Data Manipulation - Level 2

Amy Ly

4/30/2022

```
flights <- get(data("flights"))
```

Learn

Section 5.2

Find all the flights:

- Were operated by United, American, or Delta

```
flights %>%
  filter(carrier == "UA" | carrier == "AA" | carrier == "DL") %>%
  nrow()
```

```
## [1] 139504
```

There are 139504 flights operated by United, American, or Delta.

- Arrived more than two hours late, but didn't leave late

```
flights %>%
  filter(arr_delay >= 120 & dep_delay <=0) %>%
  nrow()
```

```
## [1] 29
```

There were 29 flights that arrived more than 2 hours late, but didn't leave late.

- Were delayed by at least an hour, but made up over 30 minutes in flight

```
flights %>%
  filter(arr_delay >= 60, dep_delay - arr_delay > 30) %>%
  nrow()
```

```
## [1] 1084
```

There were 1084 flightst that were delayed by at least an hour, but made up 30 minutes in flight.

Section 5.3

Which flights traveled the farthest? Which traveled the shortest?

```
head(arrange(flights, desc(distance)))
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      857            900        -3     1516           1530
## 2  2013     1     2      909            900         9     1525           1530
## 3  2013     1     3      914            900        14     1504           1530
## 4  2013     1     4      900            900         0     1516           1530
## 5  2013     1     5      858            900        -2     1519           1530
## 6  2013     1     6     1019            900        79     1558           1530
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
head(arrange(flights, distance))
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     7    27       NA            106        NA       NA            245
## 2  2013     1     3     2127           2129        -2     2222           2224
## 3  2013     1     4     1240           1200        40     1333           1306
## 4  2013     1     4     1829           1615       134     1937           1721
## 5  2013     1     4     2128           2129        -1     2218           2224
## 6  2013     1     5     1155           1200        -5     1241           1306
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

The longest flight is HA 51, JFK to HNL at 4,983 miles. The shortest flight is US 1632 from EWR to LGA at 17 miles.

Section 5.4

One way to select the variables dep_time, dep_delay, arr_time, and arr_delay from flights is:

```
flights %>% select(dep_time, dep_delay, arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
```

```
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

How could you select the same columns with starts_with()?

```
flights %>% select(starts_with("dep_"), starts_with("arr_"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

Section 5.5

Add a column to flights that is the arrival delay (arr_delay) as a percentage of the total flight time (air_time).

```
flights %>%
  mutate(pct_arr = 100*arr_delay/air_time) %>%
  head()
```

```
## # A tibble: 6 x 20
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 12 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, pct_arr <dbl>
```

Section 5.6

- Find the best traveled plane. For each plane (tailnum), find the number of unique destinations that it flies to. Which plane(s) flew to the most destinations?

```
df <- flights %>%
  group_by(tailnum) %>%
  summarise(count = n_distinct(dest))

df$tailnum[max(df$count)]
```

```
## [1] "N12172"
```

The plane that flew the most destination is N12172.

- What is the most popular route? For each route (combination of origin and dest), find the number of flights made. Which route has the highest number of flights?

```
flights %>%
  group_by(origin, dest) %>%
  summarise(count = n_distinct(flight)) %>%
  filter(count == max(count))
```

```
## `summarise()` has grouped output by 'origin'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 3 x 3
## # Groups:   origin [3]
##   origin dest  count
##   <chr>  <chr> <int>
## 1 EWR    IAH     453
## 2 JFK    LAX     123
## 3 LGA    IAH     381
```

The route with the most flight is EWR to IAH with a count of 453 flights.

- Compare the carriers' arrival delays for flights to Atlanta. For flights to "ATL", for each carrier, calculate the average arrival delay, and the 10th and 90th percentiles of arrival delay. Which carrier has the lowest average delay? Which carrier has the smallest spread between 10th and 90th percentiles of arrival delay?

