Tidyverse project

Jennifer Brosnahan

6/25/2020

This is my attempt to learn more about R's Tidyverse data wrangling functionality. I am experimenting throughout and learning that tidyr is a way to stay organized with tabular data. It also contains the following packages:

1. readr: data import

2. tidyr: data tidying

3. tibble: modern re-imagining of data frames

4. dplyr: data manipulation

5. stringr: strings

6. ggplot2: data visualization

7. purrr: functional programming

8. forcats: for dealing factors

load libraries, openxlsx is for writing excel files!

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------ tidyverse 1
```

```
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     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_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readxl)
library(openxlsx)
```

writing new df

```
write_file(x = 'a,b,c\n1,2,3\n4,5,NA', path = 'file.csv')
```

reading file back into our environment

```
tibble_1 <- read_csv('file.csv')
```

```
## Parsed with column specification:
## cols(
##   a = col_double(),
##   b = col_double(),
##   c = col_double()
## )
```

```
tibble_1
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5    NA
```

reading excel file - Titanic

```
library(titanic)

titanic <- read.csv(file.path('C:/Users/jlbro/OneDrive/Datasets', 'titanic.csv'))
head(titanic)
```

```
##   pclass survived                                            name    sex
## 1      1        1                   Allen, Miss. Elisabeth Walton female
## 2      1        1                  Allison, Master. Hudson Trevor   male
## 3      1        0                    Allison, Miss. Helen Loraine female
## 4      1        0            Allison, Mr. Hudson Joshua Creighton   male
## 5      1        0 Allison, Mrs. Hudson J C (Bessie Waldo Daniels) female
## 6      1        1                             Anderson, Mr. Harry   male
##       age sibsp parch ticket     fare   cabin embarked boat body
## 1 29.0000     0     0  24160 211.3375      B5        S    2   NA
## 2  0.9167     1     2 113781 151.5500 C22 C26        S   11   NA
## 3  2.0000     1     2 113781 151.5500 C22 C26        S        NA
## 4 30.0000     1     2 113781 151.5500 C22 C26        S       135
## 5 25.0000     1     2 113781 151.5500 C22 C26        S        NA
## 6 48.0000     0     0  19952  26.5500     E12        S    3   NA
##                         home.dest
## 1                    St Louis, MO
## 2 Montreal, PQ / Chesterville, ON
## 3 Montreal, PQ / Chesterville, ON
## 4 Montreal, PQ / Chesterville, ON
## 5 Montreal, PQ / Chesterville, ON
## 6                    New York, NY
```

Practice parsing, or labeling each column as a specific datatype

```
write_file(x = 'a,b,c,d\n1,T,3,dog\n4,FALSE,NA,cat\n6,F,5,mouse\n18,TRUE,3,moose', path= 'file2.csv')
read_csv('file2.csv')
```

```
## Parsed with column specification:
## cols(
##   a = col_double(),
##   b = col_logical(),
##   c = col_double(),
##   d = col_character()
## )
```

```
## # A tibble: 4 x 4
##       a b         c d
##   <dbl> <lgl> <dbl> <chr>
## 1     1 TRUE      3 dog
## 2     4 FALSE    NA cat
## 3     6 FALSE     5 mouse
## 4    18 TRUE      3 moose
```

Parsing manually using col_types = argument to specify columns as factors

```
x <- read_csv('file2.csv', col_types = cols(a = col_integer(), b = col_logical(), c = col_integer(), d 
x
```

```
## # A tibble: 4 x 4
##       a b         c d
##   <int> <lgl> <int> <fct>
## 1     1 TRUE      3 dog
## 2     4 FALSE    NA cat
## 3     6 FALSE     5 mouse
## 4    18 TRUE      3 moose
```

NOTE when parsing as FACTOR: it will force you to specify whether the levels are ordered or not, thus ordered = FALSE

Tibbles - a df with more enforcements

```
y <- tibble(a = c(1,4,6,18), b = c(T,FALSE,F,TRUE), c = c(3, NA, 5, 3), d = c('dog','cat','mouse','moos
y
```

```
## # A tibble: 4 x 4
##       a b         c d
##   <dbl> <lgl> <dbl> <chr>
## 1     1 TRUE      3 dog
## 2     4 FALSE    NA cat
## 3     6 FALSE     5 mouse
## 4    18 TRUE      3 moose
```

