02 Data Wrangling Homework

Data Science and Machine Learning 2187 & 2087

Max Thomasberger

11 2020

Contents

PLEASE ENTER DETAILS ABOUT YOUR GROUP HERE

Leonard Fidlin, h01352705

Daniel Jost, h01451889

Anne Valder, h11928415

Introduction

Note: this Rmarkdown file downloads a 78 MB zip file from a server when you knit it.

Please limit your outputs to the first few results using the **head()** function after filtering or re-arranging the data. If I recieve HTML files containing endless lists of names the respective group will get loose points!

Here's how you should do it:

```
# This only shows the first few rows of the data frame after knitting

babynames %>%
  head()
```

```
##   year sex      name    n       prop
## 1 1880   F      Mary 7065 0.07238359
## 2 1880   F      Anna 2604 0.02667896
## 3 1880   F      Emma 2003 0.02052149
## 4 1880   F Elizabeth 1939 0.01986579
## 5 1880   F    Minnie 1746 0.01788843
## 6 1880   F  Margaret 1578 0.01616720
```

18 points are possible for this homework. Each code chunk you enter is worth 1 point. The real world example is in total worth 6 points and a bit tricky but I am inclined to be generous if you at least try it.

The homework is due on Thursday 03-12-2020 at 23:59. Please upload your answers as HTML file to learn@WU. I will draw a random course participant on the Friday 04-12-2020 lecture to walk us through some of the exercises.

Select

Alter the code below to select just the `prop` column:

```
select(babynames, name, prop) %>%
  head()
```

```
##        name       prop
## 1      Mary 0.07238359
## 2      Anna 0.02667896
## 3      Emma 0.02052149
## 4 Elizabeth 0.01986579
## 5    Minnie 0.01788843
## 6  Margaret 0.01616720
```

Filter 1

Show:

- All names where prop is greater than or equal to 0.05 and smaller than 0.8
- All children named "Max" and "Moritz"
- All of the rows that have a missing value for `name`.

Only display the first few results using the head() function.

```
filter(babynames, 0.05<=prop, prop<=0.80)  %>%
  head()
```

```
##   year sex    name    n       prop
## 1 1880   F    Mary 7065 0.07238359
```

```
## 2 1880   M    John 9655 0.08154561
## 3 1880   M William 9532 0.08050676
## 4 1880   M   James 5927 0.05005912
## 5 1881   F    Mary 6919 0.06999140
## 6 1881   M    John 8769 0.08098299
```

```
filter(babynames, name %in% c("Max","Moritz"))  %>%
  head()
```

```
##   year sex name  n       prop
## 1 1880   M  Max 52 0.00043919
## 2 1881   M  Max 66 0.00060952
## 3 1882   M  Max 74 0.00060640
## 4 1883   M  Max 75 0.00066680
## 5 1884   M  Max 80 0.00065179
## 6 1885   M  Max 71 0.00061236
```

```
filter(babynames, is.na(name))  %>%
  head()
```

```
## [1] year sex  name n    prop
## <0 rows> (or 0-length row.names)
```

Filter 2

Use Boolean operators to return only the rows of the babyname object that contain:

- Girls named Max
- Names that were used by exactly 5 or 6 children in 1990
- Names that are one of Max, Moritz, or Wilhelm

Only display the first few results using the head() function.

```
filter(babynames, sex=="F" & name=="Max") %>%
  head()
```

```
##   year sex name  n     prop
## 1 1912   F  Max  5 8.52e-06
## 2 1914   F  Max  6 7.53e-06
## 3 1917   F  Max  7 6.23e-06
## 4 1918   F  Max  8 6.65e-06
## 5 1919   F  Max 11 9.36e-06
## 6 1920   F  Max 10 8.04e-06
```

```
babynames %>% filter(year == 1990 & n %in% c(5,6)) %>%
  head()
```

```
##   year sex     name n     prop
## 1 1990   F   Aariel 6 2.92e-06
## 2 1990   F   Aarion 6 2.92e-06
## 3 1990   F  Abagael 6 2.92e-06
## 4 1990   F    Abbye 6 2.92e-06
## 5 1990   F   Abiola 6 2.92e-06
## 6 1990   F Abreanna 6 2.92e-06
```

```
filter(babynames, name=="Max" | name=="Moritz" | name=="Wilhelm") %>%
  head()
```

```
##   year sex    name  n       prop
```

```
## 1 1880   M     Max 52 0.00043919
## 2 1880   M Wilhelm  6 0.00005068
## 3 1881   M     Max 66 0.00060952
## 4 1882   M     Max 74 0.00060640
## 5 1882   M Wilhelm  6 0.00004917
## 6 1883   M     Max 75 0.00066680
```

Arrange

Arrange babynames by `n`. Add `prop` as a second (tie breaking) variable to arrange on. Can you tell what the smallest value of `n` is?

