# Contents

## R Packages

Install These Packages ........................................ 3

## Manipulating Data Frames

**dplyr Package** ............................................ 4
Grammar of **dplyr** ............................................ 4
Grammar of **dplyr** ............................................ 4
Example: Baby Names ........................................... 4
**babynames** Object ........................................... 5
Peek at the Data .................................................. 5
**%>%** Operator .................................................. 6
**filter()** ....................................................... 6
**arrange()** ...................................................... 7
**arrange()** ...................................................... 8
renaming with **select()** ....................................... 8
**select()** ....................................................... 8
Renaming with **select()** ....................................... 9
**mutate()** ....................................................... 9
No. Individuals by Year and Sex ................................ 10
**summarize()** ................................................... 10
**group_by()** .................................................... 11
No. Individuals by Year and Sex ................................ 11
How Many Distinct Names? ...................................... 12
Most Popular Names ............................................. 12
Most Popular Names ............................................. 13
Data Visualization Basics

Plots .................................................. 24
R Base Graphics ................................. 25
Read the Documentation .................... 25
Barplot .............................................. 25
Boxplot .......................................... 26
Constructing Boxplots ....................... 27
Boxplot with Outliers ....................... 27
Histogram ......................................... 28
Histogram with More Breaks ............... 29
Density Plot ....................................... 30
Boxplot (Side-By-Side) ..................... 31
Stacked Barplot ................................. 32
Scatterplot ....................................... 33

Extras ............................................. 34
License ........................................... 34
Source Code ....................................... 34
Session Information ......................... 34

R Packages

Install These Packages

Run this code in RStudio and let us know if you experience any errors.

pkgs <- c("dplyr", "babynames", "readr", "tidyr", "reshape2",
          "hexbin", "ggplot2", "ggthemes", "broom",
          "devtools", "RColorBrewer", "rvest", "xtable")
install.packages(pkgs)
Manipulating Data Frames

dplyr Package

dplyr is a package with the following description:

A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

This package offers a “grammar” for manipulating data frames. Everything that dplyr does can also be done using basic R commands – however, it tends to be much faster and easier to use dplyr.

Grammar of dplyr

Verbs:

- **filter**: extract a subset of rows from a data frame based on logical conditions
- **arrange**: reorder rows of a data frame
- **rename**: rename variables in a data frame
- **select**: return a subset of the columns of a data frame, using a flexible notation

Partially based on *R Programming for Data Science*

Grammar of dplyr

Verbs (continued):

- **mutate**: add new variables/columns or transform existing variables
- **distinct**: returns only the unique values in a table
- **summarize**: generate summary statistics of different variables in the data frame, possibly within strata
- **group_by**: breaks down a dataset into specified groups of rows

Partially based on *R Programming for Data Science*

Example: Baby Names
> library("dplyr", verbose=FALSE)
> library("babynames")
> ls()
character(0)
> babynames <- babynames::babynames
> ls()
[1] "babynames"

babynames Object

> class(babynames)
[1] "tbl_df"  "tbl"       "data.frame"
> dim(babynames)
[1] 1825433  5

> babynames
Source: local data frame [1,825,433 x 5]

    year sex name  n  prop
  <dbl> <chr> <chr> <int> <dbl>
1  1880  F    Mary    7065 0.07238359
2  1880  F    Anna    2604 0.02667896
3  1880  F   Emma    2003 0.02052149
4  1880  F Elizabeth 1939 0.01986579
5  1880  F  Minnie  1746 0.01788843
6  1880  F  Margaret 1578 0.01616720
7  1880  F    Ida  1472 0.01508119
8  1880  F   Alice  1414 0.01448696
9  1880  F  Bertha  1320 0.01352390
10 1880  F    Sarah 1288 0.01319605
... ... ... ... ... 