```
class(y)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Notice R view the tibble as a data.frame, tbl, and tbl_df

Convert df to a tibble

```
df <- data.frame(a = c(1,4,6,18), b = c(T,FALSE,F,TRUE), c = c(3, NA, 5, 3), d = c('dog','cat','mouse',
df
```

```
##    a     b  c     d
## 1  1  TRUE  3   dog
## 2  4 FALSE NA   cat
## 3  6 FALSE  5 mouse
## 4 18  TRUE  3 moose
```

```
class(df)
```

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

Tutorial says this one doesn’t list the dimensions of the table and doesn’t specify the datatypes of each column, however it does.

Converting df to tibble

```
df <- as_tibble(df)
class(df)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

creating table4a from Tibble cheat sheet

```
table4a <- tibble(country = c('A','B','C'), '1999' = c('0.7K','37K','212K'), '2000' = c('2K','80K','213
table4a
```

```
## # A tibble: 3 x 3
##   country ‘1999‘ ‘2000‘
##   <chr>   <chr>  <chr>
## 1 A       0.7K   2K
## 2 B       37K    80K
## 3 C       212K   213K
```

making ‘year’ a variable using GATHER

```
table <- gather(table4a, '1999','2000', key = 'year', value = 'cases')
table
```

```
## # A tibble: 6 x 3
##   country year  cases
##   <chr>   <chr> <chr>
## 1 A       1999  0.7K
## 2 B       1999  37K
## 3 C       1999  212K
## 4 A       2000  2K
## 5 B       2000  80K
## 6 C       2000  213K
```

Allows you to use the year as a factor that you can filter by

doing the exact opposite with the spread() function

```
spread(table, year, cases)
```

```
## # A tibble: 3 x 3
##   country ‘1999‘ ‘2000‘
##   <chr>   <chr>  <chr>
## 1 A       0.7K   2K
## 2 B       37K    80K
## 3 C       212K   213K
```

Handling missing data

```
table <- tibble(x1= c('A','B','C','D','E'), x2 = c(1,NA,NA,3,NA))
table
```

```
## # A tibble: 5 x 2
##   x1       x2
##   <chr> <dbl>
## 1 A         1
## 2 B        NA
## 3 C        NA
## 4 D         3
## 5 E        NA
```

Removing rows containing NA’s with drop_na(data)

```
drop_na(table)
```

```
## # A tibble: 2 x 2
##   x1       x2
##   <chr> <dbl>
## 1 A         1
## 2 D         3
```

Fill in NA’s with the column’s most recent non-NA value with fill(data, . . . , .direction = c(‘down’,‘up’))

direction always defaults to UP, the data above the NA

```
fill(table, x2)
```

```
## # A tibble: 5 x 2
##   x1       x2
##   <chr> <dbl>
## 1 A         1
## 2 B         1
## 3 C         1
## 4 D         3
## 5 E         3
```

Replace NA’s by the column with a specific value, with replace_na(data, replace = list(), . . .)

```
replace_na(table, replace = list(x2 = 2))
```

```
## # A tibble: 5 x 2
##   x1       x2
##   <chr> <dbl>
## 1 A         1
## 2 B         2
## 3 C         2
## 4 D         3
## 5 E         2
```

Expand tables, by splitting cells

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>       <int> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

We want to split rate by numerator and denominator

```r
separate(table3, rate, into = c('numerator', 'denominator'), sep = '[^[:alnum:]]+')
```

```
## # A tibble: 6 x 4
##   country      year numerator denominator
##   <chr>       <int> <chr>     <chr>
## 1 Afghanistan  1999 745       19987071
## 2 Afghanistan  2000 2666      20595360
## 3 Brazil       1999 37737     172006362
## 4 Brazil       2000 80488     174504898
## 5 China        1999 212258    1272915272
## 6 China        2000 213766    1280428583
```

Separated correctly, although into characters

now separating specifying the separator = LESS HASSLE

```r
table3 <- separate(table3, rate, into = c('numerator', 'denominator'), sep = '/')
table3
```

