NC STATE UNIVERSITY

Programming in R Part I

Justin Post

August 15, 2018

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

What do we want to be able to do?

- Restructure Data/Clean Data
- Streamline repeated sections of code
- Improve efficiency of code
- Write custom functions to simplify code

Where do we start?

- Review of concepts
- Using dplyr/tidry to manipulate data
- For loops
- If/Then logic
- Vectorized Functions
- Function Writing

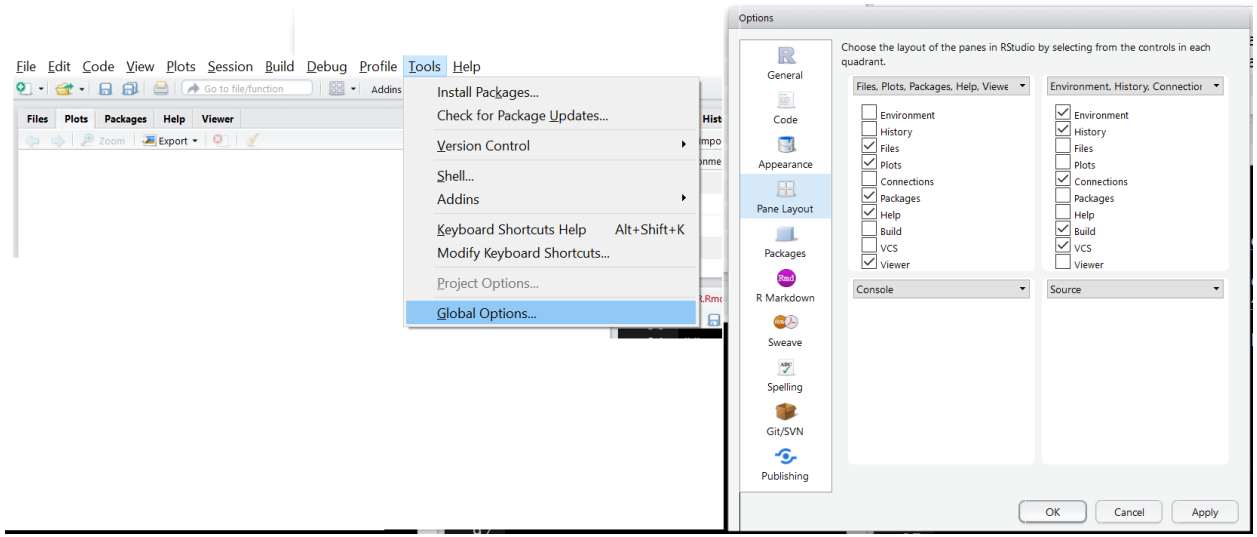
Review of Concepts

R Studio

- Great integrated development environment (IDE)
- Four main 'areas' we'll use
  - Scripting and Viewing Area
  - Environment/History
  - Plots/Packages/Help
  - Console

Review of Concepts

R Studio - Can rearrange panes



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

Review of Concepts

Data Frames

- Best R object for data sets
- 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 data2
## 1    a     1    10
## 2    b     3    11
## 3    c     4    12
## 4    d    -1    13
## 5    e     5    14
## 6    f     6    15
```

Review of Concepts

Data Frames

- Consider the built in `iris` data set

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
```

Review of Concepts

Data Frames

- Can see info about object with `str()` and `attributes()`

```
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(iris)
```

```
## $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
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
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
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
iris[1, ]
```

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

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
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
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
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
```

Review of Concepts

Packages - Many ways to accomplish the same thing in R

- How to choose?
  - Want 'fast' code
  - Want 'easy' syntax
  - Good default settings on functions
- Base R has reasonable defaults and syntax but functions are slow
- "TidyVerse" - collection of R packages that share common philosophies and are designed to work together!
  - Very efficient code
  - Common syntax

Review of Concepts

- If not installed (downloaded) on computer

```
install.packages("tidyverse")
```

Review of Concepts

- Once installed, `library()` or `require()` to load

```
library(tidyverse)
```

```
## -- Attaching packages --------
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts -----------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

- `dplry` package made for most standard data manipulation tasks
- `tidyr` handles most of the rest

