# Linear Algebra in R

ITEC 621 W1

# Outline

- Vectors
- Data Types
- Functions
- Matrices
- Linear Systems

# R is a calculator

```
7+12
```

```
[1] 19
```

```
c(1,2,3) + c(4,5,6)
```

```
[1] 5 7 9
```

```
1:3 + 4:6
```

```
[1] 5 7 9
```

```
# Try 1:10 + 1:3
list(1:10, 1:10 + 1:3)
```

```
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
 [1]  2  4  6  5  7  9  8 10 12 11
```

# Vectors

```
w = c(1,2,3,4,5)
x = 1:5
y = c('a','b','c','d')
z <- 11:20

w
```

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

```
w[4]
```

```
[1] 4
```

```
x[4]
```

```
[1] 4
```

```
y[4]
```

```
[1] "d"
```

```
z[4]
```

```
[1] 14
```

# Data Types

```
my_num = 3.6
class(my_num)
```

```
[1] "numeric"
```

```
my_string = 'R has functions, which take the form function_name(argument1, argument2, ...)'
class(my_string)
```

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

```
# Try: 3 + '1'

# Error in 3 + "1" : non-numeric argument to binary operator
```

# Functions

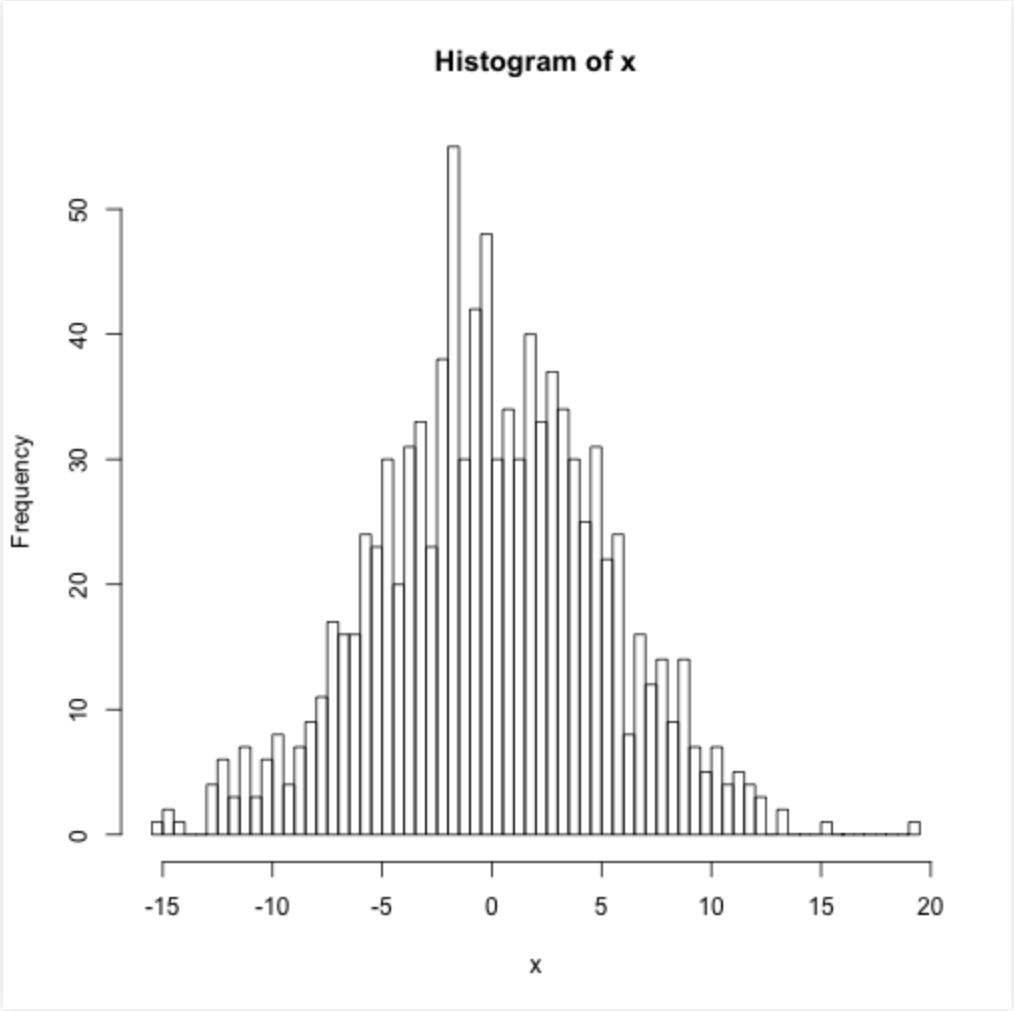
```
print(my_string)
```

```
[1] "R has functions, which take the form function_name(argument1,
argument2, ...)"
```

```
# Note: Many of these functions live inside downloadable libraries (see
next slide)
set.seed(1)
x = rnorm(1000, mean=0, sd=5)
# Or x = rnorm(1000, 0, 5)
# Or x = rnorm(1000, sd=5), but *not* rnorm(1000, 5)
x[1:10]
```

```
 [1] -3.1322691  0.9182166 -4.1781431  7.9764040  1.6475389 -4.1023419
 [7]  2.4371453  3.6916235  2.8789068 -1.5269419
```

```
hist(x, 50)
```




```
mean(x)
```

```
[1] -0.05824071
```

```
sd(x)
```

```
[1] 5.174579
```

