

NC STATE UNIVERSITY

Introduction to R for Data Science Part I

Justin Post
August 12-13, 2019

Course Schedule

Daily agenda:

- 9:30-10:40 Session
- 10-minute break
- 10:50-12:00 Session
- 12:00-1:15 Lunch
- 1:15-2:25 Session
- 10-minute break
- 2:35-3:45 Session
- 3:45 – 4:00 Q&A and Feedback

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

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!

If installed, go to console and type `install.packages("tidyverse")`

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.)
- Vibrant community readily sharing (rbloggers for instance)
- Can *easily* create pdfs, slides, reports, html files, and interactive apps.

Drawbacks of Using R

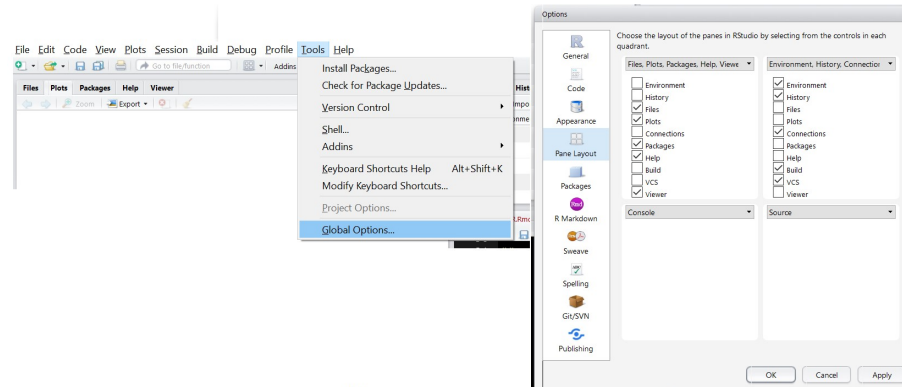
- Somewhat slow generally (although ways to speed it up)
- Code style differs greatly!
- New code not necessarily verified
- Confusing! Often many ways to do the same thing

R Studio Interface

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

R Studio Interface

To rearrange panes



- Global options → Appearance allows font/background changes
- Global options → Code allows for soft-wrap of script files

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) #exp is exponential function
```

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

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

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

Basic Use of 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+Enter (PC) or Command+Enter (MAC)
 - Highlight section and do above

Basic Use of R

- Go to file -> New File -> R Script
- Type `hist(cars$dist)`
- Submit to console using button or hot key
- Have a script open as we go so you can try things!
- Save script using file -> Save
- Note: All R code from slides available on web!

R Objects

R has strong **Object Oriented Programming (OOP)** tools

- Object: data structure with attributes
- Methods exist to act on an object type
- R functions like `print()` or `plot()` act differently depending on object type

```
str(cars)
```

```
## 'data.frame':   50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

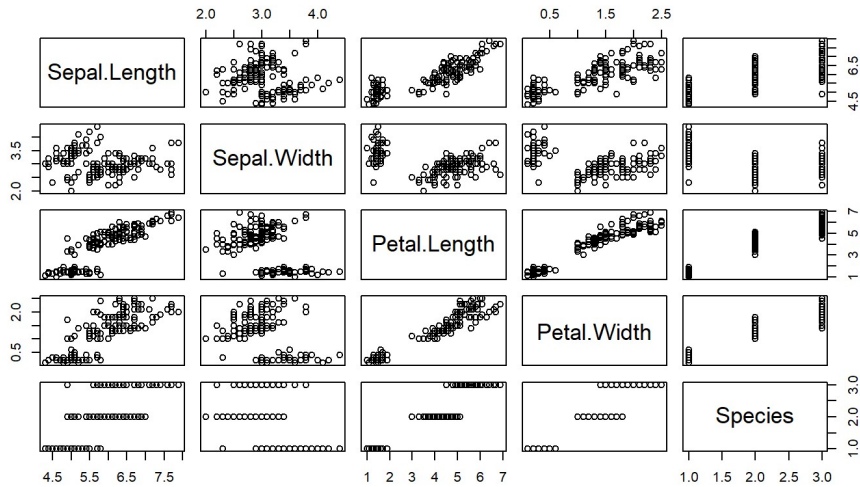
Methods

```
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 ...
```

