Introduction to Data Science Using R Part I

Justin Post

August 7-8, 2017

Course Schedule

Daily agenda:

- 10-11:10 Session
- 10-minute break
- 11:20-12:30 Session
- 12:30-1:45 Lunch
- 1:45-2:55 Session
- 10-minute break
- 3:05-4:15 Session

What do we want to be able to do?

- Read in data
- Manipulate data
- Plot data
- Summarize data
- Analyze data

Where do we start?

Day 1

- Install R/R studio
- R Studio Interface
- Classes and Objects
- Attributes and Basic Data Object Manipulation
- Reading in Data/Writing Out Data
- Logical Statements and Subsetting/Manipulating Data

Day 2

- Logical Statements and Subsetting/Manipulating Data?
- Numerical and Graphical Summaries
- Basic Analyses

Why learn R?

- It's free, open source, available on all major platforms.
- Tons of packages for modeling, visualization, data manipulation, etc.
- Access to the newest methods.
- Great community support (stackoverflow, R-help mailing list, etc.)
- Can *easily* create pdfs, slides, reports, html files, and interactive apps.

Drawbacks of Using R

- Slow generally (although ways to speed it up)
- Code style differs greatly!
- New code not necessarily verified
- Often many ways to do the same thing

Installing R

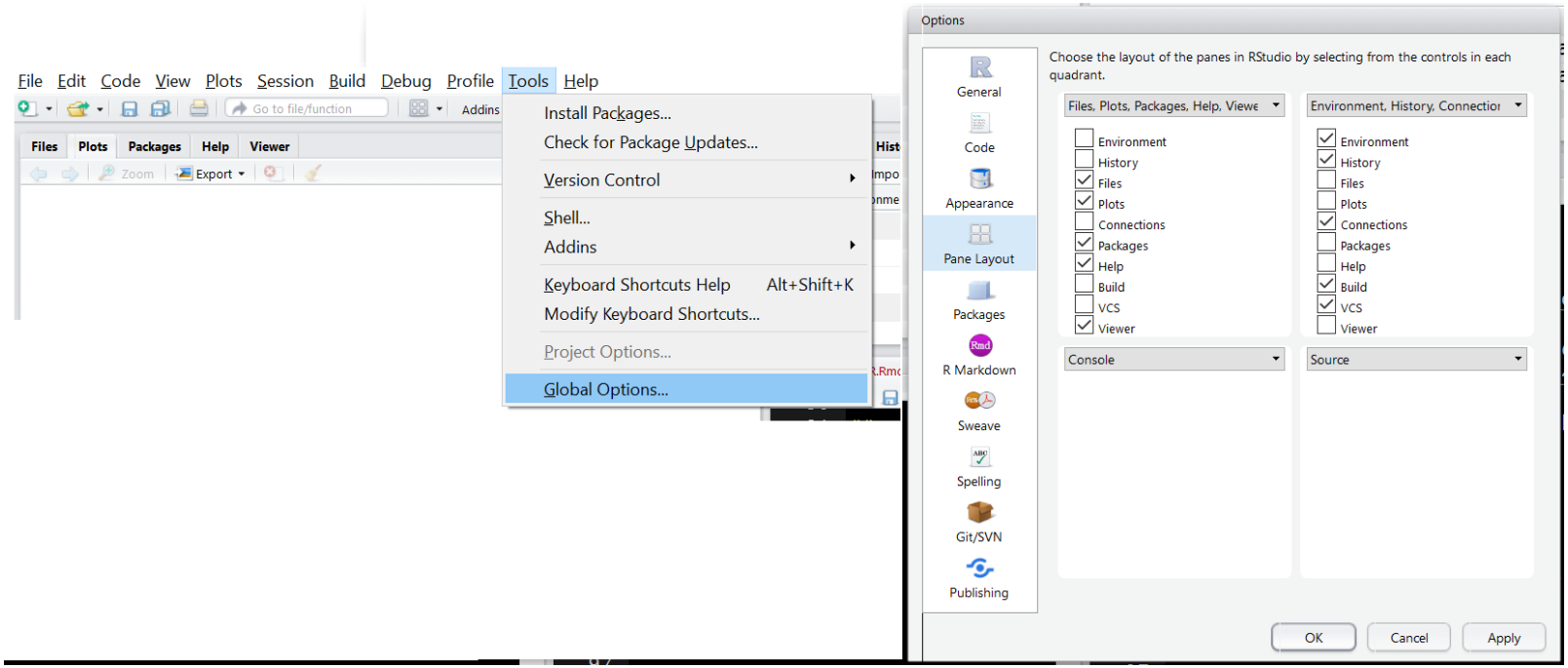
- Check out the course website
- Info on installing R and R studio available here
- Let's take a few minutes and make sure everyone has these installed and working properly!

