sas®

SQL
Database Language

STATA®

Programming and Statistics

R

Microsoft®
Visual Basic®

Install R

Goto www.r-project.org

Goto left panel and click on CRAN

Select (geographically) closest CRAN

Choose correct package for OS

Install

Also Available: Rstudio

A GUI for R

https://www.rstudio.com

Motivation

Most powerful statistical tools are programming-oriented

Applicable to all fields: math, natural sciences, computer science, economics, public policy, linguistics, psychology, sports, business, etc.

GREAT RESUME BOOSTER

Why Use R?

Open Source = available to everyone

IT’S FREE!

Cool built-in features

But still very customizable

Good stepping stone: easy to learn SAS, STATA, Python, and other scripting languages

Goals

Introduce R data structures

Using Scripts

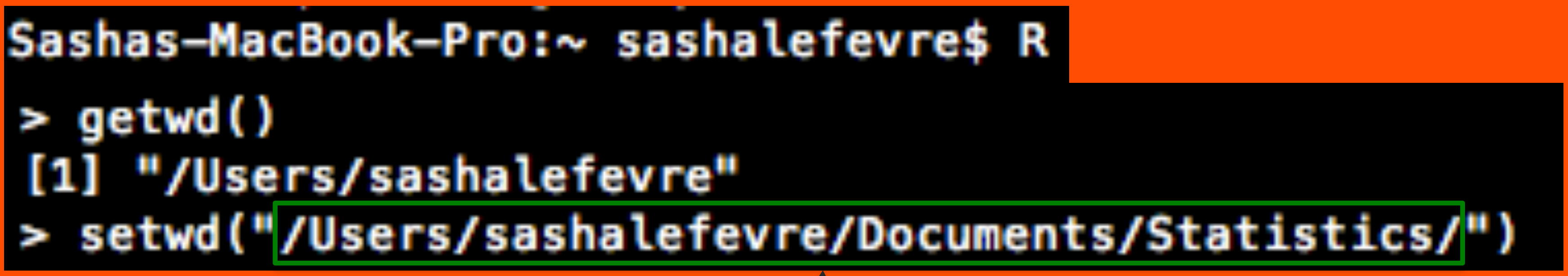
Use excel data in R

Coding in R

Getting Started:

<type R into Terminal>

Set working directory

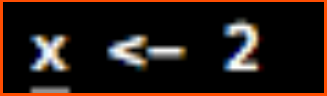


```
Sashas-MacBook-Pro:~ sashalefevre$ R
> getwd()
[1] "/Users/sashalefevre"
> setwd("/Users/sashalefevre/Documents/Statistics/")
```

Path to your folder goes here

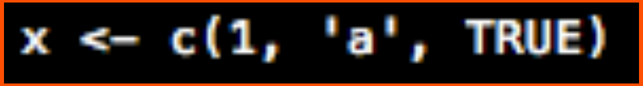
R vs C++

Variable Assignment



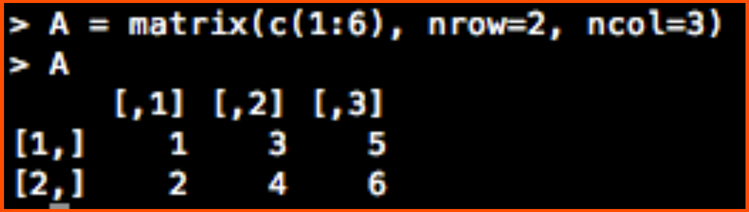
```
x <- 2
```

Vectors, sequences



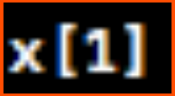
```
x <- c(1, 'a', TRUE)
```

Matrices[row,col]

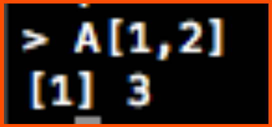


```
> A = matrix(c(1:6), nrow=2, ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

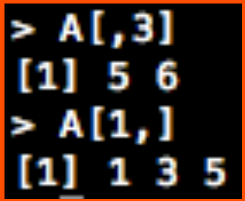
Element Access



```
x[1]
```



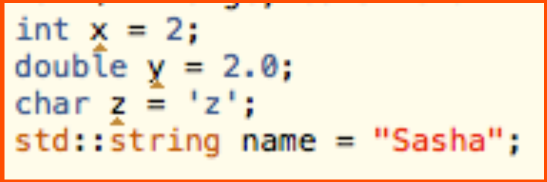
```
> A[1,2]
[1] 3
```



```
> A[,3]
[1] 5 6
> A[1,]
[1] 1 3 5
```

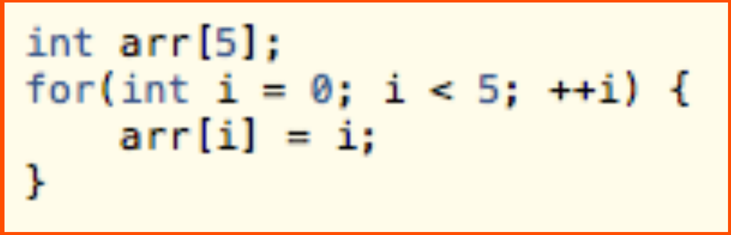
Indices start at 1!

Variable Assignment



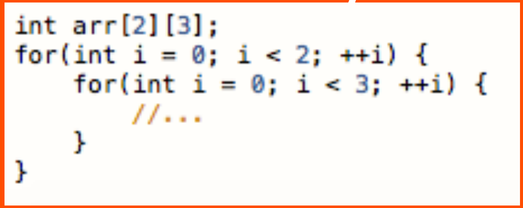
```
int x = 2;
double y = 2.0;
char z = 'z';
std::string name = "Sasha";
```

Arrays



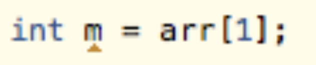
```
int arr[5];
for(int i = 0; i < 5; ++i) {
    arr[i] = i;
}
```

2D arrays



```
int arr[2][3];
for(int i = 0; i < 2; ++i) {
    for(int i = 0; i < 3; ++i) {
        //...
    }
}
```

Element Access



```
int m = arr[1];
```

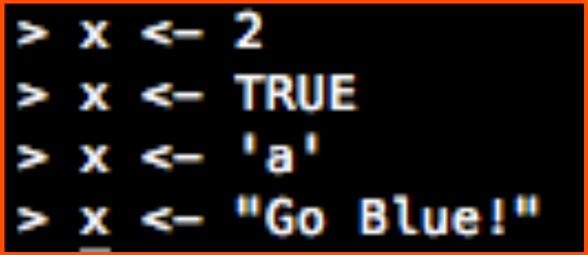
Indices start at 0!

Variable Assignment

R

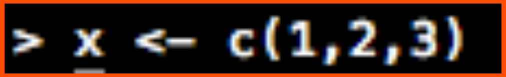
Do not have to specify type

Can overwrite values with other data types



```
> x <- 2
> x <- TRUE
> x <- 'a'
> x <- "Go Blue!"
```

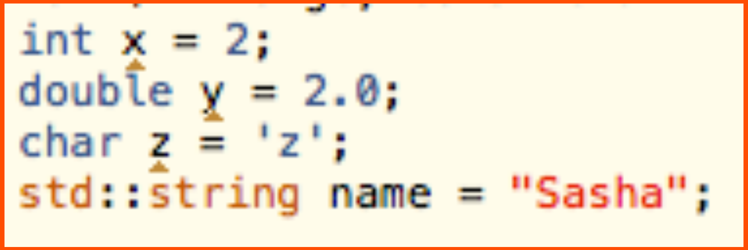
Can overwrite values with a data structure