```
flights %>%
  filter(dest == "ATL") %>%
  group_by(carrier) %>%
  summarize(avg = mean(arr_delay, na.rm = TRUE),
            tenth = quantile(arr_delay, 0.1, na.rm=TRUE),
            nintieth = quantile(arr_delay, 0.9, na.rm=TRUE),
            spread = nintieth-tenth)
```

```
## # A tibble: 7 x 5
##   carrier    avg tenth nintieth spread
##   <chr>    <dbl> <dbl>    <dbl>  <dbl>
## 1 9E       0.857 -18.5     20     38.5
## 2 DL       7.42  -21       41     62
## 3 EV      19.6   -20       82    102
```

```
## 4 FL      20.7   -15       70     85
## 5 MQ      14.0   -17       57     74
## 6 UA      10.5   -21.9     59.4   81.3
## 7 WN       6.90   -7.6     23.9   31.5
```

The carrier with the lowest average delay is 9E. The carrier with the smallest spread between the 10th and 90th percentiles of the average delay is WN.

Apply

```
# Data Import --------------------------------------------------------------
url <- "https://fred.stlouisfed.org/data/"

# U.S. Bureau of Labor Statistics, All Employees, Total Nonfarm [PAYNSA],
# retrieved from FRED, Federal Reserve Bank of St. Louis;
# https://fred.stlouisfed.org/series/PAYNSA, April 29, 2020.
employed_file <- "PAYNSA.txt"

# only download if it isn't already here
if(!file.exists(employed_file)){
  download.file(paste0(url, employed_file), employed_file,
    mode = "wb")
}

employed <- read_table2(employed_file,
  col_names = c("date", "employed"), skip = 39)
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## Please use `read_table()` instead.
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   date = col_date(format = ""),
##   employed = col_double()
## )
```

```
#creates the dataframe for all employees. Note y-axis scale is in thousands of persons

# Organization for Economic Co-operation and Development,
# Working Age Population: Aged 15-64:
# All Persons for the United States [LFWA64TTUSM647N],
# retrieved from FRED, Federal Reserve Bank of St. Louis;
# https://fred.stlouisfed.org/series/LFWA64TTUSM647N, April 30, 2020.
pop_file <- "LFWA64TTUSM647N.txt"

# only download if it isn't already here
if(!file.exists(pop_file)){
  download.file(paste0(url, pop_file), pop_file,
    mode = "wb")
}
```

```
pop <- read_table2(pop_file,
  col_names = c("date", "pop"), skip = 18)
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## Please use `read_table()` instead.
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   date = col_character(),
##   pop = col_character()
## )
```

```
#removed the first line of the dataframe
#note that this is the number of people in population 15-64 for US (all persons who are of working age)
#note that the y axis units are in scale of millionss of persons

pop <- pop[-1,]

#converted the dates from characters to dates so the dataframes are compatible

pop$date <- as.Date(pop$date)
pop$pop <- as.numeric(pop$pop)

employed$employed <- employed$employed*1000

# Put employed and population together
pop_emp <- full_join(pop, employed) %>%
  arrange(date)
```

```
## Joining, by = "date"
```

Use mutate() to create a new column employed_scaled with an approximation of the total number of employed people (careful with units!), and percent_emp with a percentage of the working population who are employed. The resulting values of percent_emp should be in a similar range to those in Spain, e.g. 50-60%.

```
pop_emp <- pop_emp %>%  replace(is.na(.), 0) %>%
  mutate(employed_scaled = sum(employed),
                   percent_emp = (100*employed/pop))

pop_emp
```

```
## # A tibble: 999 x 5
##    date         pop employed employed_scaled percent_emp
##    <date>     <dbl>    <dbl>           <dbl>       <dbl>
##  1 1939-01-01     0 29296000     90456828000         Inf
##  2 1939-02-01     0 29394000     90456828000         Inf
##  3 1939-03-01     0 29804000     90456828000         Inf
##  4 1939-04-01     0 29786000     90456828000         Inf
##  5 1939-05-01     0 30145000     90456828000         Inf
```

```
##  6 1939-06-01     0 30520000     90456828000         Inf
##  7 1939-07-01     0 30472000     90456828000         Inf
##  8 1939-08-01     0 30870000     90456828000         Inf
##  9 1939-09-01     0 31608000     90456828000         Inf
## 10 1939-10-01     0 31975000     90456828000         Inf
## # ... with 989 more rows
```

```
#what is employed_scaled supposed to be?
```

Filter pop_emp to just dates after January 2002, call this pop_emp_recent.