Tidyverse Syntax

- Reason you might choose `dplyr` and packages from the `tidyverse`
- Fast!
- Good defaults
- All packages have similar syntax! All work on `tibbles` (data frames)
- Syntax: function(data.frame, actions, …)

`dplyr` package

- Basic commands
  - `tbl_df()` - convert data frame to one with better printing
  - `filter()` - subset **rows**
  - `arrange()` - reorder **rows**
  - `select()` - subset **columns**
  - `mutate()` - add newly created **column**
  - `transmute()` - create new variable
  - `group_by()` - group **rows** by a variable
  - `summarise()` - apply basic function to data
  - `left_join()`, `right_join()`, `inner_join()`, `full_join()` - commands to combine multiple data frames

Subsetting/Manipulating Data

`tbl_df()` - convert data frame to one with better printing

- If data read in with haven, readxl, or readr already in this format!
- Just 'wrap' data frame

```
#install.packages("fivethirtyeight")
library(fivethirtyeight)
head(fandango, n = 4) #look at just first 4 observations
```

Subsetting/Manipulating Data

`tbl_df()` - convert data frame to one with better printing

```
head(fandango, n = 4) #look at just first 4 observations
```

```
## # A tibble: 4 x 23
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 Aven~  2015             74               86         66             7.1
## 2 Cind~  2015             85               80         67             7.5
## 3 Ant-~  2015             80               90         64             8.1
## 4 Do Y~  2015             18               84         22             4.7
## # ... with 17 more variables: imdb <dbl>, fandango_stars <dbl>,
## #   fandango_ratingvalue <dbl>, rt_norm <dbl>, rt_user_norm <dbl>,
## #   metacritic_norm <dbl>, metacritic_user_nom <dbl>, imdb_norm <dbl>,
## #   rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

```
fandango <- tbl_df(fandango)
fandango
```

```
## # A tibble: 146 x 23
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 Aven~  2015             74               86         66             7.1
## 2 Cind~  2015             85               80         67             7.5
## 3 Ant-~  2015             80               90         64             8.1
## 4 Do Y~  2015             18               84         22             4.7
## 5 Hot ~  2015             14               28         29             3.4
## # ... with 141 more rows, and 17 more variables: imdb <dbl>,
## #   fandango_stars <dbl>, fandango_ratingvalue <dbl>, rt_norm <dbl>,
## #   rt_user_norm <dbl>, metacritic_norm <dbl>, metacritic_user_nom <dbl>,
## #   imdb_norm <dbl>, rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

`filter()` - subset **rows**

- Use `filter()` to obtain only 2014 movies

```
filter(fandango, year == 2014)
```

```
## # A tibble: 17 x 23
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##    <chr> <dbl>          <int>            <int>      <int>           <dbl>
##  1 Top ~  2014             86               64         81             6.8
##  2 Levi~  2014             99               79         92             7.2
##  3 Unbr~  2014             51               70         59             6.5
##  4 The ~  2014             90               92         73             8.2
##  5 Nigh~  2014             50               58         47             5.8
##  6 Selma  2014             99               86         89             7.1
##  7 Wild~  2014             96               92         77             8.8
##  8 Annie  2014             27               61         33             4.8
##  9 Bird~  2014             92               78         88             8
## 10 Mr. ~  2014             98               56         94             6.6
## 11 The ~  2014             61               75         59             7
## 12 Big ~  2014             72               69         62             7.5
## 13 Song~  2014             99               92         86             8.2
## 14 Into~  2014             71               50         69             6.1
```

Subsetting/Manipulating Data

`filter()` - subset **rows**

- Multiple filters

```
filter(fandango, (year == 2014) & (rottentomatoes <= 60))
```

```
## # A tibble: 3 x 23
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 Unbr~  2014             51               70         59             6.5
## 2 Nigh~  2014             50               58         47             5.8
## 3 Annie  2014             27               61         33             4.8
## # ... with 17 more variables: imdb <dbl>, fandango_stars <dbl>,
## #   fandango_ratingvalue <dbl>, rt_norm <dbl>, rt_user_norm <dbl>,
## #   metacritic_norm <dbl>, metacritic_user_nom <dbl>, imdb_norm <dbl>,
## #   rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

`arrange()` - reorder **rows**