```
## # A tibble: 6 x 4
##   country      year numerator denominator
##   <chr>       <int> <chr>     <chr>
## 1 Afghanistan  1999 745       19987071
## 2 Afghanistan  2000 2666      20595360
## 3 Brazil       1999 37737     172006362
## 4 Brazil       2000 80488     174504898
## 5 China        1999 212258    1272915272
## 6 China        2000 213766    1280428583
```

Using unite() to make rate a ratio with ':' for a separator

```r
unite(table3, numerator, denominator, col = rate, sep = ':')
```

```
## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <int> <chr>
## 1 Afghanistan  1999 745:19987071
## 2 Afghanistan  2000 2666:20595360
## 3 Brazil       1999 37737:172006362
## 4 Brazil       2000 80488:174504898
## 5 China        1999 212258:1272915272
## 6 China        2000 213766:1280428583
```

Note, parse_function is not working for me

Dplyr

Starting with pipes, %>%, simply pushes data from whatever is before it to the function that is after it

Getting # of rows of mtcars

```
mtcars %>% nrow()
```

```
## [1] 32
```

Summarize cases

```
mtcars %>% summarise(mpg_avg = mean(mpg), mpg_median = median(mpg), mpg_ndistinct = n_distinct(mpg), hp
```

```
##    mpg_avg mpg_median mpg_ndistinct   hp_avg hp_median hp_ndistinct
## 1 20.09062       19.2            25 146.6875       123           22
```

Groupby cases

```
mtcars %>% group_by(cyl) %>%
  summarise(mean(hp), mean(mpg))
```

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

```
## # A tibble: 3 x 3
##     cyl ‘mean(hp)‘ ‘mean(mpg)‘
##   <dbl>      <dbl>       <dbl>
## 1     4       82.6        26.7
## 2     6      122.         19.7
## 3     8      209.         15.1
```

count() how many cars of each group there are

```
mtcars %>% group_by(cyl) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   cyl [3]
##     cyl     n
##   <dbl> <int>
## 1     4    11
## 2     6     7
## 3     8    14
```

Manipulating cases! FILTERING!

```
mtcars %>% filter(cyl >= 6 & hp < 150)
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
```

```
mtcars %>% filter(mpg > 25 & cyl < 6)
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

distinct() gives distinct or unique values for variable you select

```
mtcars %>% distinct(gear)
```

```
##                gear
## Mazda RX4         4
## Hornet 4 Drive    3
## Porsche 914-2     5
```

Now selecting multiple columns

```
mtcars %>% distinct(gear, hp)
```

```
##                      hp gear
## Mazda RX4           110    4
## Datsun 710           93    4
## Hornet 4 Drive      110    3
## Hornet Sportabout   175    3
## Valiant             105    3
## Duster 360          245    3
## Merc 240D            62    4
## Merc 230             95    4
## Merc 280            123    4
## Merc 450SE          180    3
## Cadillac Fleetwood  205    3
## Lincoln Continental 215    3
## Chrysler Imperial   230    3
## Fiat 128             66    4
## Honda Civic          52    4
## Toyota Corolla       65    4
## Toyota Corona        97    3
## Dodge Challenger    150    3
## Porsche 914-2        91    5
## Lotus Europa        113    5
## Ford Pantera L      264    5
## Ferrari Dino        175    5
## Maserati Bora       335    5
## Volvo 142E          109    4
```

```
mtcars %>% distinct(gear, hp) %>%
  count()
```

```
##    n
## 1 24
```

arrange() to arrange in an order, this time by displacement

Note it will automatically arrange table in ascending order unless you specify DESCENDING, desc()

```
mtcars %>% top_n(10, hp) %>%
  arrange(desc(disp))
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
```

adding rows using add_row()

```
mtcars %>% top_n(10, hp) %>%
  arrange(desc(disp)) %>%
  add_row(mpg = 56, cyl = 4, disp = 260, hp = 900)
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## ...11               56.0   4 260.0 900   NA    NA    NA NA NA   NA   NA
```

