FA1_Group13_Quijano

Julian Philip S. Quijano

2026-01-28

```
library(tidyverse)
```

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

## Warning: package 'ggplot2' 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 'purrr' was built under R version 4.4.3

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

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

## Warning: package 'lubridate' 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   1.0.1     v stringr   1.5.1
## v ggplot2   4.0.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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 become errors
```

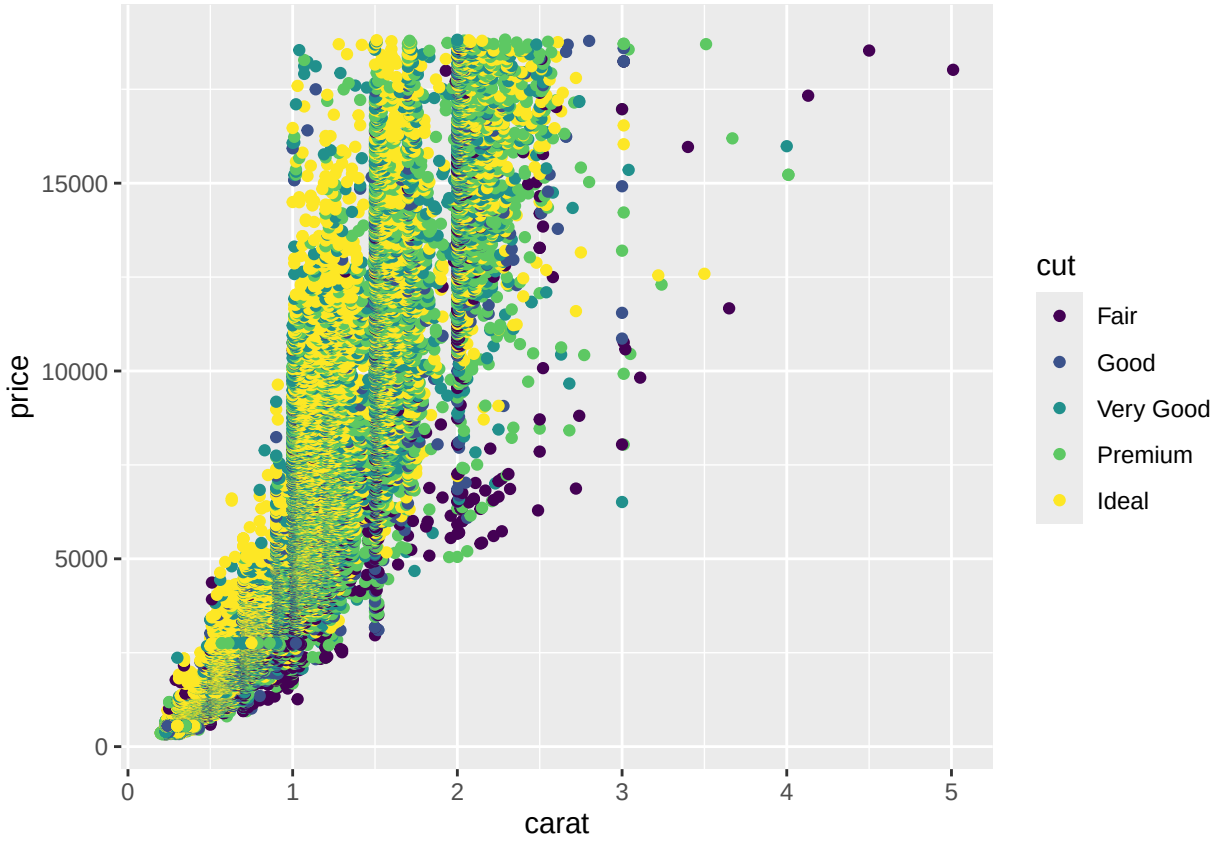
```
data <- diamonds
data
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
```