```
pop_emp_recent <- pop_emp %>%
  filter(date >= "2002-01-01")
```

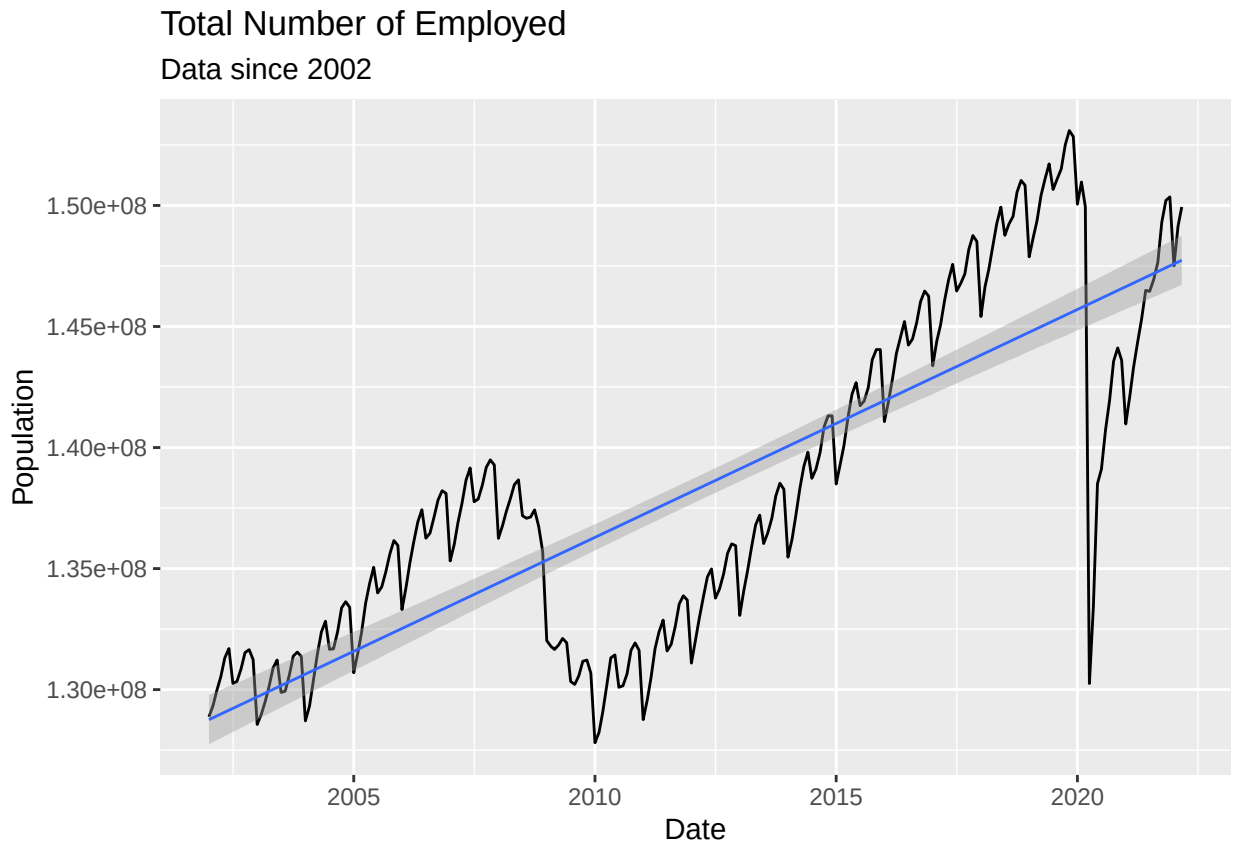
Recreate the three plots in Figure 8.4: the total number of employed, the working population, and the percent employed since 2002. How do the patterns in the U.S. compare to the patterns in Spain? You can add simple linear regression lines with geom_smooth(method = "lm").

```
ggplot
```

```
## function (data = NULL, mapping = aes(), ..., environment = parent.frame())
## {
##     UseMethod("ggplot")
## }
## <bytecode: 0x0000000020e18728>
## <environment: namespace:ggplot2>
```

```
ggplot(pop_emp_recent, aes(x=date, y=employed))+
  geom_line()+
  geom_smooth(method = "lm", size =0.5)+
  labs(y = "Population",
       x = "Date",
       title = "Total Number of Employed",
       subtitle = "Data since 2002")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