```
> x <- c(1,2,3)
```

C++

Have to specify type



```
int x = 2;
double y = 2.0;
char z = 'z';
std::string name = "Sasha";
```

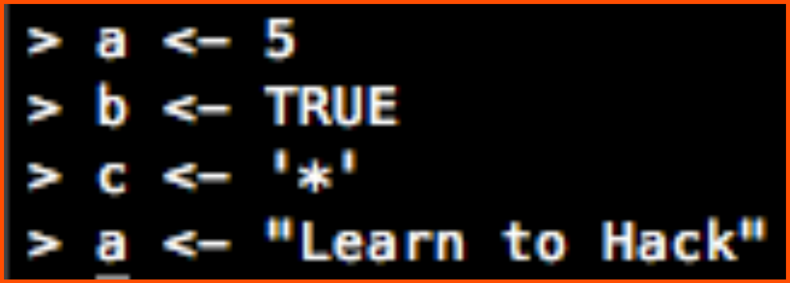
Cannot reuse variables in same scope!

Cannot assign a data structure to something that has been declared as basic data type

Class Problem #1

1. Declare a variable called ‘a’ and assign an integer to it
2. Declare a variable called ‘b’ and assign a boolean value to it
3. Declare a variable called ‘c’ and assign a character to it
4. Reassign ‘a’ to a string

Class Problem #1: Solution

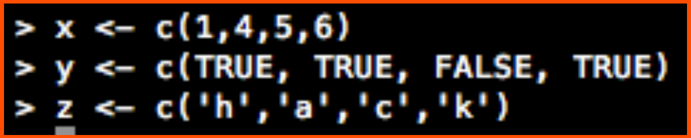


```
> a <- 5
> b <- TRUE
> c <- '*'
> a <- "Learn to Hack"
```

Vectors

R

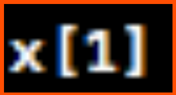
Vectors can be numerical, character, string, or logical



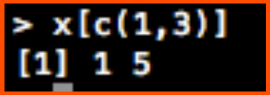
```
> x <- c(1,4,5,6)
> y <- c(TRUE, TRUE, FALSE, TRUE)
> z <- c('h','a','c','k')
```

Can have different data types in one vector

Element Access:



```
x[1]
```



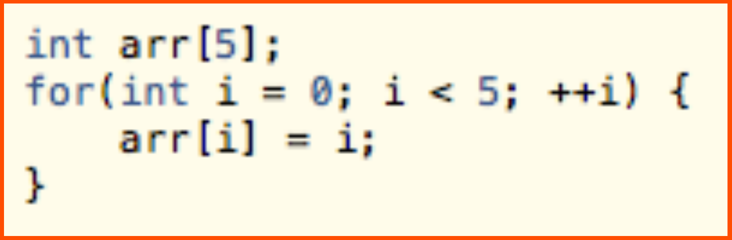
```
> x[c(1,3)]
[1] 1 5
```

Indices start at 1!

C++

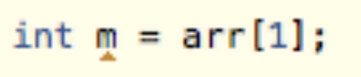
Arrays can be numerical, character, string, or logical

Can’t have different data types in one array



```
int arr[5];
for(int i = 0; i < 5; ++i) {
    arr[i] = i;
}
```

Element Access



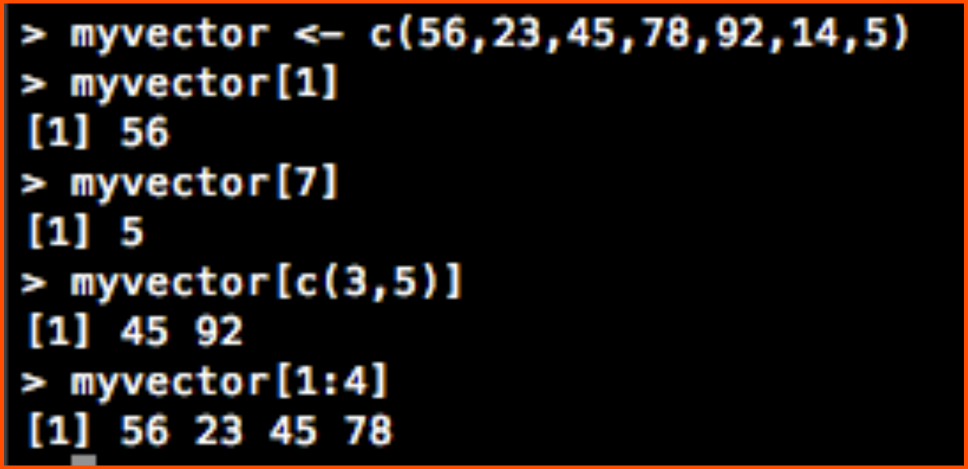
```
int m = arr[1];
```

Indices start at 0!

Class Problem #2

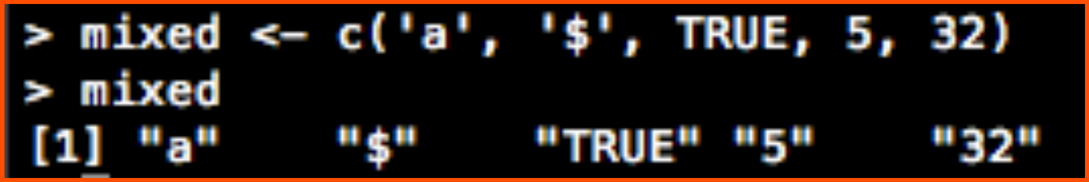
1. Declare a vector of 7 elements
2. Access the first element
3. Access the last element
4. Access the 3rd and 5th elements
5. Access elements 1-4 (inclusive)
6. Declare a vector of mixed values -> what type are the elements converted to?

Class Problem #2: Solution



```
> myvector <- c(56,23,45,78,92,14,5)
> myvector[1]
[1] 56
> myvector[7]
[1] 5
> myvector[c(3,5)]
[1] 45 92
> myvector[1:4]
[1] 56 23 45 78
```

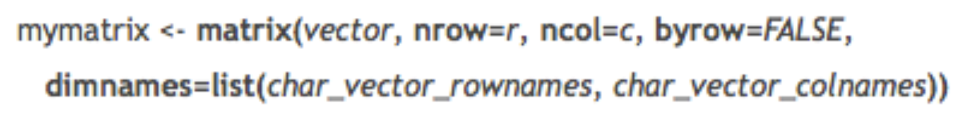
If vector has different data types, all elements get converted to strings



```
> mixed <- c('a', '$', TRUE, 5, 32)
> mixed
[1] "a"    "$"    "TRUE" "5"    "32"
```

Matrices

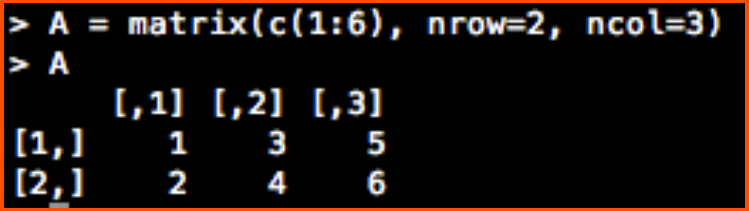
R



```
mymatrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE,
  dimnames=list(char_vector_rownames, char_vector_colnames))
```

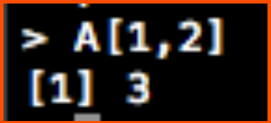
Matrices can be numerical, character, string, or logical

All elements must have same tye

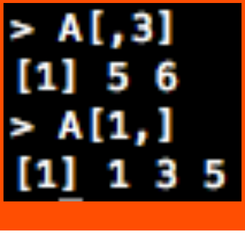


```
> A = matrix(c(1:6), nrow=2, ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Element Access:



```
> A[1,2]
[1] 3
```

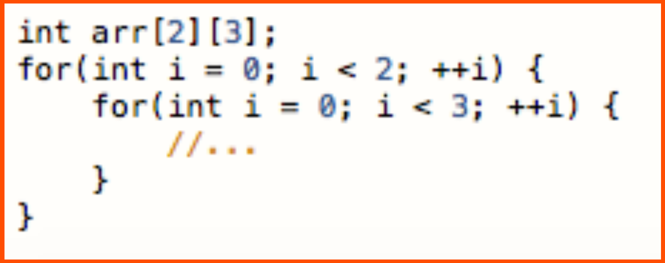


```
> A[,3]
[1] 5 6
> A[1,]
[1] 1 3 5
```