Peek at the Data

> set.seed(201)
> sample_n(babynames, 10)
Source: local data frame [10 x 5]

    year sex name n  prop
  <dbl> <chr> <chr> <int> <dbl>
1  1991  M  Esaias   5 2.359700e-06
<table>
<thead>
<tr>
<th>Year</th>
<th>Sex</th>
<th>Name</th>
<th>Age</th>
<th>Prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1933</td>
<td>F</td>
<td>Maida</td>
<td>33</td>
<td>3.155410e-05</td>
</tr>
<tr>
<td>1967</td>
<td>M</td>
<td>Alvis</td>
<td>33</td>
<td>1.853916e-05</td>
</tr>
<tr>
<td>1905</td>
<td>M</td>
<td>Gaylord</td>
<td>11</td>
<td>7.679151e-05</td>
</tr>
<tr>
<td>1993</td>
<td>F</td>
<td>Kyleigh</td>
<td>157</td>
<td>7.965969e-05</td>
</tr>
<tr>
<td>1927</td>
<td>M</td>
<td>Della</td>
<td>8</td>
<td>6.886519e-06</td>
</tr>
<tr>
<td>1908</td>
<td>F</td>
<td>Luberta</td>
<td>12</td>
<td>3.384753e-05</td>
</tr>
<tr>
<td>1968</td>
<td>F</td>
<td>Andrea</td>
<td>7086</td>
<td>4.145300e-03</td>
</tr>
<tr>
<td>1921</td>
<td>F</td>
<td>Ardelle</td>
<td>50</td>
<td>3.907288e-05</td>
</tr>
<tr>
<td>1955</td>
<td>M</td>
<td>Dainel</td>
<td>7</td>
<td>3.351657e-06</td>
</tr>
</tbody>
</table>

> "try also sample_frac(babynames, 6e-6)

**%>% Operator**

Originally from R package `magrittr`. Provides a mechanism for chaining commands with a forward-pipe operator, `%>%`.

```r
> x <- 1:10
> x %>% log(base=10) %>% sum
[1] 6.559763
> sum(log(x,base=10))
[1] 6.559763

> babynames %>% sample_n(5)
Source: local data frame [5 x 5]

<table>
<thead>
<tr>
<th>Year</th>
<th>Sex</th>
<th>Name</th>
<th>Age</th>
<th>Prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>M</td>
<td>Sunil</td>
<td>42</td>
<td>2.344364e-05</td>
</tr>
<tr>
<td>1996</td>
<td>F</td>
<td>Kelina</td>
<td>5</td>
<td>2.608857e-06</td>
</tr>
<tr>
<td>1991</td>
<td>F</td>
<td>Gimena</td>
<td>7</td>
<td>3.443326e-06</td>
</tr>
<tr>
<td>1979</td>
<td>M</td>
<td>Neilson</td>
<td>9</td>
<td>5.023636e-06</td>
</tr>
<tr>
<td>1984</td>
<td>F</td>
<td>Romelia</td>
<td>5</td>
<td>2.774045e-06</td>
</tr>
</tbody>
</table>

**filter()**

```r
> filter(babynames, year==1880, sex="F")
Source: local data frame [942 x 5]

<table>
<thead>
<tr>
<th>Year</th>
<th>Sex</th>
<th>Name</th>
<th>Age</th>
<th>Prop</th>
</tr>
</thead>
</table>

6
```r
> filter(babynames, year==1880, sex=='F', n > 5000)
Source: local data frame [1 x 5]

    year sex name     n     prop
   (dbl) (chr) (chr) (int)   (dbl)
1 1880   F  Mary  7065 0.07238359

arrange()

> arrange(babynames, name, year, sex)
Source: local data frame [1825433 x 5]

    year sex name     n     prop
   (dbl) (chr) (chr) (int)   (dbl)
1  2007   M  Aaban   5 2.260251e-06
2  2009   M  Aaban   6 2.834029e-06
3  2010   M  Aaban   9 4.390297e-06
4  2011   M  Aaban  11 5.429927e-06
5  2012   M  Aaban  11 5.440091e-06
6  2013   M  Aaban  14 6.961721e-06
7  2014   M  Aaban  16 7.882569e-06
8  2011   F  Aabha   7 3.622491e-06
9  2012   F  Aabha   5 2.587144e-06
10 2014   F  Aabha   9 4.642664e-06
```
arrange()

```r
> arrange(babynames, desc(name), desc(year), sex)
Source: local data frame [1,825,433 x 5]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>name</th>
<th>n</th>
<th>prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>M</td>
<td>Zzyzx</td>
<td>5</td>
<td>2.439054e-06</td>
</tr>
<tr>
<td>2014</td>
<td>M</td>
<td>Zyyon</td>
<td>6</td>
<td>2.955964e-06</td>
</tr>
<tr>
<td>2010</td>
<td>F</td>
<td>Zyyanna</td>
<td>6</td>
<td>3.067323e-06</td>
</tr>
<tr>
<td>2009</td>
<td>M</td>
<td>Zytavious</td>
<td>5</td>
<td>2.361691e-06</td>
</tr>
<tr>
<td>2010</td>
<td>M</td>
<td>Zytavious</td>
<td>6</td>
<td>2.926865e-06</td>
</tr>
<tr>
<td>2009</td>
<td>M</td>
<td>Zytavious</td>
<td>7</td>
<td>3.306368e-06</td>
</tr>
<tr>
<td>2007</td>
<td>M</td>
<td>Zytavious</td>
<td>6</td>
<td>2.712301e-06</td>
</tr>
<tr>
<td>2006</td>
<td>M</td>
<td>Zytavious</td>
<td>7</td>
<td>3.196664e-06</td>
</tr>
<tr>
<td>2005</td>
<td>M</td>
<td>Zytavious</td>
<td>5</td>
<td>2.352830e-06</td>
</tr>
<tr>
<td>2004</td>
<td>M</td>
<td>Zytavious</td>
<td>6</td>
<td>2.841628e-06</td>
</tr>
</tbody>
</table>
```

