tidyverse intro

Jeremy Mikecz

Introduction to Tidyverse

Tidyverse is a system of R packages for data wrangling and analysis. It provides a different method and syntax for working with data tables (often called "data frames" in data science; but, tidyverse dataframes are known as "tibbles") from base R.

For additional help learning how to use R's tidyverse system of packages, see:

- Wickham, Çetinkaya-Rundel, and Grolemund, *R for Data Science*
- Grolemund, A Tidyverse Cookbook (2020).!
- package documentation: tidyverse
- tidyr cheatsheet
- *Tidyverse style guide*

For a more detailed introduction to working with tidy data in R see:

- Wickham, Çetinkaya-Rundel, and Grolemund, *R for Data Science*, Ch. 5 "Data Tidying".
- Grolemund, A Tidyverse Cookbook (2020).
- Silge and Robinson, Text Mining with R, Ch. 1 "The tidy text format".

Getting Started with Tidyverse

Data analysis usually involves working with two-dimensional datasets known as dataframes. There are different ways to work with dataframes in R, these include:

- using core R dataframe functions
- using the **tibble** a type of dataframe used with the tidyverse collection of packages
- using other dataframe types like **data.table** for speed and to facilitate working with large datasets (i.e. with millions of rows)

In these lessons, we will work with tidyverse a collection of packages designed for data science. Tidyverse works with **tibbles**, a customized and newer form of dataframes. For more on the differences between tibbles and dataframes see the explanation here. Key packages in tidyverse include (borrowing from Kyle Walker's **Analyzing US Census Data: Methods, Maps, and Models in R**):

* readr (Wickham and Hester 2021), which contains tools for importing and exporting data

* dplyr (Wickham et al. 2021), a powerful framework for data wrangling tasks;

* tidyr (Wickham 2021b), a package for reshaping data;

* purrr (Henry and Wickham 2020), a comprehensive framework for functional programming a

* ggplot2 (Wickham 2016), a data visualization package based on the Grammar of Graphics

```
## install tidyverse with:
#install.packages("tidyverse")

## update tidyverse with:
#tidyverse_update()

## import tidyverse with:
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.4.3

Warning: package 'tibble' was built under R version 4.4.3

Warning: package 'tidyr' was built under R version 4.4.3

Warning: package 'readr' was built under R version 4.4.3

Warning: package 'dplyr' was built under R version 4.4.3

Warning: package 'stringr' was built under R version 4.4.3

Warning: package 'forcats' was built under R version 4.4.3

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
## see what packages are included with tidyverse:
tidyverse_packages()
```

```
 [1] "broom"         "conflicted"    "cli"           "dbplyr"
 [5] "dplyr"         "dtplyr"        "forcats"       "ggplot2"
 [9] "googledrive"   "googlesheets4" "haven"         "hms"
[13] "httr"          "jsonlite"      "lubridate"     "magrittr"
[17] "modelr"        "pillar"        "purrr"         "ragg"
[21] "readr"         "readxl"        "reprex"        "rlang"
[25] "rstudioapi"    "rvest"         "stringr"       "tibble"
[29] "tidyr"         "xml2"          "tidyverse"
```

Combine functions into a pipe

Using the symbol `|>` we can chain multiple functions together.

Previously, `%>%` was commonly used for pipes. Now, it is recommended to use `|>` instead. In R Studio, you can use the keyboard shortcut `CTRL/CMD + SHIFT + M` to create a pipe. However, you can update R Studio's default pipe using the instructions in *R for Data Science*, Ch. 3.4.

Each new function in the pipeline operates on the results produced by the previous function.