R Studio Interface

- Four main 'areas' we'll use
  - Scripting and Viewing Area
  - Workspace/History
  - Plots/Help
  - Console

R Studio Interface

To rearrange panes



- Global options –> Appearance allows font/background changes

Basic Use of R

- You can type directly into the console to evaluate code
- R is the fanciest calculator you could ever want!

```
#simple math operations (# is a comment, not evaluated)
3 + 7
```

```
## [1] 10
```

```
10 * exp(3)
```

```
## [1] 200.8554
```

```
log(pi^2) #log is natural log by default
```

```
## [1] 2.28946
```

Basic Use or R

- Usually want to keep code for later use
- Write code in a 'script'
- Save code script
- Send lines of code to console via:
  - "Run" button (runs current line)
  - CTRL+r (PC) or Command+Enter (MAC)
  - Highlight section and do above

Objects and Common Classes

- Often want to save result for later use
- Can store output in an R 'object'

```
#save for later
avg <- (5 + 7 + 6) / 3
#call avg object
avg
```

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

```
#strings can be saved as well
words <- "Hello there!"
words
```

```
## [1] "Hello there!"
```

Objects and Common Classes

- Five major data structures used

1. Atomic Vector (1d)
2. Matrix (2d)
3. Array (nd) (we'll skip)
4. Data Frame (2d)
5. List (1d)

Objects and Common Classes

1. Atomic Vector (a set of elements with an ordering)

- `c()` function "combines" values together

```
#vectors (1 dimensional) objects
#all elements of the same 'type'
x <- c(1, 3, 10, -20, sqrt(2))
y <- c("cat", "dog", "bird", "floor")
x
```

```
## [1]   1.000000   3.000000  10.000000 -20.000000   1.414214
```

```
y
```

```
## [1] "cat"   "dog"   "bird"  "floor"
```

Objects and Common Classes

- Many ways to populate a numeric vector

```
1:20 / 20
```

```
##  [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
## [15] 0.75 0.80 0.85 0.90 0.95 1.00
```

```
seq(from = 1, to = 10, by = 2)
```

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

```
runif(4, min = 0, max = 1)
```

```
## [1] 0.2764794 0.2943467 0.8020882 0.6270074
```

Help Files

- Functions are ubiquitous in R!
- To find out about a function's arguments use `help()`
- Understanding the help files is key to using code!
- For instance we can try:
  - `help(seq)`
  - `help(runif)`

Objects and Common Classes

2. Matrix

- collection of vectors of the same **type and length**

```
#populate vectors
x <- rep(0.2, times = 6)
y <- c(1, 3, 4, -1, 5, 6)

#check 'type'
is.numeric(x)
```

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

```
is.numeric(y)
```

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

Objects and Common Classes

2. Matrix

- collection of vectors of the same **type and length**

```
#populate vectors
x <- rep(0.2, times = 6)
y <- c(1, 3, 4, -1, 5, 6)

#check 'length'
length(x)
```

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

```
length(y)
```

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

Objects and Common Classes

2. Matrix

- collection of vectors of the same **type and length**

```
#populate vectors
x <- rep(0.2, times = 6)
y <- c(1, 3, 4, -1, 5, 6)
#combine in a matrix (check help(matrix))
matrix(c(x, y), ncol = 2)
```

```
##      [,1] [,2]
## [1,]  0.2    1
## [2,]  0.2    3
## [3,]  0.2    4
## [4,]  0.2   -1
## [5,]  0.2    5
## [6,]  0.2    6
```

Objects and Common Classes

2. Matrix

- collection of vectors of the same **type and length**

```
x <- c("Hi", "There", "!"); y <- c("a", "b", "c"); z <- c("One", "Two", "Three")
is.character(x)
```

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

```
matrix(c(x, y, z), nrow = 3)
```

```
##      [,1]    [,2] [,3]
## [1,] "Hi"    "a"  "One"
## [2,] "There" "b"  "Two"
## [3,] "!"     "c"  "Three"
```

Objects and Common Classes

4. Data Frame

- collection (list) of vectors of the same **length**

```
x <- c("a", "b", "c", "d", "e", "f")
y <- c(1, 3, 4, -1, 5, 6)
z <- 10:15
data.frame(x, y, z)
```

```
##   x  y  z
## 1 a  1 10
## 2 b  3 11
## 3 c  4 12
## 4 d -1 13
## 5 e  5 14
## 6 f  6 15
```

Objects and Common Classes

4. Data Frame

- collection (list) of vectors of the same **length**

```
x <- c("a", "b", "c", "d", "e", "f")
y <- c(1, 3, 4, -1, 5, 6)
z <- 10:15
data.frame(char = x, data1 = y, data2 = z)
```

