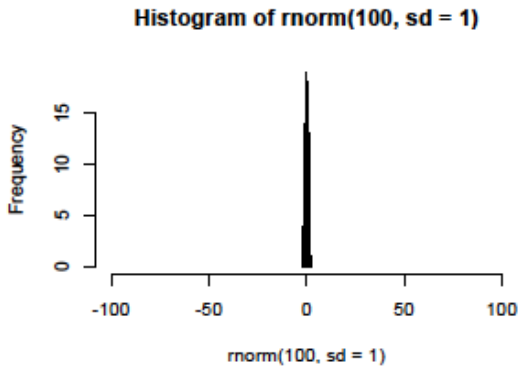


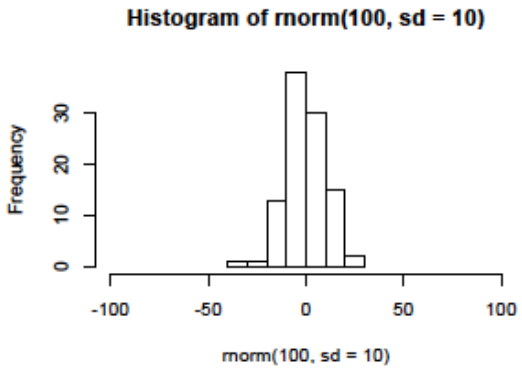


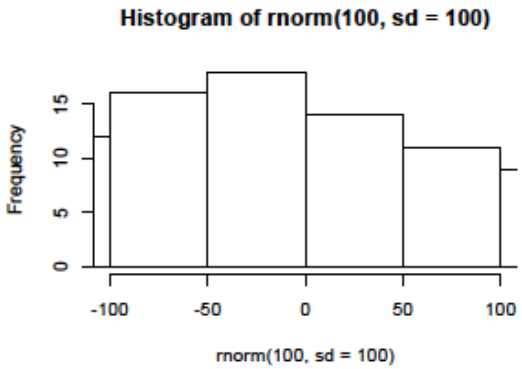


Variance, Standard deviation & data spread

What is the difference between the following distributions?







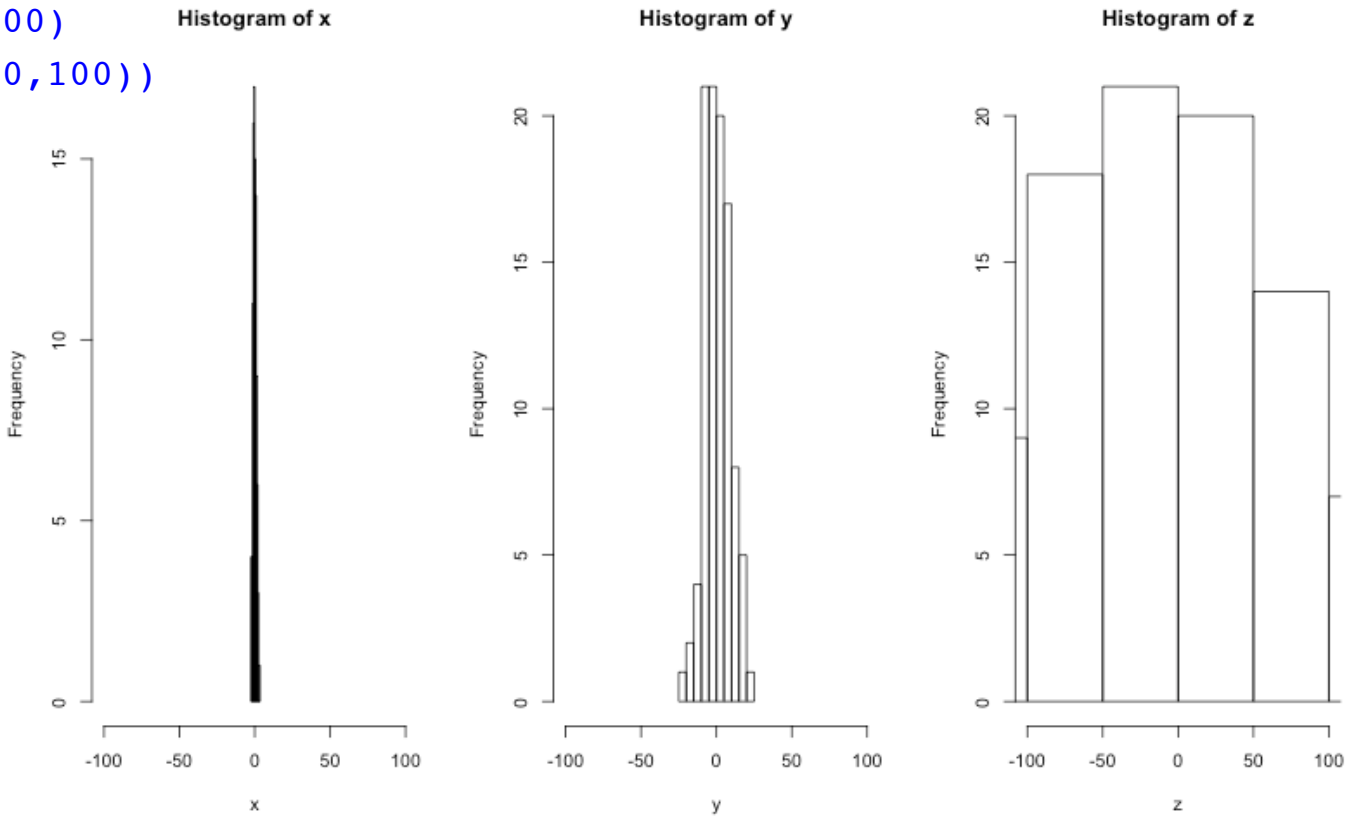
Different variances of the data.

Variance, Standard deviation & data spread

```
> par(mfrow=c(1,3))
> X = rnorm(100,sd=1)
> hist(x,xlim=c(-100,100))
> Y = rnorm(100,sd=10)
> hist(y,xlim=c(-100,100))
> Z = rnorm(100,sd=100)
> hist(z, xlim=c(-100,100))
```

Note:
`xlim=c(min,max)`

sets the limits on the x-axis



Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

Reading a table

Please download the following table:

http://biocore.cri.uchicago.edu/training/GSE47561.txt

Reading a table

```
> tab = read.table("GSE47561.txt")
> head(tab)
         V1          V2          V3          V4          V5
1    ID_REF   GSM107072   GSM107073   GSM107075   GSM107076
2 1007_s_at  9.32636963  9.50907773 9.470700565 9.579381316
3   1053_at 5.770975829 6.871806536   4.8116242 5.896438495
4    117_at 4.837099985 4.100548813 3.025137107 3.930648404
5    121_at 5.320876995 5.263375928 5.132676878  5.14460312
6 1255_g_at 2.221334895 2.221334895 2.221334895 2.221334895
> tab = read.table("GSE47561.txt",header=T)
> head(tab)
     ID_REF GSM107072 GSM107073 GSM107075 GSM107076
1 1007_s_at  9.326370  9.509078  9.470701  9.579381
2   1053_at  5.770976  6.871807  4.811624  5.896438
3    117_at  4.837100  4.100549  3.025137  3.930648
4    121_at  5.320877  5.263376  5.132677  5.144603
5 1255_g_at  2.221335  2.221335  2.221335  2.221335
6   1294_at  4.727786  4.673524  5.425090  4.765807
> tab = read.table("GSE47561.txt",header=T,row.names=1)
> head(tab)
          GSM107072 GSM107073 GSM107075 GSM107076
1007_s_at  9.326370  9.509078  9.470701  9.579381
1053_at    5.770976  6.871807  4.811624  5.896438
117_at     4.837100  4.100549  3.025137  3.930648
121_at     5.320877  5.263376  5.132677  5.144603
1255_g_at  2.221335  2.221335  2.221335  2.221335
1294_at    4.727786  4.673524  5.425090  4.765807
```

Note: `head(tab)` returns the first 6 rows of the table. This is a good way to get a snapshot of your data.

