Plan for the week

- **M: Introduction to R**
- **W: dplyr**
  - More advanced R functionality (sort, filter, group by, etc.)
- **F: Section**
  - Visualization (i.e., plotting) in R
Data Wrangling
Some definitions

- A **data table** is a collection of measurements
- A **variable** is a column in a data table
- An **observation** is a row in a data table
- A **measurement**, or **scalar**, is a value for each (observation, variable) pair
Data wrangling

- **Reduction/Aggregation**: reduces a variable to a scalar
  - Summarizing (e.g., How many students are in the class? And how many had perfect attendance?)
- **Transformation/Mutation**: creates a new variable based on one or more existing variables
  - From a student’s total absences and total number of school days, compute the proportion of days the student was absent
Data wrangling (cont’d)

- **Data verb**: transforms a data table into a new one, usually by applying a reduction or transformation
  - Adding or deleting variables or observations
  - Sorting a variable (e.g., names) in, say, ascending order
  - Filtering a variable (e.g., who was absent on the day of the test?)

- **Grouping data** is another data verb whereby data are grouped before a reduction or transformation is applied
  - “Group by” students whose last name begins with A to M and N to Z
Programming with data

- Figure out what you want to do
- Describe those tasks in code, i.e., as a computer program
- Execute your program and interpret the output
- Repeat until your program is bug-free
- Repeat again, because what you want to do has inevitably changed (for the better)
Introduction to R
Statistical tools are central to data science

- One could define data science as a set of methods that can be used to draw robust conclusions from partial information
- But before data science, this goal was already a goal of statistics, which explains why statistics is integral to data science
- R is a tool for statistical computation; it is a facilitator for both data science and statistics to achieve this goal
What is R, more specifically?

- Some nifty things R can do include:
  - Basic maths (arithmetic, probability, statistics)
  - Machine learning (clustering, classification, regression)
  - Numerical optimization and mathematical programming
  - Visualizations: static and dynamic graphics

- In this course, we will use R for almost all of the above (not so much mathematical programming)
Before we start...

● **Style matters in programming!**  
  ○ But you don’t want to be original!  
  ○ Code is hard to read, even for expert programmers.  
  ○ Abide by this style guide to make it easier for other R programmers (including your later self!) to read what you write.

● **Testing is essential!**  
  ○ You must test every single line of code you write.  
  ○ We will test our code manually, by running each and every line in turn, and observing the outputs, one after another.

● So is commenting! (Code is written for computers to read, not people!)
Values in R

The most basic R values (or data types) are: numerics, characters, and logicals.

>> TRUE
   # expression
TRUE
   # value

>> 100
   # expression
100
   # value

>> “fun”
   # expression
“fun”
   # value

Note: In other programming languages, logicals are called booleans.
Values in R

The most basic R values (or data types) are: numerics, characters, and logicals.

```r
>> TRUE # expression  
TRUE # value

Error: object true not found

>> 100 # expression

100 # value

>> "fun" # expression

"fun" # value

>> "true" # expression

"true" # value

Note: In other programming languages, logicals are called booleans.
Expressions in R

Expressions: Programs are made of up expressions, which built up from values, and are the sentences the language can “understand,” and hence evaluate.

```
>> 3 + 4
7
# expression
# value

>> 3 - 4
-1
# expression
# value

>> 3 * 4
12
# expression
# value

>> 3 / 4
0.75
# expression
# value
```
String Manipulations in R

- To find the length of a string:

  ```
  >>> nchar("hello")  # expression
  5  # value
  ```

- To combine (concatenate) strings:

  ```
  >>> paste("Mary", "had", "a", "little", "lamb")
  "Mary had a little lamb"

  >>> paste("Mary", "had", "a", "little", "lamb", sep = "-")
  "Mary-had-a-little-lamb"
  ```
Variables in R

Variables are names used to store, and then later reference, data