Only display the first few results using the head() function.

```
arrange(babynames, n, prop) %>%
  head()
```

```
##   year sex       name n     prop
## 1 2007   M      Aaban 5 2.26e-06
## 2 2007   M     Aareon 5 2.26e-06
## 3 2007   M      Aaris 5 2.26e-06
## 4 2007   M        Abd 5 2.26e-06
## 5 2007   M Abdulazeez 5 2.26e-06
## 6 2007   M  Abdulhadi 5 2.26e-06
```

```
min(babynames$n)
```

```
## [1] 5
```

Desc

Use `desc()` to find the names with the highest prop.

Then, use `desc()` to find the names with the highest n.

Only display the first few results using the head() function.

```
arrange(babynames, desc(prop)) %>%
  head()
```

```
##   year sex    name    n       prop
## 1 1880   M    John 9655 0.08154561
## 2 1881   M    John 8769 0.08098299
## 3 1880   M William 9532 0.08050676
## 4 1883   M    John 8894 0.07907394
## 5 1881   M William 8524 0.07872038
## 6 1882   M    John 9557 0.07831617
```

```
arrange(babynames, desc(n)) %>%
  head()
```

```
##   year sex    name     n       prop
## 1 1947   F   Linda 99686 0.05483812
## 2 1948   F   Linda 96209 0.05521079
## 3 1947   M   James 94756 0.05101589
## 4 1957   M Michael 92695 0.04237565
## 5 1947   M  Robert 91642 0.04933934
## 6 1949   F   Linda 91016 0.05184643
```

Steps and the pipe

Use `%>%` to write a sequence of functions that:

1. Filter babynames to just the boys that were born in 1990
2. Select the `prop` and `name` columns
3. Arrange the results so that the most popular names are near the top.
4. Only show the first few results using the head() function

```
babynames %>%
filter(sex=="M", year==1990) %>%
  select(prop, name) %>%
  arrange(desc(prop)) %>%
  head()
```

```
##         prop        name
## 1 0.03034735     Michael
## 2 0.02432734 Christopher
## 3 0.02082597     Matthew
## 4 0.02008963      Joshua
## 5 0.01571943      Daniel
## 6 0.01568549       David
```