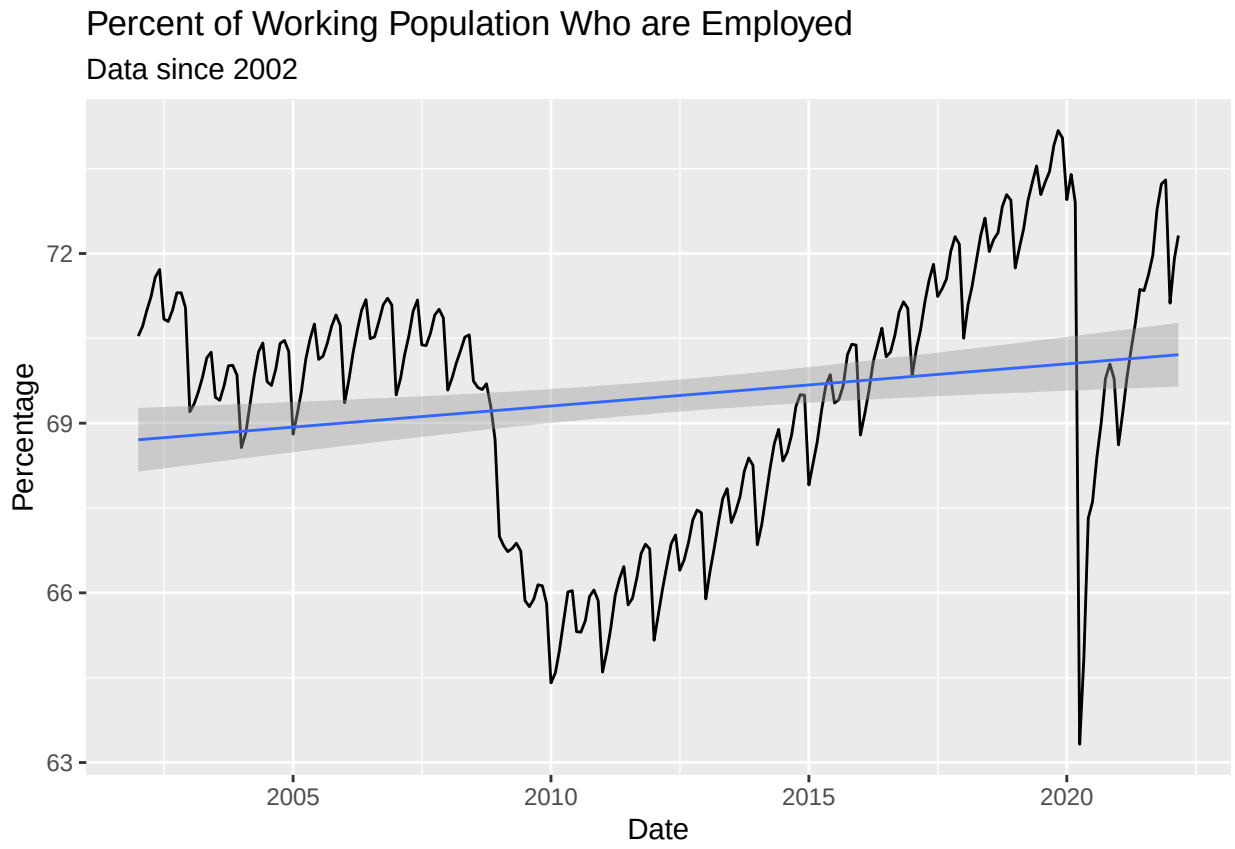



```
ggplot(pop_emp_recent, aes(x=date, y=pop))+
  geom_line()+
  geom_smooth(method = "lm", size =0.5)+
  labs(y = "Population",
       x = "Date",
       title = "Population of People of Working Age (15-64 year olds)",
       subtitle = "Data since 2002")
```

```
## `geom_smooth()` using formula 'y ~ x'
```


```
ggplot(pop_emp_recent, aes(x=date, y=percent_emp))+
  geom_line()+
  geom_smooth(method = "lm", size =0.5)+
  labs(y = "Percentage",
       x = "Date",
       title = "Percent of Working Population Who are Employed",
       subtitle = "Data since 2002")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Add a column to pop_emp_recent containing the average percent employed for the whole period in pop_emp_recent and use it to create Figure 8.5.

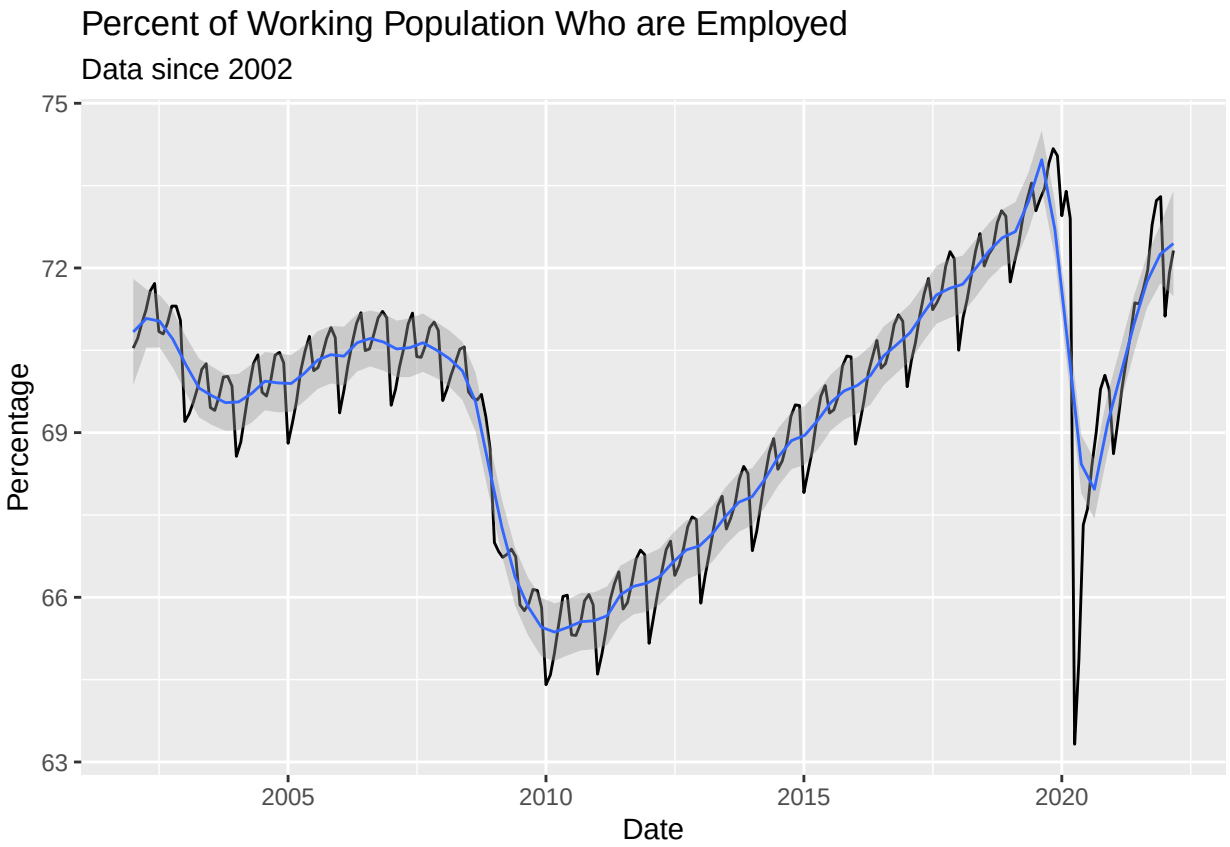
```
updated_pop <- pop_emp_recent %>%
  mutate(average = mean(percent_emp),
         diff = percent_emp-average)

