ESTP R - Basics

R is functional

```r
x <- 1
x # has value of 1
```

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

Always asign values to variables!

Don’t do this. . .

```r
read.csv("test.csv")
```

Do this

```r
test <- read.csv("test.csv")
```

Data types

Available data types: logical, numeric, integer, complex, character, and raw (not discussed)

```
x <- 1
class(x)
```

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

```
y <- TRUE
class(y)
```

```
## [1] "logical"
```

```
z <- "Some text"
class(z)
```

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

Variables are vectors

Each variable is also a *vector*, i.e. a sequence of data elements of the same class:

```
numbers <- 1:10
numbers
```

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

```
my_text <- c("Julia", "Python", "R")
my_text
```

```
## [1] "Julia"  "Python" "R"
```

Vectorized operations

Operations work on vectors

```
numbers
```

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

```
numbers + 10
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```
numbers^2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Operation work on vectors (2)

```
log(numbers)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944
##  [5] 1.6094379 1.7917595 1.9459101 2.0794415
##  [9] 2.1972246 2.3025851
```

```
mean(numbers)
```

```
## [1] 5.5
```

```
paste0("My language is: ", my_text)
```

```
## [1] "My language is: Julia"
## [2] "My language is: Python"
## [3] "My language is: R"
```

Retrieve/set items with index

```r
my_text <- c("Julia", "Python", "R")
my_text[3]
```

```
## [1] "R"
```

```r
my_text[3:1]
```

```
## [1] "R"      "Python" "Julia"
```

```r
my_text[2] <- "C++"
my_text
```

```
## [1] "Julia" "C++"   "R"
```

Vector generating functions

Combine: c

```
c(1, 5, 3, 8, 5, 3)
```

```
## [1] 1 5 3 8 5 3
```

```
# Also works for vectors as input
x <- c(1, 5)
y <- c(8, 5, 3)
c(x, y)
```

```
## [1] 1 5 8 5 3
```

Repeat: `rep`

```r
# repeat 2, 5 times
rep(2, 5)
```

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

```r
# repeat vector (1,3) 5 times
rep(c(1, 3), 5)
```

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

```r
# repeat vector (1,3) until length output is 5
rep(c(1, 3), length.out = 5)
```

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

Sequence generation: seq

```r
# numbers 2 to (and including) 5
seq(2, 5)
```

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

```r
# 1 to 10 step size 2
seq(1, 10, by = 2)
```

```
## [1] 1 3 5 7 9
```

```r
# 1 to 10 where output consists of 20 numbers
seq(1, 10, length.out = 20)
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053
##  [5]  2.894737  3.368421  3.842105  4.315789
##  [9]  4.789474  5.263158  5.736842  6.210526
## [13]  6.684211  7.157895  7.631579  8.105263
## [17]  8.578947  9.052632  9.526316 10.000000
```

Comparison operators

Expression	`TRUE` when
`x == y`	`x` equal to `y`
`x <= y`	`x` less than or equal to `y`
`x < y`	`x` less than `y`
`x > y`	`x` greater than `y`
`x >= y`	`x` greater than or equal to `y`
`x != y`	`x` not equal `y`
`x %in% y`	`x` is element of `y`

Example: %in%

```
x <- c("Jolien", "Edwin", "John")
y <- c("Jolien", "Richard")
x %in% y
```

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

Logical/binary operators

Operator	Means
`&`	AND
`|`	OR (en/of)
`!`	NOT
`all(x)`	all `x` equal to `TRUE`?
`any(x)`	at least one `x` equal to `TRUE`?

Data frames

A `data.frame` is a tabular format. Technically, it is a list of vectors of the same length

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.
##  $ Species     : Factor w/ 3 levels "setosa","versicolor
```

Data frames summary

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length
##  Min.   :4.300   Min.   :2.000   Min.   :1.000
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
##  Median :5.800   Median :3.000   Median :4.350
##  Mean   :5.843   Mean   :3.057   Mean   :3.758
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
##  Max.   :7.900   Max.   :4.400   Max.   :6.900
##   Petal.Width          Species
##  Min.   :0.100   setosa    :50
##  1st Qu.:0.300   versicolor:50
##  Median :1.300   virginica :50
##  Mean   :1.199
##  3rd Qu.:1.800
##  Max.   :2.500
```

Handy functions

Function	**description**
`summary`	statistical summary
`str`	technical summary
`colMeans`, `rowMeans`	mean per column/row
`colSums`, `rowSums`	sum per column/row
`names`	column names
`ncol nrow`	number of columns, rows
`dim`	vector with `nrow, ncol`

Selecting data

Retrieve colum with $