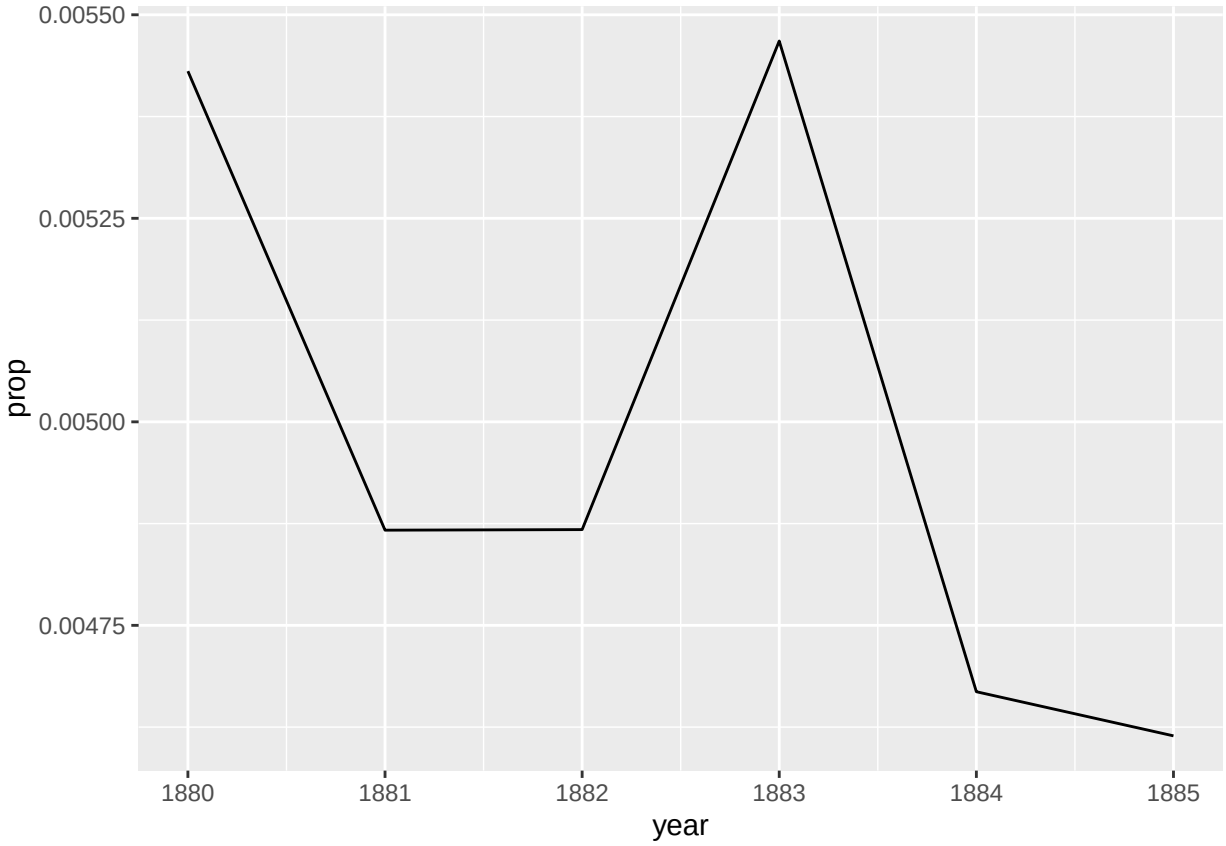
Steps and the pipe

Chain the following steps together using the pipe `%>%` operator.

1. Trim `babynames` to just the rows that contain one of your `name`s and your `sex` (if your name is not in there use the name and sex of your favorite movie star)
2. Using ggplot2 plot the results as a line graph with `year` on the x axis and `prop` on the y axis

```
dan <- babynames %>%
  filter(name=="Daniel", sex=="M") %>%
  head()

ggplot(dan, aes(year, prop)) +
  geom_line()
```



Summarise

Here some code to remind you how summarise works:

```
pollution %>%
 summarise(mean = mean(amount), sum = sum(amount), n = n())
```

```
## # A tibble: 1 x 3
##    mean   sum     n
##   <dbl> <dbl> <int>
## 1    42   252     6
```

Now use summarise() to compute three statistics about the babynames data:

1. The first (minimum) year in the dataset
2. The last (maximum) year in the dataset
3. The total number of children represented in the data

```
babynames %>%
  summarise(min(year), max(year), sum = sum(n))  %>%
  head()
```

```
##   min(year) max(year)       sum
## 1      1880      2017 348120517
```

Filtering and wrangling

Extract the rows where `name == "Khaleesi"`. Then use `summarise()` and a summary functions to find:

1. The total number of children named Khaleesi
2. The first year Khaleesi appeared in the data

```
babynames %>%
filter(name == "Khaleesi") %>%
  summarise(sum = sum(n), min(year))  %>%
  head()
```

```
##    sum min(year)
## 1 1964      2011
```

Split apply combine

Here some code to remind you how `group_by()` and `summarise()` work:

```
pollution %>%
  group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

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

```
## # A tibble: 3 x 4
##   city      mean   sum     n
##   <chr>    <dbl> <dbl> <int>
## 1 Beijing   88.5   177     2
## 2 London    19      38     2
## 3 New York  18.5    37     2
```

Now use `group_by()`, `summarise()`, and `arrange()` to display the ten most popular baby names using the `head(10)` command. Compute popularity as the total number of children of a single gender given a name.