```
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # i 53,930 more rows
```

LECTURE 2

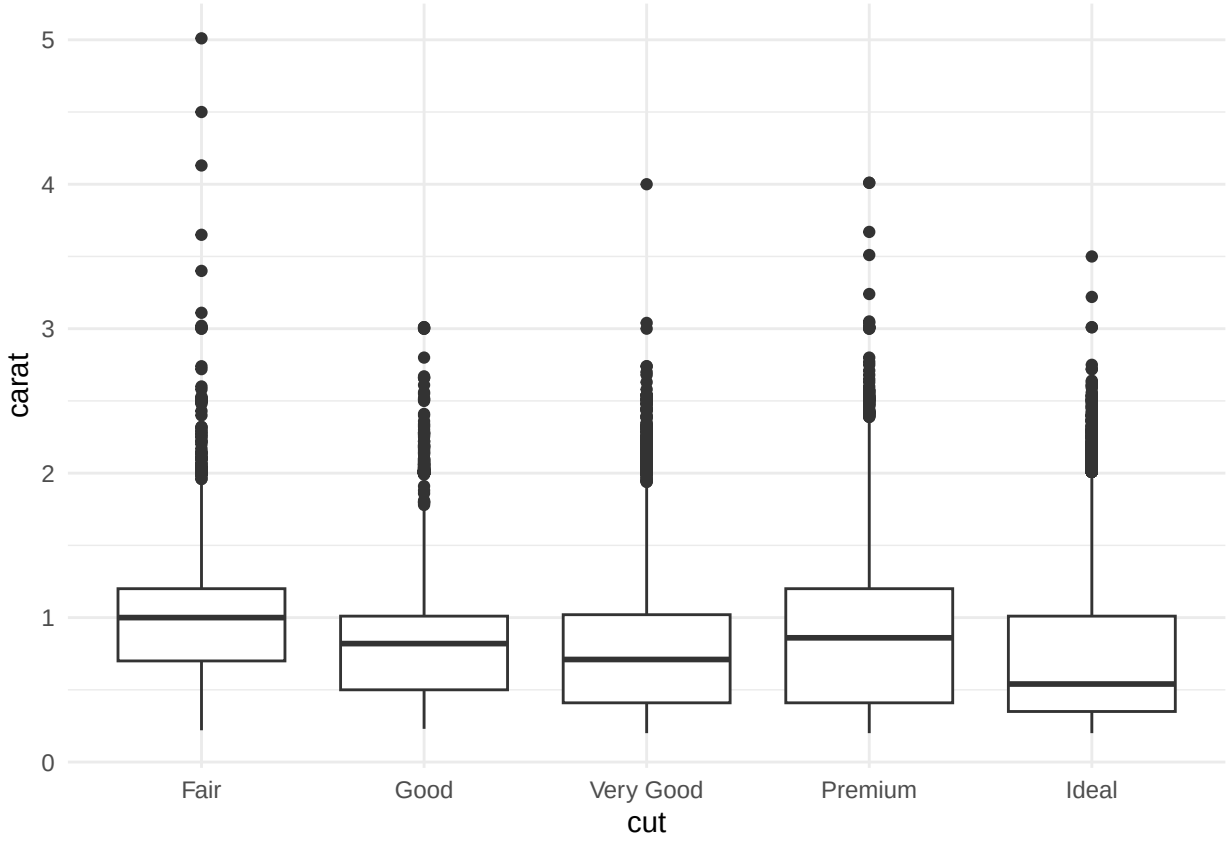
```
data %>%
  ggplot(aes(x = carat, y = price, color = cut)) +
  geom_point()
```



What relationship does it suggest between carat and cut? This plot makes it appear as though worse cuts are more expensive than better cuts.

Create a plot to directly visualize this relationship.

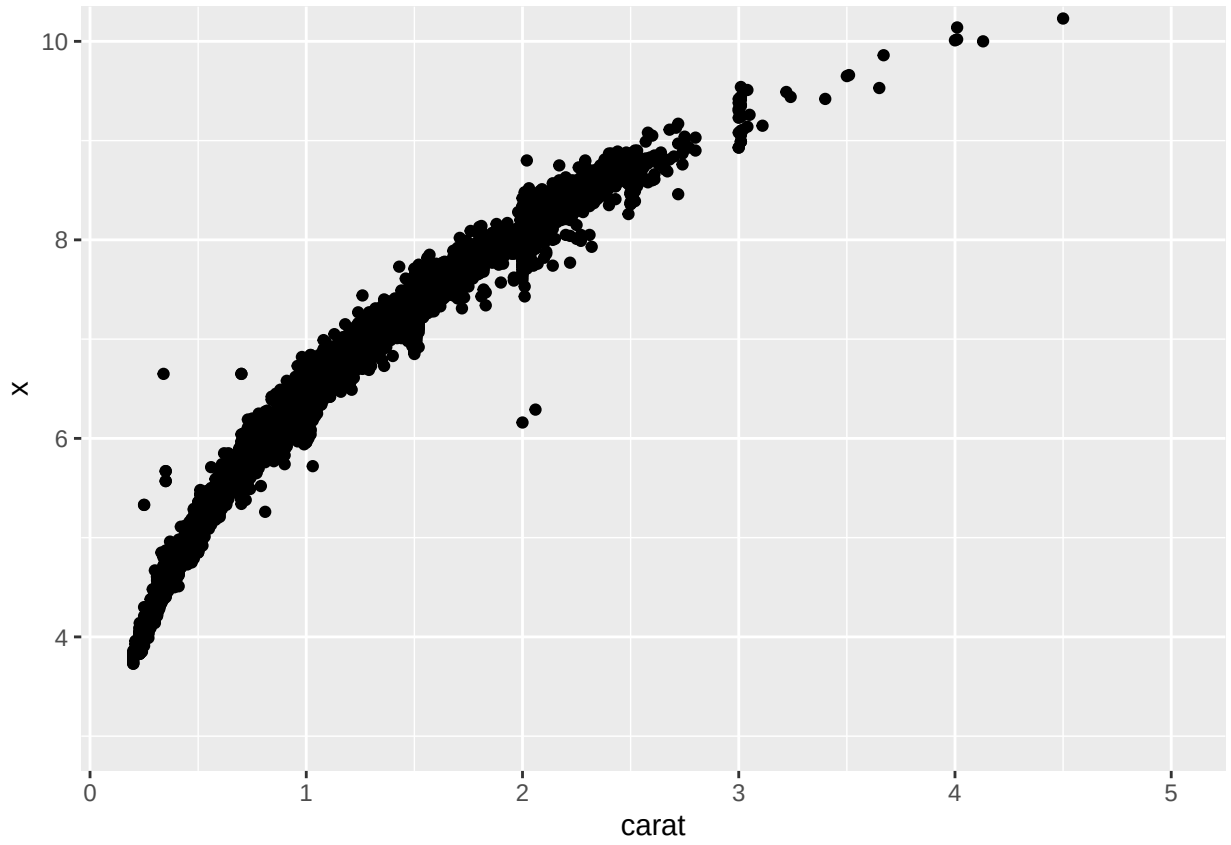
```
data %>%
  ggplot(aes(x = cut, y = carat)) +
  geom_boxplot() +
  theme_minimal()
```



What do you conclude? How does this explain the paradoxical trend we found in the plot below? Lower quality cuts (Fair, Good) have, in average larger carat sizes. Meanwhile, higher quality cuts (Premium, Ideal) have smaller carat sizes in general. So we can conclude that as cut quality increases, carat size