```
>> x <- 5  # assigns value of x
>> y <- 10 # assigns value of y
>> x * y  # expression
50        # value

>> z <- 5  # assigns value of z
>> z <- z + 1 # updates value of z
value of z
>> z       #
expression
6         # value
>> z <- z - 1 + y # updates value of z
>> z         #
```
A conditional expression, or just conditional for short, is used to write expressions that depend on a logical.

A conditional in R looks like this:

```r
if (logical) {
  expression
}
```

The expression is evaluated only if the logical is `TRUE`.
Examples in R

```r
>>> if (TRUE) {
    a <- 100
# if TRUE
value 100
}

>>> a
# what is a’s value?
100
# a is equal to 100

to 100

>>> if (!TRUE) {
    a <- a - 100
# if not TRUE (i.e., FALSE)
# update a’s value
}
```
Predicates

- A predicate is a special kind of expression that evaluates to a logical, meaning true or false
- Examples:
  - It is raining today
  - The value of $x$ is greater than 0
- They are used, generally, to test a condition to decide whether or not to do something
  - If it is raining today, then I should carry an umbrella
  - If the value of $x$ is greater than 0, then I can withdraw money from my account
Conditionals, more generally

- A conditional expression, or just conditional for short, is used to write expressions that depend on a predicate
- A conditional in R looks like this:

```r
if (predicate) {
  expression
}
```

- The expression is evaluated only if the predicate is TRUE
More complicated examples in R

```r
>> x <- 5
# sets value of x to 5
>> y <- 10
# sets value of y to 10
>> y
# what is y’s value?
10
# y is equal to 10
>> if (x == 5) {
    y <- y + 20
# if x is equal to 5,
# update y’s value to be
}
```

More complicated conditionals

- It is possible to include an `else` clause in a conditional

```java
if (condition) {
    trueExpression
} else {
    falseExpression
}
```

```java```
if (It is a weekday) {
    Get up early
} else {
    Sleep late
}
```
More complicated examples in R

```r
if (x != 5) {
# if x is NOT equal to 5,
y <- y + 20
# update y’s value to be its original value plus 20
} else {
  y <- y - 10
# update y’s value to be its original value minus 10
}

>> y
# what is y’s value?
```
Beyond Values: Data Structures
Data Frames
Data frames

- Used for storing databases
- R has plenty of built-in data frames
  - iris, mtcars (motor trend cars), USArrests, ToothGrowth, etc.
Manipulating data frames in R

- Use `head` to see the first few entries of a data frame
- Use `tail` to see the last few

```r
> head(mtcars)
                   mpg cyl disp  hp drat    wt qsec vs am gear carb
Mazda RX4         21.0  6 160 110 3.90 2.620 16.46  0  1   4    4
Mazda RX4 Wag     21.0  6 160 110 3.90 2.875 17.02  0  1   4    4
Datsun 710        22.8  4 108  93 3.85  2.320 21.40  1  1   4    2
Hornet 4 Drive    21.4  6 258 110 3.08 3.215 17.05  0  1   4    4
Hornet Sportabout 18.7  8 360 175 3.15 3.440 18.10  0  1   4    2
Valiant           18.1  6 225 105 2.76 3.770 20.22  1  0   4    2
```

```r
> tail(mtcars)
                   mpg cyl disp  hp drat    wt qsec vs am gear carb
Porsche 914-2     26.0  4 120.3 91 4.43 2.140 16.70  0  1   5    2
Lotus Europa      30.4  4  95.1 113 3.77 1.513 16.90  1  1   5    2
Ford Pantera L    15.8  8 351.0 264 4.22 3.170 14.50  0  1   5    4
Ferrari Dino      19.7  6 145.0 175 3.62 2.770 15.50  0  1   5    6
Maserati Bora     15.0  8 301.0 335 3.54 3.570 14.60  0  1   5    8
Volvo 142E        21.4  4 121.0 109 4.11 2.780 18.60  1  1   4    2
```
• Use `str` to see the overall structure