Properties of the table

```
> colnames(tab)
[1] "GSM107072" "GSM107073" "GSM107075"
"GSM107076"
```

- Challenge 3:

1. Get the row names of the table
2. Display the first 10 row names.

Challenge 3: Row names

- Challenge 3:

1. Get the row names of the table

```
> row.names(tab) OR rownames(tab)
```

2. Display the first 10 row names.

```
> row.names(tab)[1:10]
 [1] "1007_s_at" "1053_at"   "117_at"
"121_at"    "1255_g_at" "1294_at"   "1316_at"
 [8] "1320_at"   "1405_i_at" "1431_at"
```

Summary statistics

Get the means of the colums

```
> colMeans(tab)
GSM107072 GSM107073 GSM107075 GSM107076
 9.943704  9.907088  9.740273  9.883658
> colSums(tab)
GSM107072 GSM107073 GSM107075 GSM107076
 1968.853  1961.603  1928.574  1956.964
```

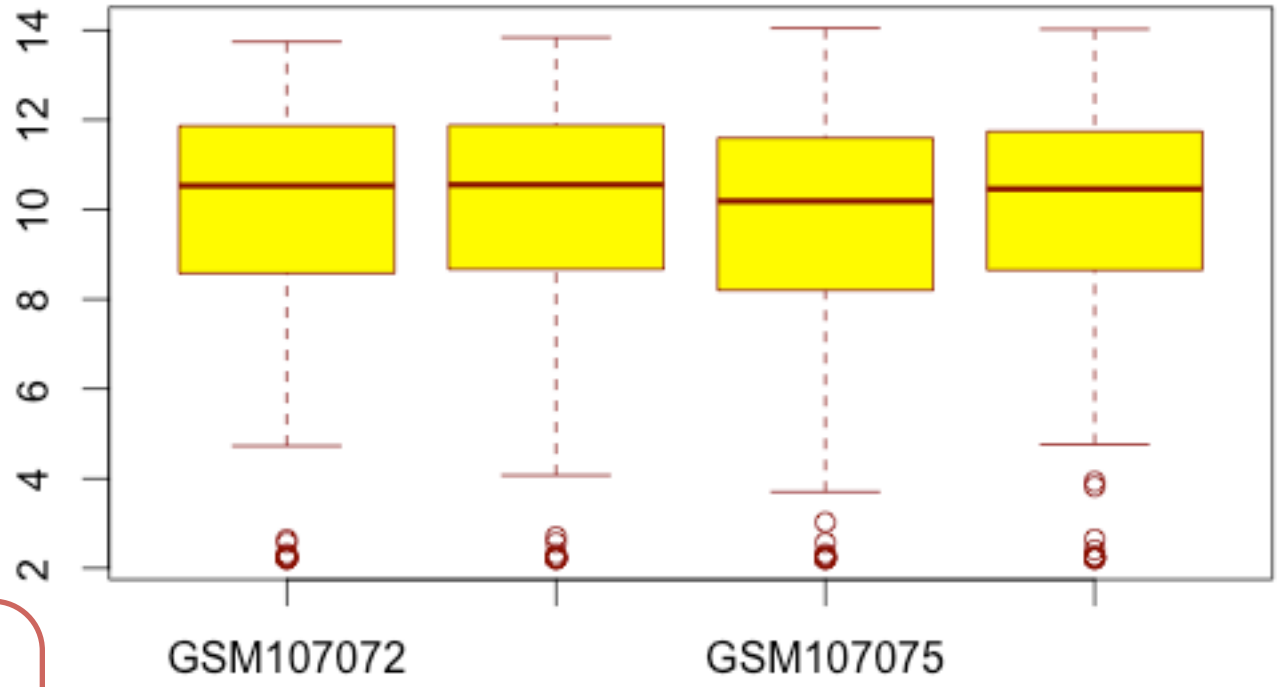
The same operations can be done on the rows:

Try:

```
rowMeans(tab)
rowSums(tab)
```

Graphics for grouped data: Boxplots

```
> boxplot(tab,col="yellow",border="darkred")
```



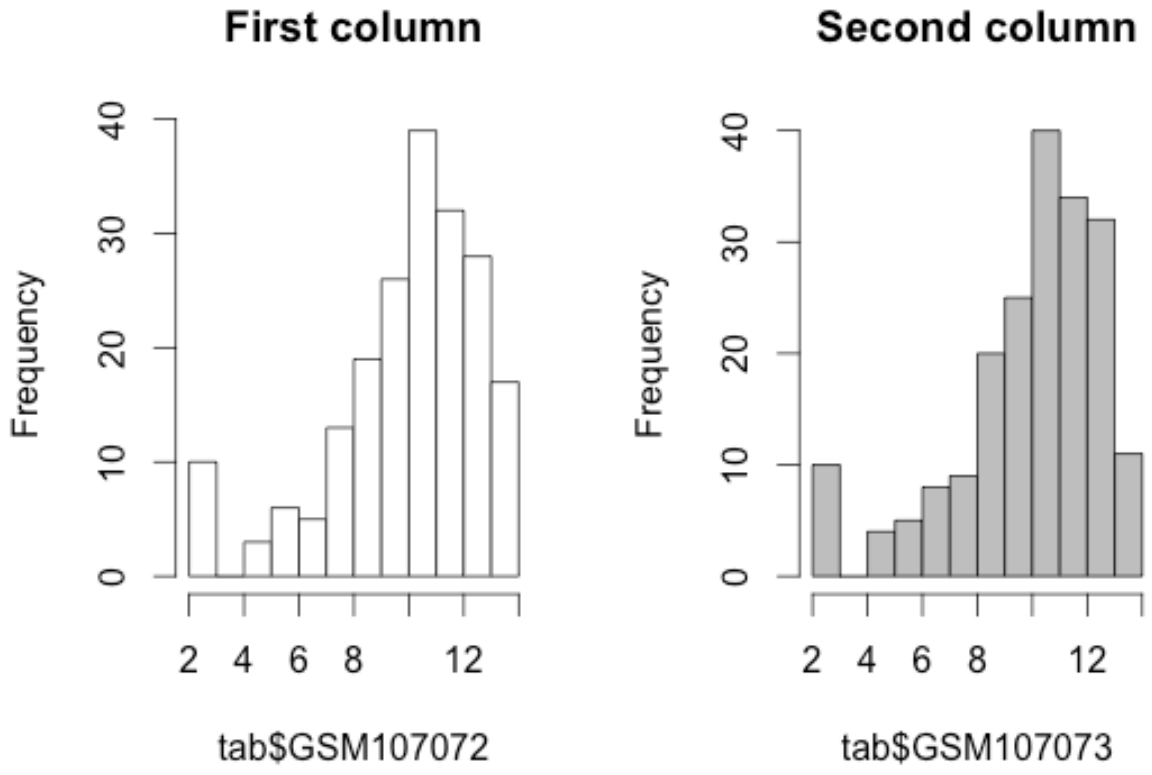
Note: `colors()`

Shows all available color names

Graphics for grouped data: Histograms

Plotting histograms of the first 2 columns of tab

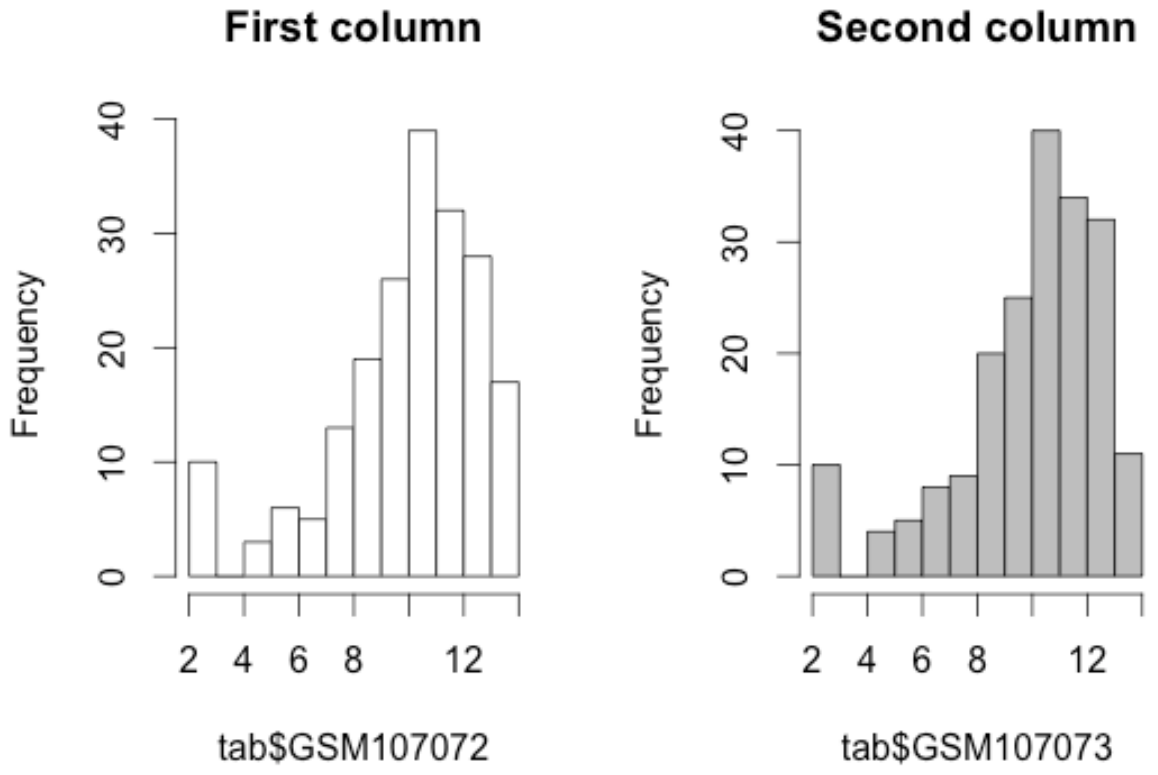
```
> par(mfrow=c(1,2))
> hist(tab[,1],col="white",main="First column")
> hist(tab[,2],col="grey",main="Second column")
```



You can also refer to the columns by the column name

Plotting histograms of the first 2 columns of tab

```
> par(mfrow=c(1,2))
> hist(tab$GSM107072,col="white",main="First column")
> hist(tab$GSM107073,col="grey",main="Second column")
```



