4, Code

Lynn Huang

August 17, 2020

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

4.1 Manipulating Data: Logicals

Logical statements are comparisons between 2 quantities.

```
"hi" == " hi"
```

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

```
"hi" == "hi"
```

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

```
4 == 1
```

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

```
4 != 1
```

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

Package tidyverse has package dplyr, useful for comparisons (esp. numeric).
Due to loss of precision, first one will be FALSE even though it's TRUE!

```
sqrt(3)^2 == 3
```

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

```
dplyr::near(sqrt(3)^2, 3)
```

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

Check type of object using is functions.

```
is.numeric("Word")
```

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

```
is.numeric(10)
```

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

```
is.character("10")
```

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

```
is.na(c(1:2, NA, 3))
```

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

Can use Boolean vector to index (which elements to include/exclude). Index using [], subset(), tidyverse/dplyr filter().

```
# Tibble is like dataframe, just nicer printing properties
iris <- tbl_df(iris)
```

```
## Warning: ‘tbl_df()‘ is deprecated as of dplyr 1.0.0.
## Please use ‘tibble::as_tibble()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```
iris
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           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           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
## # ... with 140 more rows
```

```r
# How to get only "setosa" irises?
iris$Species=="setosa"
```

```
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [49]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
iris[iris$Species=="setosa",]
```

```
## # A tibble: 50 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           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           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
## # ... with 40 more rows
```

```r
subset(iris, Species=="setosa")
```

```
## # A tibble: 50 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           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           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
## # ... with 40 more rows
```

```
dplyr::filter(iris, Species=="setosa")    # NOTE: dplyr:: not necessary if only 1 filter() function
```

```
## # A tibble: 50 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           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           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
## # ... with 40 more rows
```

Beware implicit or explicit coercion, when R changes element type from less to more flexible: logical, integer, double, character.

```
# Implicit coercion using c()
c("hi", 10)
```

```
## [1] "hi" "10"
```

```
c(TRUE, FALSE) + 0
```

```
## [1] 1 0
```

```
c(TRUE, "hi")
```

```
## [1] "TRUE" "hi"
```

```
mean(c(TRUE, FALSE, TRUE))
```

```
## [1] 0.6666667
```

```
# Explicit coercion using as functions
as.numeric(c(TRUE, FALSE, TRUE))
```

```
## [1] 1 0 1
```

```
as.character(c(1, 2, 3.5, TRUE))
```

```
## [1] "1"   "2"   "3.5" "1"
```

Compound logic available with operators.

```r
# And using &. Or using |
set.seed(3)
(x <- runif(n=10, min=0, max=1))
```

```
##  [1] 0.1680415 0.8075164 0.3849424 0.3277343 0.6021007 0.6043941 0.1246334
##  [8] 0.2946009 0.5776099 0.6309793
```

```r
(x < 0.25) | (x > 0.75)
```

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

```r
# Double operators only check 1st comparison if given a vector
(x < 0.25) || (x > 0.75)
```

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

```r
# Use logical operators to do multiple subsets on data
filter(iris, (Petal.Length>1.5) & (Petal.Width>0.3) & (Species=="setosa"))
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.4         3.9          1.7         0.4 setosa
## 2          5.1         3.3          1.7         0.5 setosa
## 3          5           3.4          1.6         0.4 setosa
## 4          5           3.5          1.6         0.6 setosa
## 5          5.1         3.8          1.9         0.4 setosa
```

```r
iris[(iris$Petal.Length>1.5) & (iris$Petal.Width>0.3) & (iris$Species=="setosa"), ]
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.4         3.9          1.7         0.4 setosa
## 2          5.1         3.3          1.7         0.5 setosa
## 3          5           3.4          1.6         0.4 setosa
## 4          5           3.5          1.6         0.6 setosa
## 5          5.1         3.8          1.9         0.4 setosa
```