```r
> str(mtcars)
'data.frame': 32 obs. of 11 variables:
$ mpg: num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ... 
$ cyl: num 6 6 4 6 8 6 8 4 4 6 ... 
$ disp: num 160 160 108 258 360 ... 
$ hp: num 110 110 93 110 175 105 245 62 95 123 ... 
$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ... 
$ wt: num 2.62 2.88 2.32 3.21 3.44 ... 
$ qsec: num 16.5 17 18.6 19.4 17 ... 
$ vs: num 0 0 1 1 0 1 0 1 1 1 ... 
$ am: num 1 1 0 0 0 0 0 0 0 0 ... 
$ gear: num 4 4 4 3 3 3 4 4 4 4 ... 
$ carb: num 4 4 1 1 2 1 4 2 2 4 ... 
```
• Use **names** to see the variable names (i.e., column headers)

```r
> names(mtcars)
[1] "mpg"  "cyl"  "disp"  "hp"   "drat"  "wt"   "qsec"  "vs"   "am"   "gear"  "carb"
```

• Use **dim** to see the dimensions (number of rows and columns)

```r
> dim(mtcars)
[1] 32 11
```

• Or, if you want the number of rows and columns as individual integers, use **nrow** and **ncol**

```r
> nrow(mtcars)
[1] 32
> ncol(mtcars)
[1] 11
```
Use `summary` to summarize the values of each variable (min, 1st quartile, median, mean, 3rd quartile, max)

```
> summary(mtcars)

       mpg    cyl     disp    hp       drat      wt     qsec      vs
Min. 10.400  4.000  71.100  52.00  2.7600  1.5130  14.500  0.0000
1st Qu.15.430  6.000 120.800  96.50  3.0800  2.5810  16.890  0.0000
Median 19.200  6.000  196.300 123.00  3.6950  3.3250  17.710  0.0000
Mean  20.090  6.188  230.700 146.70  3.5970  3.2170  17.850  0.4375
3rd Qu.22.800  8.000  326.000 180.00  3.9200  3.6100  18.900  1.0000
Max.  33.900  8.000  472.000 335.00 10.8800  5.7900  20.000  2.2000
```
• Use $ to select a single column in a data frame

> mtcars$mpg
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
[31] 15.0 21.4

Everything in the mpg column

• Apply a summarization function to a single column

> mean(mtcars$mpg)
[1] 20.09062

> median(mtcars$mpg)
[1] 19.2
You can also select a portion of the data frame.

```r
> mtcars[31:32, 1:4]

       mpg cyl disp  hp
Maserati Bora 15.0   8  301 335
Volvo 142E   21.4   4  121 109
```

Selection of rows 31 and 32, and columns 1 through 4, only.
• You can also select a single row, or a few rows

```r
> mtcars[3,]
       mpg cyl disp  hp drat    wt  qsec vs am gear carb
Datsun 710    710  22.8  108  93 3.85  2.32 18.61    1    1  4    1
```
Selection of row 3 only

```r
> mtcars[2:5,]
       mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4 Wag    21  6 160 110 3.90  2.875 17.02    0    1  4    4
Datsun 710    22.8  4 108  93 3.85  2.320 18.61    1    1  4    1
Hornet 4 Drive    21.4  6 258 110 3.08  3.215 19.44    1    0  3    1
Hornet Sportabout    18.7  8 360 175 3.15  3.440 17.02    0    0  3    2
```
Selection of rows 2 through 5
• Here is the analog of selecting rows: selecting columns

```
Multiple columns

> mtcars[,1:3]
                      mpg  cyl disp
Mazda RX4             21.0  6 160.0
Mazda RX4 Wag         21.0  6 160.0
Datsun 710            22.8  4 108.0
Hornet 4 Drive        21.4  6 258.0
Hornet Sportabout     18.7  8 360.0
Valiant               18.1  6 225.0
Duster 360            14.3  8 360.0
```
You can also select all but a single row or column with –