rename()

```r
> rename(babynames, number=n)
Source: local data frame [1,825,433 x 5]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>name</th>
<th>number</th>
<th>prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1880</td>
<td>F</td>
<td>Mary</td>
<td>7065</td>
<td>0.07238359</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Anna</td>
<td>2604</td>
<td>0.02667896</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Emma</td>
<td>2003</td>
<td>0.02052149</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Elizabeth</td>
<td>1939</td>
<td>0.01986579</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Minnie</td>
<td>1746</td>
<td>0.01788843</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Margaret</td>
<td>1578</td>
<td>0.01616720</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Ida</td>
<td>1472</td>
<td>0.01508119</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Alice</td>
<td>1414</td>
<td>0.01448696</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Bertha</td>
<td>1320</td>
<td>0.01352390</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Sarah</td>
<td>1288</td>
<td>0.01319605</td>
</tr>
</tbody>
</table>
```

select()
> `select(babynames, sex, name, n)`
Source: local data frame [1,825,433 x 3]

<table>
<thead>
<tr>
<th>sex</th>
<th>name</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr</td>
<td>(chr)</td>
<td>(int)</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>Mary</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>Anna</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>Emma</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>Elizabeth</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>Minnie</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>Margaret</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>Ida</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>Alice</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>Bertha</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>Sarah</td>
</tr>
</tbody>
</table>

> # same as `select(babynames, sex:n)`

Renaming with `select()`

> `select(babynames, sex, name, number=n)`
Source: local data frame [1,825,433 x 3]

<table>
<thead>
<tr>
<th>sex</th>
<th>name</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(chr)</td>
<td>(chr)</td>
<td>(int)</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>Mary</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>Anna</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>Emma</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>Elizabeth</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>Minnie</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>Margaret</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>Ida</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>Alice</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>Bertha</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>Sarah</td>
</tr>
</tbody>
</table>

`mutate()`

> `mutate(babynames, total_by_year=round(n/prop))`
Source: local data frame [1,825,433 x 6]
No. Individuals by Year and Sex

Let's put a few things together now adding the function `distinct()`...

```r
> babynames %>%
mutate(total_by_year=round(n/prop)) %>%
  select(sex, year, total_by_year) %>%
distinct()
Source: local data frame [270 x 3]

   sex year total_by_year
  <chr> <dbl>            
1    F  1880            97605
2    M  1880           118400
3    F  1881            98856
4    M  1881           108284
5    F  1882           115698
6    M  1882           120033
7    F  1883           120064
8    M  1883           112480
9    F  1884           137588
10   M  1884           122741
```

`summarize()`

```r
> summarize(babynames, mean_n = mean(n), median_n = median(n),
+ number_sex = n_distinct(sex),
+ distinct_names = n_distinct(name))
```