4.2 Manipulating Data: Aside: R Packages

4.3 Manipulating Data: dplyr Package Lahman contains Major League Baseball data from 1871-2019 in 4 tables + others:
* People = Player names, dates of birth, death, biographical info
* Batting = Batting statistics
* Pitching = Pitching statistics
* Fielding = Fielding statistics
Other tables about teams, post-season play, awards, Hall of Fame, etc. also included. Use `help(Lahman)` to see more details.

```r
#install.packages("Lahman")
library(Lahman)
head(Batting, n=4)
```

```
##    playerID yearID stint teamID lgID  G  AB  R  H X2B X3B HR RBI SB CS BB SO
## 1 abercda01   1871     1    TRO   NA  1   4  0  0   0   0  0   0  0  0  0  0
## 2  addybo01   1871     1    RC1   NA 25 118 30 32   6   0  0  13  8  1  4  0
## 3 allisar01   1871     1    CL1   NA 29 137 28 40   4   5  0  19  3  1  2  5
## 4 allisdo01   1871     1    WS3   NA 27 133 28 44  10   2  2  27  1  1  0  2
##   IBB HBP SH SF GIDP
## 1  NA  NA NA NA    0
## 2  NA  NA NA NA    0
## 3  NA  NA NA NA    1
## 4  NA  NA NA NA    0
```

```r
# Tibble prints nicer
Batting <- tbl_df(Batting)
Batting
```

```
## # A tibble: 107,429 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 abercda~   1871     1 TRO    NA        1     4     0     0     0     0     0
##  2 addybo01   1871     1 RC1    NA       25   118    30    32     6     0     0
##  3 allisar~   1871     1 CL1    NA       29   137    28    40     4     5     0
##  4 allisdo~   1871     1 WS3    NA       27   133    28    44    10     2     2
##  5 ansonca~   1871     1 RC1    NA       25   120    29    39    11     3     0
##  6 armstbo~   1871     1 FW1    NA       12    49     9    11     2     1     0
##  7 barkeal~   1871     1 RC1    NA        1     4     0     1     0     0     0
##  8 barnero~   1871     1 BS1    NA       31   157    66    63    10     9     0
##  9 barrebi~   1871     1 FW1    NA        1     5     1     1     1     0     0
## 10 barrofr~   1871     1 BS1    NA       18    86    13    13     2     1     0
## # ... with 107,419 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

```r
# Subset to get only PIT data, only 2000
filter(Batting, teamID=="PIT")
```

```
## # A tibble: 4,871 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 barklsa~   1887     1 PIT    NL       89   340    44    76    10     4     1
##  2 beeched~   1887     1 PIT    NL       41   169    15    41     8     0     2
##  3 bishobi~   1887     1 PIT    NL        3     9     0     0     0     0     0
##  4 brownto~   1887     1 PIT    NL       47   192    30    47     3     4     0
##  5 carrofr~   1887     1 PIT    NL      102   421    71   138    24    15     6
##  6 colemjo~   1887     1 PIT    NL      115   475    75   139    21    11     2
##  7 dalryab~   1887     1 PIT    NL       92   358    45    76    18     5     2
##  8 fieldjo~   1887     1 PIT    NL       43   164    26    44     9     2     0
##  9 galvipu~   1887     1 PIT    NL       49   193    10    41     7     3     2
## 10 kuehnbi~   1887     1 PIT    NL      102   402    68   120    18    15     1
## # ... with 4,861 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

```
filter(Batting, teamID=="PIT" & yearID==2000)
```

```
## # A tibble: 46 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 anderji~   2000     1 PIT    NL       27    50     5     7     1     0     0
##  2 arroybr~   2000     1 PIT    NL       21    21     2     3     2     0     0
##  3 avenbr01   2000     1 PIT    NL       72   148    18    37    11     0     5
##  4 benjami~   2000     1 PIT    NL       93   233    28    63    18     2     2
##  5 bensokr~   2000     1 PIT    NL       32    65     3     6     2     0     0
##  6 brownad~   2000     1 PIT    NL      104   308    64    97    18     3     4
##  7 brownem~   2000     1 PIT    NL       50   119    13    26     5     0     3
##  8 chrisja~   2000     1 PIT    NL       44     0     0     0     0     0     0
##  9 clontbr~   2000     1 PIT    NL        5     0     1     0     0     0     0
## 10 cordewi~   2000     1 PIT    NL       89   348    46    98    24     3    16
## # ... with 36 more rows, and 10 more variables: RBI <int>, SB <int>, CS <int>,
## #   BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>, GIDP <int>
```

```
# Re-order rows (default by ascending order)
arrange(Batting, teamID)
```

```
## # A tibble: 107,429 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 berrych~   1884     1 ALT    UA        7    25     2     6     0     0     0
##  2 brownji~   1884     1 ALT    UA       21    88    12    22     2     2     1
##  3 carropa~   1884     1 ALT    UA       11    49     4    13     1     0     0
##  4 connojo~   1884     1 ALT    UA        3    11     0     1     0     0     0
##  5 crosscl~   1884     1 ALT    UA        2     7     1     4     1     0     0
##  6 daisege~   1884     1 ALT    UA        1     4     0     0     0     0     0
##  7 doughch~   1884     1 ALT    UA       23    85     6    22     5     0     0
##  8 gradyjo~   1884     1 ALT    UA        9    36     5    11     3     0     0
##  9 harrifr~   1884     1 ALT    UA       24    95    10    25     2     1     0
## 10 koonsha~   1884     1 ALT    UA       21    78     8    18     2     1     0
## # ... with 107,419 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