Manipulate using select() = selects only the columns that you choose. Output will be tibble with those selected variables

```
mtcars %>%
  select(qsec, hp) %>%
  head()
```

```
##                    qsec  hp
## Mazda RX4         16.46 110
## Mazda RX4 Wag     17.02 110
## Datsun 710        18.61  93
## Hornet 4 Drive    19.44 110
## Hornet Sportabout 17.02 175
## Valiant           20.22 105
```

deselecting columns

```
mtcars %>%
  select(-qsec, -hp) %>%
  head()
```

```
##                    mpg cyl disp drat    wt vs am gear carb
## Mazda RX4         21.0   6  160 3.90 2.620  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 3.90 2.875  0  1    4    4
## Datsun 710        22.8   4  108 3.85 2.320  1  1    4    1
## Hornet 4 Drive    21.4   6  258 3.08 3.215  1  0    3    1
## Hornet Sportabout 18.7   8  360 3.15 3.440  0  0    3    2
## Valiant           18.1   6  225 2.76 3.460  1  0    3    1
```

Computing new columns using MUTATE!!!!

```
mtcars %>% mutate(gpm = 1/mpg) %>%
head()
```

```
##    mpg cyl disp  hp drat    wt  qsec vs am gear carb        gpm
## 1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 0.04761905
## 2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 0.04761905
## 3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 0.04385965
## 4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 0.04672897
## 5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 0.05347594
## 6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 0.05524862
```

Now adding new column using add_column, first selecting only a few columns

```
mtcars2 <- mtcars %>% select(disp, hp, qsec) %>%
  add_column(engine_size = NA)
head(mtcars2)
```

```
##                    disp  hp  qsec engine_size
## Mazda RX4           160 110 16.46          NA
## Mazda RX4 Wag       160 110 17.02          NA
## Datsun 710          108  93 18.61          NA
## Hornet 4 Drive      258 110 19.44          NA
## Hornet Sportabout   360 175 17.02          NA
## Valiant             225 105 20.22          NA
```

Now subsetting out the ones that fit in each category

```
mtcars2$engine_size[mtcars2$disp <= 120.8] <- 'small'
mtcars2$engine_size[mtcars2$disp > 120.8 & mtcars2$disp <= 326] <- 'medium'
mtcars2$engine_size[mtcars2$disp > 326] <- 'large'
mtcars2 %>% head()
```

```
##                    disp  hp  qsec engine_size
## Mazda RX4           160 110 16.46      medium
## Mazda RX4 Wag       160 110 17.02      medium
## Datsun 710          108  93 18.61       small
## Hornet 4 Drive      258 110 19.44      medium
## Hornet Sportabout   360 175 17.02       large
## Valiant             225 105 20.22      medium
```

Vector functions.

The column of a table is functionally the same thing as a VECTOR!

cumsum() adds up the cumulative sum of a column

```
mtcars2 %>% mutate(cum_displacement = cumsum(disp)) %>%
  head()
```

```
##   disp  hp  qsec engine_size cum_displacement
## 1  160 110 16.46      medium              160
## 2  160 110 17.02      medium              320
## 3  108  93 18.61       small              428
## 4  258 110 19.44      medium              686
## 5  360 175 17.02       large             1046
## 6  225 105 20.22      medium             1271
```

Reordering table

```
mtcars2 %>% arrange(desc(hp)) %>%
  mutate(cum_dispacement = cumsum(disp)) %>%
  head()
```

```
##   disp  hp  qsec engine_size cum_dispacement
## 1  301 335 14.60      medium             301
## 2  351 264 14.50       large             652
## 3  360 245 15.84       large            1012
## 4  350 245 15.41       large            1362
## 5  440 230 17.42       large            1802
## 6  460 215 17.82       large            2262
```

