ECON 590: Class 1 Activity Solutions

Using `R` Notebooks

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
# You can write and run sample code here

x <- 2

x**3
```

```
## [1] 8
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Load and Subset Sample Data using `dplyr`

Now we want to demonstrate several data cleaning tasks using `dplyr` functions. We'll start by removing variables from our data set, then demonstrate how to create a subset of the data (a limited set of observations from our broader data set, selected based on some critieria).

```
# Load the tidyverse package

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.4     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

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

```
# Load .csv file stored via course GitHub

sample.data <- read.csv(url("https://raw.githubusercontent.com/mackaytc/econ-590-resources/main/data/ACS

# We want to make the data set a bit smaller and more manageable for today.
# Let's start by using the select command to keep only "year", "state", and
# "incwage" variables

sample.data <- select(sample.data, year, state, incwage)

# The head() command will show us a snapshot of what the data set looks like

head(sample.data)
```

```
##   year        state incwage
## 1 2004     new york   10000
## 2 2004      florida   35000
## 3 2004      florida       0
## 4 2004 north dakota     300
## 5 2004      alabama     500
## 6 2004      indiana       0
```

```
# Suppose we wanted to subset the data so that we only kept observations that
# lived in New York. To do this, we can use the filter() function to create a
# new object "sample.data.NY" that stores our subsetted data.

sample.data.NY <- filter(sample.data, state == "new york")

# Take a look at the new data set stored in the "Environment" window. Are all
# the observations from New York?

head(sample.data.NY)
```

```
##   year    state incwage
## 1 2004 new york   10000
## 2 2004 new york  999999
## 3 2004 new york   10000
## 4 2004 new york       0
## 5 2004 new york  999999
## 6 2004 new york    2500
```

```
# We can also use filter() to select observations by numeric values. This time,
# we won't create a new object - we'll just output the result using head().

head(filter(sample.data, incwage > 20000))
```

```
##   year    state incwage
## 1 2004  florida   35000
```

```
## 2 2004     iowa   30000
## 3 2004 delaware   45000
## 4 2004   oregon   40500
## 5 2004  arizona   69000
## 6 2004 michigan   63000
```

Practice Problems

Using our sample data set, answer the following questions. Before you get started on the activity, take a look at the `dplyr` cheat sheet posted on the course GitHub page.

1) Using the `mean()` function, report the average wage income for everyone in our sample

```
# Base-R approach

mean(sample.data$incwage)
```

```
## [1] 216472.7
```

```
# tidyverse using the summarize() function

incwage.stats <- summarize(sample.data, avg.incwage = mean(incwage))

# summarize() returns a data object that can store multiple summary stats (we'll
# see why this is useful later). In this case, its just storing mean wages,
# which we've named "avg.incwage". You can print the resulting data object:

incwage.stats
```

```
##   avg.incwage
## 1    216472.7
```

```
# We can all print just the value of mean wages, using the $-notation below
# after the variable name to return the average value of incwage

incwage.stats$avg.incwage
```

```
## [1] 216472.7
```

We mentioned in class that wages in the ACS are *top-coded* - income over a certain level isn't reported. Instead, the ACS reports an income value of `incwage = 999999`. We can set those values equal to missing (`NA`) and see how this changes the mean.

```
# Set top-coded wages equal to missing using the replace() function

sample.data.missings <- sample.data %>%
  mutate(incwage = replace(incwage, incwage == 999999, NA))

# Now we need to use the "na.rm = TRUE" option to tell R that we want to ignore
# missing / NA values of incwage when calculating average wages

mean(sample.data.missings$incwage, na.rm = T)
```

```
## [1] 26674.4
```

2) Using the `filter()` function, report the average wage income for everyone in our sample who lives in Florida

```
# Using tidyverse and %>% operator -- note that we'll use the data object with
# missing income values sample.data.missings

FL.incwage.stats <- sample.data.missings %>%        # Start with sample data...
  filter(state == "florida")             %>%        # keep only FL observations
  summarize(avg.wage = mean(incwage, na.rm = TRUE)) # compute average income

# summarize() stores data objects. Again, you can print the resulting data object

FL.incwage.stats
```

```
##   avg.wage
## 1 20443.17
```

```
# Or you can add `$mean.wage` to print just the numeric value of mean wages

FL.incwage.stats$avg.wage
```

```
## [1] 20443.17
```

3) Report the average wage income from observations who were surveyed between the years 2004 and 2008.

HINT: You can use `&` to add multiple restrictions in your `filter()` command.