```
arrange(Batting, teamID, G)
```

```
## # A tibble: 107,429 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 daisege~   1884     1 ALT    UA        1     4     0     0     0     0     0
##  2 crosscl~   1884     1 ALT    UA        2     7     1     4     1     0     0
##  3 manloch~   1884     1 ALT    UA        2     7     1     3     0     0     0
##  4 connojo~   1884     1 ALT    UA        3    11     0     1     0     0     0
##  5 shafff01   1884     1 ALT    UA        6    19     1     3     0     0     0
##   6 berrych~   1884     1 ALT    UA        7    25     2     6     0     0     0
##   7 noftsge~   1884     1 ALT    UA        7    25     0     1     0     0     0
##   8 learyja~   1884     1 ALT    UA        8    33     1     3     0     0     0
##   9 gradyjo~   1884     1 ALT    UA        9    36     5    11     3     0     0
##  10 carropa~   1884     1 ALT    UA       11    49     4    13     1     0     0
## # ... with 107,419 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

```
arrange(Batting, teamID, desc(G))
```

```
## # A tibble: 107,429 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##   1 smithge~   1884     1 ALT    UA       25   108     9    34     8     1     0
##   2 harrifr~   1884     1 ALT    UA       24    95    10    25     2     1     0
##   3 doughch~   1884     1 ALT    UA       23    85     6    22     5     0     0
##   4 murphjo~   1884     1 ALT    UA       23    94    10    14     1     0     0
##   5 brownji~   1884     1 ALT    UA       21    88    12    22     2     2     1
##   6 koonsha~   1884     1 ALT    UA       21    78     8    18     2     1     0
##   7 mooreje~   1884     1 ALT    UA       20    80    10    25     3     1     1
##   8 shaffta~   1884     1 ALT    UA       13    55    10    18     2     0     0
##   9 carropa~   1884     1 ALT    UA       11    49     4    13     1     0     0
##  10 gradyjo~   1884     1 ALT    UA        9    36     5    11     3     0     0
## # ... with 107,419 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

```
# Subset columns: Select cols that match certain characteristic (contain X2B)
# $ operator returns simplified vector form, select() returns same type of object (tibble)
vec <- Batting$X2B
tib <- select(Batting, X2B)

# Piping/chanining can feed one function's output into another function's input
arrange(select(filter(Batting, teamID=="PIT"), playerID, G, X2B), desc(X2B))
```