C++

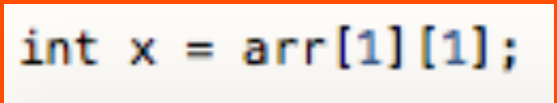
2D Arrays can be numerical, character, string, or logical

Can’t have different data types in one array



```
int arr[2][3];
for(int i = 0; i < 2; ++i) {
    for(int i = 0; i < 3; ++i) {
        //...
    }
}
```

Element Access



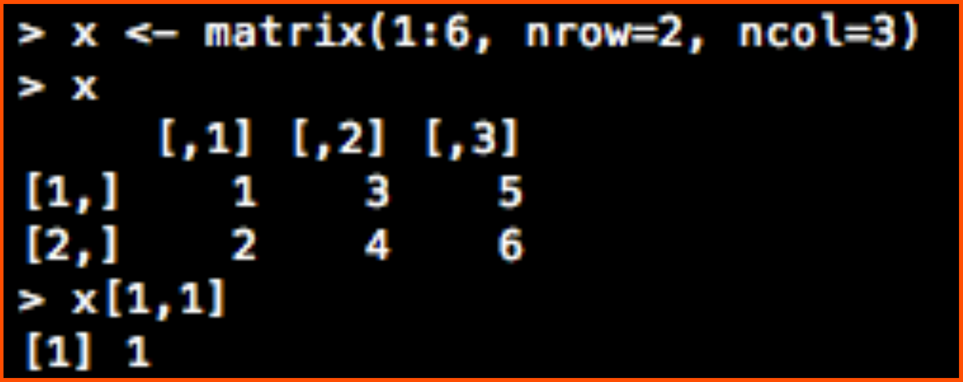
```
int x = arr[1][1];
```

Indices start at 0!

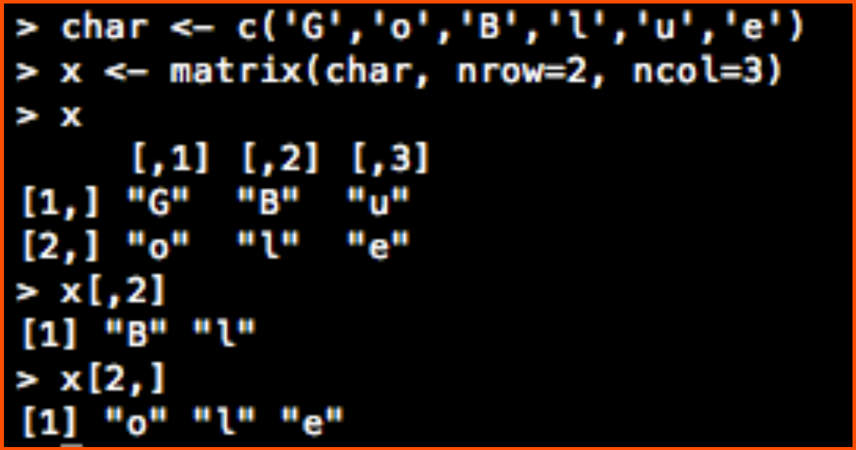
Class Problem #3

1. Declare one 2x3 matrix
2. Fill it with integers 1 through 6
3. Access element in row 1, column 1
4. Declare another 2x3 matrix
5. Fill it with a vector of characters
6. Access the entire 2nd column
7. Access the entire 2nd column

Class Problem #3: Solution



```
> x <- matrix(1:6, nrow=2, ncol=3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> x[1,1]
[1] 1
```



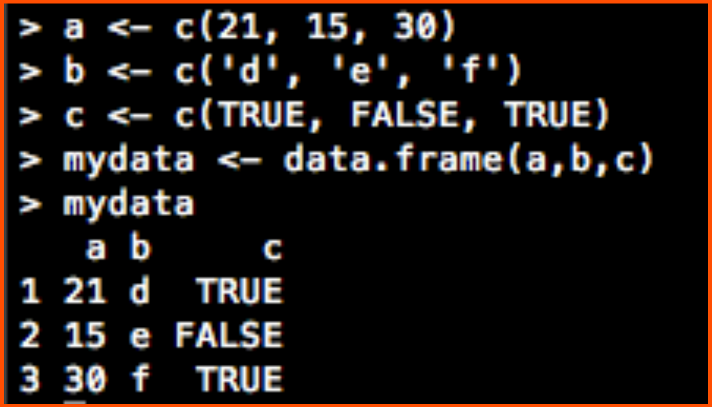
```
> char <- c('G','o','B','l','u','e')
> x <- matrix(char, nrow=2, ncol=3)
> x
     [,1] [,2] [,3]
[1,] "G"  "B"  "u" 
[2,] "o"  "l"  "e" 
> x[,2]
[1] "B" "l"
> x[2,]
[1] "o" "l" "e"
```

Data Frames

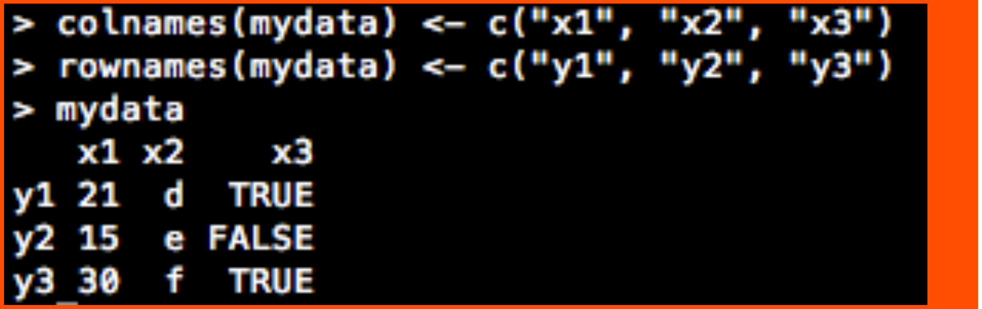
Data Frames are matrices, but columns can differ in data type

Make data frame by combining columns

Rename column/ row names with built-in functions colnames() and rownames()



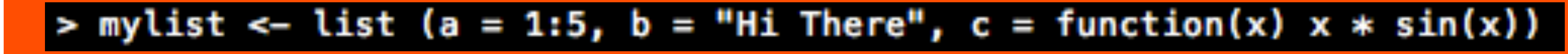
```
> a <- c(21, 15, 30)
> b <- c('d', 'e', 'f')
> c <- c(TRUE, FALSE, TRUE)
> mydata <- data.frame(a,b,c)
> mydata
   a b     c
1 21 d  TRUE
2 15 e FALSE
3 30 f  TRUE
```



```
> colnames(mydata) <- c("x1", "x2", "x3")
> rownames(mydata) <- c("y1", "y2", "y3")
> mydata
   x1 x2    x3
y1 21  d  TRUE
y2 15  e FALSE
y3 30  f  TRUE
```

Other Data Structures

Lists: structure that may contain objects of any other types



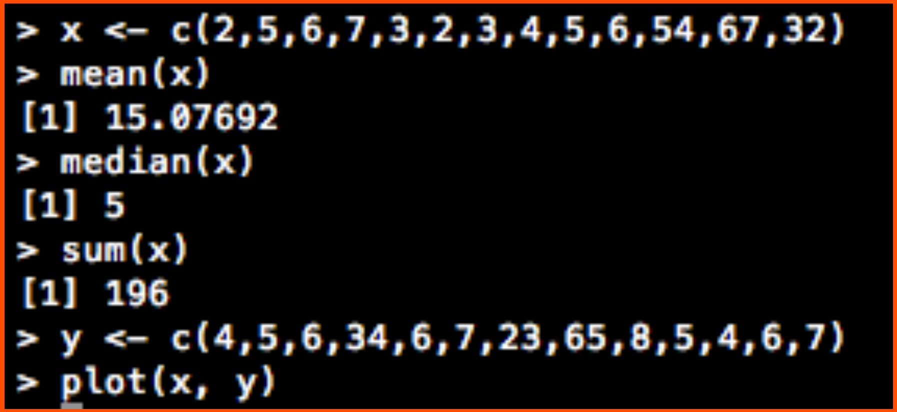
```
> mylist <- list (a = 1:5, b = "Hi There", c = function(x) x * sin(x))
```