```
# One way - use filter() to create a new data object with obsevations only from
# the years 2004 to 2008, then take the average of the incwage variable

data.04.08 <- filter(sample.data.missings, year <= 2008 & year >= 2004)

# Base-R syntax

mean(data.04.08$incwage, na.rm = TRUE)
```

```
## [1] 26008.1
```

```
# Another way - skip creating a new intermediate data object and just directly
# calculate average wages using the %>% operator

incwage.04.08.stats <- sample.data.missings %>%
  filter(year <= 2008 & year >= 2004)    %>%
  summarize(avg.wage = mean(incwage, na.rm = TRUE))

# tidyverse syntax using summarize function -- returns a data object named
# incwage.04.08.stats - the specific value we want is avg.wage, so we can
# type the following:

incwage.04.08.stats$avg.wage
```

```
## [1] 26008.1
```

4) Report the average wage for everyone who lived in Florida, Alabama, or Georgia.

HINT: You could use the `&`-syntax from (3) above, but that would get cumbersome if you had a longer list of states. Instead, take a look how we defined `b.vector` above. You can create a similar object named state.list that stores the values of the three states (with each state name in parentheses and separated by commas). Because we're now telling R to look over a list of states as opposed to a single state, use `%in%` instead of `==`.

```
# Define the list of states we want to include in our sub-sample

state.list <- c("florida", "alabama", "georgia")

# Tell R to filter the data by looking for every observation that lived in one
# of the states listed in the state.list vector using (using %in% instead of ==)

data.FL.AL.GA <- filter(sample.data.missings, state %in% state.list)

# Now calculate average wages

mean(data.FL.AL.GA$incwage, na.rm = T)
```

```
## [1] 21445.06
```

5) Using the `dplyr` cheat sheet on GitHub, calculate the median, minimum and maximum values of the `incwage` variable

```
# In the example below, we can why using summarize() and returning a data object
# of results is useful - here, summary.stats gives us a data object of all the
# summary stats we want to calculate.

summary.stats <- sample.data.missings %>%
  summarize(min.wage = min(incwage, na.rm = TRUE),
            max.wage = max(incwage, na.rm = TRUE),
            med.wage = median(incwage, na.rm = TRUE))

# We can then either print the full set of summary stat values

summary.stats
```

```
##   min.wage max.wage med.wage
## 1        0   710000     8600
```

```
# Or print specific values as follows

summary.stats$min.wage # minimum wage across the entire sample
```

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

```
summary.stats$max.wage # maximum wage
```

```
## [1] 710000
```

```
summary.stats$med.wage # median wage
```

```
## [1] 8600
```

6) Calculate the average wage for each year in the data.

HINT: Take a look at the Group Data section of the `dplyr` cheat sheet. We want to group the data by year, then calculate average wages using the `summarise()` command.

```
# Now we can see another use for the summarize() function. We can use group_by()
# to tell R that we want to calculate summary stats by year.

yearly.summary.stats <- sample.data.missings %>%
  group_by(year) %>%
  summarize(avg.annual.wage = mean(incwage, na.rm = TRUE))

# Now our summary stats data object stores mean values by year

yearly.summary.stats
```

```
## # A tibble: 16 x 2
##     year avg.annual.wage
##    <int>           <dbl>
##  1  2004          25235.
##  2  2005          23109.
##  3  2006          23342.
##  4  2007          27802.
##  5  2008          30483.
##  6  2009          24944.
##  7  2010          28044.
##  8  2011          20775.
##  9  2012          18968.
## 10  2013          24375.
## 11  2014          27391.
## 12  2015          27854.
## 13  2016          28262.
## 14  2017          39553.
## 15  2018          28476.
## 16  2019          27621.
```