decreases. This maybe because larger diamonds are harder to cut perfectly, while smaller diamonds are likelier to have better cuts. The apparent price advantage of lower-quality cuts is explained by their larger typical carat sizes; once carat is accounted for, higher cut quality corresponds to higher prices.

Create a plot to visualize the relationship between the carat and length of a diamond. Zoom in to exclude any outliers. What sort of relationship does your plot suggest? How would you explain this relationship?

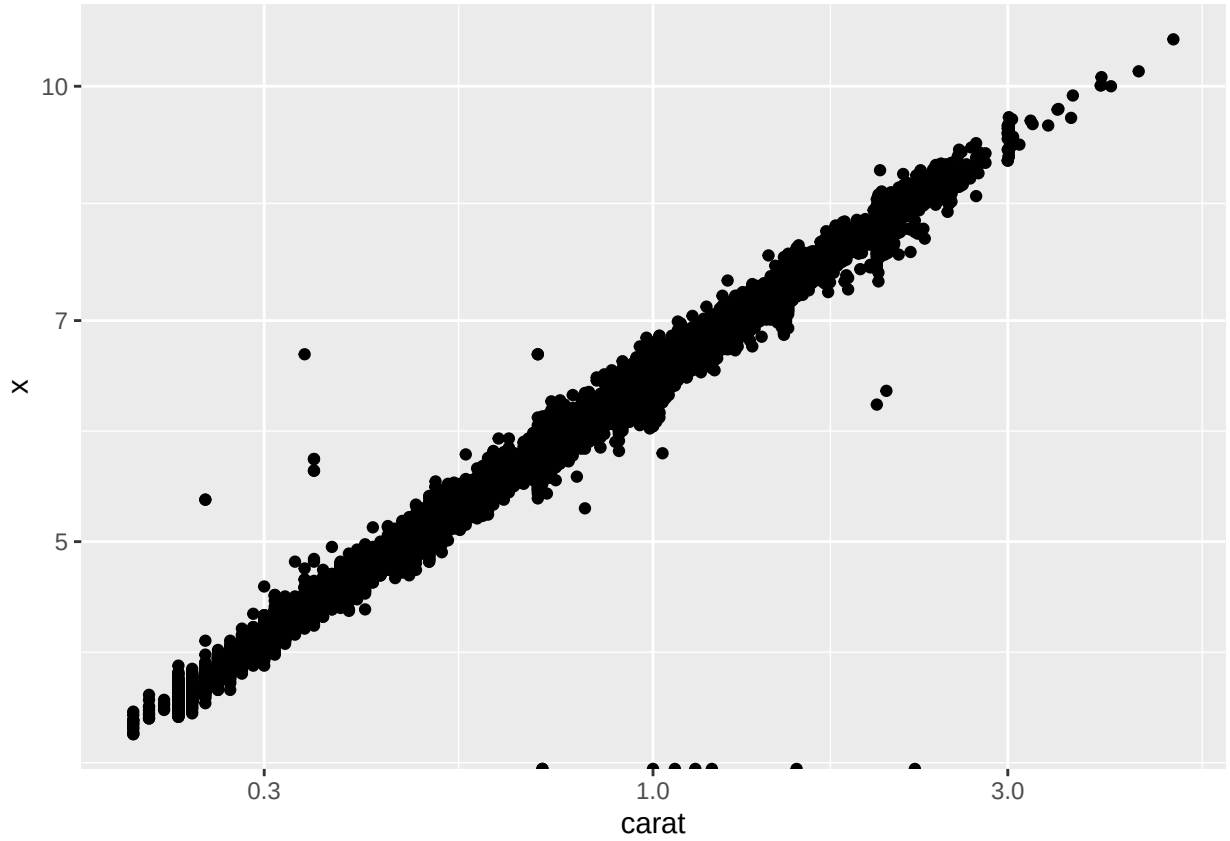
```
data %>%
  ggplot(aes(x = carat, y = x)) +
  geom_point() +
  coord_cartesian(ylim = c(3,10))
```



The carat, which is weight, of the diamonds appear to have a logarithmic relationship with the length of the diamonds. To confirm this, we can change the scale of the axes with log_10 to create this:

```
data %>%
  ggplot(aes(x = carat, y = x)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```



The linear relationship now confirms that, indeed, the relation between carat and length is logarithmic. This can be explained by the fact that weight is a 3-dimensional value, whilst length is 1-dimensional. So as a diamond grows, its weight grows faster exponentially relative to its length.

LECTURE 3

Exercise: Filter diamonds to those with ideal cut and at least 3 carats. How many such diamonds are there?

```
filter(data, carat <= 3 & cut %in% ('Ideal'))
```

```
## # A tibble: 21,547 x 10
##    carat cut   color clarity depth table price     x     y     z
##    <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.23 Ideal E     SI2      61.5    55   326  3.95  3.98  2.43
##  2  0.23 Ideal J     VS1      62.8    56   340  3.93  3.9   2.46
##  3  0.31 Ideal J     SI2      62.2    54   344  4.35  4.37  2.71
##  4  0.3  Ideal I     SI2      62      54   348  4.31  4.34  2.68
##  5  0.33 Ideal I     SI2      61.8    55   403  4.49  4.51  2.78
##  6  0.33 Ideal I     SI2      61.2    56   403  4.49  4.5   2.75
##  7  0.33 Ideal J     SI1      61.1    56   403  4.49  4.55  2.76
##  8  0.23 Ideal G     VS1      61.9    54   404  3.93  3.95  2.44
##  9  0.32 Ideal I     SI1      60.9    55   404  4.45  4.48  2.72
```

```
## 10  0.3  Ideal I     SI2      61      59   405  4.3   4.33  2.63
## # i 21,537 more rows
```

21,547 Observations, so 21,547 diamonds.