Factors: vector of values corresponding to another vector of values (used for categorical variables)

Built-In Functions

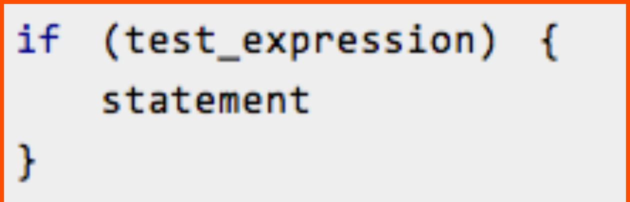
Use help() to get useful information about built-in functions (including parameters and examples of use)

Additional functions available through packages

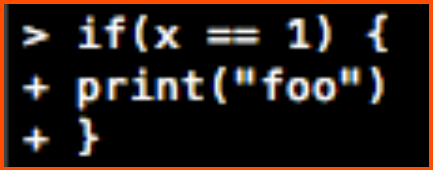


```
> x <- c(2,5,6,7,3,2,3,4,5,6,54,67,32)
> mean(x)
[1] 15.07692
> median(x)
[1] 5
> sum(x)
[1] 196
> y <- c(4,5,6,34,6,7,23,65,8,5,4,6,7)
> plot(x, y)
```

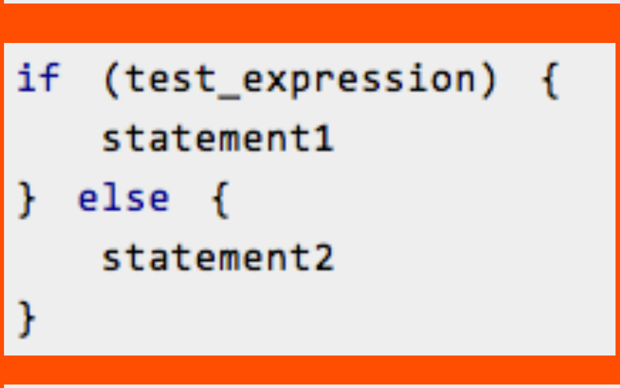
Function Syntax: If Statements



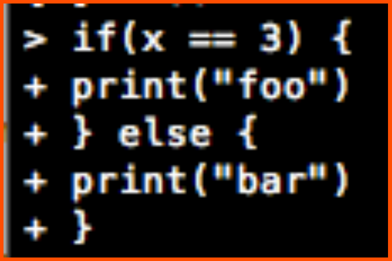
```
if (test_expression) {
    statement
}
```



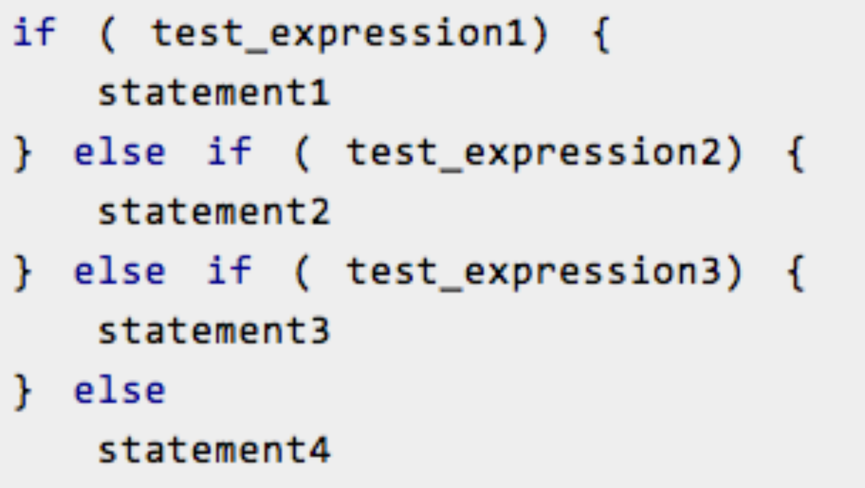
```
> if(x == 1) {
+ print("foo")
+ }
```



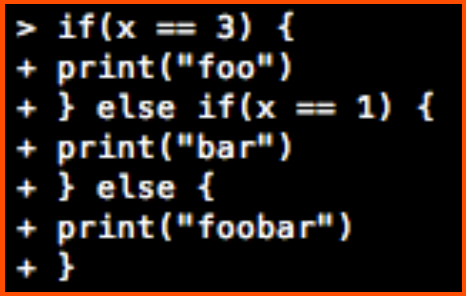
```
if (test_expression) {
    statement1
} else {
    statement2
}
```



```
> if(x == 3) {
+ print("foo")
+ } else {
+ print("bar")
+ }
```



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



```
> if(x == 3) {
+ print("foo")
+ } else if(x == 1) {
+ print("bar")
+ } else {
+ print("foobar")
+ }
```

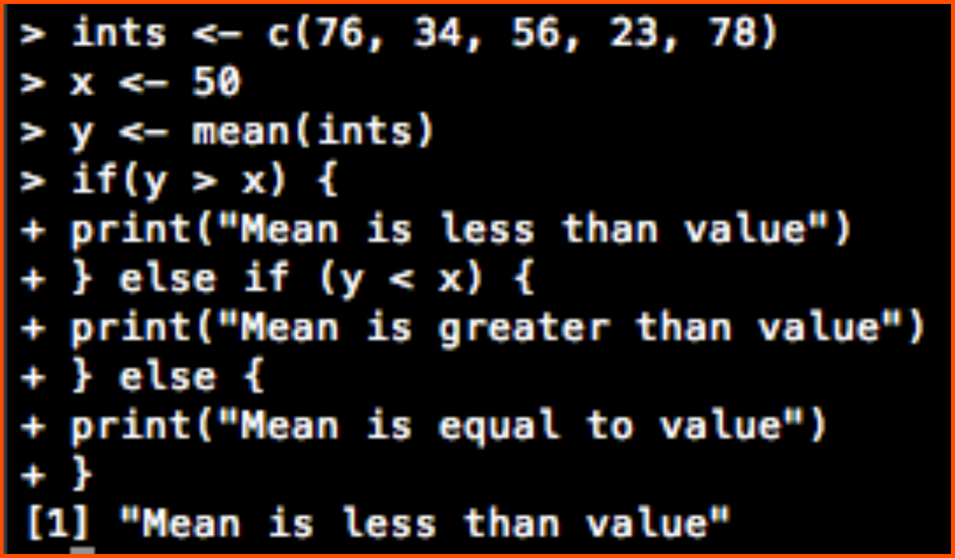
Class Problem #5

Create a vector of integer values

Create a conditional statement about the mean of these values: pick another value, and test to see if the mean if less than, greater than, or equal to that value.