```
#reorder by film title
arrange(fandango, film)
```

```
## # A tibble: 146 x 23
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 '71    2015             97               82         83             7.5
## 2 5 Fl~  2015             52               47         55             6.8
## 3 A Li~  2015             40               47         51             7
## 4 A Mo~  2014             90               69         79             7
## 5 Abou~  2015             97               86         87             9.6
## # ... with 141 more rows, and 17 more variables: imdb <dbl>,
## #   fandango_stars <dbl>, fandango_ratingvalue <dbl>, rt_norm <dbl>,
## #   rt_user_norm <dbl>, metacritic_norm <dbl>, metacritic_user_nom <dbl>,
## #   imdb_norm <dbl>, rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

`arrange()` - reorder **rows**

```
#get secondary arrangement as well
arrange(fandango, year, film)
```

```
## # A tibble: 146 x 23
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 A Mo~  2014             90               69         79             7
## 2 Annie  2014             27               61         33             4.8
## 3 Big ~  2014             72               69         62             7.5
## 4 Bird~  2014             92               78         88             8
## 5 Inhe~  2014             73               52         81             7.4
## # ... with 141 more rows, and 17 more variables: imdb <dbl>,
## #   fandango_stars <dbl>, fandango_ratingvalue <dbl>, rt_norm <dbl>,
## #   rt_user_norm <dbl>, metacritic_norm <dbl>, metacritic_user_nom <dbl>,
## #   imdb_norm <dbl>, rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

`arrange()` - reorder **rows**

```
#descending instead
arrange(fandango, year, desc(film))
```

```
## # A tibble: 146 x 23
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##    <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 Wild~  2014             96               92         77             8.8
## 2 Unbr~  2014             51               70         59             6.5
## 3 Two ~  2014             97               78         89             8.8
## 4 Top ~  2014             86               64         81             6.8
## 5 The ~  2014             90               92         73             8.2
## # ... with 141 more rows, and 17 more variables: imdb <dbl>,
## #   fandango_stars <dbl>, fandango_ratingvalue <dbl>, rt_norm <dbl>,
## #   rt_user_norm <dbl>, metacritic_norm <dbl>, metacritic_user_nom <dbl>,
## #   imdb_norm <dbl>, rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>
```

Subsetting/Manipulating Data

Piping or Chaining

- Applying multiple functions: nesting hard to parse!
- Piping or Chaining with %>% operator helps

```
arrange(filter(fandango, year == 2014), desc(film))
```

```
## # A tibble: 17 x 23
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##    <chr> <dbl>          <int>            <int>      <int>           <dbl>
##  1 Wild~  2014             96               92         77             8.8
##  2 Unbr~  2014             51               70         59             6.5
##  3 Two ~  2014             97               78         89             8.8
##  4 Top ~  2014             86               64         81             6.8
##  5 The ~  2014             90               92         73             8.2
##  6 The ~  2014             61               75         59             7
##  7 Song~  2014             99               92         86             8.2
##  8 Selma  2014             99               86         89             7.1
##  9 Nigh~  2014             50               58         47             5.8
## 10 Mr. ~  2014             98               56         94             6.6
## 11 Levi~  2014             99               79         92             7.2
## 12 Into~  2014             71               50         69             6.1
```

Subsetting/Manipulating Data

Piping or Chaining

- Applying multiple functions: nesting hard to parse!
- Piping or Chaining with %>% operator helps

```
fandango %>% filter(year == 2014) %>% arrange(desc(film))
```

```
## # A tibble: 17 x 23
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##    <chr> <dbl>          <int>            <int>      <int>           <dbl>
##  1 Wild~  2014             96               92         77             8.8
##  2 Unbr~  2014             51               70         59             6.5
##  3 Two ~  2014             97               78         89             8.8
##  4 Top ~  2014             86               64         81             6.8
##  5 The ~  2014             90               92         73             8.2
##  6 The ~  2014             61               75         59             7
##  7 Song~  2014             99               92         86             8.2
##  8 Selma  2014             99               86         89             7.1
##  9 Nigh~  2014             50               58         47             5.8
## 10 Mr. ~  2014             98               56         94             6.6
## 11 Levi~  2014             99               79         92             7.2
## 12 Into~  2014             71               50         69             6.1
```

Subsetting/Manipulating Data

Piping or Chaining

- Applying multiple functions: nesting hard to parse!
- Piping or Chaining with %>% operator helps
- If `dplyr` or `magrittr` package loaded, can often use