```r
> head(mtcars[-1,])

<table>
<thead>
<tr>
<th></th>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazda RX4</td>
<td>21.0</td>
<td>6</td>
<td>160</td>
<td>110</td>
<td>3.90</td>
<td>2.875</td>
<td>17.02</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Datsun 710</td>
<td>22.8</td>
<td>4</td>
<td>108</td>
<td>93</td>
<td>3.85</td>
<td>2.320</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Hornet 4 Drive</td>
<td>21.4</td>
<td>6</td>
<td>258</td>
<td>110</td>
<td>3.08</td>
<td>3.215</td>
<td>19.44</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Hornet Sportabout</td>
<td>18.7</td>
<td>8</td>
<td>360</td>
<td>175</td>
<td>3.15</td>
<td>3.440</td>
<td>17.02</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Valiant</td>
<td>18.1</td>
<td>6</td>
<td>225</td>
<td>105</td>
<td>2.76</td>
<td>3.460</td>
<td>20.22</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Duster 360</td>
<td>14.3</td>
<td>8</td>
<td>360</td>
<td>245</td>
<td>3.21</td>
<td>3.570</td>
<td>15.84</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```

Selection of all but row 1: Mazda RX4

```r
> head(mtcars[, -1])

<table>
<thead>
<tr>
<th></th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazda RX4</td>
<td>6</td>
<td>160</td>
<td>3.90</td>
<td>2.620</td>
<td>16.46</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Mazda RX4 Wag</td>
<td>6</td>
<td>160</td>
<td>3.90</td>
<td>2.875</td>
<td>17.02</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Datsun 710</td>
<td>4</td>
<td>108</td>
<td>3.85</td>
<td>2.320</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Hornet 4 Drive</td>
<td>6</td>
<td>258</td>
<td>3.08</td>
<td>3.215</td>
<td>19.44</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Hornet Sportabout</td>
<td>8</td>
<td>360</td>
<td>3.15</td>
<td>3.440</td>
<td>17.02</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Valiant</td>
<td>6</td>
<td>225</td>
<td>2.76</td>
<td>3.460</td>
<td>20.22</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

Selection of all but column 1: mpg
Vectors

- A vector is a sequence of objects (can be numbers, strings, etc.)
  - Points in the Cartesian plane are two-dimensional vectors

- Vectors can also be 3, 4, 5, etc. dimensions:
  - (1, 2, 3) is a 3-dimensional vector
  - (10, -20, 30, -40) is a 4-dimensional vector
  - (1.1, -2.2, -3.3, -4.4, 5.5) is a 5-dimensional vector
Representing vectors in R

We use the \texttt{c} function to create a vector in R:

\begin{verbatim}
>> x <- c(1, 2, 3, 4)  # creates vector x
  1  2  3  4

>> y <- c(-1, -2, -3, -4)  # creates vector y
  -1 -2 -3 -4

>> z <- c("hello", "world")  # creates vector z
  "hello" "world"

>> w <- c(TRUE, TRUE, FALSE)  # creates vector w
  TRUE TRUE FALSE
\end{verbatim}
Computing with numerical vectors in R

Many common mathematical functions apply to (i.e., across) vectors:

```r
>> x <- c(1, 2, 3, 4)  # creates vector x
>> y <- c(-1, -2, -3, -4)  # creates vector y
>> x + y  # sums two vectors
  0 0 0 0

>> y * -1  # multiples vector by -1
  1 2 3 4
>> x * y  # multiplies two vectors
0 0 0 0
```
Computing with logical vectors in R

Logical functions can also be applied to (i.e., across) vectors:

```r
>> x <- c(TRUE, FALSE, FALSE)
>> y <- c(FALSE, FALSE, TRUE)

>> x & y
vector logical AND
FALSE FALSE FALSE
```

```r
>> x | y
vector logical OR
TRUE FALSE TRUE
```
Summarizing numerical vectors in R

Other mathematical functions summarize vectors: 
sum, mean, min, and max, etc.