```
babynames %>%
group_by(sex, name) %>%
summarise(sum = sum(n)) %>%
arrange(desc(sum)) %>%
head(10)
```

```
## `summarise()` regrouping output by 'sex' (override with `.groups` argument)
```

```
## # A tibble: 10 x 3
## # Groups:   sex [2]
##    sex   name        sum
##    <chr> <chr>     <int>
##  1 M     James   5150472
##  2 M     John    5115466
##  3 M     Robert  4814815
##  4 M     Michael 4350824
##  5 F     Mary    4123200
##  6 M     William 4102604
##  7 M     David   3611329
##  8 M     Joseph  2603445
##  9 M     Richard 2563082
## 10 M     Charles 2386048
```

Mutate

Here some code to remind you how `mutate()` works:

```
babynames %>%
  mutate(percent = round(prop*100, 2)) %>%
  head()
```

```
##   year sex      name    n       prop percent
## 1 1880   F      Mary 7065 0.07238359    7.24
## 2 1880   F      Anna 2604 0.02667896    2.67
## 3 1880   F      Emma 2003 0.02052149    2.05
## 4 1880   F Elizabeth 1939 0.01986579    1.99
## 5 1880   F    Minnie 1746 0.01788843    1.79
## 6 1880   F  Margaret 1578 0.01616720    1.62
```

Now use `min_rank()` and `mutate()` to rank each row in `babynames` from largest `n` to smallest `n`.

Only display the first few results using the head() function.

```
babynames %>%
  mutate(rank = min_rank(desc(n))) %>%
  arrange(rank) %>%
  head()
```

```
##   year sex    name     n       prop rank
## 1 1947   F   Linda 99686 0.05483812    1
## 2 1948   F   Linda 96209 0.05521079    2
## 3 1947   M   James 94756 0.05101589    3
## 4 1957   M Michael 92695 0.04237565    4
## 5 1947   M  Robert 91642 0.04933934    5
## 6 1949   F   Linda 91016 0.05184643    6
```

Compute each name's rank *within its year and sex*. Then compute the median rank *for each combination of name and sex*, and arrange the results from highest median rank to lowest.

```
# 1
babynames %>%
  group_by(year, sex) %>%
  mutate(rank_ys = min_rank(desc(n))) %>%
  arrange(rank_ys) %>%
  head()
```

```
## # A tibble: 6 x 6
## # Groups:   year, sex [6]
##    year sex   name      n   prop rank_ys
##   <dbl> <chr> <chr> <int>  <dbl>   <int>
## 1  1880 F     Mary   7065 0.0724       1
## 2  1880 M     John   9655 0.0815       1
## 3  1881 F     Mary   6919 0.0700       1
## 4  1881 M     John   8769 0.0810       1
## 5  1882 F     Mary   8148 0.0704       1
## 6  1882 M     John   9557 0.0783       1
```

```
# 2
babynames %>%
  group_by(year, sex) %>%
  mutate(rank_ys = min_rank(desc(n))) %>%
  summarise(median = median(rank_ys)) %>%
```

```
  arrange(desc(median)) %>%
  head()
```

```
## `summarise()` regrouping output by 'year' (override with `.groups` argument)
```

```
## # A tibble: 6 x 3
## # Groups:   year [6]
##    year sex   median
##   <dbl> <chr>  <dbl>
## 1  2007 F      10121
## 2  2008 F      10070
## 3  2009 F       9962
## 4  2006 F       9894
## 5  2010 F       9779
## 6  2011 F       9712
```

Joining data

Here some code to remind you of the types of joins we looked at in class:

```
band %>% left_join(instrument, by = "name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

```
band %>% right_join(instrument, by = "name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

```
band %>% full_join(instrument, by = "name")
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

```
band %>% inner_join(instrument, by = "name")
```

```
## # A tibble: 2 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
```

Left join

Which airlines had the largest arrival delays?

1. Join `airlines` to `flights`
2. Compute and order the average arrival delays by airline. Display full names, no codes.

```
join <- flights %>%  left_join(airlines, by="carrier")

join %>%
  group_by(name) %>%
  summarise(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(avg_delay) %>%
  head(10)
```

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

```
## # A tibble: 10 x 2
##    name                    avg_delay
##    <chr>                       <dbl>
##  1 Alaska Airlines Inc.       -9.93
##  2 Hawaiian Airlines Inc.     -6.92
##  3 American Airlines Inc.      0.364
##  4 Delta Air Lines Inc.        1.64
##  5 Virgin America              1.76
##  6 US Airways Inc.             2.13
##  7 United Air Lines Inc.       3.56
##  8 Endeavor Air Inc.           7.38
##  9 JetBlue Airways             9.46
## 10 Southwest Airlines Co.      9.65
```

