# hw10 R_slides

YI-Hsuan Lo



# 1. general calculating

| Operator | Description |
|---|---|
| + | Addition |
| – | Subtraction |
| * | Multiplication |
| / | Division |
| ^ | Exponent |
| %% | Modulus (Remainder from division) |
| %/% | Integer Division |

```
20*3
```

```
## [1] 60
```

| Operator | Description |
|---|---|
| > | bigger than |
| < | smaller than |
| >= | bigger than or equal to |
| <= | smaller then or equal to |
| == | equal to |
| != | not equal to |

```
8>=9
```

```
## [1] FALSE
```

# 2. Data Types

## vector

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
d <- a[c(2,4)] # 2nd and 4th elements of vector
```

```
print(a)
```

```
## [1]  1.0  2.0  5.3  6.0 -2.0  4.0
```

```
print(b)
```

```
## [1] "one"   "two"   "three"
```

```
print(c)
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE
```

```
print(d)
```

```
## [1] 2 6
```

## matrix

```
# generates 5 x 4 numeric matrix
x=matrix(1:20, nrow=5,ncol=4)
print(x)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

$$x = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}$$

```
x[,4] # 4th column of matrix
```

```
## [1] 16 17 18 19 20
```

```
x[3,] # 3rd row of matrix
```

```
## [1]  3  8 13 18
```

```
x[2:4,1:3] # rows 2,3,4 of columns 1,2,3
```

```
##      [,1] [,2] [,3]
## [1,]    2    7   12
## [2,]    3    8   13
## [3,]    4    9   14
```

## array

```
array(1:24, dim = c(4, 3, 2))
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   13   17   21
## [2,]   14   18   22
## [3,]   15   19   23
## [4,]   16   20   24
```

# factor

```
food <- c("cake", "cookie", "fish", "cake", "fish")
food <- factor(food)
food
```

```
## [1] cake   cookie fish   cake   fish
## Levels: cake cookie fish
```

```
levels(food)
```

```
## [1] "cake"   "cookie" "fish"
```

# data frame

```
d <- c(1,2,3,4)
e <- c("red", "white", "red", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
mydata <- data.frame(d,e,f)
names(mydata) <- c("ID","Color","Passed") # variable names
```

```
print(mydata)
```

```
##   ID Color Passed
## 1  1   red   TRUE
## 2  2 white   TRUE
## 3  3   red   TRUE
## 4  4  <NA>  FALSE
```

# list

```
student <- list(gender="woman", age=15)
student
```

```
## $gender
## [1] "woman"
##
## $age
## [1] 15
```

```
str(student)
```

```
## List of 2
##  $ gender: chr "woman"
##  $ age   : num 15
```

# 3. Functions

## `if()` statements

> if (test_expression) { statement }

```
x <- 5
if(x > 0){
print("Positive number")
}
```

```
## [1] "Positive number"
```

> if (test_expression) { statement1 } else { statement2 }

```
x <- -5
if(x > 0){
print("Non-negative number")
} else {
print("Negative number")
}
```

```
## [1] "Negative number"
```

```
if(x > 0) print("Non-negative number") else print("Negative number")
```

```
## [1] "Negative number"
```

> if ( test_expression1) { statement1 } else if ( test_expression2) { statement2 } else if ( test_expression3) { statement3 } else { statement4 }

```
x <- 0
if (x < 0) {
print("Negative number")
} else if (x > 0) {
print("Positive number")
} else
print("Zero")
```

```
## [1] "Zero"
```

# `for()` loops

> for (val in sequence) { statement }

```
x <- c(2,5,3,9,8,11,6)
count <- 0
for (val in x) {
if(val %% 2 == 0)  count = count+1
}
print(count)
```

```
## [1] 3
```

# `wile()` loops

> while (test_expression) { statement }

```
i <- 1
while (i < 6) {
print(i)
i = i+1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

# `repeat loop` and `break``next`

`repeat loop`

> repeat { statement }

```
x <- 1
repeat {
print(x)
x = x+1
if (x == 6){
break
}
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

`break` >if (test_expression) { break }

```
x <- 1:5
for (val in x) {
if (val == 3){
break
}
print(val)
}
```

```
## [1] 1
## [1] 2
```

`next`

> if (test_condition) { next }

```
x <- 1:5
for (val in x) {
if (val == 3){
next
}
print(val)
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 5
```

# useful functions

```
object='hello'

length(object) # number of elements or components
```

```
## [1] 1
```

```
str(object)    # structure of an object
```

```
##  chr "hello"
```

```
class(object)  # class or type of an object
```

```
## [1] "character"
```

```
names(object)  # names
```

```
## NULL
```

```
c(object,object)       # combine objects into a vector
```

```
## [1] "hello" "hello"
```

```
cbind(object, object) # combine objects as columns
```

```
##      object  object
## [1,] "hello" "hello"
```

```
rbind(object, object) # combine objects as rows
```

```
##        [,1]
## object "hello"
## object "hello"
```

```
object     # prints the object
```

```
## [1] "hello"
```

```
ls()       # list current objects
```

```
##  [1] "a"       "b"       "c"       "count"   "d"       "e"       "f"
##  [8] "food"    "i"       "mydata"  "object"  "student" "val"     "x"
```