Graphics for grouped data:Barplots

```
> caff.marital <-
matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67),nrow=3,byrow
=T)
> caff.marital
     [,1] [,2] [,3] [,4]
[1,]  652 1537  598  242
[2,]   36   46   38   21
[3,]  218  327  106   67
> colnames(caff.marital) <- c("0","1-150","151-300",">300")
> caff.marital
       0 1-150 151-300 >300
[1,] 652  1537     598  242
[2,]  36    46      38   21
[3,] 218   327     106   67
```

Note: Assign the column names

Graphics for grouped data:Barplots

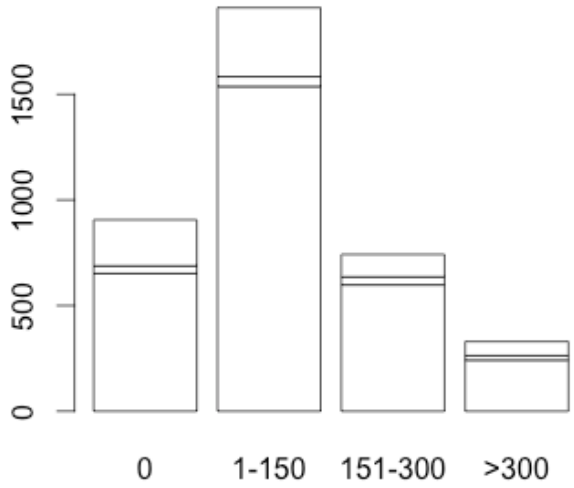
```
> rownames(caff.marital) <- c("Married","Prev.married","Single")
> caff.marital
                0 1-150 151-300 >300
Married       652  1537     598  242
Prev.married   36    46      38   21
Single        218   327     106   67
> t(caff.marital)
             Married Prev.married Single
0                652           36    218
1-150           1537           46    327
151-300          598           38    106
>300             242           21     67
```

Note: `t(table)` transposes the table.

Graphics for grouped data:Barplots

```
> par(mfrow=c(2,2))
> barplot(caff.marital, col="white")
```

```
                  0 1-150 151-300 >300
Married         652  1537     598  242
Prev.married     36    46      38   21
Single          218   327     106   67
```

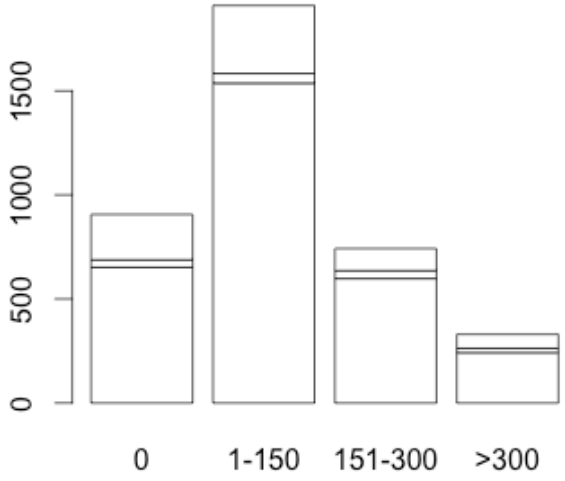


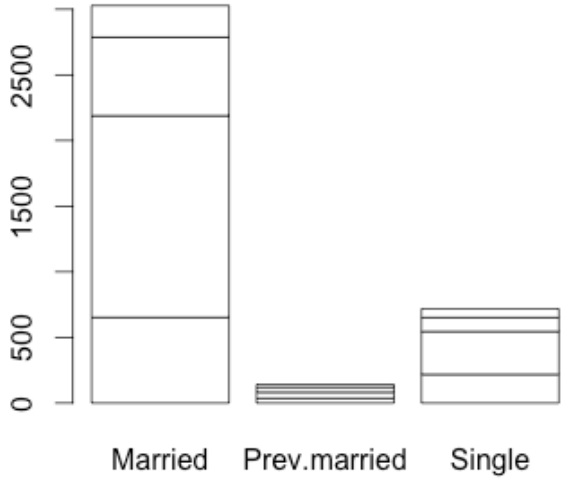
Graphics for grouped data:Barplots

```
> par(mfrow=c(2,2))
> barplot(caff.marital, col="white")
> barplot(t(caff.marital), col="white")
```

	Married	Prev.married	Single
0	652	36	218
1-150	1537	46	327
151-300	598	38	106
>300	242	21	67

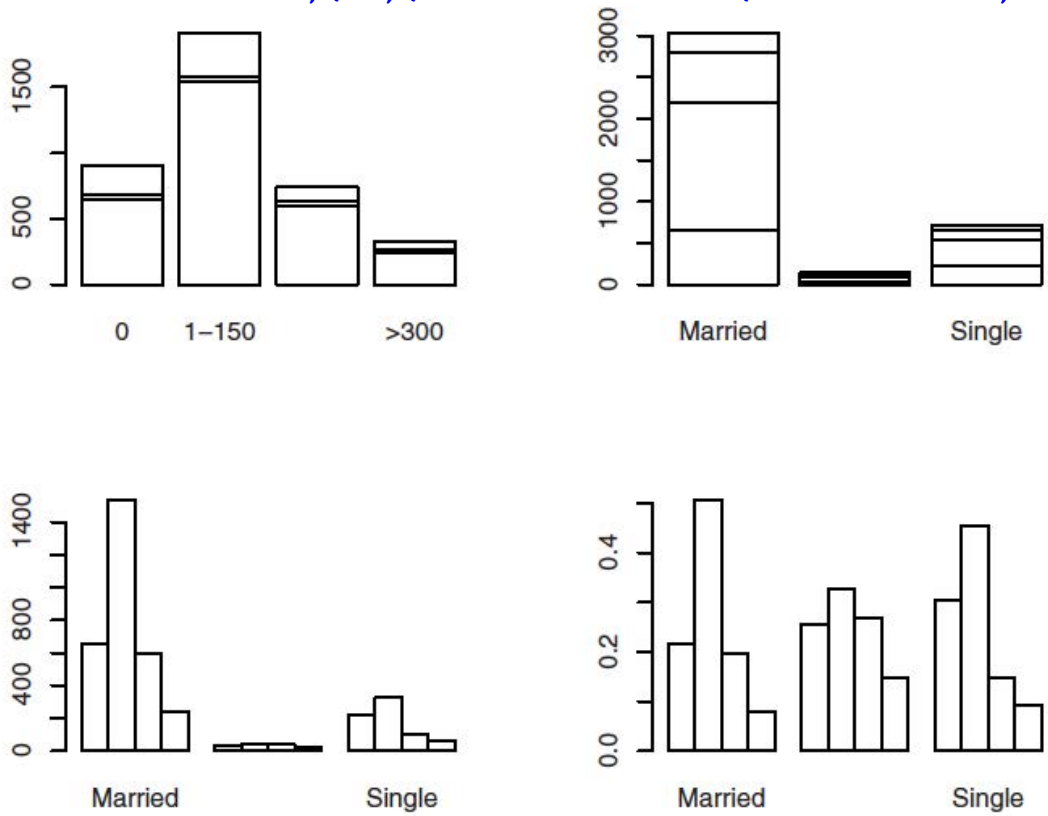
Note: `t(table)` transposes the table.





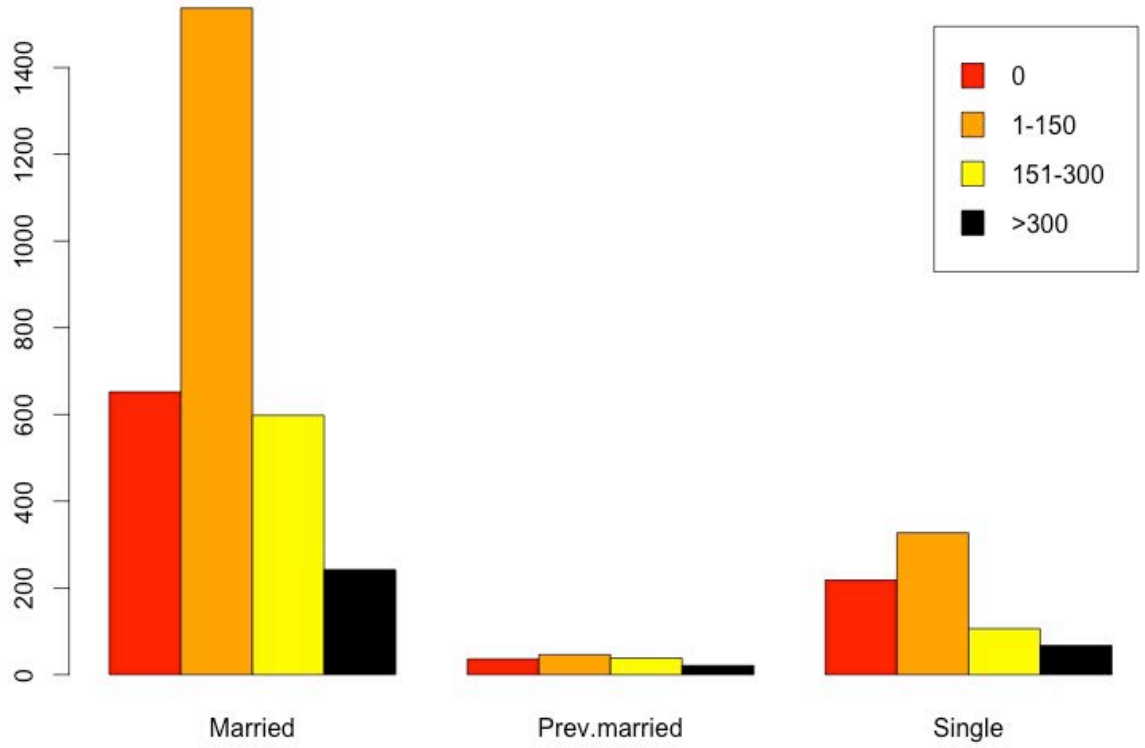
Graphics for grouped data:Barplots

```
> par(mfrow=c(2,2))
> barplot(caff.marital, col="white")
> barplot(t(caff.marital), col="white")
> barplot(t(caff.marital), col="white", beside=T)
> barplot(prop.table(t(caff.marital),2), col="white", beside=T)
> par(mfrow=c(1,1))
```