A real world example

Look at the code below. What does it do exactly? Try to understand each line of code.

```
# create the download path for the zip file
# we will download the file into the temporary directory of your computer

path <- file.path(tempdir(),"intrvw19.zip")

# downloading the zip file if it does not exist in the temporary folder yet

url <- "https://www.bls.gov/cex/pumd/data/comma/intrvw19.zip"

if(! file.exists(path)) {

download.file(url=url,
              destfile=path,
              mode="wb",
              method="libcurl")
}


# unzip the files containing the string "fmli" in the name into the temporary directory

files <- unzip(path,list=TRUE)
```

```r
files <- files[grepl("fmli",files$Name),]$Name


unzip(path,
      files=files,
      #exdir="./data/rds",
      exdir=tempdir(),
      junkpaths=TRUE)

# read in the household file for 2019 Q4

household <- read_csv(file.path(tempdir(),"fmli194.csv")) %>%

    as.tibble() %>%
    # Change all variable names to lower case
    rename_all(tolower)


# unzip the files containing the string "memi" in the name into the temporary directory

files <- unzip(path,list=TRUE)

files <- files[grepl("memi",files$Name),]$Name


unzip(path,
      files=files,
      #exdir="./data/rds",
      exdir=tempdir(),
      junkpaths=TRUE)

# read in the person file for 2019 Q4

person <- read_csv(file.path(tempdir(),"memi194.csv")) %>%

    as.tibble() %>%
    # Change all variable names to lower case
    rename_all(tolower)
```

We (hopefully) just downloaded and unzipped two files of the American Consumer Expenditure Survey. If this does not work please let me know on teams. For more information see: https://www.bls.gov/cex/pumd.htm

The files contain **A LOT** of information about US household spending. The survey is conducted quarterly and each quarterly data file is representative for the whole US population.

We now should have two objects in RAM:

- The `household` object containing information about the whole household for Quarter 4 of 2019
- The `person` object containing information about the household members for Quarter 4 of 2019

Now have a look at the objects. As you can see there are a lot of variables in both object. To find more information about these variables consult the codebook: https://www.bls.gov/cex/pumd/ce_pumd_interview_diary_dictionary.xlsx

Average consumption per tenure status

We want to calculate the population weighted average per capita consumption (consumption per household member) for all the different tenure status category for Quarter 4

The goal of this exercise is to write a sequence of functions using the pipe operator %>% for the following data wrangling steps:

Select the following variables from the household object:

- Household ID: **newid**
- Total expenditure current quarter: **totexpcq**
- Household Size: **fam_size**
- Tenure status: **cutenure**
- Houshold weight: **finlwt21**

Calculate the per capita consumption per household

Create a new column called "exp_pc" using the variables *totexpcq* and *fam_size*.

Recode the tenure status variable using the case_when() command from dplyr

Use the codes shown below and read up on the command here: https://dplyr.tidyverse.org/reference/case_when.html

- Owned with mortgage = 1
- Owned without morgage = 2,3
- Rented = 4,6
- Occupied without payment of rent = 5

Calculate the population weighted mean per capita consumption per tenure status

Use this formula to calculate the population weighted mean per tenure status. $\bar{X} = \frac{\sum_{i=1}^{n} x_i * w_i}{\sum_{i=1}^{n} w_i}$

- Use the variable *finlwt21* as population weight.
- Do not use packages, do it on your own using dplyr synthax.
- How can the variable *finlwt21* be interpreted?

```
household_clean <- household %>%
  select(newid, totexpcq, fam_size, cutenure, finlwt21) %>%
  mutate(exp_pc = (totexpcq / fam_size)) %>%
  mutate (cutenure = case_when (cutenure == 1 ~ "Owned with mortgage",
    cutenure == 2 | cutenure == 3 ~ "Owned without mortgage",
    cutenure == 4 | cutenure == 6 ~ "Rented",
    cutenure == 5 ~ "Occupied without payment of rent")) %>%
  group_by(cutenure) %>%
  mutate(pop_weighted_mean =  weighted.mean(exp_pc,finlwt21))

head(household_clean)
```

```
## # A tibble: 6 x 7
## # Groups:   cutenure [3]
##   newid    totexpcq fam_size cutenure           finlwt21 exp_pc pop_weighted_me~
##   <chr>       <dbl>    <dbl> <chr>                 <dbl>  <dbl>            <dbl>
## 1 04069474        0        3 Owned with mortga~   36019.      0            2634.
## 2 04069484        0        2 Owned without mor~    2990.      0            2251.
## 3 04069524        0        4 Owned with mortga~   25697.      0            2634.
```

```
## 4 04069534        0        2 Owned without mor~   18046.      0            2251.
## 5 04069564        0        4 Rented               21586.      0            2018.
## 6 04069584        0        1 Owned with mortga~   19245.      0            2634.
```