```
## # A tibble: 4,871 x 3
##    playerID      G   X2B
##    <chr>     <int> <int>
##   1 wanerpa01   154    62
##   2 wanerpa01   148    53
##   3 sanchfr01   157    53
##   4 wanerpa01   152    50
##   5 comorad01   152    47
##   6 mclouna01   152    46
##   7 wagneho01   135    45
##   8 parkeda01   158    45
##   9 vanslan01   154    45
##  10 wagneho01   132    44
## # ... with 4,861 more rows
```

```
Batting %>% filter(teamID=="PIT") %>% select(playerID, G, X2B) %>% arrange(desc(X2B))
```

```
## # A tibble: 4,871 x 3
##    playerID      G   X2B
##    <chr>     <int> <int>
##  1 wanerpa01   154    62
##  2 wanerpa01   148    53
##  3 sanchfr01   157    53
##  4 wanerpa01   152    50
##  5 comorad01   152    47
##  6 mclouna01   152    46
##  7 wagneho01   135    45
##  8 parkeda01   158    45
##  9 vanslan01   154    45
## 10 wagneho01   132    44
## # ... with 4,861 more rows
```

```
# Select columns using multiple types of criteria
Batting %>% select(X2B:HR)
```

```
## # A tibble: 107,429 x 3
##      X2B   X3B    HR
##    <int> <int> <int>
##  1     0     0     0
##  2     6     0     0
##  3     4     5     0
##  4    10     2     2
##  5    11     3     0
##  6     2     1     0
##  7     0     0     0
##  8    10     9     0
##  9     1     0     0
## 10     2     1     0
## # ... with 107,419 more rows
```

```
Batting %>% select(contains("X"))
```

```
## # A tibble: 107,429 x 2
##      X2B   X3B
##    <int> <int>
##  1     0     0
##  2     6     0
##  3     4     5
##  4    10     2
##  5    11     3
##  6     2     1
##  7     0     0
##  8    10     9
##  9     1     0
## 10     2     1
## # ... with 107,419 more rows
```

```
Batting %>% select(starts_with("X"), ends_with("ID"), G)
```

```
## # A tibble: 107,429 x 7
##      X2B   X3B playerID  yearID teamID lgID      G
##    <int> <int> <chr>      <int> <fct>  <fct> <int>
##  1     0     0 abercda01   1871 TRO    NA        1
##  2     6     0 addybo01    1871 RC1    NA       25
##  3     4     5 allisar01   1871 CL1    NA       29
##  4    10     2 allisdo01   1871 WS3    NA       27
##  5    11     3 ansonca01   1871 RC1    NA       25
##  6     2     1 armstbo01   1871 FW1    NA       12
##  7     0     0 barkeal01   1871 RC1    NA        1
##  8    10     9 barnero01   1871 BS1    NA       31
##  9     1     0 barrebi01   1871 FW1    NA        1
## 10     2     1 barrofr01   1871 BS1    NA       18
## # ... with 107,419 more rows
```

```
# Rename variables
# NOTE: THis renaming isn't permanent, b/c we don't save output
Batting %>%
  select(starts_with("X"), ends_with("ID"), G) %>%
  rename("Doubles"=X2B, "Triples"=X3B)
```

```
## # A tibble: 107,429 x 7
##    Doubles Triples playerID  yearID teamID lgID      G
##      <int>   <int> <chr>      <int> <fct>  <fct> <int>
##  1       0       0 abercda01   1871 TRO    NA        1
##  2       6       0 addybo01    1871 RC1    NA       25
##  3       4       5 allisar01   1871 CL1    NA       29
##  4      10       2 allisdo01   1871 WS3    NA       27
##  5      11       3 ansonca01   1871 RC1    NA       25
##  6       2       1 armstbo01   1871 FW1    NA       12
##  7       0       0 barkeal01   1871 RC1    NA        1
##  8      10       9 barnero01   1871 BS1    NA       31
##  9       1       0 barrebi01   1871 FW1    NA        1
## 10       2       1 barrofr01   1871 BS1    NA       18
## # ... with 107,419 more rows
```