```
mean(iris$Sepal.Length)
```

```
## [1] 5.843333
```

```
# or
# number of rows
nrow(iris)
```

```
## [1] 150
```

```
colnames(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length"
## [4] "Petal.Width"  "Species"
```

Retrieve rows with index

```r
# first row (before comma)
iris[1,]
```

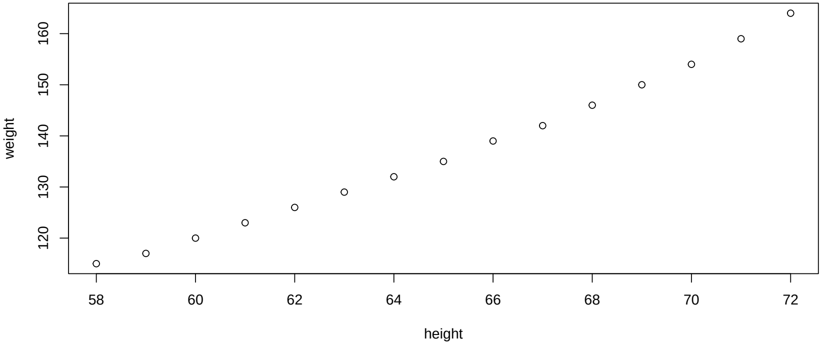
```
##   Sepal.Length Sepal.Width Petal.Length
## 1          5.1         3.5          1.4
##   Petal.Width Species
## 1         0.2  setosa
```

```r
# row 2, 6 and 3 and column 1 and 2
iris[c(2,6,3),1:2]
```

```
##   Sepal.Length Sepal.Width
## 2          4.9         3.0
## 6          5.4         3.9
## 3          4.7         3.2
```

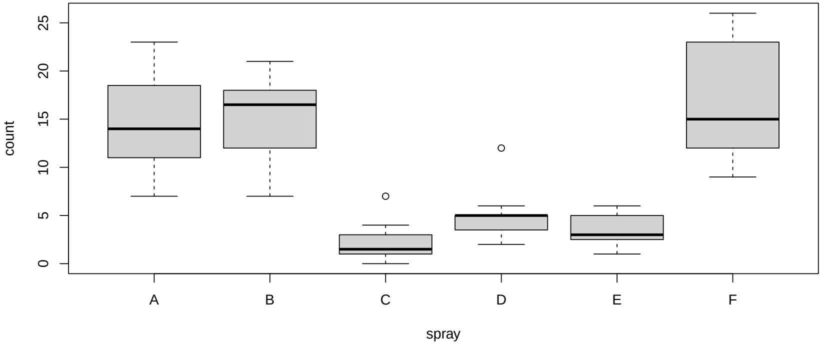
Plotting (1)

```r
plot(weight ~ height, data=women)
```



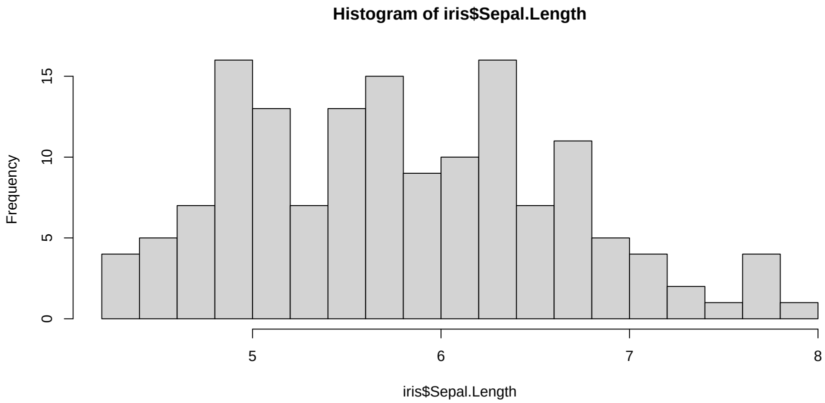
Plotting (2)

```
plot(count ~ spray, data=InsectSprays)
```



Plotting (3)

```
hist(iris$Sepal.Length, breaks=20)
```



Workspace

List variables

```
ls()
```

```
## [1] "my_text" "numbers" "x"       "y"
## [5] "z"
```

Remove variable(s)

```
rm("x")
ls()
```

```
## [1] "my_text" "numbers" "y"       "z"
```

RStudio Shortcuts

- **CTRL + R** or **CTRL + Enter** Run code
- **F1** (cursor at a function name) Help
- **F2** (cursor at a function name) Go to source