Methods

```
plot(iris)
```



R Objects

- Store **data/info/function/etc.** in R objects

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

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

```
#strings (text) can be saved as well  
words = "Hello there!"  
words
```

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


Objects and Common Classes

- You have data...
- 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

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 'functions' output 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] 0.8390161 0.3873480 0.2367200 0.4803183
```

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!
 - `help(seq)`
 - `help(runif)`

Objects and Common Classes

1. Atomic Vector (a set of elements with an ordering)
 - Vectors useful to know about
 - Not usually useful for your data
 - Often 'building blocks' for other data types

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)
```

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**

```
#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), nrow = 2, byrow = TRUE)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 0.2  0.2  0.2  0.2  0.2  0.2
## [2,] 1.0  3.0  4.0 -1.0  5.0  6.0
```

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

2. Matrix

- collection of vectors of the same **type and length**
- Useful for some data
- Often some variables with numbers, some with text

| brand | tar | nicotine | weight | co |
|--------------|------|----------|--------|------|
| Alpine | 14.1 | 0.86 | 0.9853 | 13.6 |
| Benson | 16.0 | 1.06 | 1.0938 | 16.6 |
| CamelLights | 8.0 | 0.67 | 0.9280 | 10.2 |
| Carlton | 4.1 | 0.40 | 0.9462 | 5.4 |
| Chesterfield | 15.0 | 1.04 | 0.8885 | 15.0 |
| GoldenLights | 8.8 | 0.76 | 1.0267 | 9.0 |
| Kent | 12.4 | 0.95 | 0.9225 | 12.3 |
| Kool | 16.6 | 1.12 | 0.9372 | 16.3 |
| L&M | 14.9 | 1.02 | 0.8858 | 15.4 |
| MarkLights | 13.7 | 1.01 | 0.9643 | 13.0 |

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

4. Data Frame

- collection (list) of vectors of the same **length**
- Perfect for most data sets!
- Most functions that read data in store it as a data frame

Objects and Common Classes

5. List

- a vector that can have differing elements

```
list("Hi", 1:3, rnorm(2), c("!", "?"))
```

```
## [[1]]  
## [1] "Hi"  
##  
## [[2]]  
## [1] 1 2 3  
##  
## [[3]]  
## [1] -0.7036555 0.1470195  
##  
## [[4]]  
## [1] "!" "?"
```

Objects and Common Classes

5. List

- a vector that can have differing elements
- More flexible than a Data Frame!
- Useful for more complex types of data

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"
##
## $class
## [1] "data.frame"
##
## $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
```

Basic Data Manipulation

- How do we access different parts of our object?

Basic Data Manipulation

- How do we access different parts of our object?
- For data may want
 - Data value
 - Just a column
 - Multiple columns
 - Just a row
 - Multiple rows
 - Access to values of an attribute

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"
```

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"
```

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
```

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
```


Basic Data Manipulation

Matrices

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

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
```

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"
```

Basic Data Manipulation

Data Frames

- '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 ...
```

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
```

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
## 14          4.3      setosa
## 15          5.8      setosa
## 16          5.7      setosa
## 17          5.4      setosa
## 18          5.1      setosa
## 19          5.7      setosa
## 20          5.1      setosa
## 21          5.4      setosa
## 22          5.1      setosa
## 23          4.6      setosa
## 24          5.1      setosa
## 25          4.8      setosa
## 26          5.0      setosa
## 27          5.0      setosa
## 28          5.2      setosa
## 29          5.2      setosa
## 30          4.7      setosa
## 31          4.8      setosa
## 32          5.4      setosa
## 33          5.2      setosa
## 34          5.5      setosa
## 35          4.9      setosa
## 36          5.0      setosa
## 37          5.5      setosa
## 38          4.9      setosa
## 39          4.4      setosa
## 40          5.1      setosa
```

Basic Data Manipulation

Data Frames

- Dollar sign allows access to columns! (Returns a vector.)

```
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
```