Using min_rank

For the race we want low 'mpg', high 'hp', low 'disp', low 'qsec'

```
# first selecting only variables of interest

mtcars3 <- mtcars %>% select(mpg, hp, disp, qsec) %>%
  mutate(mpg_rank = min_rank(desc(mpg)), hp_rank = min_rank(desc(hp)), qsec_rank = min_rank(qsec), disp
  mutate(total_rank = (mpg_rank + hp_rank + qsec_rank + disp_rank)) %>%

arrange(total_rank) %>%

# now putting cumulative displacement back in there
  mutate(cum_displacement = cumsum(disp))
mtcars3
```

```
##     mpg  hp  disp  qsec mpg_rank hp_rank qsec_rank disp_rank total_rank
## 1  30.4 113  95.1 16.90        3      18         9         5         35
## 2  19.7 175 145.0 15.50       15      11         4        11         41
## 3  26.0  91 120.3 16.70        6      27         7         8         48
## 4  21.0 110 160.0 16.46       13      19         6        13         51
## 5  15.8 264 351.0 14.50       23       2         1        26         52
## 6  15.0 335 301.0 14.60       27       1         2        22         52
## 7  21.0 110 160.0 17.02       13      19        10        13         55
## 8  30.4  52  75.7 18.52        3      32        21         2         58
## 9  32.4  66  78.7 19.47        2      28        27         3         60
## 10 33.9  65  71.1 19.90        1      30        28         1         60
## 11 13.3 245 350.0 15.41       30       3         3        25         61
## 12 27.3  66  79.0 18.90        5      28        24         4         61
## 13 22.8  93 108.0 18.61        8      26        23         6         63
## 14 16.4 180 275.8 17.40       22       8        14        19         63
## 15 14.3 245 360.0 15.84       29       3         5        27         64
## 16 17.3 180 275.8 17.60       21       8        16        19         64
## 17 21.4 109 121.0 18.60       11      22        22         9         64
## 18 18.7 175 360.0 17.02       18      11        10        27         66
## 19 19.2 123 167.6 18.30       16      16        20        15         67
## 20 19.2 175 400.0 17.05       16      11        12        29         68
## 21 15.5 150 318.0 16.87       24      14         8        24         70
## 22 15.2 180 275.8 18.00       25       8        19        19         71
## 23 21.5  97 120.1 20.01       10      24        30         7         71
## 24 21.4 110 258.0 19.44       11      19        26        18         74
## 25 22.8  95 140.8 22.90        8      25        32        10         75
## 26 17.8 123 167.6 18.90       20      16        24        15         75
## 27 15.2 150 304.0 17.30       25      14        13        23         75
## 28 14.7 230 440.0 17.42       28       5        15        30         78
## 29 24.4  62 146.7 20.00        7      31        29        12         79
## 30 10.4 215 460.0 17.82       31       6        17        31         85
## 31 10.4 205 472.0 17.98       31       7        18        32         88
## 32 18.1 105 225.0 20.22       19      23        31        17         90
##    cum_displacement
## 1              95.1
## 2             240.1
## 3             360.4
## 4             520.4
## 5             871.4
## 6            1172.4
## 7            1332.4
## 8            1408.1
## 9            1486.8
## 10           1557.9
## 11           1907.9
## 12           1986.9
## 13           2094.9
## 14           2370.7
## 15           2730.7
## 16           3006.5
## 17           3127.5
## 18           3487.5
## 19           3655.1
## 20           4055.1
## 21           4373.1
## 22           4648.9
## 23           4769.0
## 24           5027.0
## 25           5167.8
## 26           5335.4
## 27           5639.4
## 28           6079.4
## 29           6226.1
## 30           6686.1
## 31           7158.1
## 32           7383.1
```

if_else()

Now label the cars good or bad based on their total rank. Mean total_rank is 65.125, so above is bad, below is good

```
mtcars4 <- mtcars3 %>% mutate(good_bad = if_else(total_rank < 65.125, 'good', 'bad'))
mtcars4 %>% head()
```

```
##    mpg  hp  disp  qsec mpg_rank hp_rank qsec_rank disp_rank total_rank
## 1 30.4 113  95.1 16.90        3      18         9         5         35
## 2 19.7 175 145.0 15.50       15      11         4        11         41
## 3 26.0  91 120.3 16.70        6      27         7         8         48
## 4 21.0 110 160.0 16.46       13      19         6        13         51
## 5 15.8 264 351.0 14.50       23       2         1        26         52
## 6 15.0 335 301.0 14.60       27       1         2        22         52
##   cum_displacement good_bad
## 1             95.1     good
## 2            240.1     good
## 3            360.4     good
## 4            520.4     good
## 5            871.4     good
## 6           1172.4     good
```

More ways to use summarise() and group_by() functions

Comparing good/bad cars by 3 different things:

‘qsec’ avg quarter mile; max mpg; variance of displacement

```
# first use group_by function
mtcars4 %>% group_by(good_bad) %>%
  summarise(mean_qsec = mean(qsec), max_mpg = max(mpg), disp_variance = var(disp))
```

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

```
## # A tibble: 2 x 4
##   good_bad mean_qsec max_mpg disp_variance
##   <chr>        <dbl>   <dbl>         <dbl>
## 1 bad           18.6    24.4        14928.
## 2 good          17.2    33.9        11746.
```

Changing row names_to_columns

If you import a dataset which has the index in the first column, or actual data in the row_names instead of in the first column

```
mtcars %>% head()
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Notice names of cars are the row names rather than the first column in table. Put them in column as their own variable.