Chaining functions together in a pipe like this:

```
starwars |>
  group_by(species) |>
  summarise(avg_height = mean(height, na.rm = TRUE)) |>
  arrange(avg_height)
```

```
starwars |>
  group_by(species) |>
  summarise(avg_height = mean(height, na.rm = TRUE)) |>
  arrange(avg_height)
```

```
# A tibble: 38 x 2
   species        avg_height
   <chr>               <dbl>
 1 Yoda's species         66
 2 Aleena                 79
 3 Ewok                   88
 4 Vulptereen             94
 5 Dug                   112
 6 Xexto                 122
 7 Droid                 131.
 8 Toydarian             137
```

```
 9 Sullustan            160
10 Toong                163
# i 28 more rows
```

produces the same results as nesting a series of functions within another:

```
arrange(
  summarise(
    group_by(starwars, species),
    avg_height = mean(height, na.rm = TRUE)
  ),
  avg_height
)
```

or calling each function in order:

```
x1 <- starwars
x2 <- group_by(x1, species)
x3 <- summarise(x3, avg_height = mean(height, na.rm = TRUE))
arrange(x3, avg_height)
```

By default `|>` passes the result of the left hand side to the the first unnamed argument of the function on the right hand side. To override this default, use `_` as a placeholder within the function call on the right hand side. `|>` will evaluate `_` as the result of the left hand side, instead of passing the result to the first unnamed argument.

```
starwars |>
  lm(mass ~ height, data = _)
```

```
Call:
lm(formula = mass ~ height, data = starwars)

Coefficients:
(Intercept)       height
    -11.487        0.624
```

```
# the old piping syntax
#starwars %>%
#  lm(mass ~ height, data = .)
```

Tidy Data

From the Tidyverse cookbook:

Data tidying refers to reshaping your data into a tidy data frame or tibble.

Data tidying is an important first step for your analysis because every tidyverse function will expect your data to be stored as **Tidy Data**.

Tidy data is tabular data organized so that:

1. Each column contains a single variable
2. Each row contains a single observation

Tidy data is not an arbitrary requirement of the tidyverse; it is the ideal data format for doing data science with R. Tidy data makes it easy to extract every value of a variable to build a plot or to compute a summary statistic. Tidy data also makes it easy to compute new variables; when your data is tidy, you can rely on R's rowwise operations to maintain the integrity of your observations. Moreover, R can directly manipulate tidy data with R's fast, built-in vectorised observations, which lets your code run as fast as possible.

The definition of Tidy Data isn't complete until you define variable and observation, so let's borrow two definitions from *R for Data Science*:

1. A **variable** is a quantity, quality, or property that you can measure.
2. An **observation** is a set of measurements made under similar conditions (you usually make all of the measurements in an observation at the same time and on the same object).

As you work with data, you will be surprised to realize that what is a variable (or observation) will depend less on the data itself and more on what you are trying to do with it. With enough mental flexibility, you can consider anything to be a variable. However, some variables will be more useful than others for any specific task. In general, if you can formulate your task as an equation (math or code that contains an equals sign), the most useful variables will be the names in the equation.

Create a tibble from scratch

Rarely will you ever create a dataframe or tibble from scratch except to create small practice datasets.

Here, we create a small practice dataset about the popular murder mystery TV show, *Only Murders in the Building*

```
omitb <- tribble(~name, ~occupation, ~apartment, ~is_suspect, ~is_dead,
            "Charles", "washed-up actor", "14C", "no", "no",
            "Mabel", "unemployed never-was", "12E", "maybe", "no",
            "Oliver", "theater director", "10D", "no", "no",
            "Howard", "childless cat guy", "3D", "yes", "no",
            "Bunny", "petty despot", "12A", "no", "maybe"
)
omitb
```

```
# A tibble: 5 x 5
  name    occupation           apartment is_suspect is_dead
  <chr>   <chr>                <chr>     <chr>      <chr>
1 Charles washed-up actor      14C       no         no
2 Mabel   unemployed never-was 12E       maybe      no
3 Oliver  theater director     10D       no         no
4 Howard  childless cat guy    3D        yes        no
5 Bunny   petty despot         12A       no         maybe
```