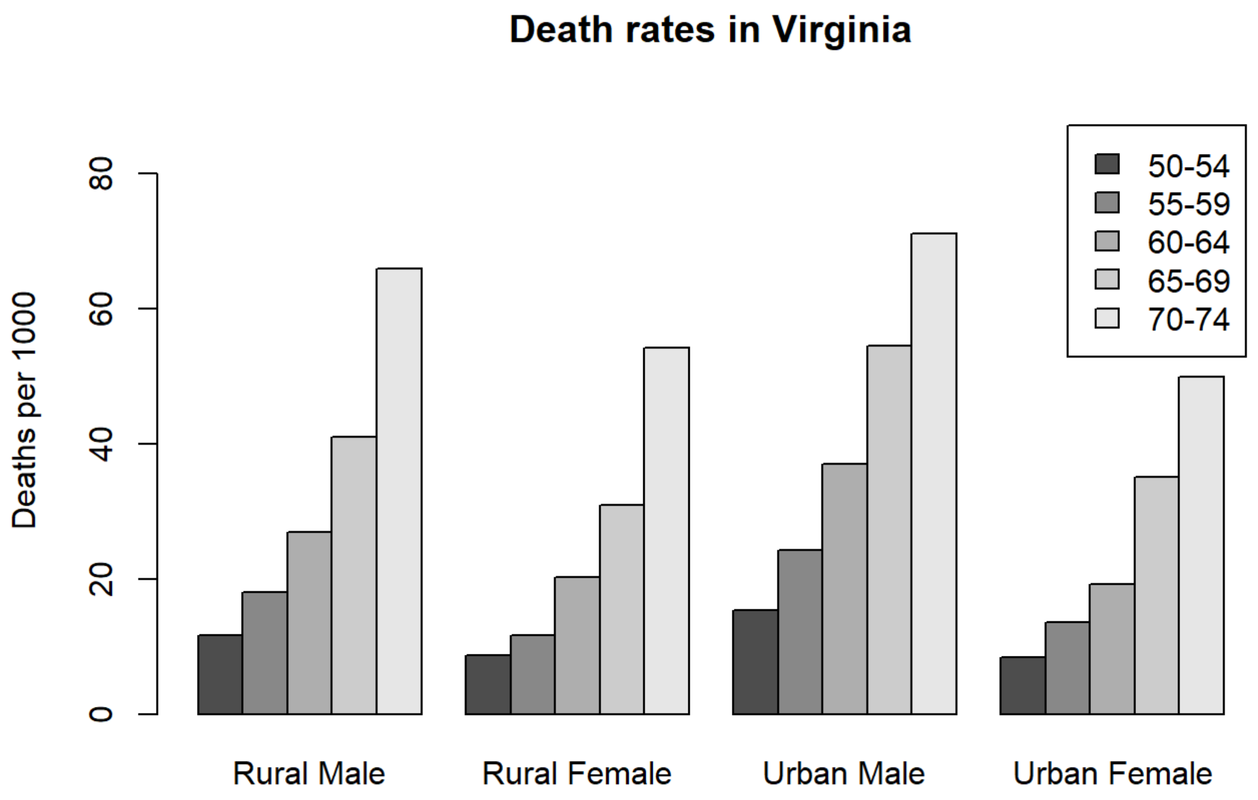
```
rm(object) # delete an object

fix(object)               # edit in place
```

# 4. Graphics

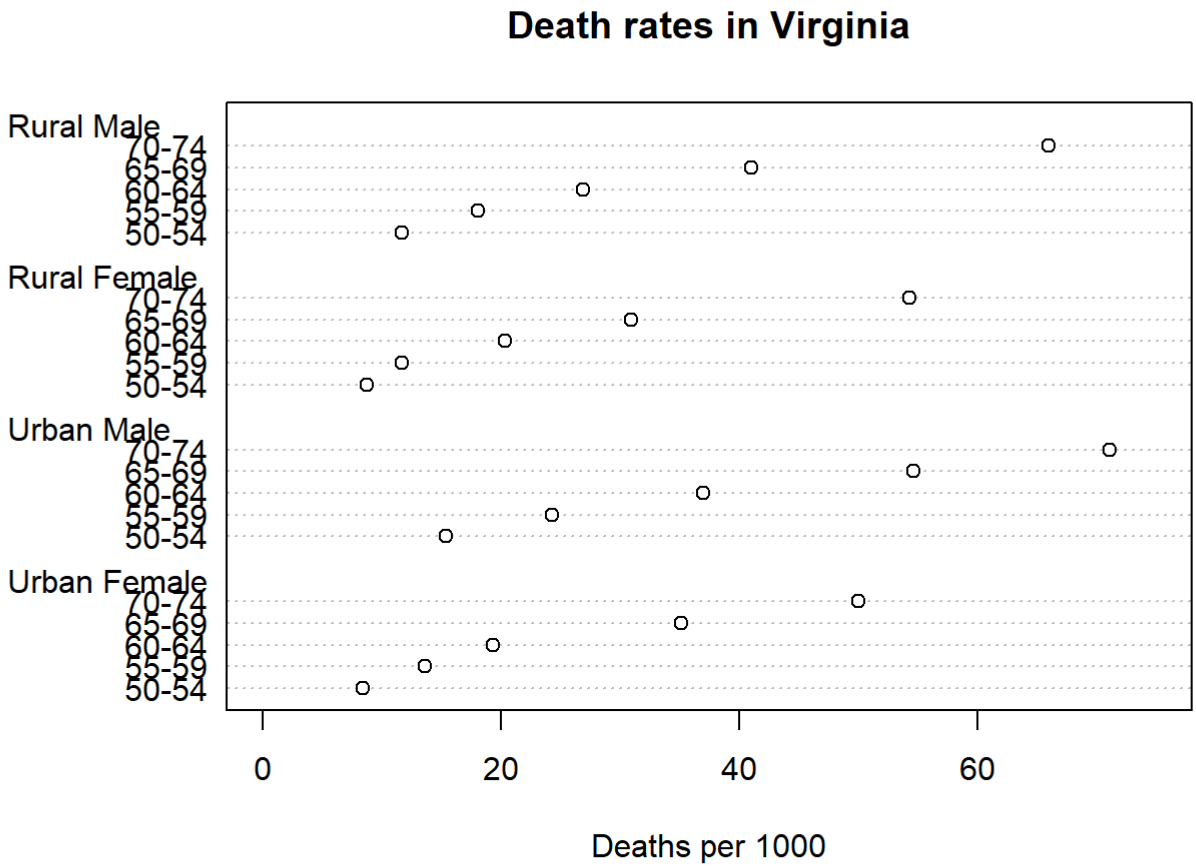
## bar charts

```
barplot(VADeaths, beside=TRUE, legend=TRUE, ylim=c(0, 90),
ylab="Deaths per 1000",
main="Death rates in Virginia")
```