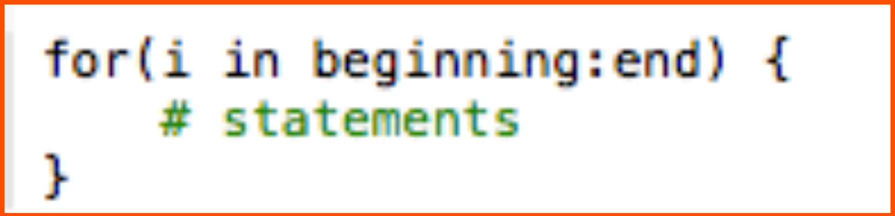
Print an informative statement for each case

Class Problem #5: Solution

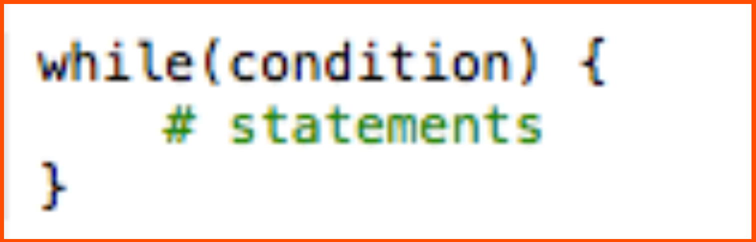


```
> ints <- c(76, 34, 56, 23, 78)
> x <- 50
> y <- mean(ints)
> if(y > x) {
+ print("Mean is less than value")
+ } else if (y < x) {
+ print("Mean is greater than value")
+ } else {
+ print("Mean is equal to value")
+ }
[1] "Mean is less than value"
```

Function Syntax: Loops

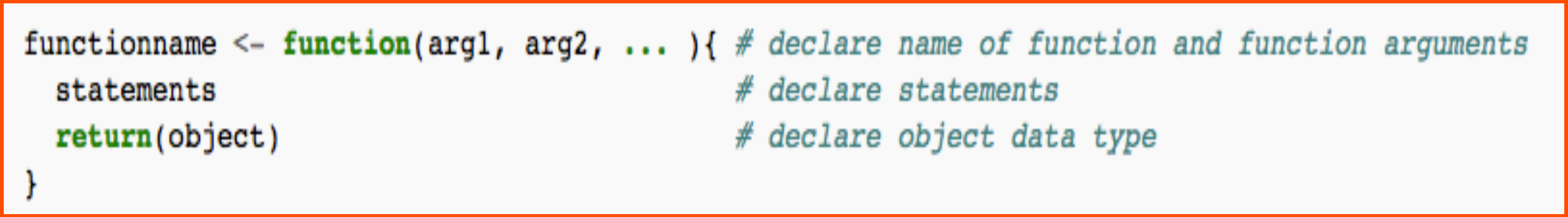


```
for(i in beginning:end) {
    # statements
}
```

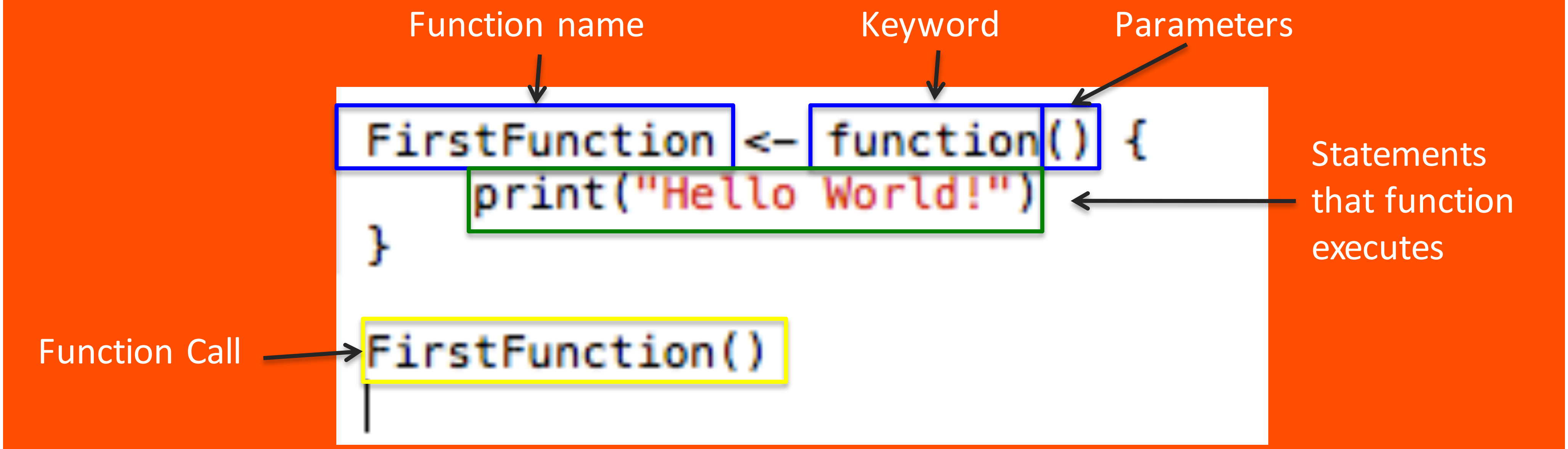


```
while(condition) {
    # statements
}
```

Create Own Function



```
functionname <- function(arg1, arg2, ... ){ # declare name of function and function arguments
  statements                                # declare statements
  return(object)                            # declare object data type
}
```



Function name

Keyword

Parameters

Statements that function executes

Function Call

```
 FirstFunction <- function() {
     print("Hello World!")
 }

FirstFunction()
```

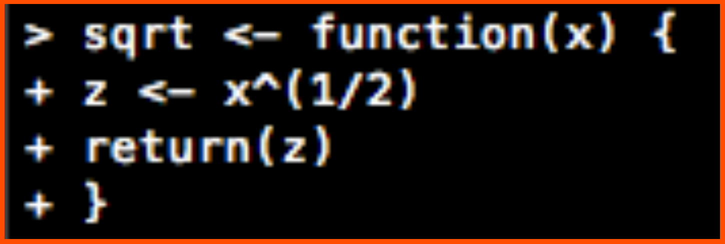
Class Problem

Write a basic function that calculates the square root of x

Bonus: Apply this function over a vector of integer values (HINT: look up built-in function **lapply()**)

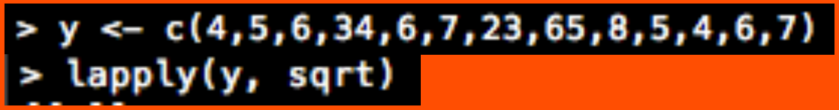
Class Problem: Solution

The sqrt() function:



```
> sqrt <- function(x) {
+ z <- x^(1/2)
+ return(z)
+ }
```

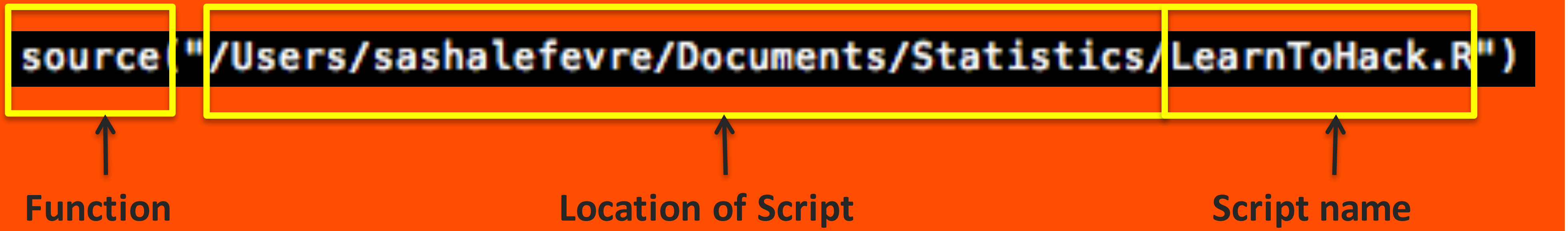
Applied over a vector:



```
> y <- c(4,5,6,34,6,7,23,65,8,5,4,6,7)
> lapply(y, sqrt)
```

Scripts

Instead of typing into command line, create text files with commands and functions that you want

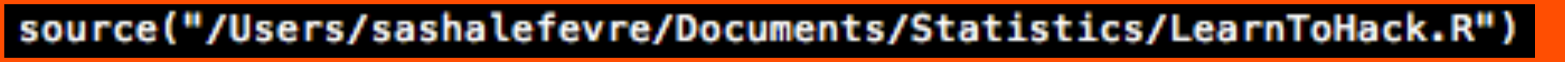


```
source("/Users/sashalefevre/Documents/Statistics/LearnToHack.R")
```