We can add a new row by piping the function **add_row.** To figure out how do so, we can review the function's documentation. Note: the default is to add the new row to the bottom of the tibble. To add it before the first row we need to set the argument `.before = 1`.

```
omitb <- omitb |>
  add_row(name="Jan", occupation = "2nd-chair basoonist",
          is_suspect="yes", is_dead="no", .before=1)

omitb
```

```
# A tibble: 6 x 5
  name    occupation           apartment is_suspect is_dead
  <chr>   <chr>                <chr>     <chr>      <chr>
1 Jan     2nd-chair basoonist  <NA>      yes        no
2 Charles washed-up actor      14C       no         no
3 Mabel   unemployed never-was 12E       maybe      no
4 Oliver  theater director     10D       no         no
5 Howard  childless cat guy    3D        yes        no
6 Bunny   petty despot         12A       no         maybe
```

Above we used the **tribble** function to create a **tibble** (tribble vs. tibble - confusing right?) by using a syntax that lines up the rows neatly in a way that will preview the result.

You can also create the same tibble using the **tibble** function and passing in a list with the values of column:

```
tibble(name = c("Charles", "Mabel", "Oliver", "Howard", "Bunny"),
       occupation = c("washed-up actor", "unemployed never-was", "theater director", "childl
       apartment = c("14C", "12E", "10D", "3D", "12A"),
       is_suspect = c("no", "no", "no", "yes", "no"),
       is_dead = c("no", "no", "no", "no", "maybe"),
       )
```

```
# A tibble: 5 x 5
  name    occupation           apartment is_suspect is_dead
  <chr>   <chr>                <chr>     <chr>      <chr>
1 Charles washed-up actor      14C       no         no
2 Mabel   unemployed never-was 12E       no         no
```

```
3 Oliver  theater director     10D       no         no
4 Howard  childless cat guy    3D        yes        no
5 Bunny   petty despot         12A       no         maybe
```