```
rownames_to_column(mtcars, var = 'car_model') %>% head()
```

```
##           car_model  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1         Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2     Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3        Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4    Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5 Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6           Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Combining Tables!!!

option 1: bind_cols() - pretty rare

```
mtcars1 <- rownames_to_column(mtcars, var = 'car_model') %>%
  select(car_model, mpg, cyl, disp)
mtcars2 <- rownames_to_column(mtcars, var = 'car_model') %>%
  select(car_model, hp, drat, wt, qsec)
mtcars1 %>% head()
```

```
##           car_model  mpg cyl disp
## 1         Mazda RX4 21.0   6  160
## 2     Mazda RX4 Wag 21.0   6  160
## 3        Datsun 710 22.8   4  108
## 4    Hornet 4 Drive 21.4   6  258
## 5 Hornet Sportabout 18.7   8  360
## 6           Valiant 18.1   6  225
```

```
mtcars2 %>% head()
```

```
##           car_model  hp drat    wt  qsec
## 1         Mazda RX4 110 3.90 2.620 16.46
## 2     Mazda RX4 Wag 110 3.90 2.875 17.02
## 3        Datsun 710  93 3.85 2.320 18.61
## 4    Hornet 4 Drive 110 3.08 3.215 19.44
## 5 Hornet Sportabout 175 3.15 3.440 17.02
## 6           Valiant 105 2.76 3.460 20.22
```

Binding back together, always make sure df matches row-wise

```r
mtcars3 <- bind_cols(mtcars1, mtcars2) %>% head()
```

```
## New names:
## * car_model -> car_model...1
## * car_model -> car_model...5
```

```r
mtcars3
```

```
##        car_model...1  mpg cyl disp     car_model...5  hp drat    wt  qsec
## 1          Mazda RX4 21.0   6  160         Mazda RX4 110 3.90 2.620 16.46
## 2      Mazda RX4 Wag 21.0   6  160     Mazda RX4 Wag 110 3.90 2.875 17.02
## 3         Datsun 710 22.8   4  108        Datsun 710  93 3.85 2.320 18.61
## 4     Hornet 4 Drive 21.4   6  258    Hornet 4 Drive 110 3.08 3.215 19.44
## 5  Hornet Sportabout 18.7   8  360 Hornet Sportabout 175 3.15 3.440 17.02
## 6            Valiant 18.1   6  225           Valiant 105 2.76 3.460 20.22
```

Notice how we have two columns for car_model. It added column # after so no two columns have same name.

deselect car_model...5 using bind_rows

```r
mtcars4 <- bind_rows(mtcars3, mtcars3)
mtcars4
```

```
##         car_model...1  mpg cyl disp     car_model...5  hp drat    wt  qsec
## 1           Mazda RX4 21.0   6  160         Mazda RX4 110 3.90 2.620 16.46
## 2       Mazda RX4 Wag 21.0   6  160     Mazda RX4 Wag 110 3.90 2.875 17.02
## 3          Datsun 710 22.8   4  108        Datsun 710  93 3.85 2.320 18.61
## 4      Hornet 4 Drive 21.4   6  258    Hornet 4 Drive 110 3.08 3.215 19.44
## 5   Hornet Sportabout 18.7   8  360 Hornet Sportabout 175 3.15 3.440 17.02
## 6             Valiant 18.1   6  225           Valiant 105 2.76 3.460 20.22
## 7           Mazda RX4 21.0   6  160         Mazda RX4 110 3.90 2.620 16.46
## 8       Mazda RX4 Wag 21.0   6  160     Mazda RX4 Wag 110 3.90 2.875 17.02
## 9          Datsun 710 22.8   4  108        Datsun 710  93 3.85 2.320 18.61
## 10     Hornet 4 Drive 21.4   6  258    Hornet 4 Drive 110 3.08 3.215 19.44
## 11  Hornet Sportabout 18.7   8  360 Hornet Sportabout 175 3.15 3.440 17.02
## 12            Valiant 18.1   6  225           Valiant 105 2.76 3.460 20.22
```