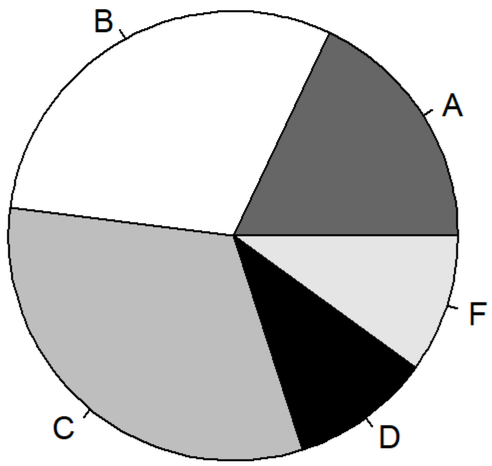

## dot charts

```
dotchart(VADeaths, xlim=c(0, 75),
 xlab="Deaths per 1000",
 main="Death rates in Virginia")
```



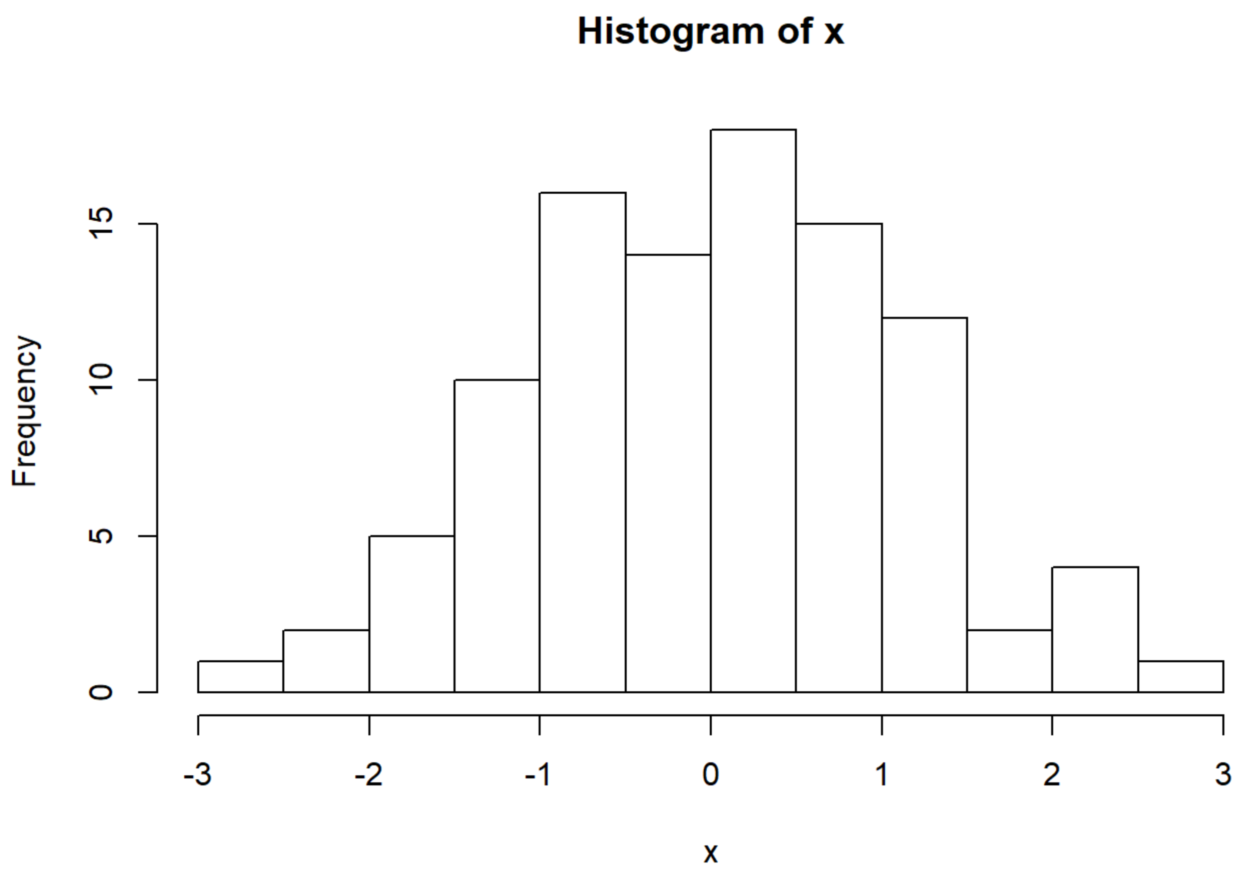

## pie charts

```
groupsizes <- c(18, 30, 32, 10, 10)
labels <- c("A", "B", "C", "D", "F")
pie(groupsizes, labels, col=c("grey40", "white", "grey", "black", "grey90"))
```



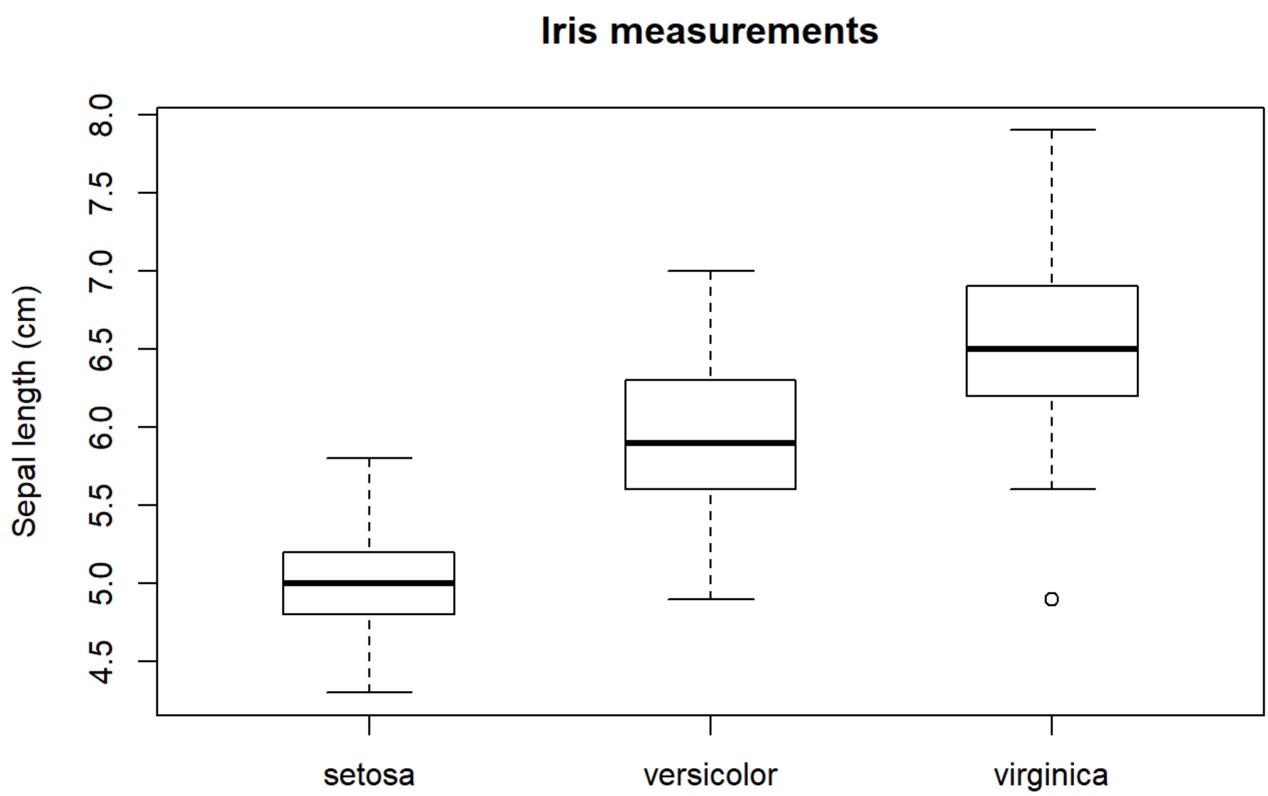

## histograms

```
x <- rnorm(100)
 hist(x)
```




## box plots

```
boxplot(Sepal.Length~Species, data = iris,ylab = "Sepal length (cm)", main = "Iris measuremen
ts",
boxwex = 0.5)
```