Legends

```
> par(mfrow=c(1,1))
> barplot(t(caff.marital),beside=T,
col=c("red","orange","yellow","black"),
legend=(colnames(caff.marital)))
```



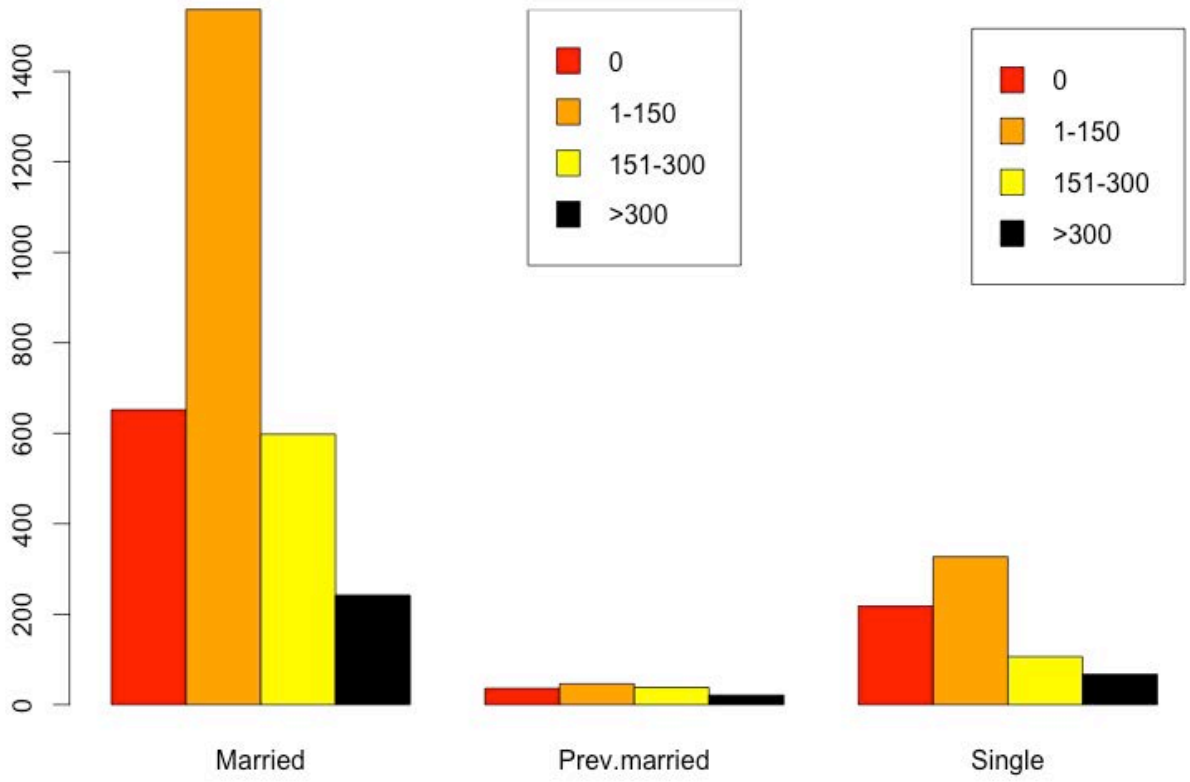
Legends

You can also use a separate function to display legends

Try:

```
> legend("top", colnames(caff.marital),
fill=c("red","orange","yellow","black"))
```

You can use
"top", "bottom",
"topleft","topright", etc.



Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

Built-in distributions in R

The common standard distributions have been built into R , and it can completely replace traditional statistical tables.

Four fundamental items can be calculated for a statistical distribution:

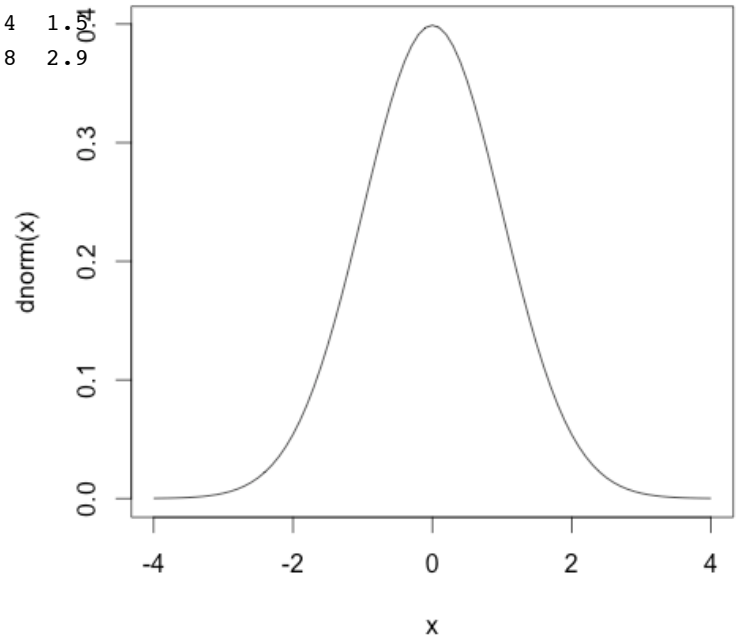
1. Density or point probability
2. Cumulated probability, distribution function
3. Quantiles
4. Pseudo-random numbers

For all distributions implemented in R , there is a function for each of the four items listed above. For example, for the normal distribution, these are named `dnorm` , `pnorm` , `qnorm` , and `rnorm` (density, probability, quantile, and random, respectively).

Continuous distribution: Normal (Gaussian) - Densities

The density for a continuous distribution is a measure of the relative probability of "getting a value close to x ". The probability of getting a value in a particular interval is the area under the corresponding part of the curve.

```
> x <- seq(-4,4,0.1)
> x
 [1] -4.0 -3.9 -3.8 -3.7 -3.6 -3.5 -3.4 -3.3 -3.2 -3.1 -3.0 -2.9 -2.8 -2.7
[15] -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3
[29] -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1
[43]  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4  1.5
[57]  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
[71]  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0
> plot(x,dnorm(x),type="l")
```



Normal (Gaussian): cumulative probability

The cumulative distribution function describes the probability of "hitting" x or less in a given distribution.

A normal distribution with a mean of 132 and a standard deviation of 13. Then, if one has a value of 160:

```
> 1-pnorm(160,mean=132,sd=13)
[1] 0.01562612
```

Meaning: 1.5% of the general population, that has that value or higher.

pnorm returns the probability of getting a value smaller than its first argument in a normal distribution with the given mean and standard deviation.

Normal (Gaussian): quantile

The quantile function is the inverse of the cumulative distribution function. The p-quantile is the value with the property that there is probability p of getting a value less than or equal to it.

```
> qnorm(0.025)
[1] -1.959964
> qnorm(0.01)
[1] -2.326348
```

Normal (Gaussian) – Random sampling

The use of the functions that generate random numbers is straightforward. The first argument specifies the number of random numbers to compute, and the subsequent arguments are similar to those for other functions related to the same distributions.

```
> rnorm(10)
[1] -0.2996466 -0.1718510 -0.1955634 1.2280843 -2.6074190
[6] -0.2999453 -0.4655102 -1.5680666 1.2545876 -1.8028839
> rnorm(10)
[1] 1.7082495 0.1432875 -1.0271750 -0.9246647 0.6402383
[6] 0.7201677 -0.3071239 1.2090712 0.8699669 0.5882753
> rnorm(10,mean=7,sd=5)
[1] 8.934983 8.611855 4.675578 3.670129 4.223117 5.484290
[7] 12.141946 8.057541 -2.893164 13.590586
> rbinom(10,size=20,prob=.5)
[1] 12 11 10 8 11 8 11 8 8 13
```