---

*Note: The text continues here with more details on the analysis.*
Source: local data frame [1 x 4]

```
mean_n median_n number_sex distinct_names
  (dbl)   (int)     (int)          (int)
1  184.6879    12        2             93889
```

group_by()

```r
> babynames %>% group_by(year, sex)
Source: local data frame [1,825,433 x 5]
Groups: year, sex [270]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>name</th>
<th>n</th>
<th>prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1880</td>
<td>F</td>
<td>Mary</td>
<td>7065</td>
<td>0.07238359</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Anna</td>
<td>2604</td>
<td>0.02667896</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Emma</td>
<td>2003</td>
<td>0.02052149</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Elizabeth</td>
<td>1939</td>
<td>0.01986579</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Minnie</td>
<td>1746</td>
<td>0.01788843</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Margaret</td>
<td>1578</td>
<td>0.01616720</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Ida</td>
<td>1472</td>
<td>0.01508119</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Alice</td>
<td>1414</td>
<td>0.01448696</td>
</tr>
<tr>
<td>1880</td>
<td>F</td>
<td>Bertha</td>
<td>1320</td>
<td>0.01352390</td>
</tr>
</tbody>
</table>
```  

No. Individuals by Year and Sex

```r
> babynames %>%
+ group_by(year, sex)
+ summarize(total_by_year=sum(n))
Source: local data frame [270 x 3]
Groups: year [?]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>total_by_year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1880</td>
<td>F</td>
<td>90993</td>
</tr>
<tr>
<td>1880</td>
<td>M</td>
<td>110491</td>
</tr>
<tr>
<td>1881</td>
<td>F</td>
<td>91954</td>
</tr>
<tr>
<td>1881</td>
<td>M</td>
<td>100745</td>
</tr>
<tr>
<td>1882</td>
<td>F</td>
<td>107850</td>
</tr>
<tr>
<td>1882</td>
<td>M</td>
<td>113688</td>
</tr>
<tr>
<td>1883</td>
<td>F</td>
<td>112321</td>
</tr>
</tbody>
</table>
```
Compare to earlier slide. Why the difference?

How Many Distinct Names?

```r
> babynames %>%
  group_by(sex) %>%
  summarize(mean_n = mean(n),
            distinct_names_sex = n_distinct(name))
```

Source: local data frame [2 x 3]

<table>
<thead>
<tr>
<th>sex</th>
<th>mean_n</th>
<th>distinct_names_sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>154.45</td>
<td>64911</td>
</tr>
<tr>
<td>M</td>
<td>228.66</td>
<td>39199</td>
</tr>
</tbody>
</table>

Most Popular Names

```r
> top_names <- babynames %>%
  group_by(year, sex) %>%
  summarize(top_name = name[which.max(n)])
> head(top_names)
```

Source: local data frame [6 x 3]

Groups: year [3]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>top_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1880</td>
<td>F</td>
<td>Mary</td>
</tr>
<tr>
<td>1880</td>
<td>M</td>
<td>John</td>
</tr>
<tr>
<td>1881</td>
<td>F</td>
<td>Mary</td>
</tr>
<tr>
<td>1881</td>
<td>M</td>
<td>John</td>
</tr>
<tr>
<td>1882</td>
<td>F</td>
<td>Mary</td>
</tr>
<tr>
<td>1882</td>
<td>M</td>
<td>John</td>
</tr>
</tbody>
</table>
Most Popular Names

Recent Years

> tail(top_names, n=10)
Source: local data frame [10 x 3]
Groups: year [5]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>top_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>F</td>
<td>Isabella</td>
</tr>
<tr>
<td>2010</td>
<td>M</td>
<td>Jacob</td>
</tr>
<tr>
<td>2011</td>
<td>F</td>
<td>Sophia</td>
</tr>
<tr>
<td>2011</td>
<td>M</td>
<td>Jacob</td>
</tr>
<tr>
<td>2012</td>
<td>F</td>
<td>Sophia</td>
</tr>
<tr>
<td>2012</td>
<td>M</td>
<td>Jacob</td>
</tr>
<tr>
<td>2013</td>
<td>F</td>
<td>Sophia</td>
</tr>
<tr>
<td>2013</td>
<td>M</td>
<td>Noah</td>
</tr>
<tr>
<td>2014</td>
<td>F</td>
<td>Emma</td>
</tr>
</tbody>
</table>