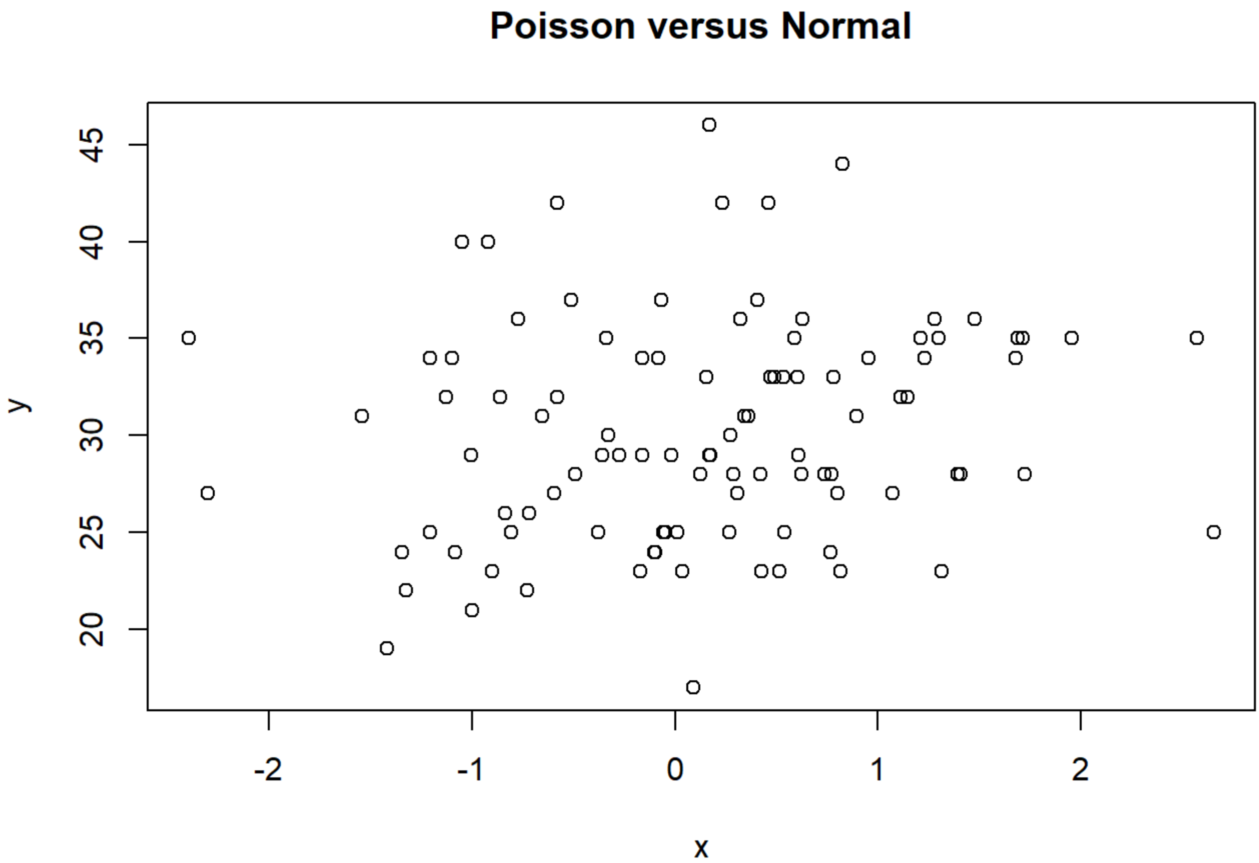



## scatter plots

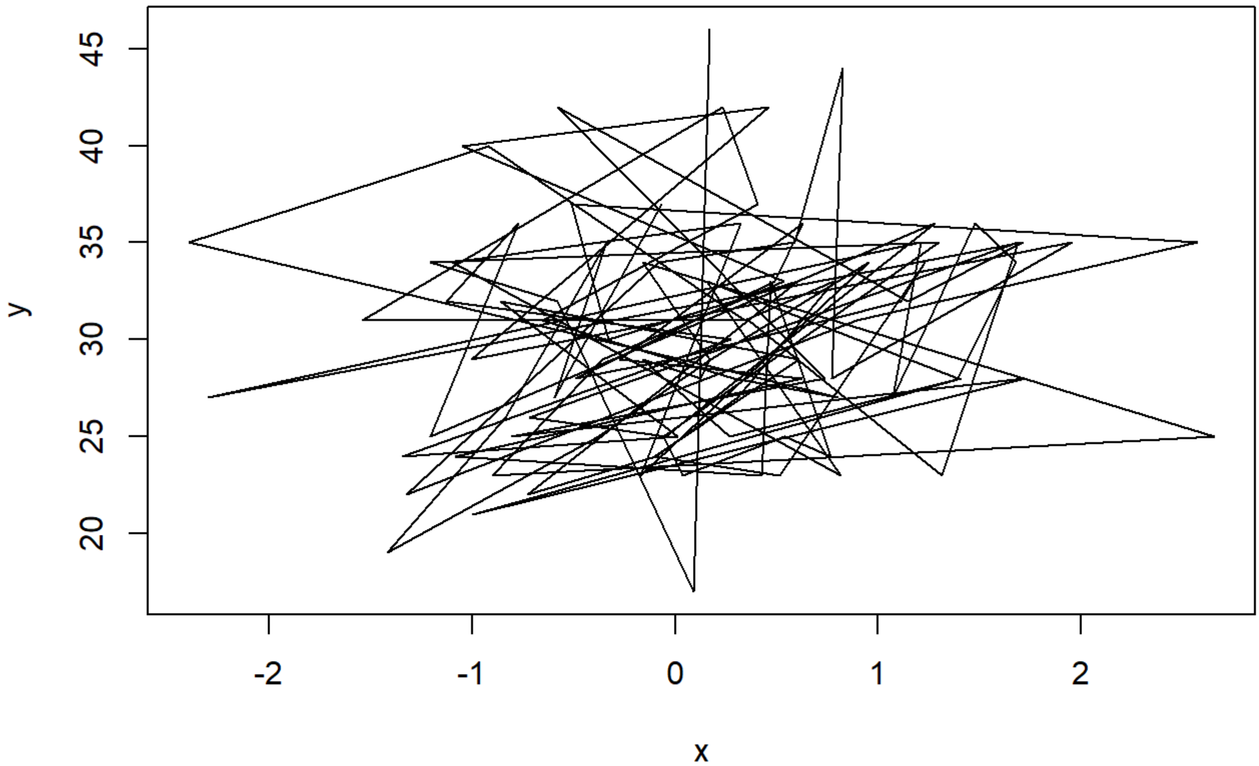
```
x <- rnorm(100) # assigns 100 random normal observations to x
y <- rpois(100, 30) # assigns 100 random Poisson observations
# to y; mean value is 30
mean(y)
```

```
## [1] 30.19
```