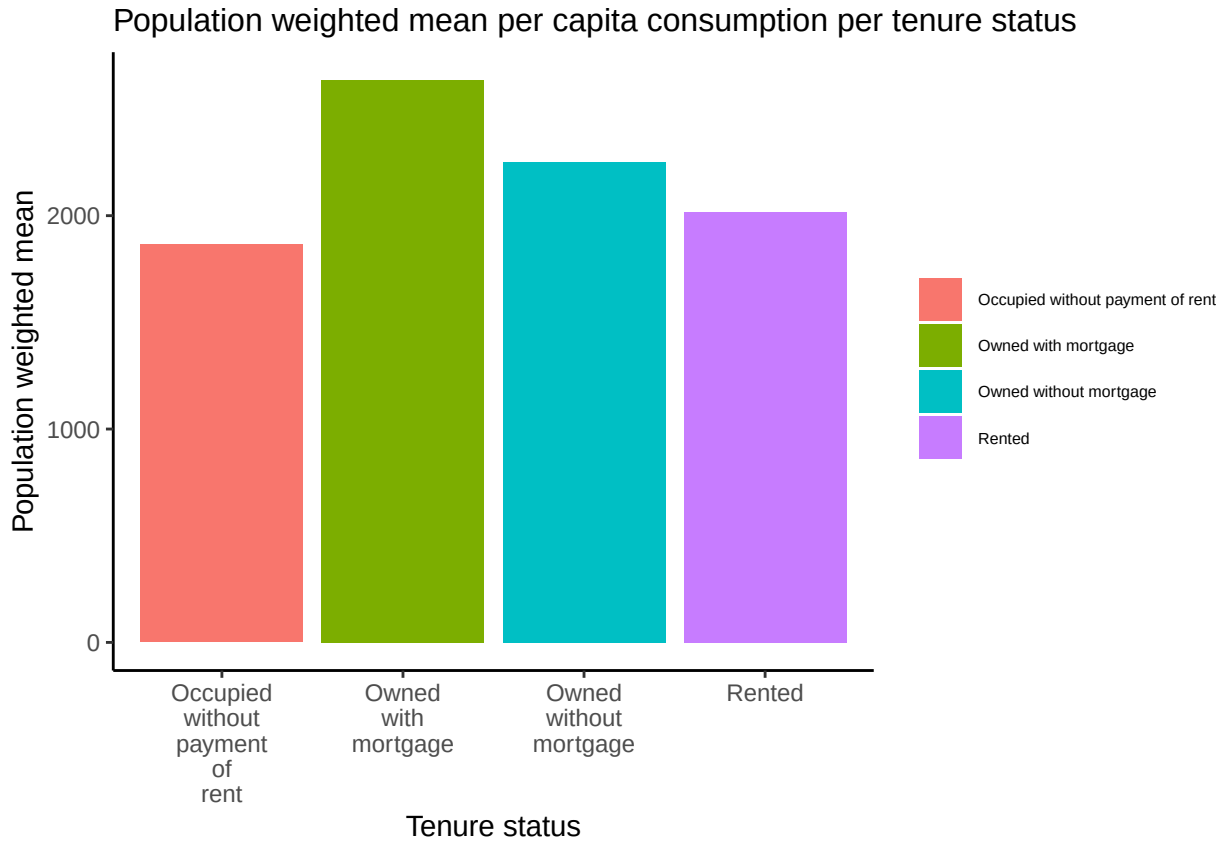
Create a bar plot that compares the population weighted mean per capita consumption per tenure status.