Most Popular Female Names

1990s

> top_names %>% filter(year >= 1990 & year < 2000, sex="F")
Source: local data frame [10 x 3]
Groups: year [10]

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>top_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>F</td>
<td>Jessica</td>
</tr>
<tr>
<td>1991</td>
<td>F</td>
<td>Ashley</td>
</tr>
<tr>
<td>1992</td>
<td>F</td>
<td>Ashley</td>
</tr>
<tr>
<td>1993</td>
<td>F</td>
<td>Jessica</td>
</tr>
<tr>
<td>1994</td>
<td>F</td>
<td>Jessica</td>
</tr>
<tr>
<td>1995</td>
<td>F</td>
<td>Jessica</td>
</tr>
<tr>
<td>1996</td>
<td>F</td>
<td>Emily</td>
</tr>
<tr>
<td>1997</td>
<td>F</td>
<td>Emily</td>
</tr>
<tr>
<td>1998</td>
<td>F</td>
<td>Emily</td>
</tr>
<tr>
<td>1999</td>
<td>F</td>
<td>Emily</td>
</tr>
</tbody>
</table>
Most Popular Male Names

1990s

```r
> top_names %>% filter(year >= 1990 & year < 2000, sex="M")
Source: local data frame [10 x 3]
Groups: year [10]

    year sex top_name
   (dbl) (chr)   (chr)
1   1990   M  Michael
2   1991   M  Michael
3   1992   M  Michael
4   1993   M  Michael
5   1994   M  Michael
6   1995   M  Michael
7   1996   M  Michael
8   1997   M  Michael
9   1998   M  Michael
10  1999   M    Jacob
```

> # Analyzing the name 'John'
> john <- babynames %>% filter(sex="M", name="John")
> plot(john$year, john$prop, type="l")
> # Analyzing the name 'Bella'
> bella <- babynames %>% filter(sex == "F", name == "Bella")
> plot(bella$year, bella$prop, type = "l")
Additional Examples

You should study additional tutorials of \texttt{dplyr} that utilize other data sets:

- Read the \texttt{dplyr} introductory vignette
- Read the examples given in \textit{R Programming for Data Science}, the “Managing Data Frames with the \texttt{dplyr} Package” chapter

Additional \texttt{dplyr} Features

- We’ve only scratched the surface – many interesting demos of \texttt{dplyr} can be found online
- \texttt{dplyr} can work with other data frame backends such as SQL databases
- There is an SQL interface for relational databases via the \texttt{DBI} package
- \texttt{dplyr} can be integrated with the \texttt{data.table} package for large fast tables
- There is a healthy rivalry between \texttt{dplyr} and \texttt{data.table}
Getting Data In and Out of R

Some Functions for Data In/Out

- We have already used the `load()` function to load `.Rdata` files; the `save()` function saves R objects to `.RData` files
- The function `read.table()` is a standard function for reading in data from a text file
- Similarly `write.table()` is a standard function for writing data to a text file
- Study the help files:

```r
> ?read.table
> ?write.table
```

Key Arguments

For `read.table`:

- `file` – the name of a file, or a connection
- `header` – logical indicating if the file has a header line
- `sep` – character string indicating how the values are separated
- `colClasses` – character vector indicating the class of each column
- `nrows` – maximum number of rows to be read in
- `skip` – number of lines to skip from beginning of file
- `stringsAsFactors` – a logical indicating if character strings should be coerced to factors

There are similar arguments for `write.table`.

CSV Files

- A CSV file is a “comma separated value” file, meaning the entries are separated by commas
- The functions `read.csv()` and `write.csv()` are specialized versions of `read.table()` and `write.table()` where essentially it sets `sep="","`
- Many data sets are distributed as `.csv` files, so these are worth knowing about
- Read the help files, `?read.csv` and `?write.csv`
Let’s write the `airquality` data frame to a tab-delimited text file (aka TSV) and a CSV file.

```r
> data("airquality", package="datasets")
> head(airquality, n=8)

Ozone Solar.R Wind Temp Month Day
1 41 190 7.4 67 5 1
2 36 118 8.0 72 5 2
3 12 149 12.6 74 5 3
4 18 313 11.5 62 5 4
5 NA NA 14.3 56 5 5
6 28 NA 14.9 66 5 6
7 23 299 8.6 65 5 7
8 19 99 13.8 59 5 8
>
> write.table(airquality, file="../data/airquality.txt",
+ sep="\t", row.names=FALSE)
> write.csv(airquality, file="../data/airquality.csv",
+ row.names=FALSE)
```