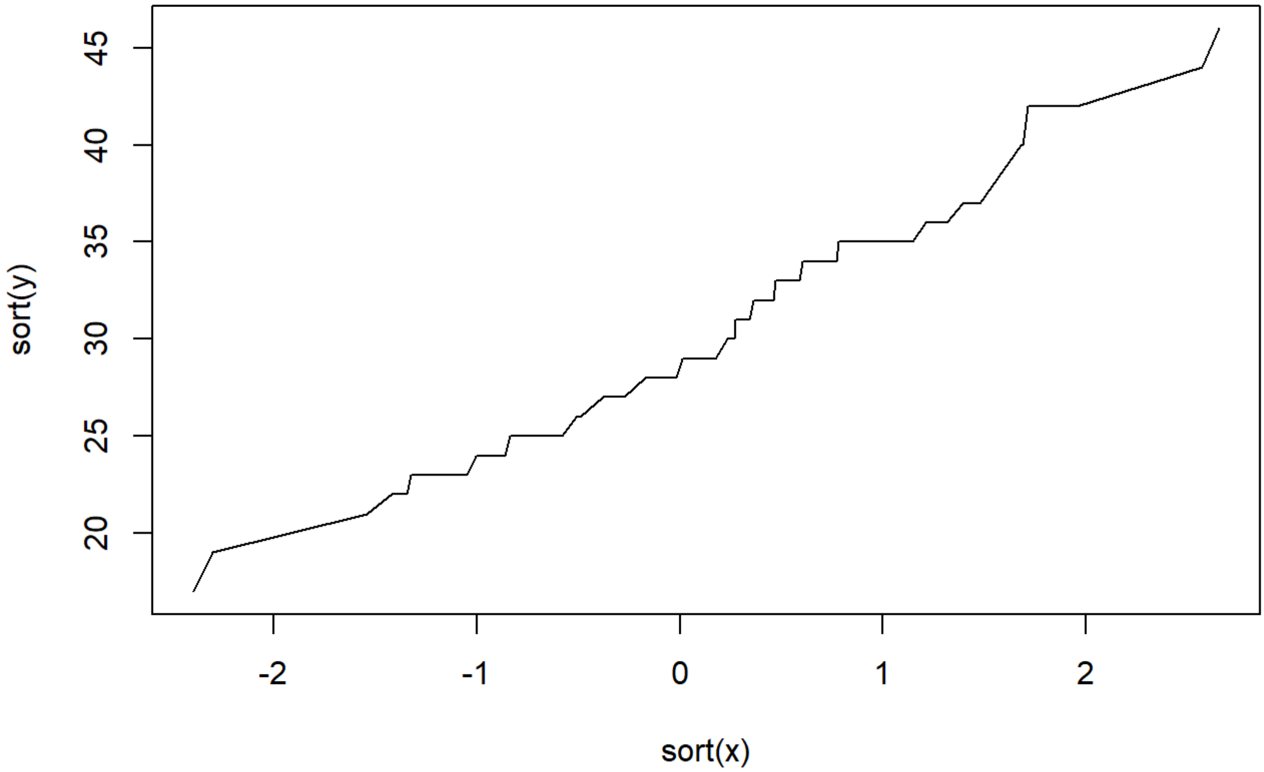
```
plot(x, y, main = "Poisson versus Normal")
```




```
plot(x, y, type="l") # plots a broken line (a dense tangle of line segments here)
```

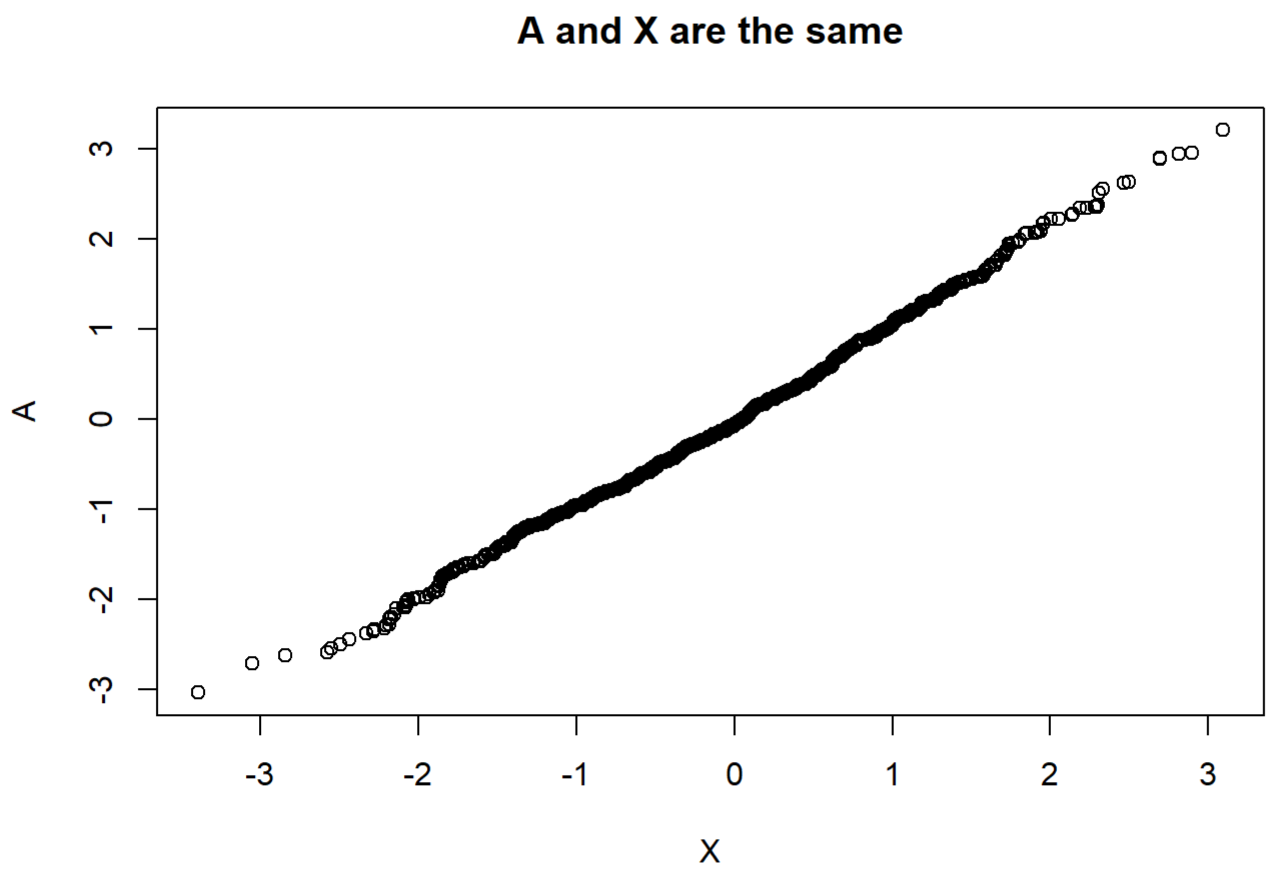


```
plot(sort(x), sort(y), type="l") # a plot of the sample "quantiles"
```