Now use your result from above to create a bar plot using ggplot2. The nicer it looks the better.

```
addline_format <- function(x,...){
    gsub('\\s','\n',x)
}

household_clean %>% ggplot +
  geom_bar(mapping = aes(x = cutenure, y =pop_weighted_mean, fill = cutenure), stat = "summary") +
  labs(title = "Population weighted mean per capita consumption per tenure status", x = "Tenure status"
  theme_classic() +
  theme(legend.title=element_blank()) +
  theme(plot.title = element_text(
          size = 12,
          hjust = 0)) +
  theme(legend.text = element_text(size = 6)) +
  scale_x_discrete(breaks=unique(household_clean$cutenure), labels=addline_format(c("Owned with mortgag
```

```
## No summary function supplied, defaulting to `mean_se()`
```



Average age per tenure status

We now want to calculate the average age per tenure status. For this we need to calculate the average age per household member and join this information to the household file. Do not bother with any population weighting for this data wrangling step.

Calculate the average age per household using the person file

The age variable is called "age" in the person object.

Join the average age per household to the household file

The foreign key is called newid

Calculate the average age per tenure status

Use the population weights for this step

```
person_clean <- person %>% group_by(newid) %>% mutate(average_age = mean(age))

household_clean <- (left_join(household_clean, person_clean, by="newid"))

household_clean <- household_clean %>%
    select(newid, totexpcq, fam_size, cutenure, finlwt21, average_age, pop_weighted_mean, exp_pc) %>%
    group_by(cutenure) %>%
    mutate(age_weighted_mean = weighted.mean(average_age,finlwt21))
head(household_clean)
```

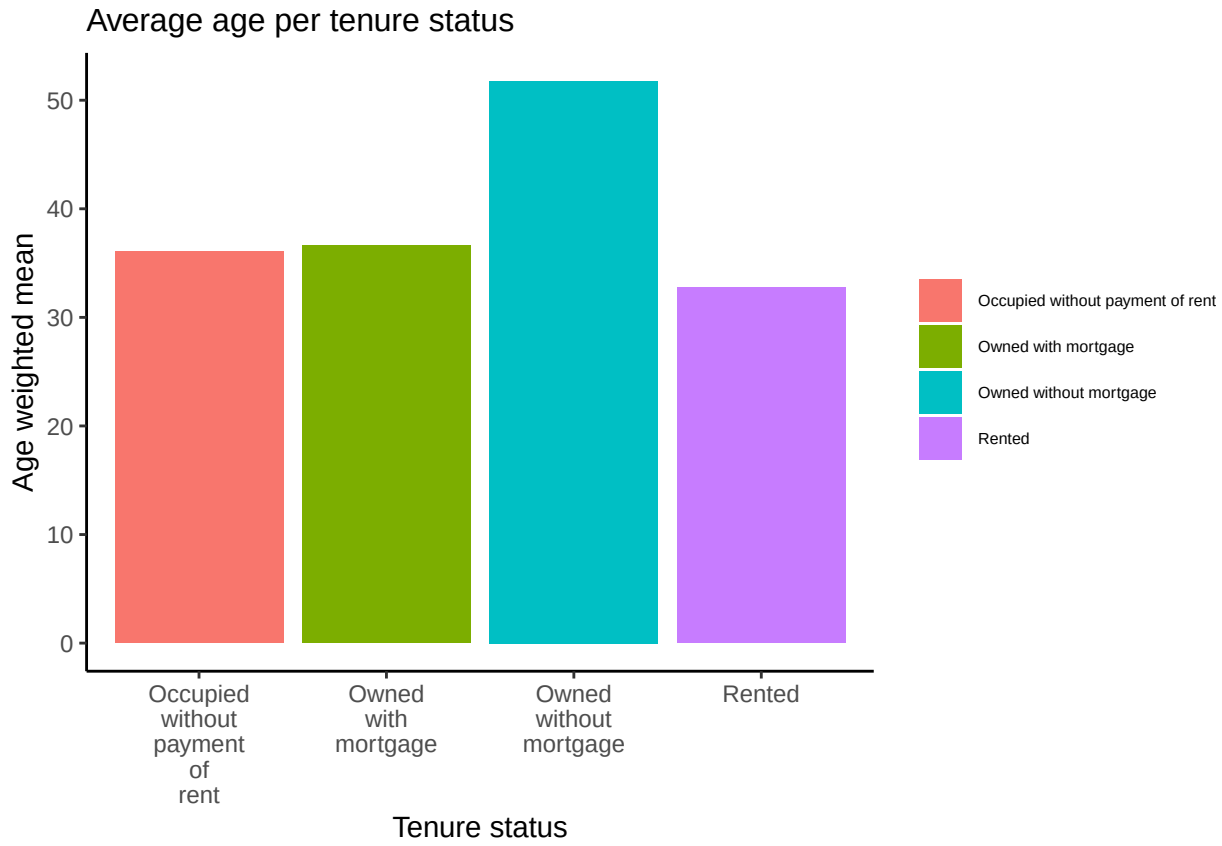
```
## # A tibble: 6 x 9
## # Groups:   cutenure [2]
##   newid totexpcq fam_size cutenure finlwt21 average_age pop_weighted_me~ exp_pc
##   <chr>    <dbl>    <dbl> <chr>       <dbl>       <dbl>            <dbl>  <dbl>
## 1 0406~        0        3 Owned w~   36019.        42              2634.      0
## 2 0406~        0        3 Owned w~   36019.        42              2634.      0
## 3 0406~        0        3 Owned w~   36019.        42              2634.      0
## 4 0406~        0        2 Owned w~    2990.        52.5            2251.      0
## 5 0406~        0        2 Owned w~    2990.        52.5            2251.      0
## 6 0406~        0        4 Owned w~   25697.        22.5            2634.      0
## # ... with 1 more variable: age_weighted_mean <dbl>
```