Dataframes vs. tibbles

```
df <- read.csv("../data/census1970.csv")
```

Convert a dataframe into a tibble

```
head(df)
```

```
  rownum STATE COUNTY              NAME TOTPOP   WPOP NEGTOT OTHRACES   MTOT
1      1     1     10 FAIRFIELD          792814 732304  56408     4102 381603
2      2     1     30 HARTFORD           816737 758086  54645     4006 394728
3      3     1     50 LITCHFIELD         144091 142649   1126      316  70176
4      4     1     70 MIDDLESEX          114816 111102   3327      387  56465
5      5     1     90 NEW HAVEN          744948 684743  56630     3575 359204
6      6     1    110 NEW LONDON         230348 221073   7390     1885 115752
    M04   M56   M79 M1013  M14  M15 M1617 M1819  M20  M21 M2224 M2534 M3544
1 32152 15536 24082 33345 8129 8059 15054 11245 5006 4485 13869 44111 48524
2 34432 15927 24454 33771 8118 7984 15238 11862 5168 5129 17665 50392 47027
3  6172  2833  4302  5993 1491 1488  2784  1975  753  719  2631  8513  7935
4  4952  2305  3506  4709 1133 1116  2017  1883  825  840  2404  7483  6622
5 31329 14023 21708 29890 7122 7156 13528 12637 5528 5240 16988 44185 40199
6 10946  4976  7329  9386 2192 2186  3964  4027 2274 2470  7584 16524 12821
  M4554 M5559 M6061 M6264 M6574   M75   FTOT   F04   F56   F79 F1013  F14  F15
1 51181 21292  7003  9010 18823 10697 411211 30932 15026 23313 32321 7773 7446
2 50696 20465  6991  8801 19346 11262 422009 33550 15350 23335 32498 8082 7688
3  8618  3960  1398  1819  4289  2503  73915  5896  2778  4196  5636 1423 1403
4  6733  2825  1036  1304  3033  1739  58351  4845  2209  3362  4553 1029 1083
5 45636 18725  6420  8196 19212 11482 385744 30522 13707 20665 28656 7213 6771
6 12164  4783  1828  2179  5119  3000 114596 10492  4623  7098  8922 2142 2010
  F1617 F1819  F20  F21 F2224 F2534 F3544 F4554 F5559 F6061 F6264 F6574   F75
1 14651 11814 5736 5411 16418 49023 51666 54853 22422  7681 10120 26181 18424
2 14579 13353 6771 6786 20770 51528 49512 53062 21459  7759 10232 27049 18646
3  2602  1908  884  873  3038  8688  8098  9366  4243  1536  2080  5358  3909
4  1953  1508  785  799  2741  7527  6593  7239  2887  1104  1408  3829  2897
5 13060 11985 6097 6060 18958 46105 43485 49626 20698  7375  9759 26603 18399
6  3880  3890 2047 2043  5826 14816 12645 12821  5094  1939  2479  6794  5035
  NEGMTOT NEGM04 NEGM514 NEGM1524 NEGM2534 NEGM3544 NEGM4554 NEGM5564 NEGM65
1   26161   3344    6917     4332     3977     2943     2262     1427    959
2   26069   3507    7105     4427     3790     2845     2281     1248    866
3     592     59     137       97       69       76       57       52     45
```

```
4    1630    203     367      358      271      169      123       71     68
5   26538   3665    7537     4888     3434     2745     2140     1188    941
6    3644    416     979      734      465      462      310      160    118
  NEGFTOT NEGF04 NEGF514 NEGF1524 NEGF2534 NEGF3544 NEGF4554 NEGF5564 NEGF65
1   30247   3513    7143     5187     4901     3450     2783     1848   1422
2   28576   3442    6717     5582     4574     3227     2385     1380   1269
3     534     62      98       87       62       71       43       68     43
4    1697    202     433      369      262      167      117       72     75
5   30092   3917    7436     5544     4565     3386     2498     1460   1286
6    3746    431     939      744      510      454      348      158    162
  TOTPOP2  HHPOP HHHEADS HHPRIM HHMHEAD HHFHEAD HHFSPOUS HHOTHREL HHUNREL GROUP
1  792814 778212  243806  41569  181747   20490   176172   346686   11548 14602
2  816737 799712  255437  48242  184924   22271   179067   352227   12981 17025
3  144091 142582   45550   8085   34330    3135    33245    62151    1636  1509
4  114816 109867   34758   5830   26409    2519    25706    48076    1327  4949
5  744948 725581  231754  42156  168111   21487   162404   321253   10170 19367
6  230348 214257   67618  11833   49180    6605    47778    96592    2269 16091
  INMATES GROUPOTH NEGTOT2 HHNEG HHNEGHEA HHNEGPRI HHNEGMHE HHNEGFHE HHNEGFSP
1    6556     8046   56408 55207    15685     3162     8879     3644     8260
2    8116     8909   54645 53442    15533     3408     8322     3803     7659
3     878      631    1126  1075      306       62      214       30      193
4    3079     1870    3327  2952      819      178      480      161      453
5    8467    10900   56630 55416    15613     3096     8481     4036     7948
6    3736    12355    7390  6793     1857      329     1128      400     1042
  HHNEGOTR HHNEGUNR GROUPNEG INMATNEG GROUNEGO REGION1 REGION2 STATEFIP FIPS
1    29100     2162     1201      582      619       1       1        9 9001
2    28526     1724     1203      656      547       1       1        9 9003
3      512       64       51       29       22       1       1        9 9005
4     1592       88      375      276       99       1       1        9 9007
5    30421     1434     1214      742      472       1       1        9 9009
6     3794      100      597      298      299       1       1        9 9011
  LEVEL
1     1
2     1
3     1
4     1
5     1
6     1
```

```
df2tib <- as_tibble(df)
```

3. Tidyverse / dplyr verbs for Data Wrangling

To modify and "wrangle" datasets, the tidyverse commonly uses the following verbs / functions:

- **select()**: to select specific columns by their names or data types
- **arrange()**: to order rows by one or more columns
- **rename()**: to rename columns
- **mutate()**: to create columns
- **filter()**: to filter out rows by a given condition
- **distinct()**: to keep only distinct / unique rows
- **gather()**: to make "wide" data longer
- **spread()**: to make "long" data wider
- **separate()**: to split a single column into multiple columns
- **unite()**: to combine multiple columns into one

More tidyverse verbs and functions here.