```r
>> x <- c(1, 2, 3, 4)  # creates vector x
>> sum(x)              # sums elements of x
  10
>> mean(x)             # calculates mean of x
  2.5
>> min(x)              # calculates min of x
  1
```
What can we learn about cars?

- The mean $\text{mpg}$ is roughly 20
  
  ```
  > mean(mtcars$mpg)
  [1] 20.09062
  ```

- The heaviest car weighs in at 5424 lbs
  
  ```
  > max(mtcars$wt)
  [1] 5.424
  ```
Computing with string vectors in R

We can also apply functions across vectors of strings:

```r
>> days <- c("Mon", "Tues", "Wednes", "Thurs", "Fri")
>> week <- paste(days, "day", sep = "")
>> week
("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

>> week <- paste("day", days, sep = "")
>> week
("dayMon", "dayTues", "dayWednes", "dayThurs", "dayFri")
```
Logical vectors in R

`>> mtcars$mpg > 23`

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[20] TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
```

`>> mtcars$cyl == 4`

```
[1] FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
[20] TRUE TRUE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE
```
Computing with logical vectors in R

```r
>> mtcars[mtcars$mpg > 23, ]

<table>
<thead>
<tr>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.4</td>
<td>4</td>
<td>146.7</td>
<td>62</td>
<td>3.69</td>
<td>3.190</td>
<td>20.00</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>32.4</td>
<td>4</td>
<td>78.7</td>
<td>66</td>
<td>4.08</td>
<td>2.200</td>
<td>19.47</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>30.4</td>
<td>4</td>
<td>75.7</td>
<td>52</td>
<td>4.93</td>
<td>1.615</td>
<td>18.52</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>33.9</td>
<td>4</td>
<td>71.1</td>
<td>65</td>
<td>4.22</td>
<td>1.835</td>
<td>19.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>27.3</td>
<td>4</td>
<td>79.0</td>
<td>66</td>
<td>4.08</td>
<td>1.935</td>
<td>18.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>26.0</td>
<td>4</td>
<td>120.3</td>
<td>91</td>
<td>4.43</td>
<td>2.140</td>
<td>16.70</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>30.4</td>
<td>4</td>
<td>95.1</td>
<td>113</td>
<td>3.77</td>
<td>1.513</td>
<td>16.90</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```r
>> mtcars[mtcars$cyl == 4, ]

<table>
<thead>
<tr>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.8</td>
<td>4</td>
<td>108.0</td>
<td>93</td>
<td>3.85</td>
<td>2.320</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>24.4</td>
<td>4</td>
<td>146.7</td>
<td>62</td>
<td>3.69</td>
<td>3.190</td>
<td>20.00</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>22.8</td>
<td>4</td>
<td>140.8</td>
<td>95</td>
<td>3.92</td>
<td>3.150</td>
<td>22.90</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>32.4</td>
<td>4</td>
<td>78.7</td>
<td>66</td>
<td>4.08</td>
<td>2.200</td>
<td>19.47</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>30.4</td>
<td>4</td>
<td>75.7</td>
<td>52</td>
<td>4.93</td>
<td>1.615</td>
<td>18.52</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>33.9</td>
<td>4</td>
<td>71.1</td>
<td>65</td>
<td>4.22</td>
<td>1.835</td>
<td>19.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>21.5</td>
<td>4</td>
<td>120.1</td>
<td>97</td>
<td>3.70</td>
<td>2.465</td>
<td>20.01</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>27.3</td>
<td>4</td>
<td>79.0</td>
<td>66</td>
<td>4.08</td>
<td>1.935</td>
<td>18.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>26.0</td>
<td>4</td>
<td>120.3</td>
<td>91</td>
<td>4.43</td>
<td>2.140</td>
<td>16.70</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>30.4</td>
<td>4</td>
<td>95.1</td>
<td>113</td>
<td>3.77</td>
<td>1.513</td>
<td>16.90</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>21.4</td>
<td>4</td>
<td>121.0</td>
<td>109</td>
<td>4.11</td>
<td>2.780</td>
<td>18.60</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
```
Computing with logical vectors in R

```r
>> ex1 <- c("b", "a", "c")
>> order(ex1)
2 1 3

>> ex1[order(ex1)]
"a" "b" "c"

>> sort(ex1)
"a" "b" "c"
```
Computing with logical vectors in R

```r
>> order(mtcars$mpg)

[1] 15 16 24 7 17 31 14 23 22 29 12 13 11 6 5 10 25 30 1 2 4 32 21
  3  9  8 27 26 19 28 18 20

>> mtcars$mpg[(order(mtcars$mpg))]

[1] 10.4 10.4 13.3 14.3 14.7 15.0 15.2 15.2 15.5 15.8 16.4 17.3 17.8
[14] 18.1 18.7 19.2 19.2 19.7 21.0 21.0 21.4 21.4 21.5 22.8 22.8 24.4
[27] 26.0 27.3 30.4 30.4 32.4 33.9

>> sort(mtcars$mpg)

[1] 10.4 10.4 13.3 14.3 14.7 15.0 15.2 15.2 15.5 15.8 16.4 17.3 17.8
[14] 18.1 18.7 19.2 19.2 19.7 21.0 21.0 21.4 21.4 21.5 22.8 22.8 24.4
[27] 26.0 27.3 30.4 30.4 32.4 33.9
```
Factors
Categorical data: Nominal