ggplot(updated_pop, aes(x=date, y=diff))+
  geom_line()+
  geom_hline(yintercept=0, color = "red") +
  labs(y = "Percentage",
       x = "Date",
       title = "Percent of Working Population Who are Employed",
       subtitle = "Difference between each score and the average of the period (69.5%)")
```

Add a loess smooth to your average percent employed plot to create Fig 8.6 with geom_smooth(). Make sure you adjust the span argument to geom_smooth() to get a smooth that follows the general trend quite well.

```
ggplot(pop_emp_recent, aes(x=date, y=percent_emp))+
  geom_line()+
  geom_smooth(method = "loess", size =0.5, span =0.1)+
  labs(y = "Percentage",
       x = "Date",
       title = "Percent of Working Population Who are Employed",
       subtitle = "Data since 2002")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Instead of using a loess smooth, you should really calculate a moving average. The function rollmean() in the zoo package can calculate moving averages, for example: rollmean(x, 12, fill = NA)

will calculate a 12-observation moving average. Combine this with mutate() to add a percent_ma column for a 12-month moving average of the percent employed. Then repeat Fig 8.6 using this calculated column instead.

```
updated_pop %>%
  mutate(percent_ma = rollmean(percent_emp, 12, fill = NA)) %>%
  ggplot(aes(x=date, y=percent_ma))+
  geom_line()+
  labs(y = "Percentage",
       x = "Date",
       title = "Percent of Working Population Who are Employed",
       subtitle = "Moving Average")
```