```
#silly example
fandango$imdb %>% quantile()
```

```
##   0%  25%  50%  75% 100%
##  4.0  6.3  6.9  7.4  8.6
```

```
fandango$imdb %>% quantile() %>% range()
```

```
## [1] 4.0 8.6
```

Subsetting/Manipulating Data

`select()` - subset **columns**

- Often only want select variables (saw `$` and `[ , ]`)
- `select()` function has same syntax as other `dplyr` functions!

```
#Choose a column by name
fandango %>% select(film, fandango_stars)
```

```
## # A tibble: 146 x 2
##   film                      fandango_stars
##   <chr>                              <dbl>
## 1 Avengers: Age of Ultron              5
## 2 Cinderella                           5
## 3 Ant-Man                              5
## 4 Do You Believe?                      5
## 5 Hot Tub Time Machine 2               3.5
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`select()` - subset **columns**

- Many ways to select variables

```
#all columns between
fandango %>% select(film, year:rottentomatoes_user)
```

```
## # A tibble: 146 x 4
##   film                    year rottentomatoes rottentomatoes_user
##   <chr>                  <dbl>          <int>                <int>
## 1 Avengers: Age of Ultron  2015             74                   86
## 2 Cinderella               2015             85                   80
## 3 Ant-Man                  2015             80                   90
## 4 Do You Believe?          2015             18                   84
## 5 Hot Tub Time Machine 2   2015             14                   28
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`select()` - subset **columns**

- Many ways to select variables

```
#all columns containing
fandango %>% select(film, contains("fandango"))
```

```
## # A tibble: 146 x 5
##   film     fandango_stars fandango_rating~ fandango_votes fandango_differ~
##   <chr>             <dbl>            <dbl>          <int>            <dbl>
## 1 Avenger~              5              4.5          14846              0.5
## 2 Cindere~              5              4.5          12640              0.5
## 3 Ant-Man               5              4.5          12055              0.5
## 4 Do You ~              5              4.5           1793              0.5
## 5 Hot Tub~              3.5            3             1021              0.5
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`select()` - subset **columns**

- Many ways to select variables

```
#all columns starting with
fandango %>% select(film, starts_with("imdb"))
```

```
## # A tibble: 146 x 5
##   film                 imdb imdb_norm imdb_norm_round imdb_user_vote_cou~
##   <chr>               <dbl>     <dbl>           <dbl>               <int>
## 1 Avengers: Age of Ul~   7.8      3.9               4              271107
## 2 Cinderella             7.1      3.55              3.5             65709
## 3 Ant-Man                7.8      3.9               4              103660
## 4 Do You Believe?        5.4      2.7               2.5              3136
## 5 Hot Tub Time Machin~   5.1      2.55              2.5             19560
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`select()` - subset **columns**

- Many ways to select variables

```
#all columns ending with
fandango %>% select(film, ends_with("user"))
```

```
## # A tibble: 146 x 3
##    film                    rottentomatoes_user metacritic_user
##    <chr>                                 <int>           <dbl>
## 1 Avengers: Age of Ultron                   86             7.1
## 2 Cinderella                                80             7.5
## 3 Ant-Man                                   90             8.1
## 4 Do You Believe?                           84             4.7
## 5 Hot Tub Time Machine 2                    28             3.4
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`mutate()` - add newly created **column**