Function

Location of Script

Script name



```
source("/Users/sashalefevre/Documents/Statistics/LearnToHack.R")
```

Organizing Your Data

Need to have data organized before reading into R

<image>

No blank cells -> put NA

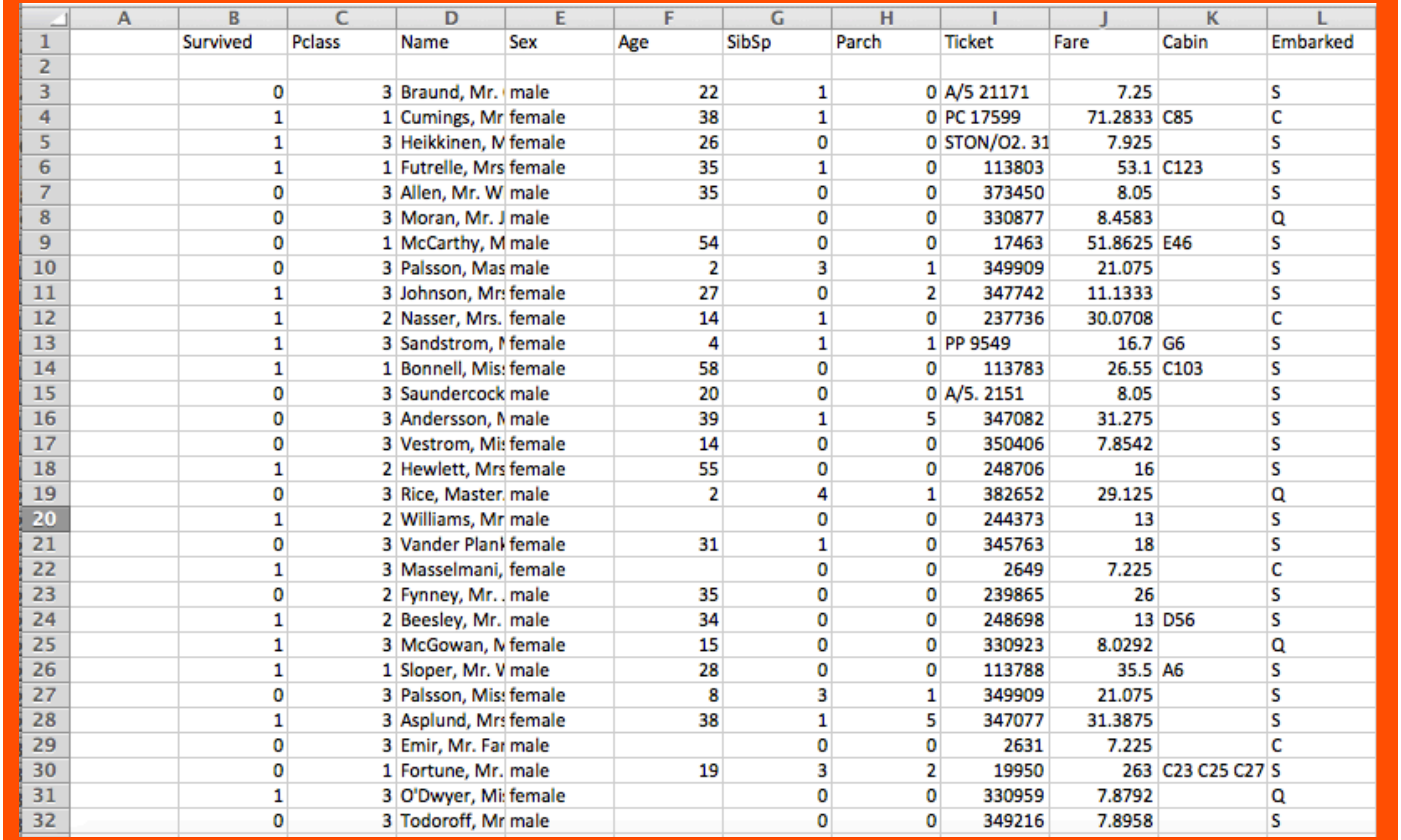
First Row reserved for headers

Variable names should be unique. Do not begin name with number and don’t use special symbols.

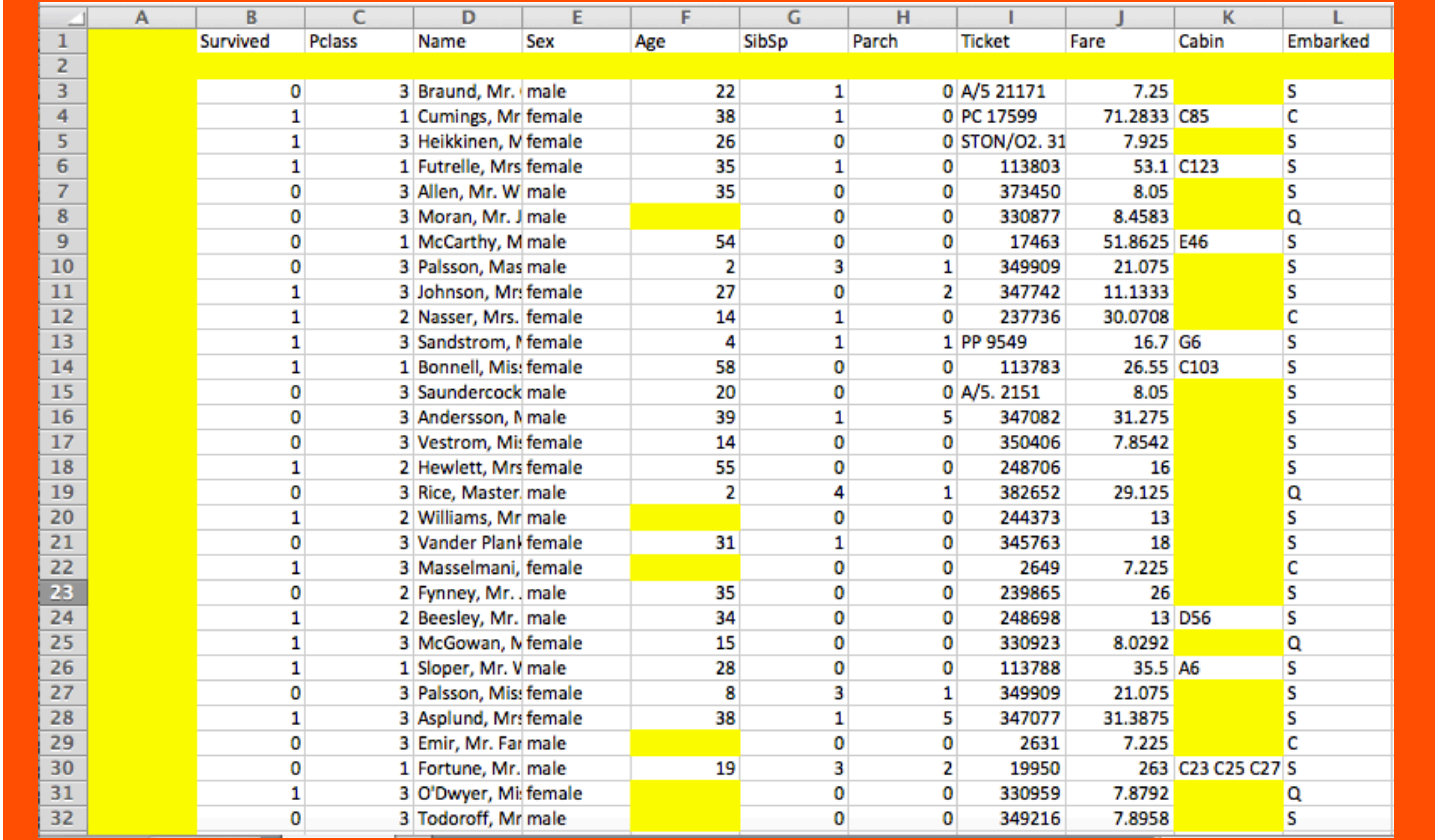
Each row should be single subject/ sample