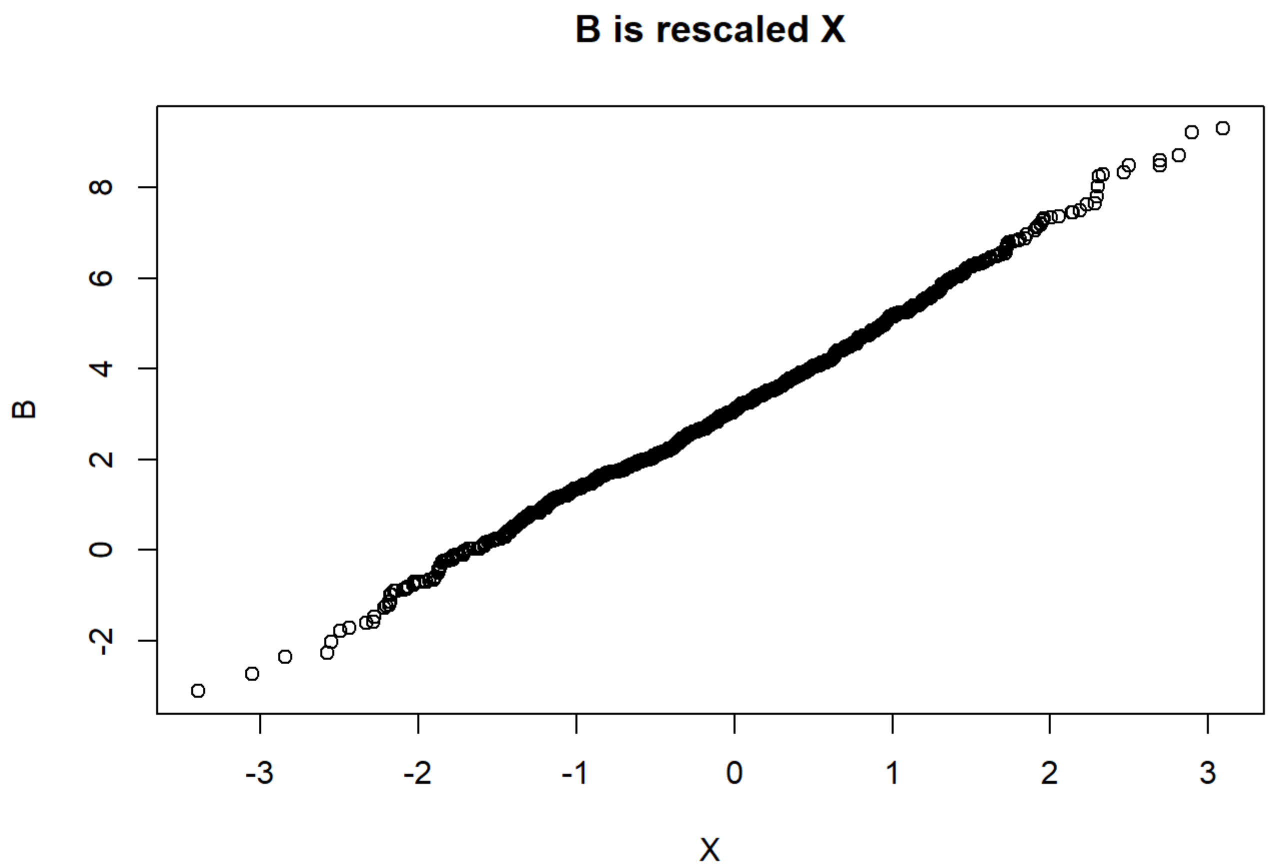



## QQ plots

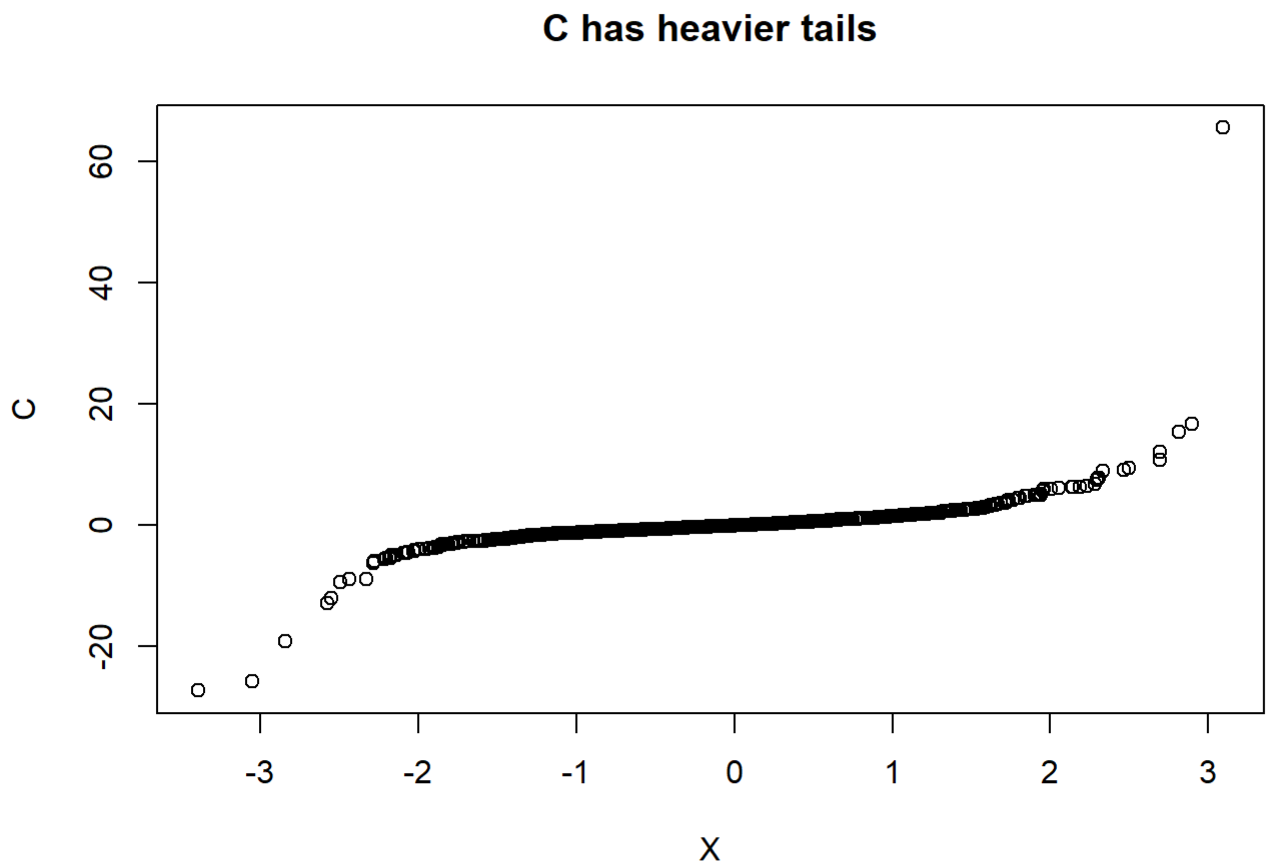
```
X <- rnorm(1000)
A <- rnorm(1000)
qqplot(X, A, main="A and X are the same")
```



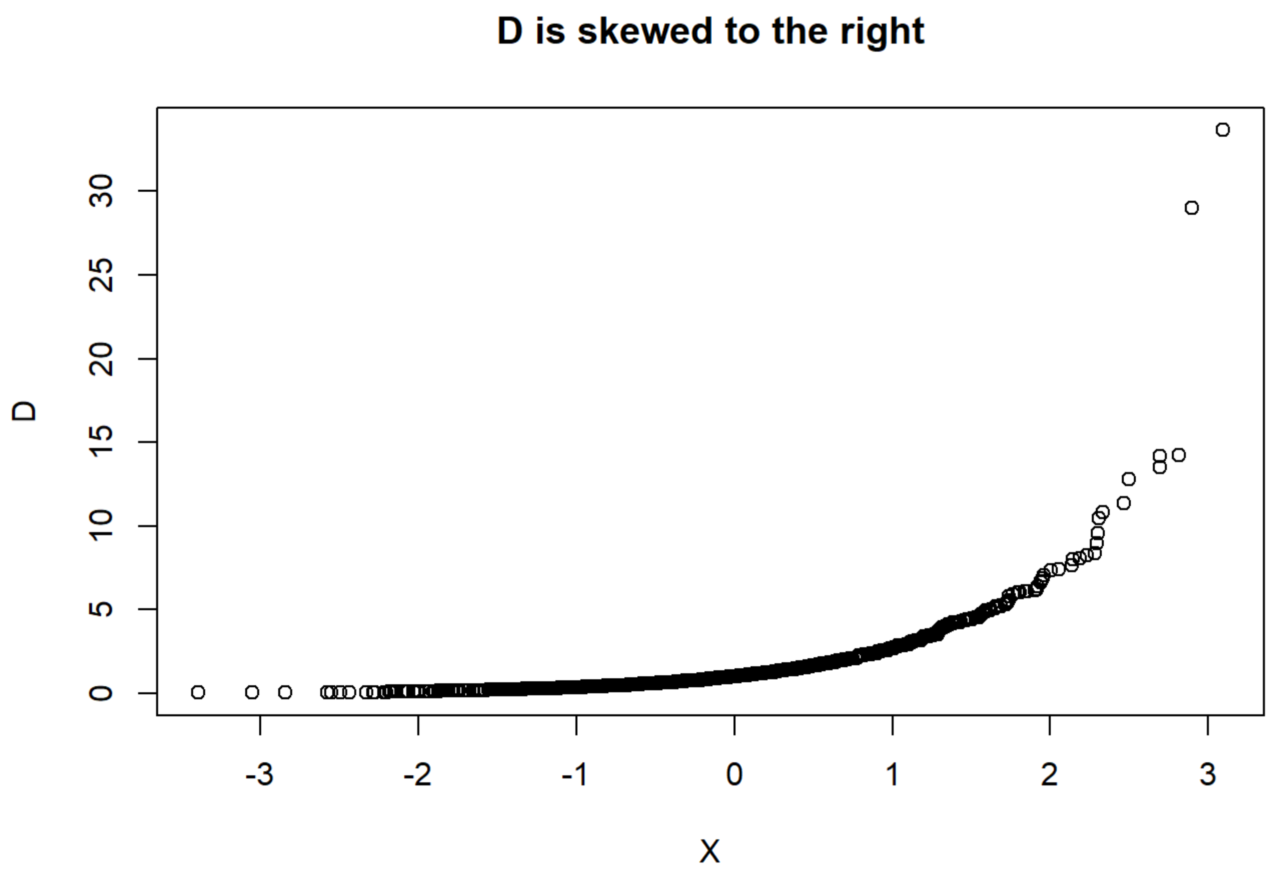

```
B <- rnorm(1000, mean=3, sd=2)
qqplot(X, B, main="B is rescaled X")
```



```
C <- rt(1000, df=2)
qqplot(X, C, main="C has heavier tails")
```




```
D <- exp(rnorm(1000))
qqplot(X, D, main="D is skewed to the right")
```