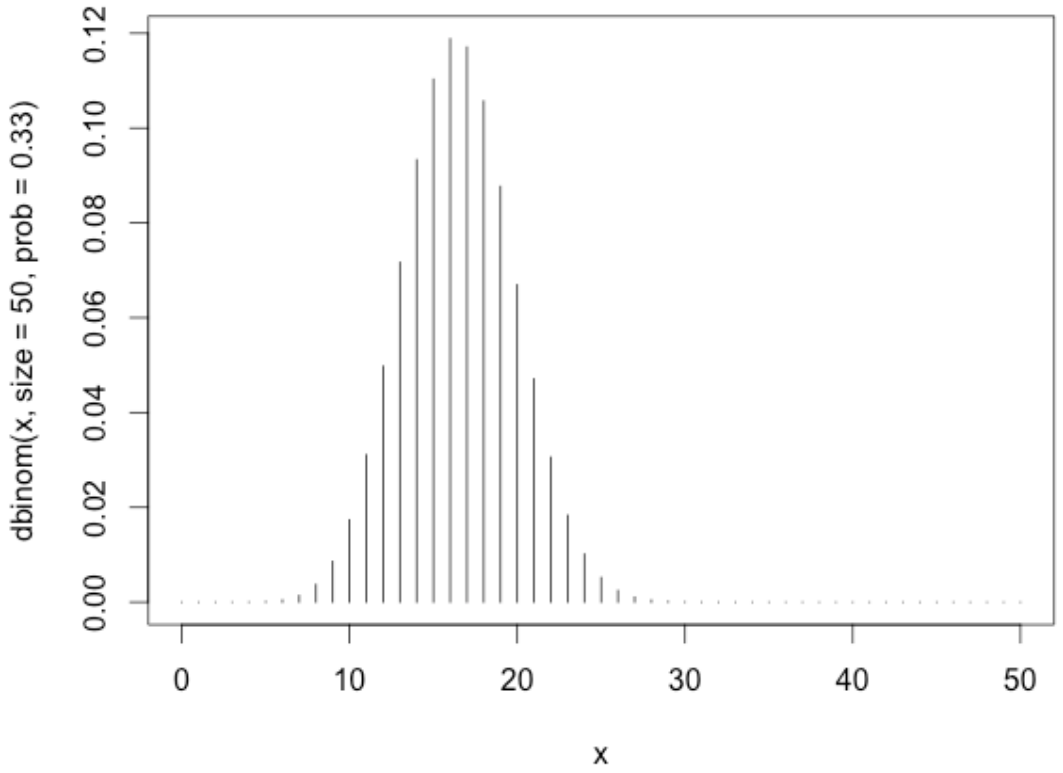
Try it yourself: Distributions

Function	Distribution
`pnorm(x,mean,sd)`	*Normal*
`plnorm(x,mean,sd)`	*Lognormal*
`pt(x,df)`	*Student's t*
`pf(x,n1,n2)`	*F distribution*
`pchisq(x,df)`	*Chi-square*
`pbinom(x,n,p)`	*Binomial*
`ppois(x,lambda)`	*Poisson*
`punif(x,min,max)`	*Uniform*
`pexp(x,rate)`	*Exponential*
`pgamma(x,shape,scale)`	*Gamma*
`pbeta(x,a,b)`	*Beta*

Discrete distribution: Binomial

```
> x <- 0:50
> plot(x,dbinom(x,size=50,prob=.33),type="h")
```

In addition to x, you have to specify the number of trials n and the probability parameter p.



Binomial distribution

Twenty patients are given two treatments each (blindly and in randomized order) and then asked whether treatment A or B worked better. It turned out that 16 patients liked A better. If there was no difference between the two treatments, then we would expect the number of people favouring treatment A to be binomially distributed with p = 0.5 and n = 20. How (im)probable would it then be to obtain what we have observed?

```
> 1-pbinom(15,size=20,prob=.5)
[1] 0.005908966
```

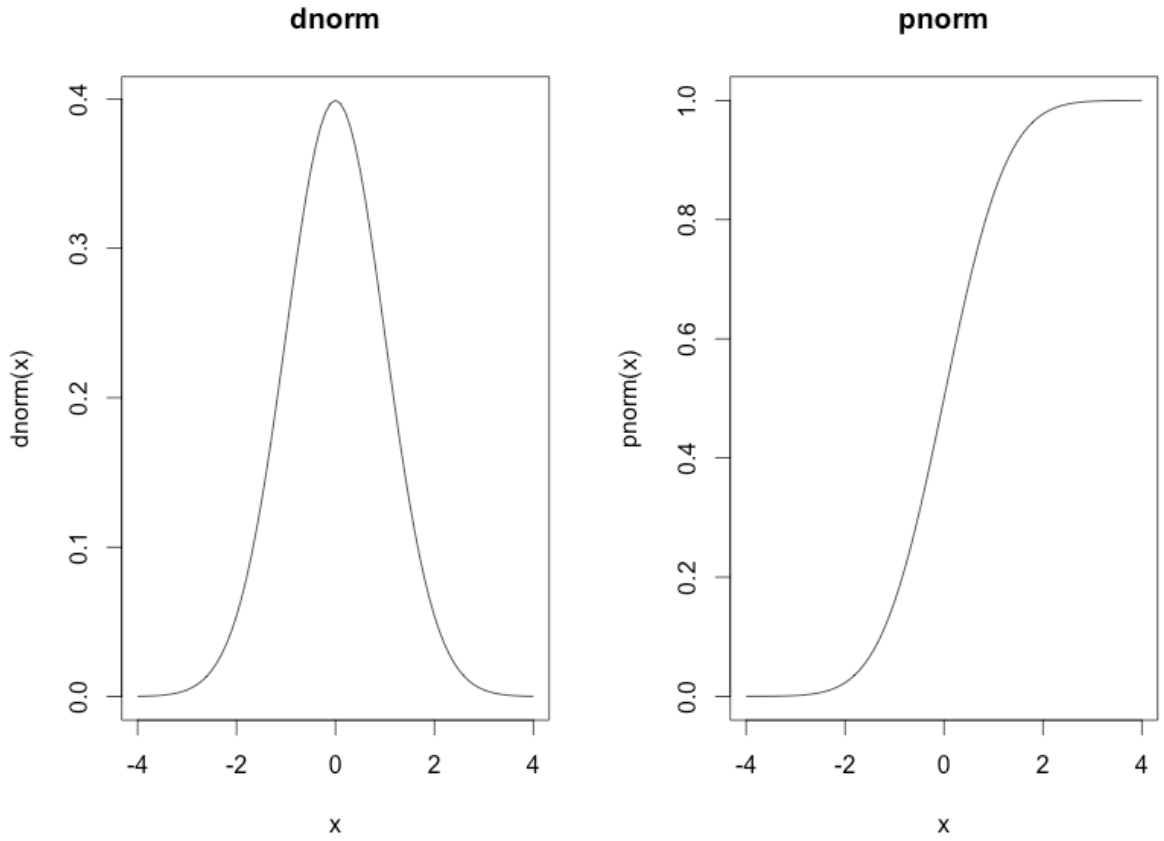
What we need is the probability of the observed or more extreme, and pbinom is giving the probability of 16 or less. We need to use “15 or less” instead.

Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

Plotting Normal distribution

```
> par(mfrow=c(1,2))
> plot(x,dnorm(x),type="l",main="dnorm")
> plot(x,pnorm(x),type="l",main="pnorm")
```