Save as a .txt or .csv

Check out package Excel package for R: readxlsx



	A	B	C	D	E	F	G	H	I	J	K	L
1		Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2												
3		0	3	Braund, Mr. (	male	22	1	0	A/5 21171	7.25		S
4		1	1	Cumings, Mr	female	38	1	0	PC 17599	71.2833	C85	C
5		1	3	Heikkinen, M	female	26	0	0	STON/O2. 31	7.925		S
6		1	1	Futrelle, Mrs	female	35	1	0	113803	53.1	C123	S
7		0	3	Allen, Mr. W	male	35	0	0	373450	8.05		S
8		0	3	Moran, Mr. J	male		0	0	330877	8.4583		Q
9		0	1	McCarthy, M	male	54	0	0	17463	51.8625	E46	S
10		0	3	Palsson, Mas	male	2	3	1	349909	21.075		S
11		1	3	Johnson, Mrs	female	27	0	2	347742	11.1333		S
12		1	2	Nasser, Mrs.	female	14	1	0	237736	30.0708		C
13		1	3	Sandstrom, M	female	4	1	1	PP 9549	16.7	G6	S
14		1	1	Bonnell, Mis	female	58	0	0	113783	26.55	C103	S
15		0	3	Saundercock	male	20	0	0	A/5. 2151	8.05		S
16		0	3	Andersson, M	male	39	1	5	347082	31.275		S
17		0	3	Vestrom, Mis	female	14	0	0	350406	7.8542		S
18		1	2	Hewlett, Mrs	female	55	0	0	248706	16		S
19		0	3	Rice, Master	male	2	4	1	382652	29.125		Q
20		1	2	Williams, Mr	male		0	0	244373	13		S
21		0	3	Vander Plank	female	31	1	0	345763	18		S
22		1	3	Masselmani,	female		0	0	2649	7.225		C
23		0	2	Fynney, Mr. J	male	35	0	0	239865	26		S
24		1	2	Beesley, Mr.	male	34	0	0	248698	13	D56	S
25		1	3	McGowan, M	female	15	0	0	330923	8.0292		Q
26		1	1	Sloper, Mr. W	male	28	0	0	113788	35.5	A6	S
27		0	3	Palsson, Mis	female	8	3	1	349909	21.075		S
28		1	3	Asplund, Mrs	female	38	1	5	347077	31.3875		S
29		0	3	Emir, Mr. Far	male		0	0	2631	7.225		C
30		0	1	Fortune, Mr.	male	19	3	2	19950	263	C23 C25 C27	S
31		1	3	O'Dwyer, Mis	female		0	0	330959	7.8792		Q
32		0	3	Todoroff, Mr	male		0	0	349216	7.8958		S