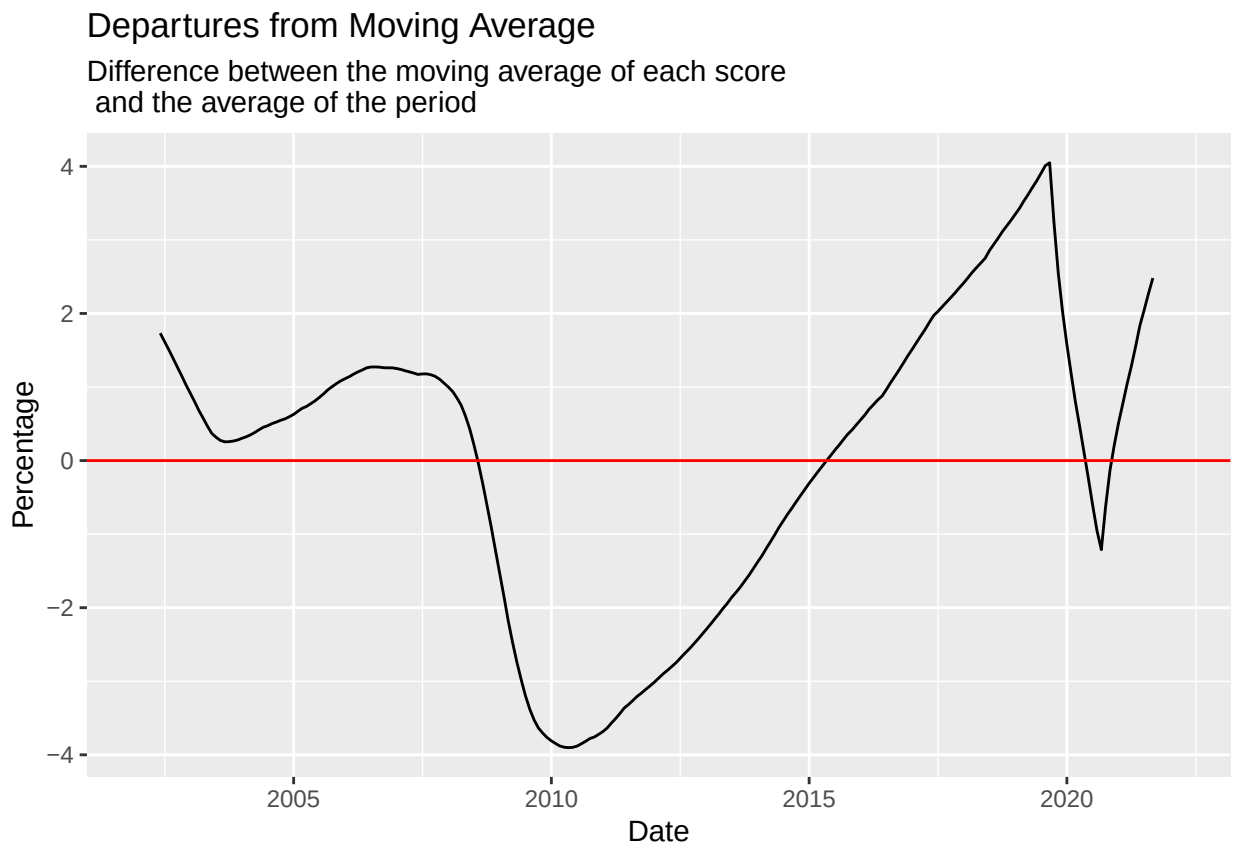
```
## Warning: Removed 11 row(s) containing missing values (geom_path).
```

Use mutate() again to find the departures from the moving average and create Fig 8.7.

```
updated_pop2 <- pop_emp_recent %>%
  mutate(percent_ma = rollmean(percent_emp, 12, fill = NA),
         avg_ma = mean(percent_ma, na.rm = TRUE),
         diff_ma = percent_ma - avg_ma)

  ggplot(updated_pop2, aes(x=date, y=diff_ma))+
    geom_line()+
  geom_hline(yintercept = 0, color = "red")+
 labs(y = "Percentage",
       x = "Date",
       title = "Departures from Moving Average",
       subtitle = "Difference between the moving average of each score \n and the average of the period
```

```
## Warning: Removed 11 row(s) containing missing values (geom_path).
```