We will read in the two files we wrote in the previous slide. We first look at the top couple lines of each file using `readLines` to understand what is in the files.

```r
> readLines(con="../data/airquality.txt", n=2)
[1] "\"Ozone\"\t\"Solar.R\"\t\"Wind\"\t\"Temp\"\t\"Month\"\t\"Day\"
[2] "41\t190\t7.4\t67\t5\t1"
> aq1 <- read.table(file="../data/airquality.txt", header=TRUE,
+ sep="\t")
>
> readLines(con="../data/airquality.csv", n=2)
[1] "\"Ozone\",\"Solar.R\",\"Wind\",\"Temp\",\"Month\",\"Day\"
[2] "41,190,7.4,67,5,1"
> aq2 <- read.csv(file="../data/airquality.csv", header=TRUE)
>
> dim(aq1) == dim(aq2)
[1] TRUE TRUE
> sum(aq1 != aq2, na.rm=TRUE)
[1] 0
```
**readr Package**

There are a number of R packages that provide more sophisticated tools for getting data in and out of R, especially as data sets have become larger and larger.

One of those packages is **readr**. It reads and writes data quickly, provides a useful status bar for large files, and does a good job at determining data types. **readr** is organized similarly to the base R functions. For example, there are functions **read_table**, **read_csv**, **write_tsv**, and **write_csv**.

**Scraping from the Web (Ex. 1)**

There are several packages that facilitate “scraping” data from the web, including **rvest** demonstrated here.

```r
> library("rvest")
> schedule <- read_html("http://sml201.github.io/schedule/")
> first_table <- html_table(schedule)[[1]]
> names(first_table) <- c("week", "topics", "reading")
> first_table[4,"week"]
[1] 4
> first_table[4,"topics"] %>% strsplit(split=" ")
[[1]] [1] "Getting data in and out of R"
[2] "Exploring and visualizing data (part 1)"
> first_table[4,"reading"] %>% strsplit(split=" ")
[[1]] [1] "ADS Ch. 4" " EDAR, Ch. 3-9"
[3] "EDAS, Ch. 5, 10, 11" "RPDS, Ch. 6, 7"
> grep("EDAR", first_table$reading)
[1] 3 4 5 11
```

**Scraping from the Web (Ex. 2)**

The **rvest** documentation recommends **SelectorGadget**, which is “a javascript bookmarklet that allows you to interactively figure out what css selector you need to extract desired components from a page.”

```r
> usg_url <- "http://princetonusg.com/meet-your-usg-officers/"]
> usg <- read_html(usg_url)
> officers <- html_nodes(usg, ".team-member-name") %>%
+ html_text
> head(officers, n=20)
```

---

19
APIs

API stands for “application programming interface” which is a set of routines, protocols, and tools for building software and applications.

A specific website may provide an API for scraping data from that website.

There are R packages that provide an interface with specific APIs, such as the twitteR package.

**Exploratory Data Analysis**

**What is EDA?**

Exploratory data analysis (EDA) is the process of analyzing data to uncover their key features.

John Tukey pioneered this framework, writing a seminal book on the topic (called *Exploratory Data Analysis*).

EDA involves calculating numerical summaries of data, visualizing data in a variety of ways, and considering interesting data points.

Before any model fitting is done to data, some exploratory data analysis should always be performed.

*Data science seems to focus much more on EDA than traditional statistics.*

**Possible Steps to EDA**

*Exploratory Data Analysis with R* proposes some steps to EDA:

1. Formulate your question
2. Read in your data
3. Check the packaging
4. Run `str()`
5. Look at the top and the bottom of your data
6. Check your n’s (i.e., count things)
7. Validate with at least one external data source
8. Try the easy solution first
9. Challenge your solution
10. Follow up

However, EDA involves much more than these steps...