Common Data Science Operations in Tidyverse

Task	Python - Pandas	Base R	Tidyverse (R)
Sort table by column	df.sort_values('col')	sort(df$col)	df %>% arrange(col)
Filter rows	df.loc[df['col']>0,:]	df[df[col] > 0,]	df %>% filter(col > 0)
Subset columns			df %>% select(col)
Create a new column	df.loc[:, "square"] = df[col] ** 2	df$square = df[col] ^2	df %>% mutate(square = col ^ 2)
Perform calculation on column	sum(df["col"])	sum(df$col)	df %>% summarise(sumcol = sum(col))
Delete duplicate rows	df.drop_duplicates()	df[!duplicated(df),]	df %>% distinct()
Group/Apply/Combine	df.groupby("col").sum()	by(df, col, sum(x)) ???	df %>% group_by(col) %>% summarise(groupsum = sum(col2))
Reshape data from wide to long	*pd.wide_to_long()*	*melt()*	*use pivot_longer()*

Sort Table (tibble)

```
starwars |>
  arrange(homeworld)  #descending order: arrange(desc(homeworld))
```

```
# A tibble: 87 x 14
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Leia Or~    150    49 brown      light      brown             19 fema~ femin~
 2 Bail Pr~    191    NA black      tan        brown             67 male  mascu~
 3 Raymus ~    188    79 brown      light      brown             NA male  mascu~
 4 Ratts T~     79    15 none       grey, blue unknown           NA male  mascu~
 5 Lobot       175    79 none       light      blue              37 male  mascu~
 6 Jek Ton~    180   110 brown      fair       blue              NA <NA>  <NA>
 7 Nute Gu~    191    90 none       mottled g~ red               NA male  mascu~
 8 Ki-Adi-~    198    82 white      pale       yellow            92 male  mascu~
 9 Mas Ame~    196    NA none       blue       blue              NA male  mascu~
10 Mon Mot~    150    NA auburn     fair       blue              48 fema~ femin~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

Filter Table

```
starwars |>
  filter(species=="Droid")
```

```
# A tibble: 6 x 14
  name   height  mass hair_color skin_color  eye_color birth_year sex   gender
  <chr>   <int> <dbl> <chr>      <chr>       <chr>          <dbl> <chr> <chr>
1 C-3PO     167    75 <NA>       gold        yellow           112 none  masculi~
2 R2-D2      96    32 <NA>       white, blue red               33 none  masculi~
3 R5-D4      97    32 <NA>       white, red  red               NA none  masculi~
4 IG-88     200   140 none       metal       red               15 none  masculi~
5 R4-P17     96    NA none       silver, red red, blue         NA none  feminine
6 BB8        NA    NA none       none        black             NA none  masculi~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

Subset Tibble by Column Names

```
starwars |>
  select(name, homeworld, species)  #reorder names here to reorder columns
```

```
# A tibble: 87 x 3
   name               homeworld species
   <chr>              <chr>     <chr>
 1 Luke Skywalker     Tatooine  Human
 2 C-3PO              Tatooine  Droid
```

```
 3 R2-D2              Naboo     Droid
 4 Darth Vader        Tatooine  Human
 5 Leia Organa        Alderaan  Human
 6 Owen Lars          Tatooine  Human
 7 Beru Whitesun Lars Tatooine  Human
 8 R5-D4              Tatooine  Droid
 9 Biggs Darklighter  Tatooine  Human
10 Obi-Wan Kenobi     Stewjon   Human
# i 77 more rows
```