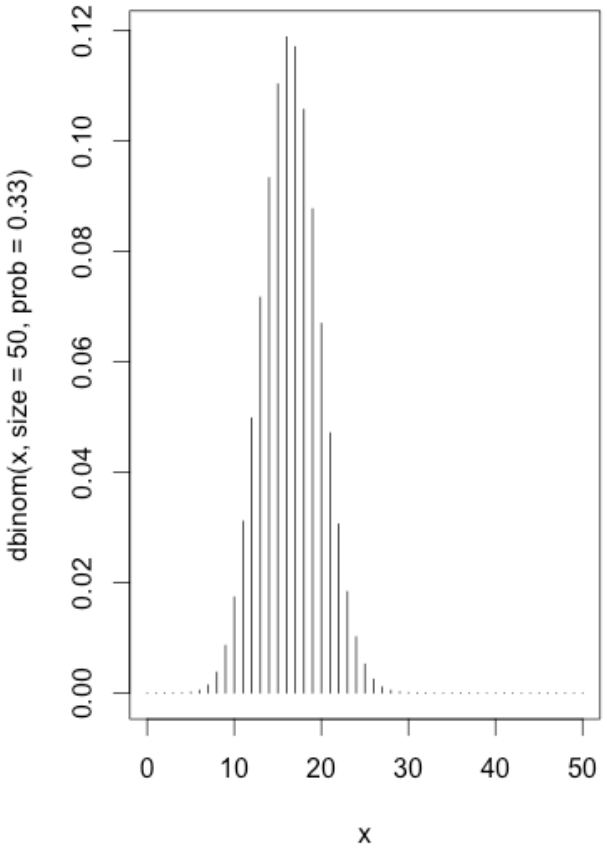
Plotting Binomial distribution

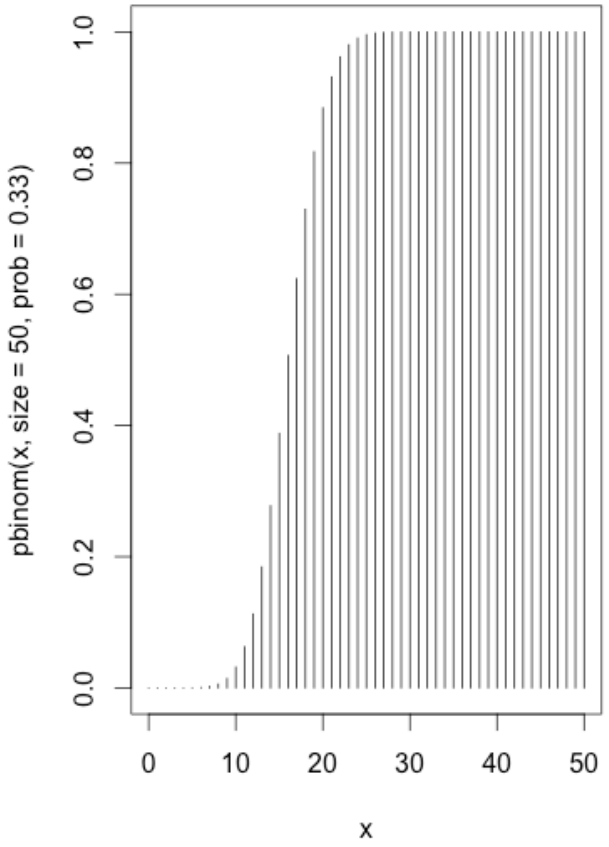
```
> x <- 0:50
```

Challenge 4: Plot the Binomial density and cumulative distributions for x (n=50, p=0.33)

Challenge 4: Binomial distribution

```
> par(mfrow=c(1,2))
> plot(x,dbinom(x,size=50,prob=.33),type="h")
> plot(x,pbinom(x,size=50,prob=.33),type="h")
```





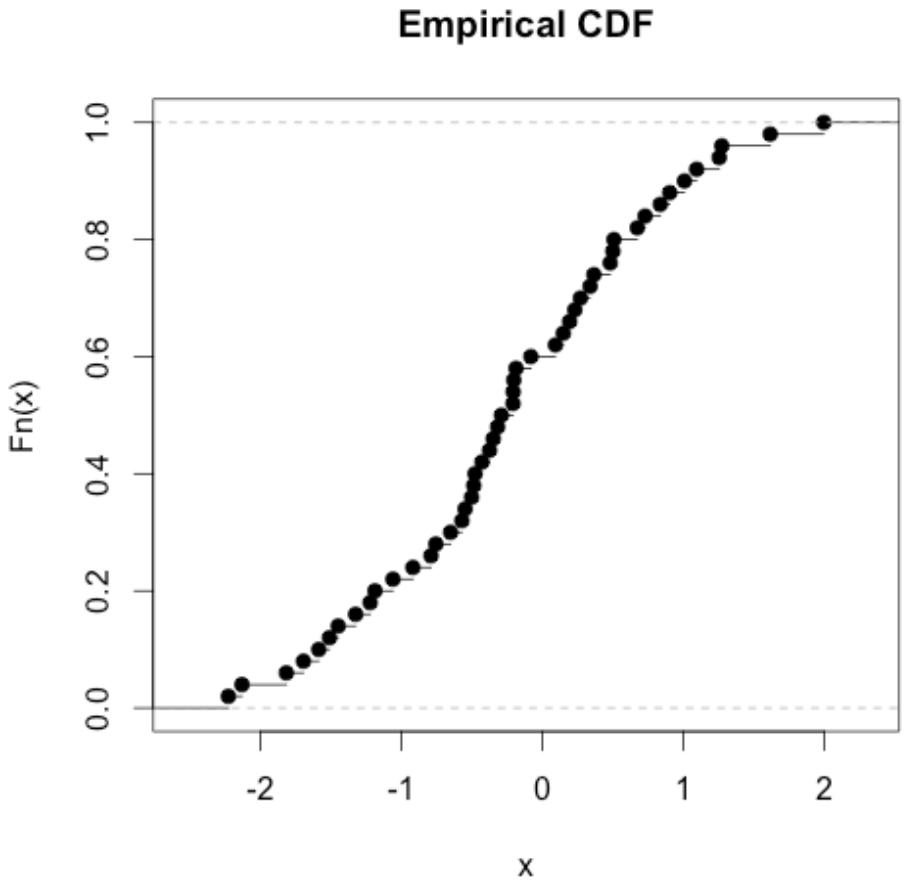
Empirical cumulative distribution function

One purpose of calculating the empirical cumulative distribution function (c.d.f.) is to see whether data can be assumed normally distributed – or distributed according to some other distribution

```
> x <- rnorm(50)
> plot(ecdf(x),main="Empirical CDF")
```

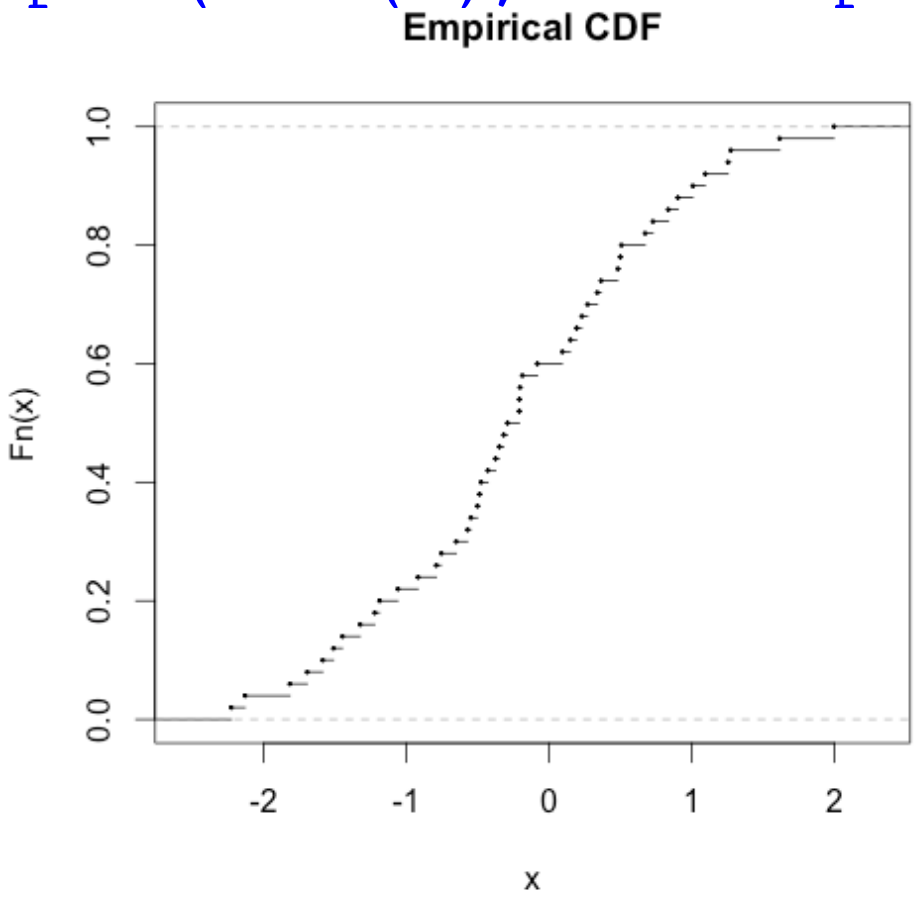
Empirical cumulative distribution function

```
> x <- rnorm(50)
> plot(ecdf(x),main="Empirical CDF")
```



Empirical cumulative distribution function

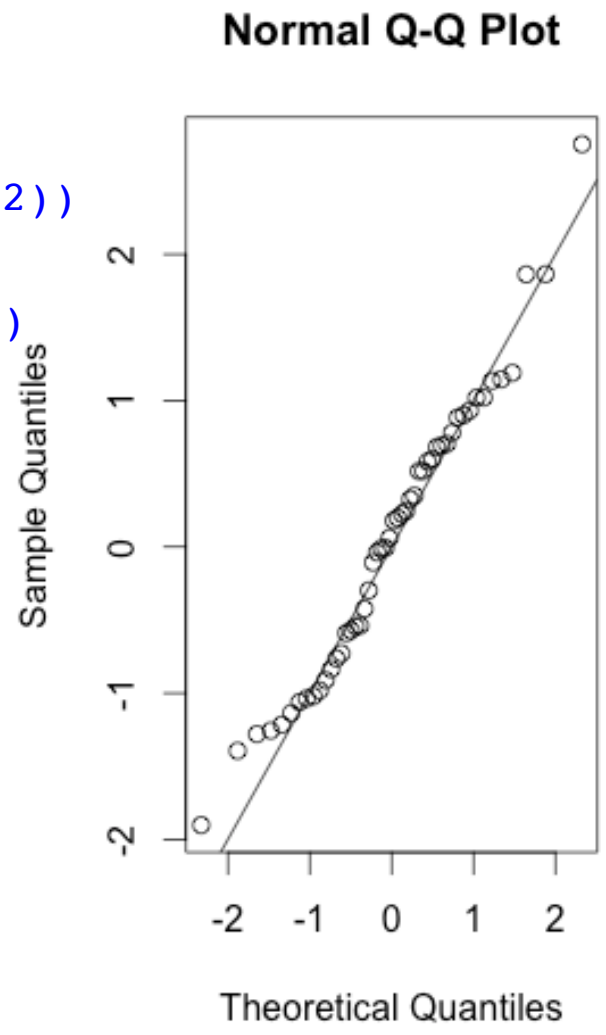
```
> x <- rnorm(50)
> plot(ecdf(x),main="Empirical CDF”,cex=0.2)
```