Exercise: Select all columns except x, y, z.

```
select(data, -x, -y, -z)
```

```
## # A tibble: 53,940 x 7
##    carat cut       color clarity depth table price
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int>
##  1  0.23 Ideal     E     SI2      61.5    55   326
##  2  0.21 Premium   E     SI1      59.8    61   326
##  3  0.23 Good      E     VS1      56.9    65   327
##  4  0.29 Premium   I     VS2      62.4    58   334
##  5  0.31 Good      J     SI2      63.3    58   335
##  6  0.24 Very Good J     VVS2     62.8    57   336
##  7  0.24 Very Good I     VVS1     62.3    57   336
##  8  0.26 Very Good H     SI1      61.9    55   337
##  9  0.22 Fair      E     VS2      65.1    61   337
## 10  0.23 Very Good H     VS1      59.4    61   338
## # i 53,930 more rows
```

Exercise: Arrange diamonds in decreasing order of their length. How long is the longest diamond?

```
arrange(data, desc(x))
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  5.01 Fair      J     I1       65.5    59 18018 10.7  10.5   6.98
##  2  4.5  Fair      J     I1       65.8    58 18531 10.2  10.2   6.72
##  3  4.01 Premium   I     I1       61      61 15223 10.1  10.1   6.17
##  4  4.01 Premium   J     I1       62.5    62 15223 10.0   9.94  6.24
##  5  4    Very Good I     I1       63.3    58 15984 10.0   9.94  6.31
##  6  4.13 Fair      H     I1       64.8    61 17329 10     9.85  6.43
##  7  3.67 Premium   I     I1       62.4    56 16193  9.86  9.81  6.13
##  8  3.51 Premium   J     VS2      62.5    59 18701  9.66  9.63  6.03
##  9  3.5  Ideal     H     I1       62.8    57 12587  9.65  9.59  6.03
## 10  3.01 Fair      H     I1       56.1    62 10761  9.54  9.38  5.31
## # i 53,930 more rows
```

10.74 mm

Exercise: Add a variable called good_color that is TRUE if the color is D, E, F, G and FALSE otherwise.

```
b <- mutate(diamonds,
       good_color =
         case_when(
           color %in% c('D','E','F','G') ~ 'TRUE',
           TRUE ~ 'FALSE'
         ))

select(b, color, good_color)
```

```
## # A tibble: 53,940 x 2
##    color good_color
##    <ord> <chr>
##  1 E     TRUE
##  2 E     TRUE
##  3 E     TRUE
##  4 I     FALSE
##  5 J     FALSE
##  6 J     FALSE
##  7 I     FALSE
##  8 H     FALSE
##  9 E     TRUE
## 10 H     FALSE
## # i 53,930 more rows
```

Exercise: Use summarise to determine if there are any diamonds of at least one carat that cost less that $1000.

```
summarise(data, c = sum(carat >= 1 & price < 1000))
```

```
## # A tibble: 1 x 1
##       c
##   <int>
## 1     0
```

There are no diamonds at least one carat that are less than $1000

Exercise: Compute the mean price for diamonds of volume at least one carat.

```
data %>%
  filter(carat >= 1) %>%
  summarise(mean_price = mean(price, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   mean_price
##        <dbl>
## 1      8142.
```

Exercise: Reproduce the output of count(diamonds, cut) via group_by() and summarise().

```
data %>%
  group_by(cut) %>%
  summarise(n = n()) %>%
  ungroup()
```

```
## # A tibble: 5 x 2
##   cut           n
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

Use dplyr to answer the following questions:

What is the minimum diamond price in this dataset? See if you can find the answer in two different ways (i.e. using two different dplyr verbs).

```
summarise(data, min_price = min(price))
```

```
## # A tibble: 1 x 1
##   min_price
##       <int>
## 1       326
```

```
data %>%
  arrange(price) %>%
  slice(1) %>%
  select(price)
```

```
## # A tibble: 1 x 1
##   price
##   <int>
## 1   326
```

$326 is the minimum diamond price.

How many diamonds have length at least one and a half times their width?

```
data %>%
  filter(x >= (3/2)*y) %>%
  summarise(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    10
```

10 diamonds.

Among diamonds with colors D, E, F, G, what is the median number of carats for diamonds of each cut?