Factors are used to represent categorical data in R

```
>> survey <- c("M", "F", "M", "O", "F")
>> survey
"M", "F", "M", "O", "F"

>> new_survey <- factor(survey)
M F M O F
Levels: F M O
```
Categorical data: Ordinal

Factors are used to represent categorical data in R

```r
>> survey <- c("small", "medium", "large", "medium")
>> survey
"small" "medium" "large" "medium"

>> new_survey <- factor(survey, ordered = TRUE, levels = c("small", "medium", "large"))
>> new_survey
small medium large medium
Levels: small < medium < large
```
NA (no answer)

NA is a special logical value

```r
>> survey <- c("M", "F", "M", NA, "F")
>> survey
"M", "F", "M", NA, "F"

>> is.na(survey)
FALSE FALSE FALSE TRUE FALSE

>> survey[!is.na(survey)]
"M", "F", "M", "F"
```
Data Wrangling

Filter, Sort, & Merge
Quick shortcut

```r
>> mean(mtcars$mpg)
20.09062

>> attach(mtcars)
>> mean(mpg)
20.09062
```
- **Filter**: select a subset of rows, depending on some condition

```r
> subset(mtcars, mpg > 23)

<table>
<thead>
<tr>
<th></th>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merc 240D</td>
<td>24.4</td>
<td>4</td>
<td>146.7</td>
<td>62</td>
<td>3.69</td>
<td>3.190</td>
<td>20.00</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Fiat 128</td>
<td>32.4</td>
<td>4</td>
<td>78.7</td>
<td>66</td>
<td>4.08</td>
<td>2.200</td>
<td>19.47</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Honda Civic</td>
<td>30.4</td>
<td>4</td>
<td>75.7</td>
<td>52</td>
<td>4.93</td>
<td>1.615</td>
<td>18.52</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Toyota Corolla</td>
<td>33.9</td>
<td>4</td>
<td>71.1</td>
<td>65</td>
<td>4.22</td>
<td>1.835</td>
<td>19.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Fiat X1-9</td>
<td>27.3</td>
<td>4</td>
<td>79.0</td>
<td>66</td>
<td>4.08</td>
<td>1.935</td>
<td>18.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Porsche 914-2</td>
<td>26.0</td>
<td>4</td>
<td>120.3</td>
<td>91</td>
<td>4.43</td>
<td>2.140</td>
<td>16.70</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Lotus Europa</td>
<td>30.4</td>
<td>4</td>
<td>95.1</td>
<td>113</td>
<td>3.77</td>
<td>1.513</td>
<td>16.90</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

> max(subset(mtcars, mpg > 23)$wt)

[1] 3.19
- Combine selecting rows and columns