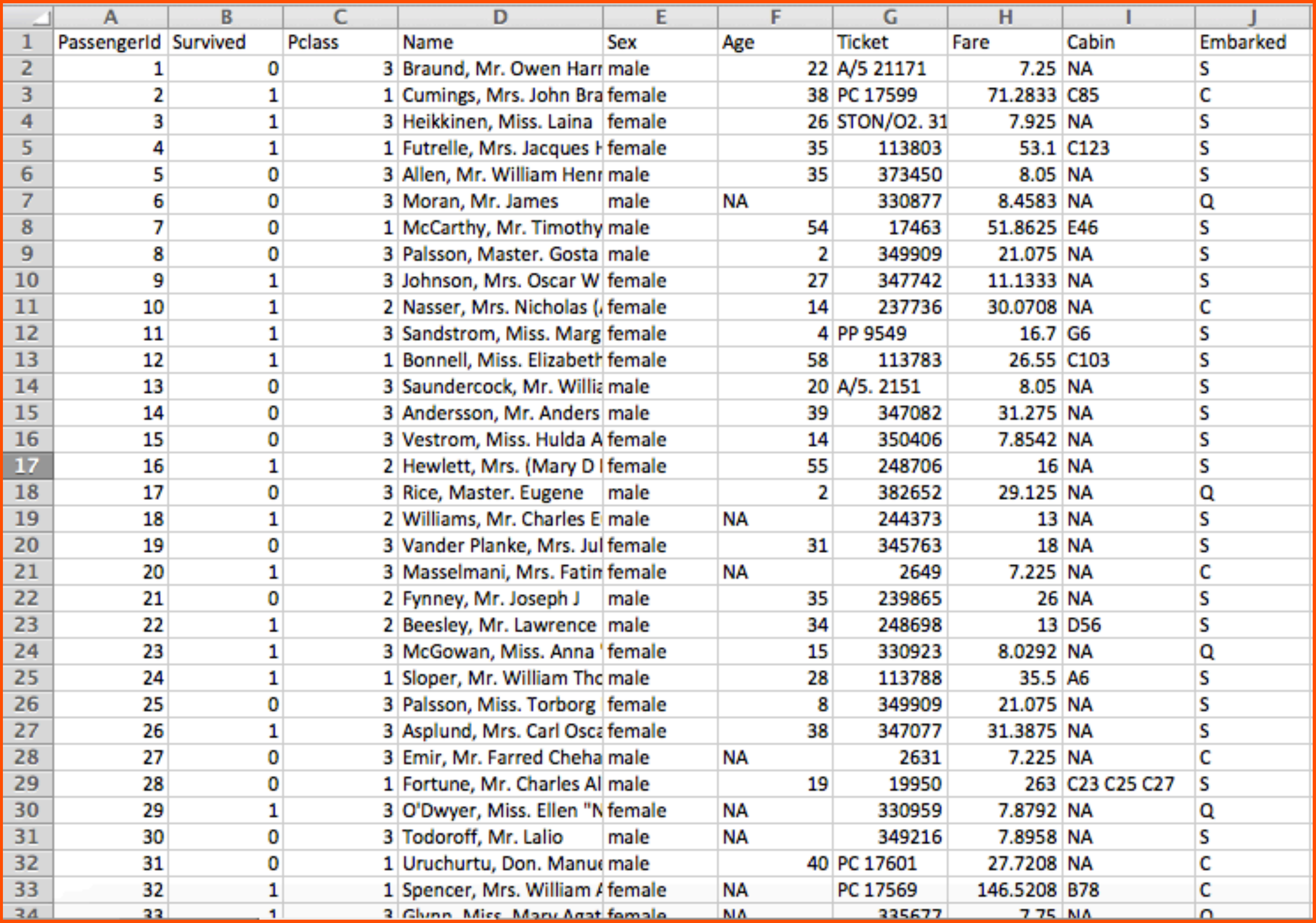


	A	B	C	D	E	F	G	H	I	J	K	L
1		Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2												
3		0	3	Braund, Mr.	male	22	1	0	A/5 21171	7.25		S
4		1	1	Cumings, Mr	female	38	1	0	PC 17599	71.2833	C85	C
5		1	3	Heikkinen, M	female	26	0	0	STON/O2. 31	7.925		S
6		1	1	Futrelle, Mrs	female	35	1	0	113803	53.1	C123	S
7		0	3	Allen, Mr. W	male	35	0	0	373450	8.05		S
8		0	3	Moran, Mr. J	male		0	0	330877	8.4583		Q
9		0	1	McCarthy, M	male	54	0	0	17463	51.8625	E46	S
10		0	3	Palsson, Mas	male	2	3	1	349909	21.075		S
11		1	3	Johnson, Mr	female	27	0	2	347742	11.1333		S
12		1	2	Nasser, Mrs.	female	14	1	0	237736	30.0708		C
13		1	3	Sandstrom, M	female	4	1	1	PP 9549	16.7	G6	S
14		1	1	Bonnell, Mis	female	58	0	0	113783	26.55	C103	S
15		0	3	Saundercock	male	20	0	0	A/5. 2151	8.05		S
16		0	3	Andersson, M	male	39	1	5	347082	31.275		S
17		0	3	Vestrom, Mi	female	14	0	0	350406	7.8542		S
18		1	2	Hewlett, Mrs	female	55	0	0	248706	16		S
19		0	3	Rice, Master	male	2	4	1	382652	29.125		Q
20		1	2	Williams, Mr	male		0	0	244373	13		S
21		0	3	Vander Plank	female	31	1	0	345763	18		S
22		1	3	Masselmani,	female		0	0	2649	7.225		C
23		0	2	Fynney, Mr. .	male	35	0	0	239865	26		S
24		1	2	Beesley, Mr.	male	34	0	0	248698	13	D56	S
25		1	3	McGowan, M	female	15	0	0	330923	8.0292		Q
26		1	1	Sloper, Mr. V	male	28	0	0	113788	35.5	A6	S
27		0	3	Palsson, Mis	female	8	3	1	349909	21.075		S
28		1	3	Asplund, Mrs	female	38	1	5	347077	31.3875		S
29		0	3	Emir, Mr. Fa	male		0	0	2631	7.225		C
30		0	1	Fortune, Mr.	male	19	3	2	19950	263	C23 C25 C27	S
31		1	3	O'Dwyer, Mi	female		0	0	330959	7.8792		Q
32		0	3	Todoroff, Mr	male		0	0	349216	7.8958		S

Variables have no numbers, spaces, or characters

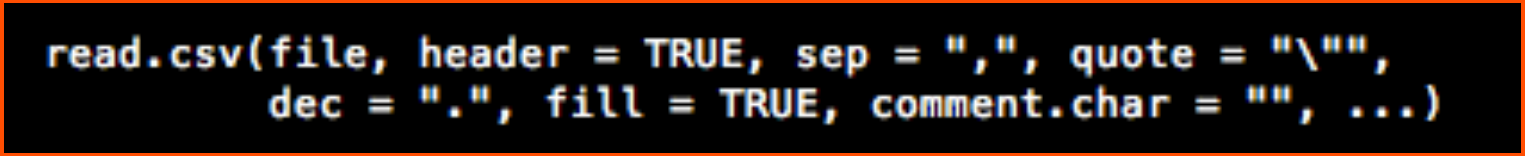
No blank cells -> write “NA”

Every row is a separate object/ sample!

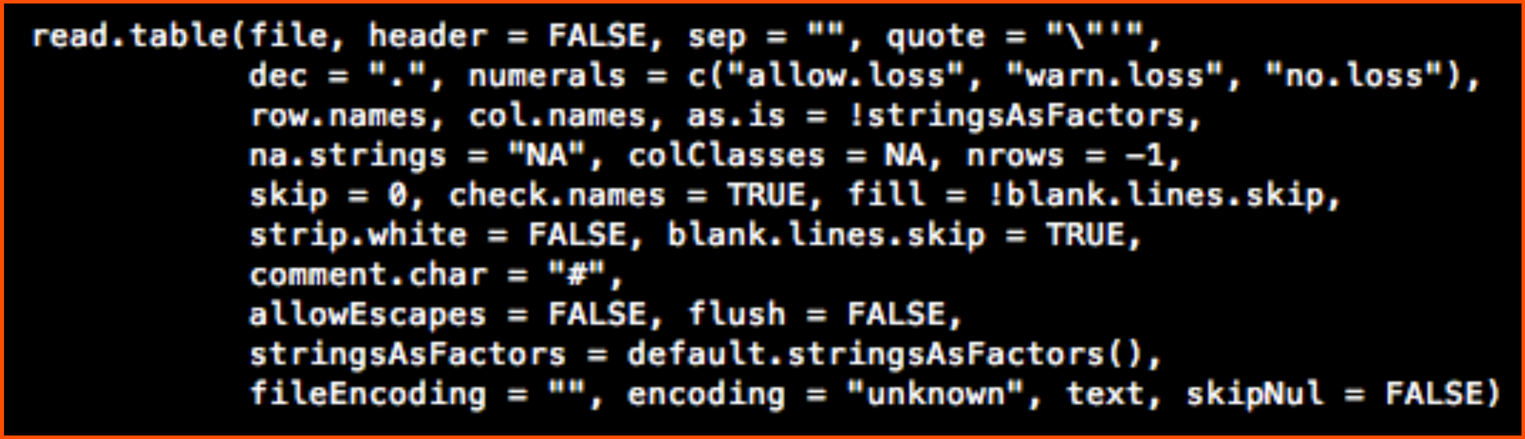


	A	B	C	D	E	F	G	H	I	J
1	PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harr	male	22	A/5 21171	7.25	NA	S
3	2	1	1	Cumings, Mrs. John Bra	female	38	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Laina	female	26	STON/O2. 31	7.925	NA	S
5	4	1	1	Futrelle, Mrs. Jacques H	female	35	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William Henr	male	35	373450	8.05	NA	S
7	6	0	3	Moran, Mr. James	male	NA	330877	8.4583	NA	Q
8	7	0	1	McCarthy, Mr. Timothy	male	54	17463	51.8625	E46	S
9	8	0	3	Palsson, Master. Gosta	male	2	349909	21.075	NA	S
10	9	1	3	Johnson, Mrs. Oscar W	female	27	347742	11.1333	NA	S
11	10	1	2	Nasser, Mrs. Nicholas (	female	14	237736	30.0708	NA	C
12	11	1	3	Sandstrom, Miss. Marg	female	4	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, Miss. Elizabeth	female	58	113783	26.55	C103	S
14	13	0	3	Saundercock, Mr. Willia	male	20	A/5. 2151	8.05	NA	S
15	14	0	3	Andersson, Mr. Anders	male	39	347082	31.275	NA	S
16	15	0	3	Vestrom, Miss. Hulda A	female	14	350406	7.8542	NA	S
17	16	1	2	Hewlett, Mrs. (Mary D I	female	55	248706	16	NA	S
18	17	0	3	Rice, Master. Eugene	male	2	382652	29.125	NA	Q
19	18	1	2	Williams, Mr. Charles E	male	NA	244373	13	NA	S
20	19	0	3	Vander Planke, Mrs. Jul	female	31	345763	18	NA	S
21	20	1	3	Masselmani, Mrs. Fatin	female	NA	2649	7.225	NA	C
22	21	0	2	Fynney, Mr. Joseph J	male	35	239865	26	NA	S
23	22	1	2	Beesley, Mr. Lawrence	male	34	248698	13	D56	S
24	23	1	3	McGowan, Miss. Anna '	female	15	330923	8.0292	NA	Q
25	24	1	1	Sloper, Mr. William Thc	male	28	113788	35.5	A6	S
26	25	0	3	Palsson, Miss. Torborg	female	8	349909	21.075	NA	S
27	26	1	3	Asplund, Mrs. Carl Osca	female	38	347077	31.3875	NA	S
28	27	0	3	Emir, Mr. Farred Cheha	male	NA	2631	7.225	NA	C
29	28	0	1	Fortune, Mr. Charles Al	male	19	19950	263	C23 C25 C27	S
30	29	1	3	O'Dwyer, Miss. Ellen "N	female	NA	330959	7.8792	NA	Q
31	30	0	3	Todoroff, Mr. Lalio	male	NA	349216	7.8958	NA	S
32	31	0	1	Uruchurtu, Don. Manue	male	40	PC 17601	27.7208	NA	C
33	32	1	1	Spencer, Mrs. William A	female	NA	PC 17569	146.5208	B78	C
34	33	1	3	Glynn, Miss. Mary Agat	female	NA	335677	7.75	NA	Q

Read In File



```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)
```



```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

Assign data to variable!

All scripts and examples can be found at github

Questions?

Additional Resources

R manuals: https://cran.r-project.org/manuals.html

Helpful packages: https://cran.r-project.org/web/views/

Statistics overview: http://www.biostathandbook.com