Basic Data Manipulation

Data Frames

- Dollar sign allows access to columns! (Returns a vector.)
- Most used method for accessing a data frame
- RStudio fills in options.
 - Type `iris$`
 - If no choices - hit tab
 - Hit tab again to choose

Basic Data Manipulation

Lists

- Use double square brackets to get at list elements

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

```
## [[1]]
## [1] "HI"
##
## [[2]]
## [1] 10 11 12 13 14 15 16 17 18 19 20
##
## [[3]]
## [1] 1
```

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
```

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
```

Basic Data Manipulation

Data Frames

- Connection: Data Frame = List of equal length vectors

```
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
##  [69] 2.2 2.5 3.2 2.8 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0
##  [86] 3.4 3.1 2.3 3.0 2.5 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7
## [103] 3.0 2.9 3.0 3.0 2.5 2.9 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6
## [120] 2.2 3.2 2.8 2.8 2.7 3.3 3.2 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6 3.0
## [137] 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2 3.3 3.0 2.5 3.0 3.4 3.0
```

Accessing Attributes

- Often want to change or modify attributes

```
str(attributes(iris))
```

```
## List of 3
## $ names      : chr [1:5] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" ...
## $ class      : chr "data.frame"
## $ row.names: int [1:150] 1 2 3 4 5 6 7 8 9 10 ...
```

- a list!

Accessing Attributes

- Often want to change or modify attributes

```
attributes(iris)$names
```

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

```
attributes(iris)$names[1] <- "Sepal_Length"  
attributes(iris)$names
```

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

Attributes and Basic Data Manipulation

- Often want to change or modify attributes
- Most commonly modified attributes have 'helper' functions

```
names(iris)
```

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

```
names(iris)[2] <- "Sepal_Width"
names(iris)
```

```
## [1] "Sepal_Length" "Sepal_Width"  "Petal.Length"  "Petal.Width"
## [5] "Species"
```

Partial Matching

With `[]` or `$` partial matching can be used

```
iris$Sp
```

```
##      [1] setosa      setosa      setosa      setosa      setosa      setosa
##      [7] setosa      setosa      setosa      setosa      setosa      setosa
##     [13] setosa      setosa      setosa      setosa      setosa      setosa
##     [19] setosa      setosa      setosa      setosa      setosa      setosa
##     [25] setosa      setosa      setosa      setosa      setosa      setosa
##     [31] setosa      setosa      setosa      setosa      setosa      setosa
##     [37] setosa      setosa      setosa      setosa      setosa      setosa
##     [43] setosa      setosa      setosa      setosa      setosa      setosa
##     [49] setosa      setosa      versicolor  versicolor  versicolor  versicolor
##     [55] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [61] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [67] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [73] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [79] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [85] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [91] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
##     [97] versicolor  versicolor  versicolor  versicolor  virginica   virginica
##    [103] virginica   virginica   virginica   virginica   virginica   virginica
##    [109] virginica   virginica   virginica   virginica   virginica   virginica
##    [115] virginica   virginica   virginica   virginica   virginica   virginica
##    [121] virginica   virginica   virginica   virginica   virginica   virginica
##    [127] virginica   virginica   virginica   virginica   virginica   virginica
##    [133] virginica   virginica   virginica   virginica   virginica   virginica
##    [139] virginica   virginica   virginica   virginica   virginica   virginica
##    [145] virginica   virginica   virginica   virginica   virginica   virginica
## Levels: setosa versicolor virginica
```


Partial Matching

With `[]` or `$` partial matching can be used

```
iris[["Petal.Len", exact = FALSE]]
```

```
##      [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3
##     [18] 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4
##     [35] 1.5 1.2 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7
##     [52] 4.5 4.9 4.0 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1
##     [69] 4.5 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5
##     [86] 4.5 4.7 4.4 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1
##    [103] 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9
##    [120] 5.0 5.7 4.9 6.7 4.9 5.7 6.0 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1
##    [137] 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9 5.7 5.2 5.0 5.2 5.4 5.1
```

Recap!

- Attributes and Structure important to understand
 - `attributes()`
 - `str()`
- 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!