left_join() = every row of the left (first listed) dataframe will be accounted for no matter what.

```r
mtcars1
```

```
##               car_model  mpg cyl  disp
## 1             Mazda RX4 21.0   6 160.0
## 2         Mazda RX4 Wag 21.0   6 160.0
## 3           Datsun 710 22.8   4 108.0
## 4       Hornet 4 Drive 21.4   6 258.0
## 5    Hornet Sportabout 18.7   8 360.0
## 6              Valiant 18.1   6 225.0
## 7           Duster 360 14.3   8 360.0
## 8            Merc 240D 24.4   4 146.7
## 9             Merc 230 22.8   4 140.8
## 10            Merc 280 19.2   6 167.6
## 11           Merc 280C 17.8   6 167.6
## 12          Merc 450SE 16.4   8 275.8
## 13          Merc 450SL 17.3   8 275.8
## 14         Merc 450SLC 15.2   8 275.8
## 15  Cadillac Fleetwood 10.4   8 472.0
## 16 Lincoln Continental 10.4   8 460.0
## 17   Chrysler Imperial 14.7   8 440.0
## 18            Fiat 128 32.4   4  78.7
## 19         Honda Civic 30.4   4  75.7
## 20      Toyota Corolla 33.9   4  71.1
## 21       Toyota Corona 21.5   4 120.1
## 22    Dodge Challenger 15.5   8 318.0
## 23         AMC Javelin 15.2   8 304.0
## 24          Camaro Z28 13.3   8 350.0
## 25    Pontiac Firebird 19.2   8 400.0
## 26           Fiat X1-9 27.3   4  79.0
## 27       Porsche 914-2 26.0   4 120.3
## 28        Lotus Europa 30.4   4  95.1
## 29      Ford Pantera L 15.8   8 351.0
## 30        Ferrari Dino 19.7   6 145.0
## 31       Maserati Bora 15.0   8 301.0
## 32          Volvo 142E 21.4   4 121.0
```

Pulling top 10 cars with best mpg in a dataset

```r
mtcars1 <- mtcars1 %>% top_n(10, mpg)
mtcars1
```

```
##         car_model  mpg cyl  disp
## 1      Datsun 710 22.8   4 108.0
## 2       Merc 240D 24.4   4 146.7
## 3        Merc 230 22.8   4 140.8
## 4        Fiat 128 32.4   4  78.7
## 5     Honda Civic 30.4   4  75.7
## 6  Toyota Corolla 33.9   4  71.1
## 7   Toyota Corona 21.5   4 120.1
## 8       Fiat X1-9 27.3   4  79.0
## 9   Porsche 914-2 26.0   4 120.3
## 10   Lotus Europa 30.4   4  95.1
```

These are top 10 cars with best gas mileage. Now we want to get the rest of the info held in mtcars2 (hp, drat, wt, qsec)

Must join tables together by common variable unique to every row - car_model

```r
left_join(mtcars1, mtcars2, by = 'car_model')
```

```
##         car_model  mpg cyl  disp  hp drat    wt  qsec
## 1      Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61
## 2       Merc 240D 24.4   4 146.7  62 3.69 3.190 20.00
## 3        Merc 230 22.8   4 140.8  95 3.92 3.150 22.90
## 4        Fiat 128 32.4   4  78.7  66 4.08 2.200 19.47
## 5     Honda Civic 30.4   4  75.7  52 4.93 1.615 18.52
## 6  Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90
## 7   Toyota Corona 21.5   4 120.1  97 3.70 2.465 20.01
## 8       Fiat X1-9 27.3   4  79.0  66 4.08 1.935 18.90
## 9   Porsche 914-2 26.0   4 120.3  91 4.43 2.140 16.70
## 10   Lotus Europa 30.4   4  95.1 113 3.77 1.513 16.90
```

```r
tinytex:::is_tinytex()
```

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