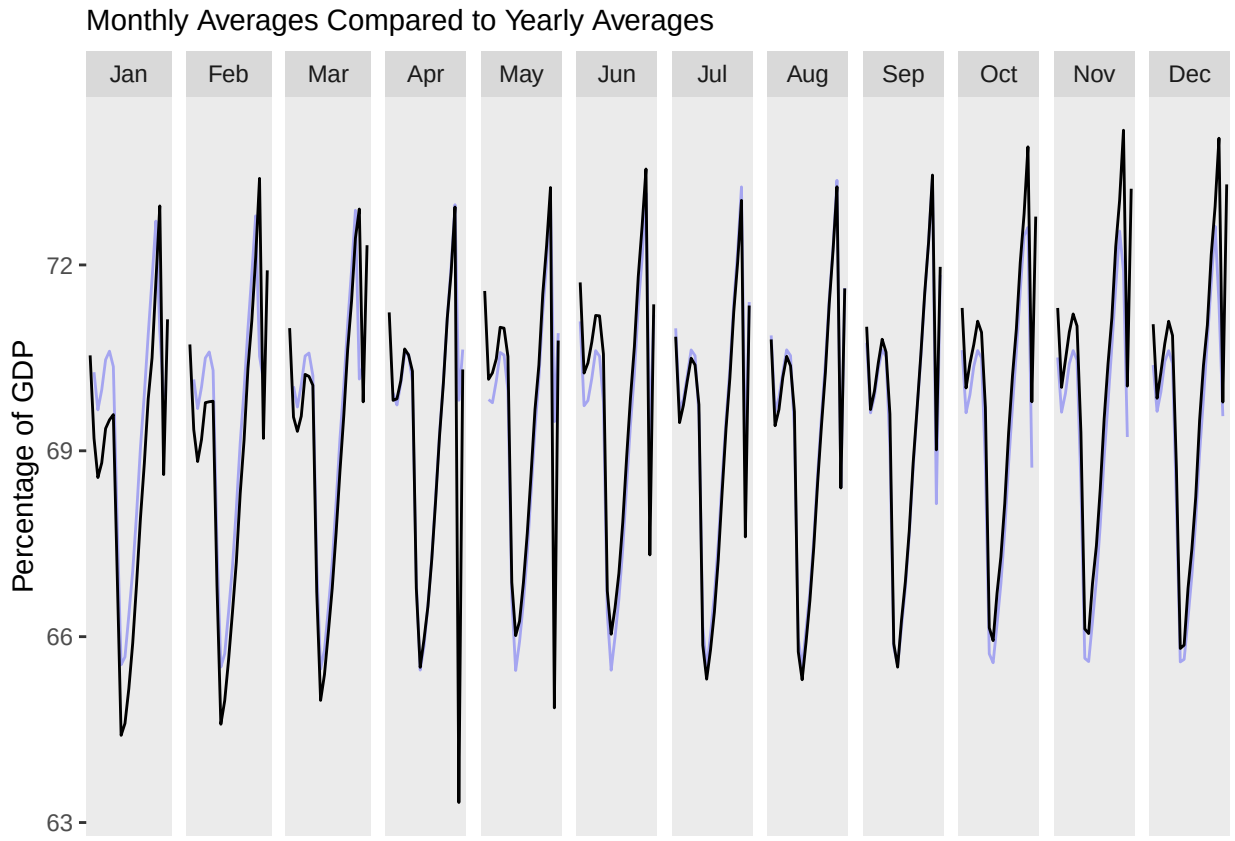
Add the month and year variables with:

```
pop_emp_recent <- pop_emp_recent %>%
  mutate(month = lubridate::month(date, label = TRUE),
         year = lubridate::year(date),
         percent_ma = rollmean(percent_emp, 12, fill = NA),
         avg_ma = mean(percent_ma, na.rm = TRUE),
         diff_ma = percent_ma - avg_ma)
```