```
> subset(mtcars, mpg > 23)[,1:2]
   mpg cyl
Merc 240D  24.4  4
Fiat 128   32.4  4
Honda Civic 30.4  4
Toyota Corolla 33.9  4
Fiat X1-9  27.3  4
Porsche 914-2 26.0  4
Lotus Europa 30.4  4
```

Selection of the first two columns of the subset of rows containing cars with mpg greater than 23

```
> subset(mtcars, mpg > 23)[1:2,]
   mpg cyl disp  hp  drat   wt  qsec vs am gear carb
Merc 240D  24.4  4 146.7 62 3.69  3.19 20.0  1  0   4   2
Fiat 128  32.4  4  78.7 66 4.08  2.20 19.4  1  1   4   1
```
Sort

`> mtcars[order(mpg), ]`

<table>
<thead>
<tr>
<th></th>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cadillac Fleetwood</td>
<td>10.4</td>
<td>8</td>
<td>472</td>
<td>205</td>
<td>2.93</td>
<td>5.250</td>
<td>17.98</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Lincoln Continental</td>
<td>10.4</td>
<td>8</td>
<td>460</td>
<td>215</td>
<td>3.00</td>
<td>5.424</td>
<td>17.82</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Camaro Z28</td>
<td>13.3</td>
<td>8</td>
<td>350</td>
<td>245</td>
<td>3.73</td>
<td>3.840</td>
<td>15.41</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Duster 360</td>
<td>14.3</td>
<td>8</td>
<td>360</td>
<td>245</td>
<td>3.01</td>
<td>3.570</td>
<td>15.84</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Chrysler Imperial</td>
<td>14.7</td>
<td>8</td>
<td>440</td>
<td>230</td>
<td>3.23</td>
<td>5.345</td>
<td>17.42</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Maserati Bora</td>
<td>15.0</td>
<td>8</td>
<td>301</td>
<td>335</td>
<td>3.54</td>
<td>3.570</td>
<td>14.60</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Sort by mpg, in ascending order
Sort (cont’d)

```r
>> mtcars[order(cyl, mpg), ]
```

```
> head(mtcars[order(cyl, mpg),])
          mpg   cyl disp  hp drat    wt  qsec vs am gear carb
Volvo 142E  21.4   4  121.0 109 4.11 2.780 18.60    1    1     4   2
Toyota Corona 21.5   4  120.1 97 3.70 2.465 20.01    1    0     3   1
Datsun 710   22.8   4  108.0 93 3.85 2.320 18.61    1    1     4   1
Merc 230     22.8   4  140.8 95 3.92 3.150 22.90    1    0     4   2
Merc 240D    24.4   4  146.7 82 3.93 3.157 20.00    1    0     4   2
Porsche 914-2 26.0   4  120.3 91 4.43 2.140 16.70    0    1     5   2
```

Sort by cyl, and then mpg (both in ascending order)

```r
>> mtcars[order(mpg, -cyl), ]
```

```
> head(mtcars[order(mpg, -cyl),])
          mpg   cyl disp  hp drat    wt  qsec vs am gear carb
Cadillac Fleetwood 10.4   8   472 205 2.93 5.250 17.98    0    0     3   4
Lincoln Continental 10.4   8   460 215 3.00 5.424 17.82    0    0     3   4
Camaro Z28        13.3   8   350 245 3.73 3.840 15.41    0    0     3   4
Duster 360        14.3   8   360 245 3.21 3.570 15.84    0    0     3   4
Chrysler Imperial 14.7   8   440 230 3.23 5.345 17.42    0    0     3   4
Maserati Bora     15.0   8   301 335 3.54 3.570 14.60    0    1     5   8
```

Sort by mpg (ascending), and then cyl (descending)
Join

INNER JOIN

FULL JOIN

LEFT JOIN

RIGHT JOIN
• **Merge**