Create a bar plot that compares the average age per tenure status

Now use your result from above to create a bar plot using ggplot2. The nicer it looks the better.

```
household_clean %>% ggplot +
  geom_bar(mapping = aes(x = cutenure, y = age_weighted_mean, fill = cutenure), stat = "summary") +
  labs(title = "Average age per tenure status", x = "Tenure status", y = "Age weighted mean") +
  theme_classic() +
  theme(legend.title=element_blank()) +
  theme(plot.title = element_text(
           size = 12,
           hjust = 0)) +
  theme(legend.text = element_text(size = 6)) +
  scale_x_discrete(breaks=unique(household_clean$cutenure), labels=addline_format(c("Owned with mortgag
```

```
## No summary function supplied, defaulting to `mean_se()`
```



Calculate the population weighted per capita consumption per average household age

- Round the average age per household to integer values
- Use the split apply combine approach to calculate the population weighted per captia consumption per average household age
- Arrange the results in an descending order
- Display only the Top 10 results

```
household_clean <- household_clean %>%
  mutate(average_age = ceiling(average_age)) %>%
  group_by(average_age) %>%
  mutate(Pwccpaha = weighted.mean(exp_pc,finlwt21)) %>%
  arrange(desc(Pwccpaha))

  household_clean %>% head(10)
```

```
## # A tibble: 10 x 10
## # Groups:   average_age [1]
##    newid totexpcq fam_size cutenure finlwt21 average_age pop_weighted_me~ exp_pc
##    <chr>    <dbl>    <dbl> <chr>       <dbl>       <dbl>            <dbl>  <dbl>
##  1 0407~       0         1 Owned w~   19762.          83            2251.     0
##  2 0410~    2046         1 Owned w~   24033.          83            2251.  2046
##  3 0417~       0         1 Owned w~   35080.          83            2251.     0
##  4 0418~       0         1 Rented     21949.          83            2018.     0
##  5 0418~       0         1 Owned w~   20885.          83            2251.     0
##  6 0418~    2441.        1 Owned w~   15392.          83            2251.  2441.
##  7 0419~    2714.        1 Rented     20208.          83            2018.  2714.
##  8 0419~   18393.        1 Owned w~   24239.          83            2251. 18393.
```

```
##  9 0421~       0         1 Owned w~   23291.          83            2634.     0
## 10 0422~       0         1 Owned w~   19913.          83            2251.     0
## # ... with 2 more variables: age_weighted_mean <dbl>, Pwccpaha <dbl>
```

Create an appropriate plot to display the data

- Use the results from above and plot the data
- What do you conclude about the functional form of this relationship?
- Run a regression to test your hypothesis

```
household_clean %>% ggplot +
  geom_bar(mapping = aes(x = cutenure, y = Pwccpaha, fill = cutenure), stat = "summary") +
  labs(title = "Population weighted per capita consumption per average household age", x = "Tenure statu
  theme_classic() +
  theme(legend.title=element_blank()) +
  theme(plot.title = element_text(
           size = 12,
           hjust = 0)) +
  theme(legend.text = element_text(size = 6)) +
  scale_x_discrete(breaks=unique(household_clean$cutenure), labels=addline_format(c("Owned with mortgag
```

```
## No summary function supplied, defaulting to `mean_se()`
```