For a range of columns

```
starwars |>
  select(name:hair_color, species)
```

```
# A tibble: 87 x 5
   name               height  mass hair_color    species
   <chr>               <int> <dbl> <chr>         <chr>
 1 Luke Skywalker        172    77 blond         Human
 2 C-3PO                 167    75 <NA>          Droid
 3 R2-D2                  96    32 <NA>          Droid
 4 Darth Vader           202   136 none          Human
 5 Leia Organa           150    49 brown         Human
 6 Owen Lars             178   120 brown, grey   Human
 7 Beru Whitesun Lars    165    75 brown         Human
 8 R5-D4                  97    32 <NA>          Droid
 9 Biggs Darklighter     183    84 black         Human
10 Obi-Wan Kenobi        182    77 auburn, white Human
# i 77 more rows
```

Rename columns

```
starwars |>
  rename(character = name, planet = homeworld)
```

```
# A tibble: 87 x 14
   character       height  mass hair_color skin_color eye_color birth_year sex
   <chr>            <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke Skywalker     172    77 blond      fair       blue            19   male
 2 C-3PO              167    75 <NA>       gold       yellow         112   none
 3 R2-D2               96    32 <NA>       white, bl~ red             33   none
 4 Darth Vader        202   136 none       white      yellow          41.9 male
 5 Leia Organa        150    49 brown      light      brown           19   fema~
 6 Owen Lars          178   120 brown, gr~ light      blue            52   male
 7 Beru Whitesun ~    165    75 brown      light      blue            47   fema~
```

```
 8 R5-D4               97    32 <NA>       white, red red             NA   none
 9 Biggs Darkligh~    183    84 black      light      brown           24   male
10 Obi-Wan Kenobi     182    77 auburn, w~ fair       blue-gray       57   male
# i 77 more rows
# i 6 more variables: gender <chr>, planet <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
starwars$vehicles[1]
```

```
[[1]]
[1] "Snowspeeder"           "Imperial Speeder Bike"
```

Split-Apply-Combine

```
starwars |>
  group_by(species)
```

```
# A tibble: 87 x 14
# Groups:   species [38]
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
 2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
 3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
 4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
 5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
 6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
 7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
 8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
 9 Biggs D~    183    84 black      light      brown           24   male  mascu~
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
starwars |>
  group_by(species) |>
  summarise(avg_height = mean(height, na.rm = TRUE))
```

```
# A tibble: 38 x 2
   species   avg_height
   <chr>          <dbl>
 1 Aleena          79
 2 Besalisk       198
 3 Cerean         198
 4 Chagrian       196
```

```
 5 Clawdite        168
 6 Droid           131.
 7 Dug             112
 8 Ewok             88
 9 Geonosian       183
10 Gungan          209.
# i 28 more rows
```

```
starwars |>
  group_by(species) |>
  summarise(avg_height = mean(height, na.rm = TRUE)) |>
  arrange(avg_height)
```

```
# A tibble: 38 x 2
   species        avg_height
   <chr>               <dbl>
 1 Yoda's species         66
 2 Aleena                 79
 3 Ewok                   88
 4 Vulptereen             94
 5 Dug                   112
 6 Xexto                 122
 7 Droid                 131.
 8 Toydarian             137
 9 Sullustan             160
10 Toong                 163
# i 28 more rows
```

CODING ASSIGNMENT Week 1, Notebooks 01-02: Exploring a Built-in Dataset

In a new Quarto document, choose another built-in dataset (run `data()` to see what datasets are available).

1. examine the dataset using the functions demonstrated in the previous notebook (Week 1 / Notebook 01).
2. Pose a question you would like to answer about this dataset?
3. To answer this question re-shape, filter, sort, and aggregate the dataset using the tidyverse functions demonstrated above.
4. Write a brief paragraph explaining what you learned from this preliminary analysis.
5. **To complete this assignment: Create a brief, clean version of this Quarto document (removing all unnecessary code). Then submit a pdf version of this document. To convert the document into a pdf see the directions below.**
6. Instructions for converting a Quarto Document (.qmd) into a PDF

(for more details see Quarto: pdf basics):

1. Place the following format instructions in the YAML header of the document:
   ```
   format:
   pdf:
   documentclass: report
   papersize: letter
   ```
2. Install **tinytex** if you haven't already. To do so, run `quarto install tinytex` in the terminal.