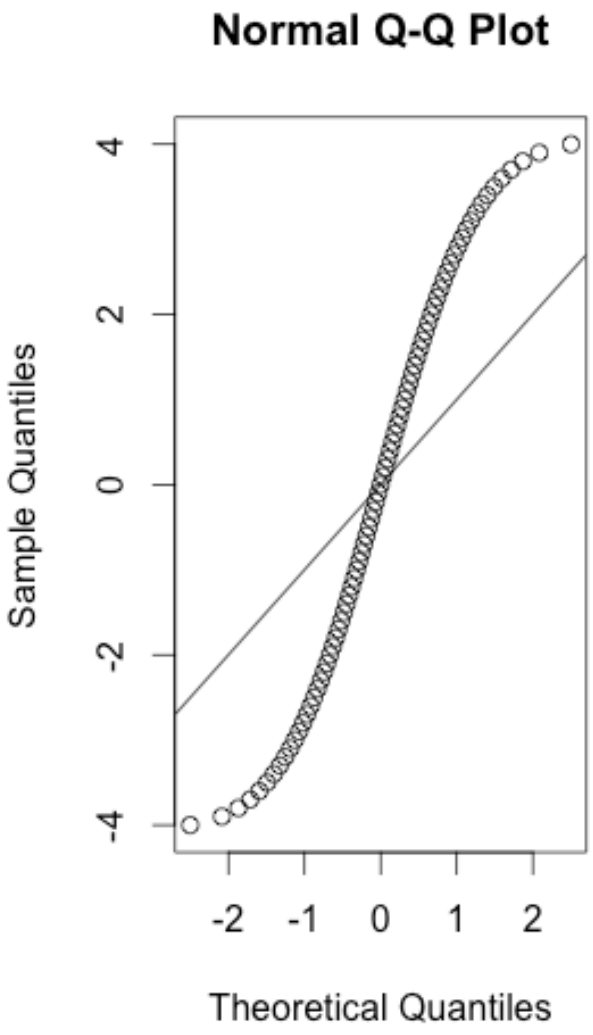


Reduce the size of the points with the “cex” parameter

qqnorm : Is the data normally distributed

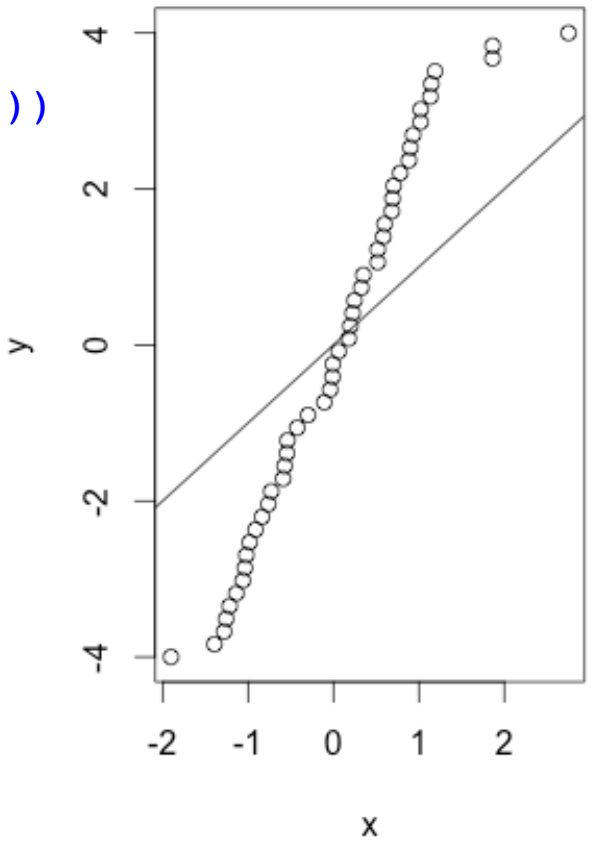
```
> par(mfrow=c(1,2))
> x <- rnorm(50)
> y=seq(-4,4,0.1)
> qqnorm(x)
> abline(0,1)
> qqnorm(y)
> abline(0,1)
```

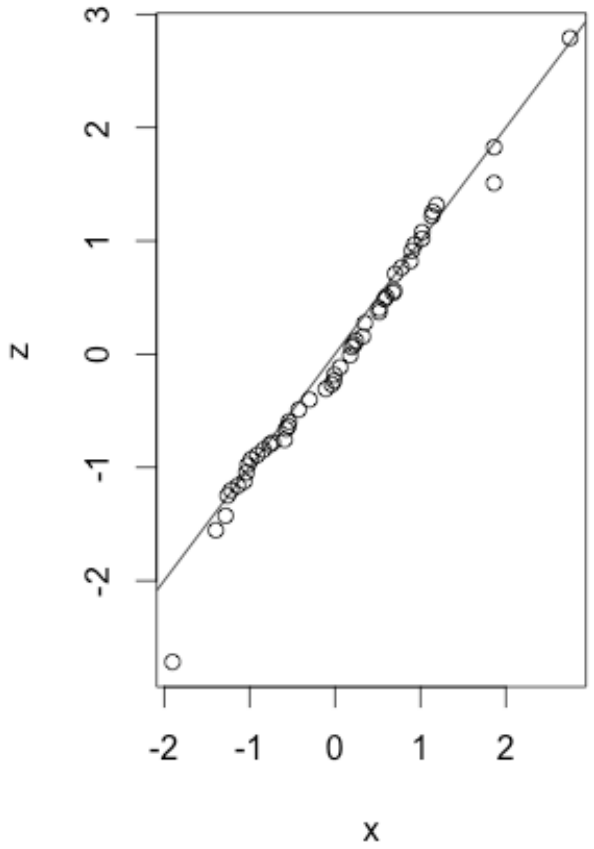




QQPlots: Are the datasets sampled from the same distribution?

```
> par(mfrow=c(1,2))
> x <- rnorm(50)
> y=seq(-4,4,0.1)
> z=rnorm(100)
> qqplot(x,y)
> abline(0,1)
> qqplot(x,z)
> abline(0,1)
```





Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

One sample t-test

Following is data for women's daily energy intake:

```
> daily.intake <- c(5260,5470,5640,6180,6390,6515,
6805,7515,7515,8230,8770)
> quantile(daily.intake)
0% 25% 50% 75% 100%
5260 5910 6515 7515 8770
```

QUESTION: You might wish to investigate whether the women's energy intake deviates systematically from a recommended value of 7725 kJ. Assuming that data come from a normal distribution, the objective is to test whether this distribution might have mean μ = 7725. This is done with t.test() as follows:

One sample t-test

QUESTION: You might wish to investigate whether the women's energy intake deviates systematically from a recommended value of 7725 kJ. Assuming that data come from a normal distribution, the objective is to test whether this distribution might have mean μ = 7725. This is done with t.test() as follows:

```
> t.test(daily.intake,mu=7725)

    One Sample t-test

data:  daily.intake
t = -2.8208, df = 10, p-value = 0.01814
alternative hypothesis: true mean is not equal to 7725
95 percent confidence interval:
 5986.348 7520.925
sample estimates:
mean of x
 6753.636
```

One sample t-test

```
> t.test(daily.intake,mu=7725)

    One Sample t-test

data:  daily.intake
t = -2.8208, df = 10, p-value = 0.01814
alternative hypothesis: true mean is not equal to 7725
95 percent confidence interval:
 5986.348 7520.925
sample estimates:
mean of x
 6753.636
```

Wilcoxon signed ranked test

Non-Parametric test:

```
> wilcox.test(daily.intake, mu=7725)

    Wilcoxon signed rank test with continuity correction

data:  daily.intake
V = 8, p-value = 0.0293
alternative hypothesis: true location is not equal to 7725

Warning message:
In wilcox.test.default(daily.intake, mu = 7725) :
  cannot compute exact p-value with ties
```

There is not as much output as from `t.test` as there is no parameter estimate in a nonparametric test and therefore no confidence limits, etc., either.

Paired t-test

Paired tests are used when there are two measurements on the same experimental unit. The theory is essentially based on taking differences and thus reducing the problem to that of a one-sample test

Remember the table we read?

```
> tab = read.table("GSE47561.txt", header=T, row.names=1)
> head(tab)
          GSM107072 GSM107073 GSM107075 GSM107076
1007_s_at  9.326370  9.509078  9.470701  9.579381
1053_at    5.770976  6.871807  4.811624  5.896438
117_at     4.837100  4.100549  3.025137  3.930648
121_at     5.320877  5.263376  5.132677  5.144603
1255_g_at  2.221335  2.221335  2.221335  2.221335
1294_at    4.727786  4.673524  5.425090  4.765807
```

Challenge 5: Assuming that column 1(GSM107072) is the gene expression of PatientA before treatment and column 2 (GSM107073) is the gene expression of the same patient after treatment, was there any difference in gene expression of the patient after treatment?