```
# Re-order variables
# everything() grabs all other variables, so this puts playerID 1st, HR 2nd, then all other cols
Batting %>% select(playerID, HR, everything())
```

```
## # A tibble: 107,429 x 22
##    playerID     HR yearID stint teamID lgID      G    AB     R     H   X2B   X3B
##    <chr>    <int>  <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 abercda~      0   1871     1 TRO    NA        1     4     0     0     0     0
##  2 addybo01      0   1871     1 RC1    NA       25   118    30    32     6     0
##  3 allisar~      0   1871     1 CL1    NA       29   137    28    40     4     5
##  4 allisdo~      2   1871     1 WS3    NA       27   133    28    44    10     2
##  5 ansonca~      0   1871     1 RC1    NA       25   120    29    39    11     3
##  6 armstbo~      0   1871     1 FW1    NA       12    49     9    11     2     1
##  7 barkeal~      0   1871     1 RC1    NA        1     4     0     1     0     0
##  8 barnero~     0    1871      1 BS1     NA        31    157     66     63     10      9
##  9 barrebi~     0    1871      1 FW1     NA         1      5      1      1      1      0
## 10 barrofr~     0    1871      1 BS1     NA        18     86     13     13      2      1
## # ... with 107,419 more rows, and 10 more variables: RBI <int>, SB <int>,
## #   CS <int>, BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