# 5. Simulations

## Uniform

> □ runif(n,min=a,max=b) default: a=0,b=1,n=隨機的n個數

```
runif(10) #隨機取十個數
```

```
##  [1] 0.3403657 0.2336291 0.5488534 0.1224356 0.9278615 0.6988193 0.8509670
##  [8] 0.2326084 0.2502140 0.5940666
```

## Bernoulli

only two possible outcomes.

```
set.seed(23207) # use this to obtain our output
guesses <- runif(20)
correct.answers <- (guesses < 0.2)
correct.answers
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
table(correct.answers)
```

```
## correct.answers
## FALSE  TRUE
##    14     6
```

# Binomial

```
dbinom(x = 4, size = 6, prob = 0.5)
```

```
## [1] 0.234375
```

```
pbinom(4,6,0.5)
```

```
## [1] 0.890625
```

```
qbinom(0.89,6,0.5)
```

```
## [1] 4
```

```
rbinom(24,15,0.1)
```

```
##  [1] 1 2 2 2 1 0 1 2 2 1 2 3 1 0 1 0 1 0 3 2 2 1 0 2
```

# Poission

```
dpois(x = 3, lambda = 0.5)
```

```
## [1] 0.01263606
```

```
ppois(3,0.5)
```

```
## [1] 0.9982484
```

```
qpois(0.03,0.5)
```

```
## [1] 0
```

```
rpois(10, 3.7)
```

```
##  [1] 5 5 2 6 0 1 4 2 2 5
```

# Exponential

```
pexp(1, rate = 3)
```

```
## [1] 0.9502129
```

```
rexp(10, rate = 3)
```

```
##  [1] 0.30811840 0.34188992 0.79840806 0.05053583 0.97660042 0.50273853
##  [7] 0.20992310 0.34392965 0.45242003 0.18683730
```

## Normal

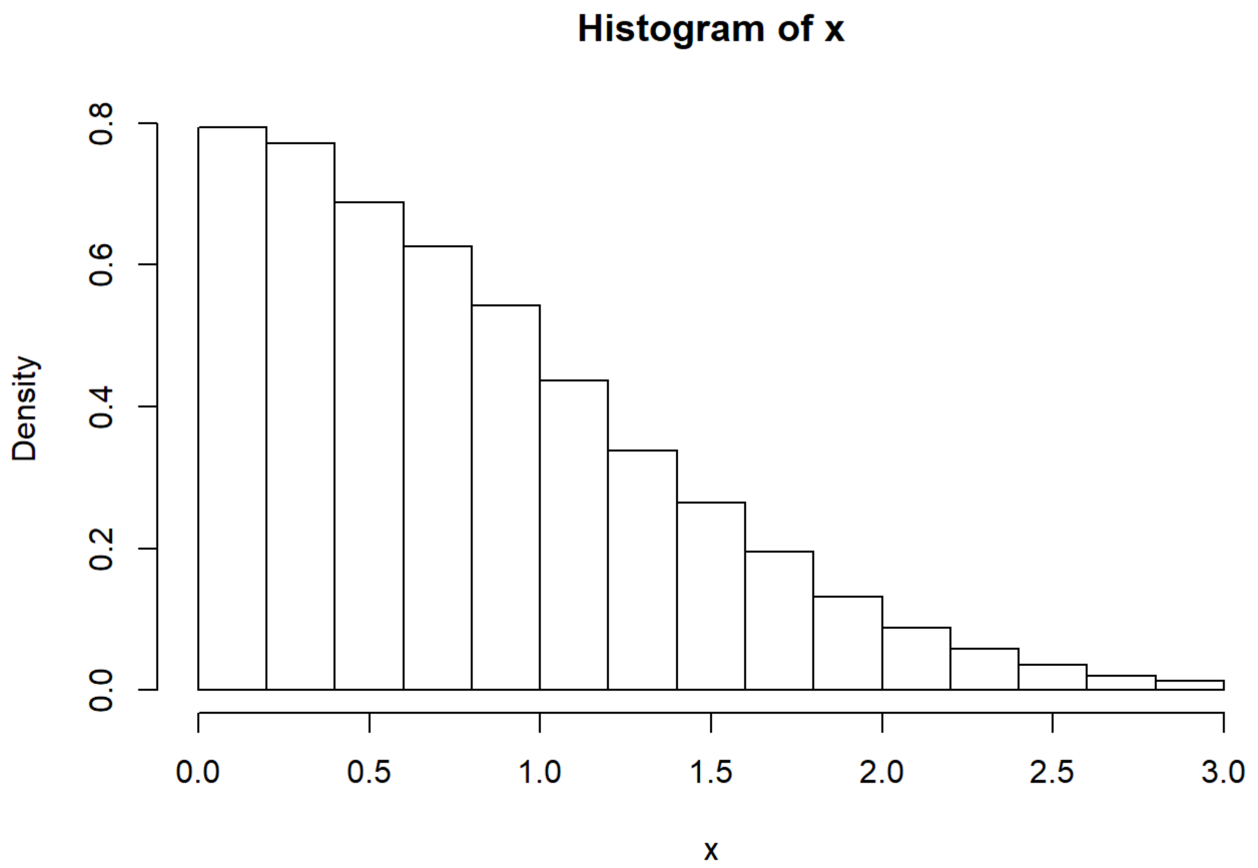
```
qnorm(0.95, mean = 2.7, sd = 3.3)
```

```
## [1] 8.128017
```

```
rnorm(10, -3, 0.5)
```

```
##  [1] -3.388241 -2.869481 -3.317109 -2.591064 -3.944068 -2.998641 -3.310168
##  [8] -3.156573 -2.497396 -3.018670
```

```
x<- rnorm(100000) # simulate from the standard normal
x <- x[(0 < x) & (x < 3)] # reject all x's outside (0,3)
hist(x, probability=TRUE) # show the simulated values
```



# 6. Linear Programming

$$minC = 5x_1 + 8x_2$$

$$x_1 + x_2 \geq 2$$

$$x_1 + 2x_2 \geq 3$$

$$x_1, x_2 \geq 0$$

```
eg.lp <- lp(objective.in=c(5, 8), const.mat=matrix(c(1, 1, 1, 2),  nrow=2), const.rhs=c(2, 3
), const.dir=c(">=", ">="))
eg.lp
```

```
## Success: the objective function is 13
```

```
x1x2=eg.lp$solution
```

(x1,x2)=(1, 1)

$$minC = 5x_1 + 8x_2 = 13$$