Challenge 5

```
> t.test(tab[,1], tab[,2])
      Welch Two Sample t-test

data:  tab[, 1] and tab[, 2]
t = 0.1349, df = 394, p-value = 0.8927
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4969145  0.5701452
sample estimates:
mean of x mean of y
 9.943704  9.907088
```

WRONG!!!!

```
> t.test(tab[,1], tab[,2], paired = T)
      Paired t-test

data:  tab[, 1] and tab[, 2]
t = 0.8597, df = 197, p-value = 0.391
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.04737259  0.12060331
sample estimates:
mean of the differences
             0.03661536
```

You have to specify paired=T explicitly in the call, indicating that you want a paired test

Outline

- Brief Introduction to R
- Statistics on vectors
- Statistics on tables
- Built-in probability distributions
- Plotting distributions in R
- Significance testing
- Correlation and regression

Correlations

- A correlation coefficient is a symmetric, scale-invariant measure of association between two random variables.
- It ranges from –1 to +1, where the extremes indicate perfect correlation and 0 means no correlation.
- The sign is negative when large values of one variable are associated with small values of the other and positive if both variables tend to be large or small simultaneously.

Function for correlation in R:

```
cor()
```

Correlations

Challenge 6: Use the "`GSE47561.txt`" table and calculate:

1. Pearson correlation between the first 2 columns
2. Spearman correlation between the first 2 column
3. Pearson correlation between all pairs of columns.

Correlations

```
> tab = read.table("GSE47561.txt", header=T, row.names=1)
```

1. Pearson correlation between the first 2 columns

Correlation of 2 columns at a time,

```
> cor(tab[,1], tab[,2])
[1] 0.9753716
```

2. Spearman correlation between the first 2 column

```
> cor(tab[,1], tab[,2], method="spearman")
[1] 0.9602081
```

NOTE: Method can be "pearson" OR "kendall" OR "spearman". Pearson is be default

Correlations

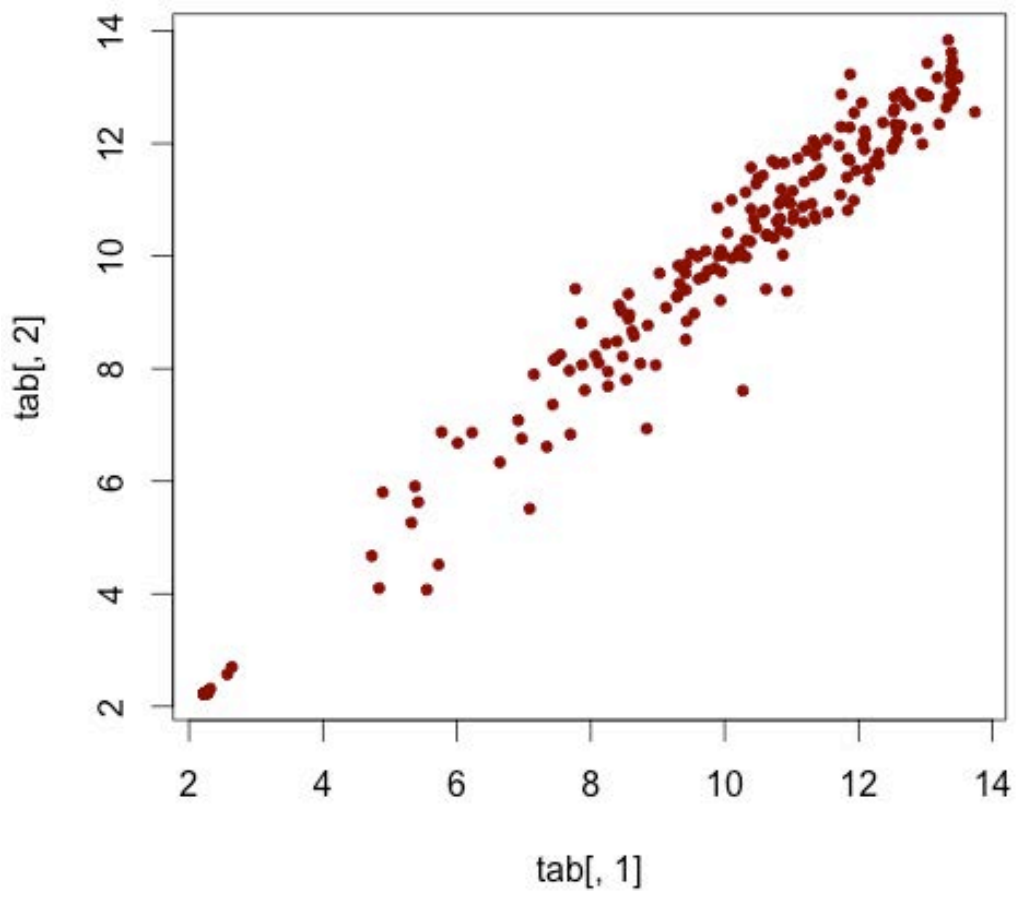
3. Pearson correlation between all pairs of columns.

Correlation of the entire table

```
> cor(tab)
           GSM107072 GSM107073 GSM107075 GSM107076
GSM107072 1.0000000 0.9753716 0.9533168 0.9829057
GSM107073 0.9753716 1.0000000 0.9363837 0.9634629
GSM107075 0.9533168 0.9363837 1.0000000 0.9751666
GSM107076 0.9829057 0.9634629 0.9751666 1.0000000
```

Plot the correlations

```
> tab=read.table("GSE47561.txt",header=T,row.names=1)
> plot(tab[,1],tab[,2],pch=20,col="darkred")
```

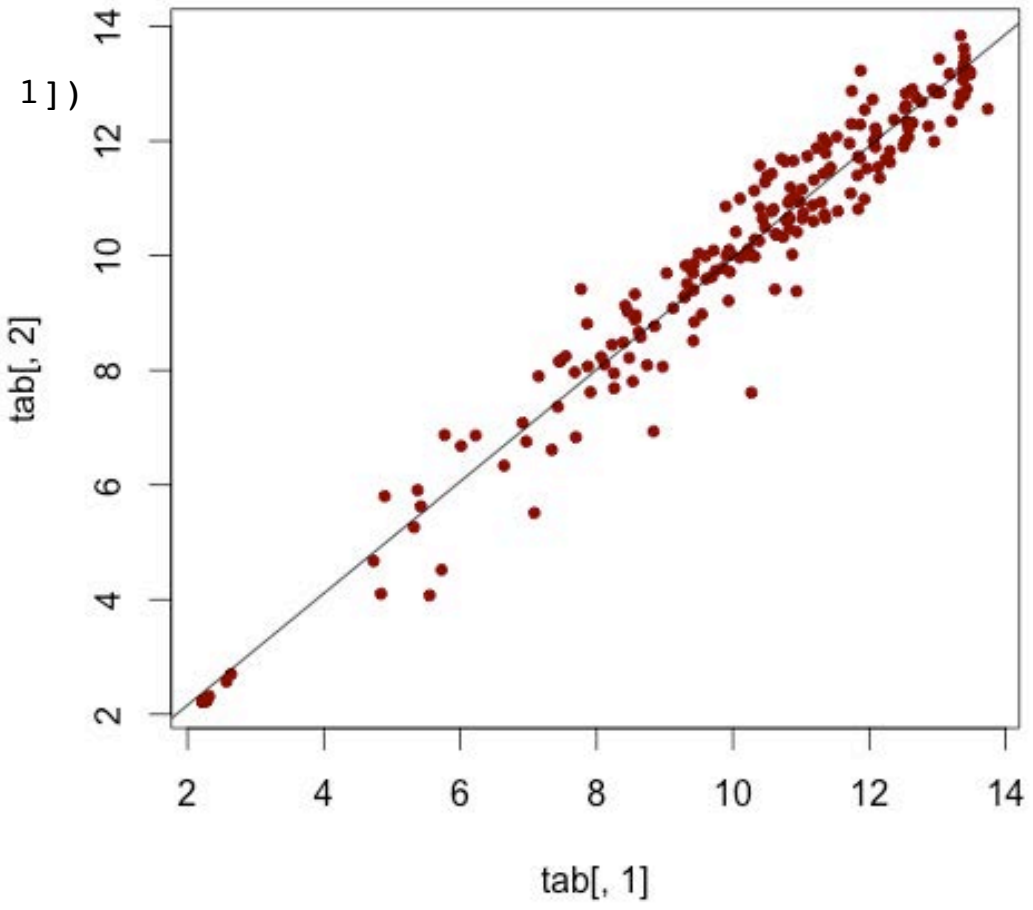


Add the linear regression line

```
> abline(lm(tab[,2] ~ tab[,1]))
> lm(tab[,2] ~ tab[,1])

Call:
lm(formula = tab[, 2] ~ tab[, 1])

Coefficients:
(Intercept)      tab[, 1]
     0.2079        0.9754
```



Please fill out a short survey

https://biocore.cri.uchicago.edu/cgi-bin/survey.cgi?id=36

Thank you!