### Motor Trend Car Road Tests Data Set

**mtcars Data Set**

Let’s use the Motor Trend Car Road Tests data set `mtcars` to establish some EDA techniques.

```r
> data("mtcars", package="datasets")
> head(mtcars)

<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>
</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>2.620</td>
<td>16.46</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mazda RX4 Wag</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>
</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>
</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>
</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>
</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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazda RX4</td>
<td>4</td>
</tr>
<tr>
<td>Mazda RX4 Wag</td>
<td>4</td>
</tr>
<tr>
<td>Datsun 710</td>
<td>4</td>
</tr>
<tr>
<td>Hornet 4 Drive</td>
<td>3</td>
</tr>
<tr>
<td>Hornet Sportabout</td>
<td>3</td>
</tr>
<tr>
<td>Valiant</td>
<td>3</td>
</tr>
</tbody>
</table>
```

### Numerical Summaries of Data

**Useful Summaries**

- **Center**: mean, median, mode
- **Quantiles**: percentiles, five number summaries
• **Spread**: standard deviation, variance, interquartile range

• **Outliers**

**Measures of Center**

Suppose we have data points \(x_1, x_2, \ldots, x_n\).

**Mean**: 
\[
\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}
\]

**Median**: Order the points \(x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}\). The median is the middle value:
- \(x_{((n+1)/2)}\) if \(n\) is odd
- \((x_{(n/2)} + x_{(n/2+1)})/2\) if \(n\) is even

**Mode**: The most frequently repeated value among the data (if any). If there are ties, then there is more than one mode.

**Mean, Median, and Mode in R**

Let’s calculate these quantities in R.

```r
> mean(mtcars$mpg)
[1] 20.09062
> median(mtcars$mpg)
[1] 19.2
> sample_mode <- function(x) {
  + as.numeric(names(which(table(x) == max(table(x)))))
  + }
> sample_mode(round(mtcars$mpg))
[1] 15 21
```

It appears there is no R base function for calculating the mode.

**Quantiles and Percentiles**

The \(p\)th **percentile** of \(x_1, x_2, \ldots, x_n\) is a number such that \(p\%\) of the data are less than this number.

The 25th, 50th, and 75th percentiles are called 1st, 2nd, and 3rd “quartiles”, respectively. These are sometimes denoted as Q1, Q2, and Q3. The median is the 50th percentile aka the 2nd quartile aka Q2.
In general, $q$-quantiles are cut points that divide the data into $q$ approximately equally sized groups. The cut points are the percentiles $1/q, 2/q, \ldots, (q-1)/q$.

**Five Number Summary**

The “five number summary” is the minimum, the three quartiles, and the maximum. This can be calculated via `fivenum()` and `summary()`. They can produce different values. Finally, `quantile()` extracts any set of percentiles.

```r
> fivenum(mtcars$mpg)
[1] 10.40 15.35 19.20 22.80 33.90
> summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 10.400  15.420  19.200  20.090  22.800  33.900
> quantile(mtcars$mpg, prob=seq(0, 1, 0.25))
   0%   25%   50%   75%  100%
10.400 15.425 19.200 22.800 33.900
```

**Measures of Spread**

The variance, standard deviation (SD), and interquartile range (IQR) measure the “spread” of the data.

**Variance:**

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

**Standard Deviation:** $s = \sqrt{s^2}$

**Interquartile Range:** $IQR = Q3 - Q1$

The SD and IQR have the same units as the observed data, but the variance is in squared units.

**Variance, SD, and IQR in R**

Variance:

```r
> var(mtcars$mpg)
[1] 36.3241
```

Standard deviation:
Identifying Outliers

An outlier is an unusual data point. Outliers can be perfectly valid but they can also be due to errors (as can non-outliers).

One must define what is meant by an outlier.

One definition is a data point that less than Q1 or greater than Q3 by $1.5 \times \text{IQR}$ or more.

Another definition is a data point whose difference from the mean is greater than $3 \times \text{SD}$ or more. For Normal distributed data (bell curve shaped), the probability of this is less than 0.27%.