`transmute()` - create new variable

```
##Create an average rottentomatoes score variable
fandango %>% mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2)
```

```
## # A tibble: 146 x 24
##   film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user
##   <chr> <dbl>          <int>            <int>      <int>           <dbl>
## 1 Aven~  2015             74               86         66             7.1
## 2 Cind~  2015             85               80         67             7.5
## 3 Ant-~  2015             80               90         64             8.1
## 4 Do Y~  2015             18               84         22             4.7
## 5 Hot ~  2015             14               28         29             3.4
## # ... with 141 more rows, and 18 more variables: imdb <dbl>,
## #   fandango_stars <dbl>, fandango_ratingvalue <dbl>, rt_norm <dbl>,
## #   rt_user_norm <dbl>, metacritic_norm <dbl>, metacritic_user_nom <dbl>,
## #   imdb_norm <dbl>, rt_norm_round <dbl>, rt_user_norm_round <dbl>,
## #   metacritic_norm_round <dbl>, metacritic_user_norm_round <dbl>,
## #   imdb_norm_round <dbl>, metacritic_user_vote_count <int>,
## #   imdb_user_vote_count <int>, fandango_votes <int>,
## #   fandango_difference <dbl>, avgRotten <dbl>
```

Subsetting/Manipulating Data

`mutate()` - add newly created **column**

`transmute()` - create new variable

```
#can't see it!
fandango %>% mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2) %>% select(avgRotten)
```

```
## # A tibble: 146 x 1
##   avgRotten
##       <dbl>
## 1      80
## 2      82.5
## 3      85
## 4      51
## 5      21
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`mutate()` - add newly created **column**

`transmute()` - create new variable

```
#transmute will keep the new variable only
fandango %>% transmute(avgRotten = (rottentomatoes + rottentomatoes_user)/2)
```

```
## # A tibble: 146 x 1
##   avgRotten
##       <dbl>
## 1      80
## 2      82.5
## 3      85
## 4      51
## 5      21
## # ... with 141 more rows
```

Subsetting/Manipulating Data

`group_by()` - group **rows** by a variable

`summarise()` - apply basic function to data

- Summarization - find avg number of fandango stars

```
fandango %>% summarise(avgStars = mean(fandango_stars))
```

```
## # A tibble: 1 x 1
##   avgStars
##      <dbl>
## 1     4.09
```

Subsetting/Manipulating Data

`group_by()` - group **rows** by a variable

`summarise()` - apply basic function to data

- Summarization - find avg fandango stars *by year*

```
fandango %>% group_by(year) %>% summarise(avgStars = mean(fandango_stars))
```

```
## # A tibble: 2 x 2
##    year avgStars
##   <dbl>    <dbl>
## 1  2014     4.12
## 2  2015     4.09
```

Subsetting/Manipulating Data

- May want to combine two data sets: `left_join()`, `right_join()`, `inner_join()`, `full_join()`

(Cite: http://rpubs.com/justmarkham/dplyr-tutorial-part-2)

```
# create two simple data frames
a <- data_frame(color = c("green", "yellow", "red"), num = 1:3)
b <- data_frame(color = c("green", "yellow", "pink"), size = c("S", "M", "L"))
```

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- Only include observations found in both "a" and "b" (automatically joins on variables that appear in both tables)

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
inner_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 2 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- include observations found in either "a" or "b"

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
full_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 4 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
## 3 red        3 <NA>
## 4 pink      NA L
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- include all observations found in "a", match with b

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
left_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
## 3 red        3 <NA>
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- include all observations found in "b", match with a

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
right_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
## 3 pink      NA L
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- `right_join(a, b)` is identical to `left_join(b, a)` except for column ordering

```
right_join(a,b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
## 3 pink      NA L
```

```
left_join(b, a)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 x 3
##   color  size    num
##   <chr>  <chr> <int>
## 1 green  S         1
## 2 yellow M         2
## 3 pink   L        NA
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- filter "a" to only show observations that match "b"

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
semi_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 2 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- filter "a" to only show observations that don't match "b"

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   color  size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
anti_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 1 x 2
##   color   num
##   <chr> <int>
## 1 red       3
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- sometimes matching variables don't have identical names

```
b <- b %>% rename(col = color)
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   col    size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

Subsetting/Manipulating Data

`left_join()`, `right_join()`, `inner_join()`, `full_join()` - combine multiple DFs

- specify that the join should occur by matching "color" in "a" with "col" in "b"