- char, data1, and data2 become the variable names for the data frame

Objects and Common Classes

5. List

- a vector that can have differing elements

```
list("Hi", 1, 2, "!")
```

```
## [[1]]
## [1] "Hi"
##
## [[2]]
## [1] 1
##
## [[3]]
## [1] 2
##
## [[4]]
## [1] "!"
```

Objects and Common Classes

5. List

- a vector that can have differing elements
- Not just differing types, but differing objects!

```
x <- c("Hi", "There", "!")
y <- c(1, 3, 4, -1, 5, 6)
list(x, y)
```

```
## [[1]]
## [1] "Hi"    "There" "!"
##
## [[2]]
## [1]  1  3  4 -1  5  6
```

- More flexible than a Data Frame!

Recap!

Review:

Dimension	Homogeneous	Heterogeneous
1d	Atomic Vector	List
2d	Matrix	Data Frame

- For most data analysis you'll use data frames!
- Next up: How do we access/change parts of our objects?

Activity

- **Objects and Common Classes Activity** instructions available on web
- Work in small groups
- Ask questions! TAs and I will float about the room
- Feel free to ask questions about anything you didn't understand as well!

Attributes and Basic Data Manipulation

- Want to know how to handle complex data sets
- R has many 'built-in' data sets

```
iris
```

Attributes and Basic Data Manipulation

- What kind of object is iris?

Attributes and Basic Data Manipulation

- **What kind of object is iris?**
- `str()` function can tell us (structure)

```
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 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Attributes and Basic Data Manipulation

- What characteristics does iris have?

Attributes and Basic Data Manipulation

- **What characteristics does iris have?**
- `attributes()` function can tell us metadata
  - Metadata = information about the data set
  - Returns a named list

```
attributes(iris)
```

Attributes and Basic Data Manipulation

```
## $names
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
##  [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##  [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
##  [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
## [103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
## [137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150
##
## $class
## [1] "data.frame"
```

Attributes and Basic Data Manipulation

- How do we access different parts of our object?

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
    - Just a column

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
  - Just a column
  - Multiple columns

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
  - Just a column
  - Multiple columns
  - Just a row

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
  - Just a column
  - Multiple columns
  - Just a row
  - Multiple rows

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
  - Just a column
  - Multiple columns
  - Just a row
  - Multiple rows
  - Access to values of an attribute

Attributes and Basic Data Manipulation

- **How do we access different parts of our object?**
- Often want things like
  - Just a column
  - Multiple columns
  - Just a row
  - Multiple rows
  - Access to values of an attribute
- Let's go through each of our common data types and work our way up!

Attributes and Basic Data Manipulation

Atomic Vectors

- Access elements of a vector using square brackets

```
letters #built in vector
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
letters[10]
```

```
## [1] "j"
```

Attributes and Basic Data Manipulation

Atomic Vectors

- Can feed R a vector of values to choose

```
letters[1:4]
```

```
## [1] "a" "b" "c" "d"
```

```
letters[c(5, 10, 15, 20, 25)]
```

```
## [1] "e" "j" "o" "t" "y"
```

```
x <- c(1, 2, 5); letters[x]
```

```
## [1] "a" "b" "e"
```

Attributes and Basic Data Manipulation

Matrices

- Access elements of a matrix using square brackets with a comma in between
- Notice the default row names and column names!

```
mat <- matrix(c(1:4, 20:17), ncol = 2)
mat
```

```
##      [,1] [,2]
## [1,]    1   20
## [2,]    2   19
## [3,]    3   18
## [4,]    4   17
```

Attributes and Basic Data Manipulation

Matrices

- Access elements using square brackets with a comma

```
mat[2, 2]
```

```
## [1] 19
```

```
mat[ , 1]
```

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

```
mat[2, ]
```

```
## [1]  2 19
```

```
mat[2:4, 1]
```

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

```
mat[c(2, 4), ]
```

```
##      [,1] [,2]
## [1,]    2   19
## [2,]    4   17
```

Attributes and Basic Data Manipulation

Matrices

- Can give columns names and use them for access
- `help(matrix)` can show us how!

Attributes and Basic Data Manipulation

Matrices

- Can give columns names and use them for access

```
mat <- matrix(c(1:4, 20:17), ncol = 2,
          dimnames = list(NULL,
                c("First", "Second"))
        )
mat
```

```
##      First Second
## [1,]     1     20
## [2,]     2     19
## [3,]     3     18
## [4,]     4     17
```

```
mat[, "First"]
```

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

Attributes and Basic Data Manipulation

Matrices

- Alternatively, we can assign the dimnames after creation

```
mat <- matrix(c(1:4, 20:17), ncol = 2)
dimnames(mat) <- list(NULL, c("First", "Second"))
mat
```

```
##      First Second
## [1,]     1     20
## [2,]     2     19
## [3,]     3     18
## [4,]     4     17
```

- dimnames a *special attribute* with its own function!

Attributes and Basic Data Manipulation

Matrices

- What about the structure and attributes of matrices?