4.4 Manipulating Data: Creating New Variables From fivethirtyeight.com, fandango dataframe of film ratings (n=146 films, p=23 cols).

```
#install.packages("fivethirtyeight")
library(fivethirtyeight)
```

```
## Some larger datasets need to be installed separately, like senators and
## house_district_forecast. To install these, we recommend you install the
## fivethirtyeightdata package by running:
## install.packages('fivethirtyeightdata', repos =
## 'https://fivethirtyeightdata.github.io/drat/', type = 'source')
```

```
fandango
```

```
## # A tibble: 146 x 23
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user  imdb
##    <chr> <dbl>          <int>            <int>      <int>           <dbl> <dbl>
##  1 Aven~  2015             74               86         66             7.1   7.8
##  2 Cind~  2015             85               80         67             7.5   7.1
##  3 Ant-~  2015             80               90         64             8.1   7.8
##  4 Do Y~  2015             18               84         22             4.7   5.4
##  5 Hot ~  2015             14               28         29             3.4   5.1
##  6 The ~  2015             63               62         50             6.8   7.2
##  7 Irra~  2015             42               53         53             7.6   6.9
##  8 Top ~  2014             86               64         81             6.8   6.5
##  9 Shau~  2015             99               82         81             8.8   7.4
## 10 Love~  2015             89               87         80             8.5   7.8
## # ... with 136 more rows, and 16 more variables: 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>
```

```
# Add new column
fandango %>% mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2)
```

```
## # A tibble: 146 x 24
##    film   year rottentomatoes rottentomatoes_~ metacritic metacritic_user  imdb
##    <chr> <dbl>          <int>            <int>      <int>           <dbl> <dbl>
##  1 Aven~  2015             74               86         66             7.1   7.8
##  2 Cind~  2015             85               80         67             7.5   7.1
##  3 Ant-~  2015             80               90         64             8.1   7.8
##  4 Do Y~  2015             18               84         22             4.7   5.4
##  5 Hot ~  2015             14               28         29             3.4   5.1
##  6 The ~  2015             63               62         50             6.8   7.2
##  7 Irra~  2015             42               53         53             7.6   6.9
##  8 Top ~  2014             86               64         81             6.8   6.5
##  9 Shau~  2015             99               82         81             8.8   7.4
## 10 Love~  2015             89               87         80             8.5   7.8
## # ... with 136 more rows, and 17 more variables: 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>
```

```
# Will be at the end (might have to scroll right in output), so select to view it up front
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
##  6      62.5
##  7      47.5
##  8      75
##  9      90.5
## 10      88
## # ... with 136 more rows
```

```
# Transmute just grabs new column only (like mutate + select)
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
##  6      62.5
##  7      47.5
##  8      75
##  9      90.5
## 10      88
## # ... with 136 more rows
```

```r
# Summarize will apply basic functions like mean and sd to data
fandango %>% summarise(avgStars = mean(fandango_stars), sdStars = sd(fandango_stars))
```

```
## # A tibble: 1 x 2
##   avgStars sdStars
##      <dbl>   <dbl>
## 1     4.09   0.540
```

```r
# Can also summarize by group_by variable
fandango %>% group_by(year) %>% summarise(avgSTars = mean(fandango_stars), sdStars = sd(fandango_stars))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
##    year avgSTars sdStars
##   <dbl>    <dbl>   <dbl>
## 1  2014     4.12   0.574
## 2  2015     4.09   0.538
```

```r
# NOTE: RUn this in console and you will see Groups: year[2] showing you've grouped by year into 2 grou

# Conditional Execution with If-Then-Else
# NOTE: Always GPP (good programming practice) to include base case if no conditions are met (else)
# How to create new variable for large setosa flowers?
# If can only take 1 comparison (see Warning)
if ((iris$Petal.Length>1.5) & (iris$Petal.Width>0.3) & (iris$Species=="setosa")) {
  "Large Setosa"
}
```

```
## Warning in if ((iris$Petal.Length > 1.5) & (iris$Petal.Width > 0.3) &
## (iris$Species == : the condition has length > 1 and only the first element will
## be used
```

```r
# Use ifelse() function for a compound logical condition
help(ifelse)
```

```
## starting httpd help server ...
```

```
##  done
```

```r
ifelse((iris$Petal.Length>1.5) & (iris$Petal.Width>0.3) & (iris$Species=="setosa"), "L-S", "NotL-S")
```

```
##   [1] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "L-S"    "NotL-S" "NotL-S"
##   [9] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [17] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "L-S"
##  [25] "NotL-S" "NotL-S" "L-S"    "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [33] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [41] "NotL-S" "NotL-S" "NotL-S" "L-S"    "L-S"    "NotL-S" "NotL-S" "NotL-S"
##  [49] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [57] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [65] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [73] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [81] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [89] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
##  [97] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [105] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [113] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [121] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [129] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [137] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
## [145] "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S" "NotL-S"
```