```
a
```

```
## # A tibble: 3 x 2
##   color    num
##   <chr>  <int>
## 1 green      1
## 2 yellow     2
## 3 red        3
```

```
b
```

```
## # A tibble: 3 x 2
##   col    size
##   <chr>  <chr>
## 1 green  S
## 2 yellow M
## 3 pink   L
```

```
inner_join(a, b,
    by = c("color" = "col"))
```

```
## # A tibble: 2 x 3
##   color    num size
##   <chr>  <int> <chr>
## 1 green      1 S
## 2 yellow     2 M
```

Overview of `dplyr` package cheatsheet

- Basic commands
  - `tbl_df()` - convert data frame to one with better printing
  - `filter()` - subset **rows**
  - `arrange()` - reorder **rows**
  - `select()` - subset **columns**
  - `mutate()` - add newly created **column**
  - `transmute()` - create new variable
  - `group_by()` - group **rows** by a variable
  - `summarise()` - apply basic function to data
  - `left_join()`, `right_join()`, `inner_join()`, `full_join()` - commands to combine multiple data frames

Manipulating Data

`tidyr` package

Easily allows for two very important actions

- **gather()** - takes multiple columns, and gathers them into key-value pairs
  - Make wide data longer
  - Most important as analysis methods often prefer this form
- **spread()** - takes two columns (key & value) and spreads in to multiple columns
  - Make "long" data wider

Manipulating Data

tidyr package

- Data in 'Wide' form

```
tempsData <- read_delim(file = "https://raw.githubusercontent.com/jbpost2/Programming-in-R/
                        master/datasets/cityTemps.txt", delim = " ")
tempsData
```

```
## # A tibble: 6 x 8
##   city        sun   mon   tue   wed   thr   fri   sat
##   <chr>     <int> <int> <int> <int> <int> <int> <int>
## 1 atlanta      81    87    83    79    88    91    94
## 2 baltimore    73    75    70    78    73    75    79
## 3 charlotte    82    80    75    82    83    88    93
## 4 denver       72    71    67    68    72    71    58
## 5 ellington    51    42    47    52    55    56    59
## 6 frankfort    70    70    72    70    74    74    79
```

Manipulating Data

tidyr package

- Switch to 'Long' form with `gather()`
  - key = new name for values in columns
  - value = new name for data values
  - columns describe which columns to take

```
gather(tempsData, key = day, value = temp, 2:8)
```

```
## # A tibble: 42 x 3
##    city      day    temp
##    <chr>     <chr> <int>
## 1 atlanta   sun      81
## 2 baltimore sun      73
## 3 charlotte sun      82
## 4 denver    sun      72
## 5 ellington sun      51
## # ... with 37 more rows
```

Manipulating Data

tidyr package

- Switch to 'Long' form with `gather()`
- Can provide columns to `gather()` in many ways!

```
newTempsData<-gather(tempsData, key = day, value = temp, sun,
mon, tue, wed, thr, fri, sat)
```

```
## # A tibble: 42 x 3
##   city      day    temp
##   <chr>     <chr> <int>
## 1 atlanta   sun      81
## 2 baltimore sun      73
## 3 charlotte sun      82
## 4 denver    sun      72
## 5 ellington sun      51
## # ... with 37 more rows
```

Manipulating Data

tidyr package

- Switch to 'Wide' form with `spread()`
  - opposite from gather
  - key = new column names
  - value = value to spread out

```
spread(newTempsData, key = day, value = temp)
```

```
## # A tibble: 6 x 8
##   city        fri   mon   sat   sun   thr   tue   wed
##   <chr>     <int> <int> <int> <int> <int> <int> <int>
## 1 atlanta      91    87    94    81    88    83    79
## 2 baltimore    75    75    79    73    73    70    78
## 3 charlotte    88    80    93    82    83    75    82
## 4 denver       71    71    58    72    72    67    68
## 5 ellington    56    42    59    51    55    47    52
## 6 frankfort    74    70    79    70    74    72    70
```

Recap!

- Tidyverse useful
- `dplyr` to manipulate data
- `tidyr` to expand, condense data

Activity

- **Manipulating Data 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!

What do we want to be able to do?

- Restructure Data/Clean Data
- **Streamline repeated sections of code**
- Improve efficiency of code
- Write custom functions to simplify code

For Loops

- Idea:
    - Run code repeatedly
    - Often change something as well
- Syntax