```
data %>%
  filter(color %in% c("D", "E", "F", "G")) %>%
  group_by(cut) %>%
  summarise(median_cut = median(carat))
```

```
## # A tibble: 5 x 2
##   cut       median_cut
##   <ord>          <dbl>
## 1 Fair            0.91
## 2 Good            0.72
## 3 Very Good       0.7
## 4 Premium         0.71
## 5 Ideal           0.52
```

0.91 for Fair, 0.72 for Good, 0.70 for Very Good, 0.71 for Premium, and 0.52 for Ideal.

LECTURE 4

Import heights2.csv

```
heights_data = read_csv("heights.csv")
```

```
## Rows: 1192 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): sex, race
## dbl (4): earn, height, ed, age
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
heights_data
```

```
## # A tibble: 1,192 x 6
##     earn height sex       ed   age race
##    <dbl>  <dbl> <chr>  <dbl> <dbl> <chr>
##  1 50000   74.4 male      16    45 white
```

```
##  2 60000   65.5 female    16    58 white
##  3 30000   63.6 female    16    29 white
##  4 50000   63.1 female    16    91 other
##  5 51000   63.4 female    17    39 white
##  6  9000   64.4 female    15    26 white
##  7 29000   61.7 female    12    49 white
##  8 32000   72.7 male      17    46 white
##  9  2000   72.0 male      15    21 hispanic
## 10 27000   72.2 male      12    26 white
## # i 1,182 more rows
```

Exercise: Using prose, describe how the variables and observations are organised in each of the sample tables.

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <dbl>      <dbl>
## 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
```

Table 1 has a column for each variable. It has a row for each observation as well. In this way, each cell is clearly defined and does not lead to confusion.

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <dbl> <chr>           <dbl>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

Table 2 doubles the observations because it merges population and cases into one variable, type. A new variable, count, is used for the number corresponding to cases and population.

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <dbl> <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
```

Table 3 is even more confusing, combining cases and population into one variable, rate. This variable contains case/population, which is confusing, since it is not stated what the numbers represent, and without prior knowledge of table1 and table2, this table would not be understandable.

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

Table 4a removes variables type, cases, population, and year in favor of creating 2 variables 1999 and 2000, which contain the number of cases from the dataset. However, this is not explicitly stated and will lead to confusion. It also removes any mention of the population of the countries from the dataset.

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
##   <chr>            <dbl>      <dbl>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

Table 4b is the exact same as 4a, but instead of cases, it has population numbers under 1999 and 2000.

Exercise: Use pivot_longer() to tidy table4b in a similar fashion. What is the difference between the code used to tidy table4a and table4b?

```
pivot_longer(table4b,
             cols = c('1999', '2000'),
             names_to = 'year',
             values_to = 'population')
```

```
## # A tibble: 6 x 3
##   country     year  population
##   <chr>       <chr>      <dbl>
## 1 Afghanistan 1999    19987071
## 2 Afghanistan 2000    20595360
## 3 Brazil      1999   172006362
## 4 Brazil      2000   174504898
## 5 China       1999  1272915272
## 6 China       2000  1280428583
```

The difference is we used 'population' for values_to instead of 'cases' for this pivot.

Exercise:

1. Why does this code fail?

table4a %>% pivot_longer(cols = c(1999, 2000), names_to = "year", values_to = "cases")

It uses 1999 and 2000 and the program treats these as int, what should be used instead is "1999" and "2000".

2. Tidy the simple tibble below. Do you need to make it wider or longer? What are the variables?

```
exc2 <- tribble(
  ~pregnant, ~male, ~female,
  "yes", NA, 10,
  "no", 20, 12
)
```

We need to use pivot_longer since the variables male and female can just be values of a single variable: gender.

```
exc2 %>%
  pivot_longer(
    cols = c(male, female),
    names_to = 'gender',
    values_to = 'count')
```

```
## # A tibble: 4 x 3
##   pregnant gender count
##   <chr>    <chr>  <dbl>
## 1 yes      male      NA
## 2 yes      female    10
## 3 no       male      20
## 4 no       female    12
```

Exercise: Consider the two tibbles below. What is the key column? Without writing any code, can you predict how many rows and columns left_join(x,y) and left_join(y,x) will have?

```
x <- tribble(
~state, ~population,
"PA", 12.8,
"TX", 28.6,
"NY", 19.5
)
y <- tribble(
~state, ~capital,
"TX", "Austin",
"CA", "Sacramento",
"NY", "New York City",
"MI", "Lansing"
)
```

The key column is 'state'. left_join(x, y) would have 3 rows and 3 columns, while left_join(y, x) would have 4 rows and 3 columns.

Github Link: https://github.com/SylTana/DSC1107---Data-Mining-and-Wrangling/tree/main/FA1