```r
> gpa <- data.frame(Name = c("Amy", "Alex", "Nina"), GPA = c(1.2, 2.3, 5.2))
> gpa
   Name GPA
1   Amy 1.2
2  Alex 2.3
3  Nina 5.2

> tas <- data.frame(Name = c("Joon", "Will", "Alex", "Nina", "Anna", "Erin", "Ben"),
       age = c(21, 20, 20, 20, 19, 19, 19))
> tas
   Name age
1  Joon 21
2   Will 20
3  Alex 20
4  Nina 20
5  Anna 19
6   Erin 19
7    Ben 19
```
### Merge (cont’d)

#### Inner join (default)

```r
> merge(gpa, tas, by = "Name")
  Name GPA age
1  Alex 2.3 20
2  Nina 5.2 20
```

#### Left outer join

```r
> merge(gpa, tas, by = "Name", all.x = TRUE)
  Name GPA age
1  Alex 2.3 20
2   Amy 1.2 NA
3  Nina 5.2 20
```

#### Outer join

```r
> merge(gpa, tas, by = "Name", all = TRUE)
  Name GPA age
1  Alex 2.3 20
2   Amy 1.2 NA
3  Nina 5.2 20
4   Anna NA 19
5   Ben NA 19
6  Erin NA 19
7  Joon NA 21
8  Will NA 20
```

#### Right

```r
> merge(gpa, tas, by = "Name", all.y = TRUE)
  Name GPA age
1  Alex 2.3 20
2  Nina 5.2 20
3   Anna NA 19
4   Ben NA 19
5  Erin NA 19
6  Joon NA 21
7  Will NA 20
```
In-class survey
A survey on the blue bear (Bluno)

Let’s imagine a survey where we ask Brown students:

- Their year
- On a scale of 1 to 5 (1 being hate, and 5 love), how much do they like the blue bear?
- Should it stay?
To create your own data frame

Here’s a sample of responses from some imaginary students:

```
year <- c(1, 3, 4, 4)
rating <- c(5, 2, 1, 2)
keep <- c(TRUE, FALSE, FALSE, FALSE)
```

Here’s how to create a data frame from these vectors:

```
df <- data.frame(year, rating, keep)
```

<table>
<thead>
<tr>
<th>year</th>
<th>rating</th>
<th>keep</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Add a column to your data frame

There are a number of ways to do this:

```r
df$name <- c("Andreas", "Monica", "Nikhil", "Alex")
df[['name']] <- c("Andreas", "Monica", "Nikhil", "Alex")
df[, "name"] <- c("Andreas", "Monica", "Nikhil", "Alex")
```

All produce the same result:

```r
> df
   year rating keep  name
  1    1     5 TRUE  Andreas
  2    3     2 FALSE Monica
  3    4     1 FALSE Nikhil
  4    4     2 FALSE  Alex
```

We can see that Andreas is a first-year who loves the bear, and Monica, Nikhil, and Alex are upperclassmen who dislike it.
iClicker Q

What is your year?

A) First year
B) Sophomore
C) Junior
D) Senior
E) Grad student
On a scale of 1 to 5 (1 being hate, and 5 love), how much do you like the blue bear?
iClicker Q

Should it stay?

A) Yes
B) No
Summary

● Basic R values: numerics, characters, logicals
● R objects: data frames, vectors
● Data wrangling, so far:
  ○ Select (rows or columns)
  ○ Sort (rearrange data)
  ○ Filter (remove rows)
  ○ Summarize (e.g., mean)
  ○ Transform (e.g., add columns)
● Still to come: Grouping