Application to mtcars Data

```r
> sd_units <- abs(mtcars$wt - mean(mtcars$wt))/sd(mtcars$wt)
> sum(sd_units > 3)
[1] 0
> max(sd_units)
[1] 2.255336
>
> iqr_outlier_cuts <- fivenum(mtcars$wt)[c(2,4)] +
+ c(-1.5, 1.5)*diff(fivenum(mtcars$wt)[c(2,4)])
> sum(mtcars$wt < iqr_outlier_cuts[1] |
+ mtcars$wt > iqr_outlier_cuts[2])
[1] 2
```

Data Visualization Basics

Plots

- Single variables:
• Barplot
• Boxplot
• Histogram
• Density plot

• Two or more variables:
  – Side-by-Side Boxplots
  – Stacked Barplot
  – Scatterplot

R Base Graphics

• We’ll be plodding through “R base graphics” this week, which means graphics functions that come with R.

• By default they are very simple. However, they can be customized a lot, but it takes a lot of work.

• Also, the syntax varies significantly among plot types and some think the syntax is not user-friendly.

• We will consider a very highly used graphics package next week, called ggplot2 that provides a “grammar of graphics”. It hits a sweet spot of “flexibility vs. complexity” for many data scientists.

Read the Documentation

For all of the plotting functions covered below, read the help files.

> ?barplot
> ?boxplot
> ?hist
> ?density
> ?plot
> ?legend

Barplot

> cyl_tbl <- table(mtcars$cyl)
> barplot(cyl_tbl, xlab="Cylinders", ylab="Count")
Boxplot

```r
> boxplot(mtcars$mpg, ylab="MPG", col="lightgray")
```
Constructing Boxplots

- The top of the box is Q3
- The line through the middle of the box is the median
- The bottom of the box is Q1
- The top whisker is the minimum of Q3 + 1.5 × IQR or the largest data point
- The bottom whisker is the maximum of Q1 - 1.5 × IQR or the smallest data point
- Outliers lie outside of (Q1 - 1.5 × IQR) or (Q3 + 1.5 × IQR), and they are shown as points
- Outliers are calculated using the fivenum() function

Boxplot with Outliers

```r
> boxplot(mtcars$wt, ylab="Weight (1000 lbs)",
+     col="lightgray")
```
Histogram

> hist(mtcars$mpg, xlab="MPG", main="", col="lightgray")
Histogram with More Breaks

> hist(mtcars$mpg, breaks=12, xlab="MPG", main="", col="lightgray")
Density Plot

```r
> plot(density(mtcars$mpg), xlab="MPG", main="")
> polygon(density(mtcars$mpg), col="lightgray", border="black")
```
Boxplot (Side-By-Side)

```r
> boxplot(mpg ~ cyl, data=mtcars, xlab="Cylinders",
+         ylab="MPG", col="lightgray")
```
Stacked Barplot

```r
> counts <- table(mtcars$cyl, mtcars$gear)
> counts

   3  4  5
4  1  8  2
6  2  4  1
8 12  0  2

> barplot(counts, main="Number of Gears and Cylinders",
+ xlab="Gears", col=c("blue","red", "lightgray"))
> legend(x="topright", title="Cyl",
+ legend = rownames(counts),
+ fill = c("blue","red", "lightgray"))
```
Number of Gears and Cylinders

Scatterplot

> plot(mtcars$wt, mtcars$mpg, xlab="Weight (1000 lbs)",
  + ylab="MPG")
Extras

License

https://github.com/SML201/lectures/blob/master/LICENSE.md

Source Code

https://github.com/SML201/lectures/tree/master/week4

Session Information

> sessionInfo()
R version 3.2.3 (2015-12-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.3 (El Capitan)
locale:

attached base packages:
[1] stats graphics grDevices utils datasets methods
[7] base

other attached packages:
[1] babynames_0.2.0 dplyr_0.4.3 rvest_0.3.1
[4] xml2_0.1.2 knitr_1.12.3 magrittr_1.5
[7] devtools_1.10.0

loaded via a namespace (and not attached):
[1] Rcpp_0.12.3 R6_2.1.2 stringr_1.0.0
[4] httr_1.1.0 highr_0.5.1 tools_3.2.3
[7] parallel_3.2.3 DBI_0.3.1 selectr_0.2-3
[10] htmltools_0.3 yaml_2.1.13 lazyeval_0.1.10
[13] digest_0.6.9 assertthat_0.1 formatR_1.2.1
[16] curl_0.9.6 memoise_1.0.0 evaluate_0.8
[19] rmarkdown_0.9.2 stringi_1.0-1 XML_3.98-1.3