```
for(index in values){
  code to be run
}
```

For Loops

- index defines 'counter' or variable that varies

```
for (i in 1:10){
  print(i)
}
```

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

For Loops

- 'values' define which values index takes on

```
for (i in 1:10){
  print(i)
}
```

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

For Loops

- 'values' define which values index takes on

```
for (value in c("cat","hat","worm")){
  print(value)
}
```

```
## [1] "cat"
## [1] "hat"
## [1] "worm"
```

For Loops

- Code in loop can change based on index
- Create small data set

```
set.seed(10)
data<-round(runif(5),2)
data
```

```
## [1] 0.51 0.31 0.43 0.69 0.09
```

For Loops

- Code in loop can change based on index

```
words<-c("first", "second", "third", "fourth", "fifth")
```

- Loop through and print out the phrase

"The (#ed) data point is (# from data vector)."

```
paste0("The ", words[1], " data point is ", data[1], ".")
```

```
## [1] "The first data point is 0.51."
```

For Loops

- Code in loop can change based on index

```
for (i in 1:5){
  print(paste0("The ", words[i], " data point is ",
               data[i], "."))
}
```

```
## [1] "The first data point is 0.51."
## [1] "The second data point is 0.31."
## [1] "The third data point is 0.43."
## [1] "The fourth data point is 0.69."
## [1] "The fifth data point is 0.09."
```

For Loops

- Example: Find `summary()` for each column of a data set
- Could loop through numeric columns
- Find `summary()` for each

- Consider smaller batting data set

```
#Load Batting data set from Lahman package
library(Lahman)
Batting2010 <- Batting %>% filter(yearID == 2010) %>%
  select(playerID, teamID, G, AB, R, H, X2B, X3B, HR)
```

For Loops

- Want to find `summary()` for each column of a data set

```
summary(Batting2010[ , 3])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   15.00   33.00   50.83   74.00  162.00
```

For Loops

- Loop through numeric columns

```
stats <- matrix(nrow = 6, ncol = 7)
for (i in 1:(dim(Batting2010)[2] - 2)){
  stats[ , i] <- summary(Batting2010[ , i + 2])
}
stats
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]       [,6]      [,7]
## [1,]   1.0000   0.0000   0.00000   0.00000  0.000000  0.0000000  0.000000
## [2,]  15.0000   0.0000   0.00000   0.00000  0.000000  0.0000000  0.000000
## [3,]  33.0000  24.5000   2.00000   4.00000  0.000000  0.0000000  0.000000
## [4,]  50.8267 121.9417  15.71386  31.38201  6.258112  0.6386431  3.401917
## [5,]  74.0000 186.0000  22.00000  45.00000  8.000000  1.0000000  3.000000
## [6,] 162.0000 680.0000 115.00000 214.00000 49.000000 14.0000000 54.000000
```

For Loops

- Add column names

```
colnames(stats) <- names(Batting2010)[3:9]
stats
```

```
##             G       AB         R         H       X2B        X3B        HR
## [1,]   1.0000   0.0000   0.00000   0.00000  0.000000  0.0000000  0.000000
## [2,]  15.0000   0.0000   0.00000   0.00000  0.000000  0.0000000  0.000000
## [3,]  33.0000  24.5000   2.00000   4.00000  0.000000  0.0000000  0.000000
## [4,]  50.8267 121.9417  15.71386  31.38201  6.258112  0.6386431  3.401917
## [5,]  74.0000 186.0000  22.00000  45.00000  8.000000  1.0000000  3.000000
## [6,] 162.0000 680.0000 115.00000 214.00000 49.000000 14.0000000 54.000000
```

Vectorized Function

- Much better way to do this type of thing
- Loops are slow, didn't keep attributes here
- Covered later today!

Recap!

- **For loops** reduce redundant code
- Syntax

```
for (index in values){
  code to execute
}
```

- Values can be a sequence of numbers or character values
- Not ideal in R

Activity

- **For Loops 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!