# Libraries

```
library(ISLR)
library(MASS)
library(car)
library(boot)
library(leaps)
library(glmnet)

# Note: You can use install.packages('ISLR'), e.g.,
# to install library ISLR but PLEASE **NEVER**
# include this in your code. You should only run
# these commands once; running them inside your
# code takes pointless extra computations, wasting time.
```

# Matrices

```
X = rbind(c(3, 2, -1), c(2, -2, 4), c(-1, 0.5, -1), c(0, 2, -3))
X
```

```
     [,1] [,2] [,3]
[1,]    3  2.0   -1
[2,]    2 -2.0    4
[3,]   -1  0.5   -1
[4,]    0  2.0   -3
```

```
dim(X)
```

```
[1] 4 3
```

```
X[2,]
```

```
[1]  2 -2  4
```

```
X[,3]
```

```
[1] -1  4 -1 -3
```

# 3-D Linear System

```
# Generate some random data
nsamples = 100
set.seed(1)
x1 = rnorm(nsamples, mean=1, sd=1)
x2 = rnorm(nsamples, mean=5, sd=5)
epsilon = rnorm(nsamples, mean=0, sd=0.3)
ones = matrix(1,nsamples,1)

# General form for linear model:
# y = X * b + noise
X = cbind(ones, x1, x2)
X[1:3,]
```

```
     x1        x2
[1,] 1 0.3735462 1.8981666
[2,] 1 1.1836433 5.2105794
[3,] 1 0.1643714 0.4453918
```

```
# Here the true model for y is
# y = 1.6 + 3*x1 - 2*x2 + epsilon
b = c(1.6, 3, -2)
y = X %*% b + epsilon
y[1:3]
```

```
[1] -0.9528741 -4.7635668  1.6783072
```

```
# hist(y)
# pairs(cbind(x1, x2, y))
# cor(x1, y)
# cor(x2, y)
# cor(x1, x2)
```

# Estimating Linear Systems

```
# Try these
# b_est = solve(X, y)
# b_est = solve(X) %*% y
# Note this is basically trying to compute inverse(X) %*% y
# Why don't they work? What if nsamples = 3?

b_est = ginv(X) %*% y
b_est
```

```
          [,1]
[1,]  1.617313
[2,]  3.006333
[3,] -2.003208
```

```
# Now try playing with parameters like mean & sd
# for x1, x2, & epsilon
```