```
# To save this new data label, use transmute() (mutate + selects new col to view) or mutate()
mutate(iris,
       Size=ifelse((Petal.Length>1.5) & (Petal.Width>0.3) & (Species=="setosa"), "L-S", "NotL-S"))
```

```
## # A tibble: 150 x 6
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species Size
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>   <chr>
##  1          5.1         3.5          1.4         0.2 setosa  NotL-S
##  2          4.9         3            1.4         0.2 setosa  NotL-S
##  3          4.7         3.2          1.3         0.2 setosa  NotL-S
##  4          4.6         3.1          1.5         0.2 setosa  NotL-S
##  5          5           3.6          1.4         0.2 setosa  NotL-S
##  6          5.4         3.9          1.7         0.4 setosa  L-S
##  7          4.6         3.4          1.4         0.3 setosa  NotL-S
##  8          5           3.4          1.5         0.2 setosa  NotL-S
##  9          4.4         2.9          1.4         0.2 setosa  NotL-S
## 10          4.9         3.1          1.5         0.1 setosa  NotL-S
## # ... with 140 more rows
```

```
# Convert from wide to long data using gather() for machine learning
(tempsData <- read_delim(file="./Data/cityTemps.txt", delim=" "))
```

```
## Parsed with column specification:
## cols(
##   city = col_character(),
##   sun = col_double(),
##   mon = col_double(),
##   tue = col_double(),
##   wed = col_double(),
##   thr = col_double(),
##   fri = col_double(),
##   sat = col_double()
## )
```

```
## # A tibble: 6 x 8
##   city        sun   mon   tue   wed   thr   fri   sat
##   <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 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
```

```
(newTempsData <- tempsData %>% gather(key=day, value=temp, 2:8))
```

```
## # A tibble: 42 x 3
##    city      day    temp
##    <chr>     <chr> <dbl>
##  1 atlanta   sun       81
##  2 baltimore sun       73
##  3 charlotte sun       82
##  4 denver    sun       72
##  5 ellington sun       51
##  6 frankfort sun       70
##  7 atlanta   mon       87
##  8 baltimore mon       75
##  9 charlotte mon       80
## 10 denver    mon       71
## # ... with 32 more rows
```

```
# Convert form long to wide data using
newTempsData %>% spread(key=day, value=temp)
```

```
## # A tibble: 6 x 8
##   city         fri   mon   sat   sun   thr   tue   wed
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 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
```

```
# NOTE: Same data set as before, although columns now alphabetically ordered

# Split 1 col into multiple cols using separate()
(chicagoData <- read_csv(file="./Data/Chicago.csv"))
```

```
## Parsed with column specification:
## cols(
##   X = col_double(),
##   city = col_character(),
##   date = col_character(),
##   death = col_double(),
##   temp = col_double(),
##   dewpoint = col_double(),
##   pm10 = col_double(),
##   o3 = col_double(),
##   time = col_double(),
##   season = col_character(),
##   year = col_double()
## )
## # A tibble: 1,461 x 11
##        X city  date      death  temp dewpoint  pm10    o3  time season  year
##    <dbl> <chr> <chr>     <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>  <dbl>
##  1  3654 chic  1/1/1997    137  36       37.5  13.1   5.66  3654 winter  1997
##  2  3655 chic  1/2/1997    123  45       47.2  41.9   5.53  3655 winter  1997
##  3  3656 chic  1/3/1997    127  40       38    27.0   6.29  3656 winter  1997
##  4  3657 chic  1/4/1997    146  51.5     45.5  25.1   7.54  3657 winter  1997
##  5  3658 chic  1/5/1997    102  27       11.2  15.3  20.8   3658 winter  1997
##  6  3659 chic  1/6/1997    127  17        5.75  9.36 14.9   3659 winter  1997
##  7  3660 chic  1/7/1997    116  16        7    20.2  11.9   3660 winter  1997
##  8  3661 chic  1/8/1997    118  19       17.8  33.1   8.68  3661 winter  1997
##  9  3662 chic  1/9/1997    148  26       24    12.1  13.4   3662 winter  1997
## 10  3663 chic  1/10/1997   121  16        5.38 24.8  10.4   3663 winter  1997
## # ... with 1,451 more rows
```

```r
chicagoData %>% separate(date, c("Day", "Month", "Year"), sep="/")
```

```
## # A tibble: 1,461 x 13
##        X city  Day   Month Year  death  temp dewpoint  pm10    o3  time season
##    <dbl> <chr> <chr> <chr> <chr> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>
##  1  3654 chic  1     1     1997    137  36       37.5  13.1   5.66  3654 winter
##  2  3655 chic  1     2     1997    123  45       47.2  41.9   5.53  3655 winter
##  3  3656 chic  1     3     1997    127  40       38    27.0   6.29  3656 winter
##  4  3657 chic  1     4     1997    146  51.5     45.5  25.1   7.54  3657 winter
##  5  3658 chic  1     5     1997    102  27       11.2  15.3  20.8   3658 winter
##  6  3659 chic  1     6     1997    127  17        5.75  9.36 14.9   3659 winter
##  7  3660 chic  1     7     1997    116  16        7    20.2  11.9   3660 winter
##  8  3661 chic  1     8     1997    118  19       17.8  33.1   8.68  3661 winter
##  9  3662 chic  1     9     1997    148  26       24    12.1  13.4   3662 winter
## 10  3663 chic  1     10    1997    121  16        5.38 24.8  10.4   3663 winter
## # ... with 1,451 more rows, and 1 more variable: year <dbl>
```