```
str(mat)
```

```
##  int [1:4, 1:2] 1 2 3 4 20 19 18 17
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "First" "Second"
```

```
attributes(mat)
```

```
## $dim
## [1] 4 2
##
## $dimnames
## $dimnames[[1]]
## NULL
##
## $dimnames[[2]]
## [1] "First"  "Second"
```

Attributes and Basic Data Manipulation

Data Frames

- Recall the built in iris data frame

```
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 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Attributes and Basic Data Manipulation

Data Frames

- Can access just like a matrix

```
iris[1:4, 2:4]
```

```
##   Sepal.Width Petal.Length Petal.Width
## 1         3.5          1.4         0.2
## 2         3.0          1.4         0.2
## 3         3.2          1.3         0.2
## 4         3.1          1.5         0.2
```

```
iris[1, ]
```

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

Attributes and Basic Data Manipulation

Data Frames

- Can use variable names

```
iris[ , c("Sepal.Length", "Species")]
```

```
##     Sepal.Length    Species
## 1            5.1     setosa
## 2            4.9     setosa
## 3            4.7     setosa
## 4            4.6     setosa
## 5            5.0     setosa
## 6            5.4     setosa
## 7            4.6     setosa
## 8            5.0     setosa
## 9            4.4     setosa
## 10           4.9     setosa
## 11           5.4     setosa
## 12           4.8     setosa
## 13           4.8     setosa
```

Attributes and Basic Data Manipulation

Data Frames

- Dollar sign allows access to columns!

```
iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

Attributes and Basic Data Manipulation

Lists

- Use double square brackets to get at list elements

```
x <- list("HI", c(10:20), 1)
x[[1]]
```

```
## [1] "HI"
```

```
x[[3]]
```

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

```
x[[2]]
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20
```

```
x[[2]][4:5]
```

```
## [1] 13 14
```

Attributes and Basic Data Manipulation

Lists

- If named list elements, can use $

```
x <- list("HI", c(10:20), 1)
str(x)
```

```
## List of 3
##  $ : chr "HI"
##  $ : int [1:11] 10 11 12 13 14 15 16 17 18 19 ...
##  $ : num 1
```

```
x <- list(First="Hi", Second=c(10:20), Third=1)
x$Second
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20
```

Attributes and Basic Data Manipulation

Data Frames

- A list of equal length vectors, can use square brackets

```
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 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
iris[[2]]
```

```
##   [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9
##  [18] 3.5 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2
##  [35] 3.1 3.2 3.5 3.6 3.0 3.4 3.5 2.3 3.2 3.5 3.8 3.0 3.8 3.2 3.7 3.3 3.2
##  [52] 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7
```

Attributes and Basic Data Manipulation

Atomic Vectors

- Note: Atomic vectors by default don't have any attributes

```
x <- seq(from = 1, to = 10, by = 2)
str(x)
```

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

```
attributes(x)
```

```
## NULL
```

Attributes and Basic Data Manipulation

Atomic Vectors

- Can add attributes to **any** object

```
names(x) <- letters[1:length(x)]
str(x)
```

```
##  Named num [1:5] 1 3 5 7 9
##  - attr(*, "names")= chr [1:5] "a" "b" "c" "d" ...
```

```
attributes(x)
```

```
## $names
## [1] "a" "b" "c" "d" "e"
```

Attributes and Basic Data Manipulation

Atomic Vectors

- New attribute via `attr()`
- Attributes returns a list!

```
attr(x, which = "MyAttr") <- "Best vector ever"
str(attributes(x))
```

```
## List of 2
##  $ names : chr [1:5] "a" "b" "c" "d" ...
##  $ MyAttr: chr "Best vector ever"
```

```
attributes(x)$MyAttr
```

```
## [1] "Best vector ever"
```

Recap!

- Attributes and Structure important to understand
  - Tells us how to access the object!
  - `attributes()`
  - `str()`
- Set new attribute with `attr()`
- Accessing common data structures
  - Atomic vectors - `x[ ]`
  - Matrices - `x[ , ]`
  - Data Frames - `x[ , ]` or `x$name`
  - Lists - `x[[ ]]` or `x$name`

Activity

- **Attributes and Basic Data Manipulation Activity** instructions available on web
- Work in small groups
- Ask questions! TAs and I will float about the room
- Feel free to ask questions about anything you didn't understand as well!