Recreate 8.12

```
ggplot(pop_emp_recent, aes(x=date, y=percent_emp)) +
  geom_line(aes(x=date, y=percent_ma), alpha = 0.3, color="blue") +
  geom_line()+
  labs(main = "Population of Working Age who are Employed",
      subtitle = "Monthly Averages Compared to Yearly Averages",
      y = "Percentage of GDP")+
   facet_grid(.~month, scale="free", space="free") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```

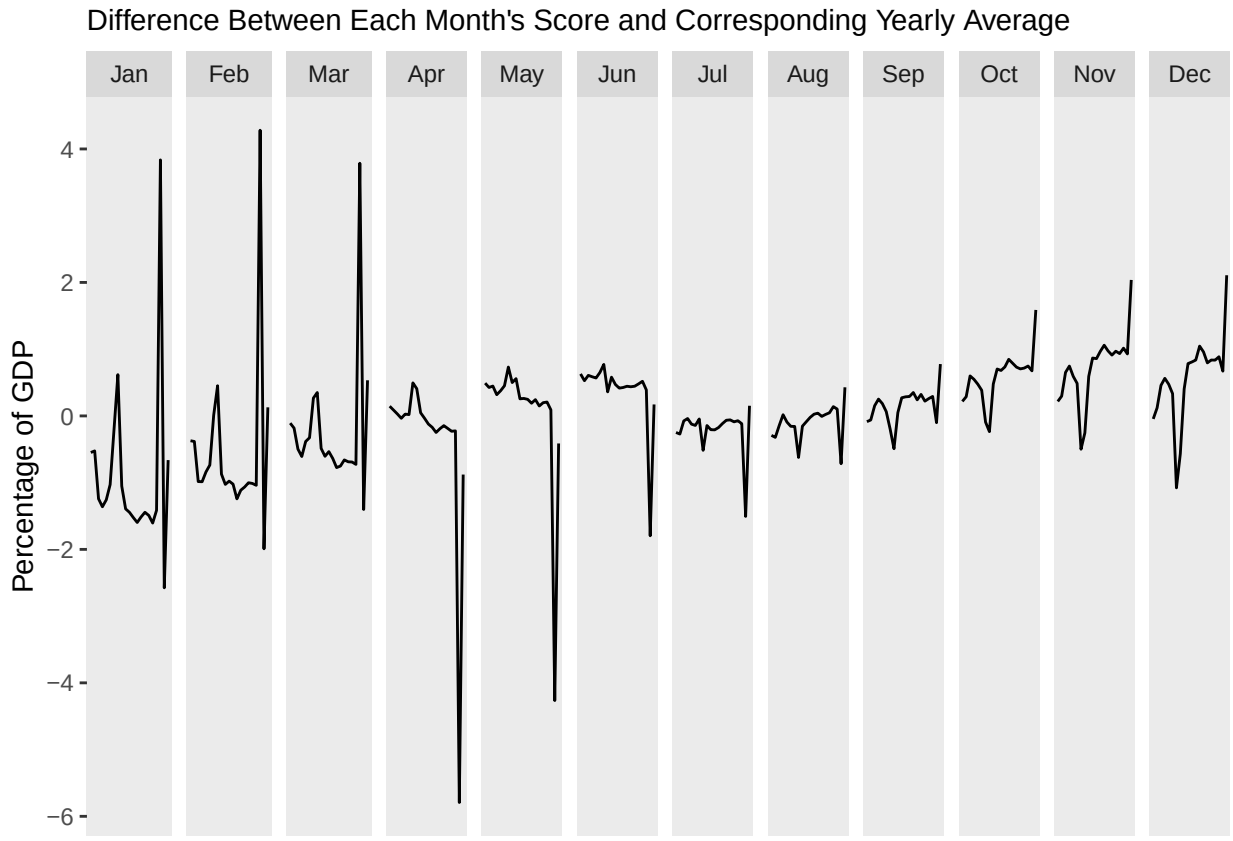
```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



Recreate 8.13 (Hint: you'll need to combine group_by() with mutate())

```
pop_emp_recent2 <- pop_emp_recent %>%
  group_by(year) %>%
  mutate(avg = mean(percent_emp, na.rm=TRUE),
         diff = percent_emp-avg)

ggplot(pop_emp_recent2, aes(x=date, y=diff)) +
  geom_line()+
  labs(main = "Population of Working Age who are Employed",
      subtitle = "Difference Between Each Month's Score and Corresponding Yearly Average",
      y = "Percentage of GDP")+
   facet_grid(.~month, scale="free", space="free") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



(Optional) Challenge Question Recreate Figures 8.9 and 8.10

The forecast package has an implementation of the STL decomposition, that is relatively amenable to manipulation. The following code creates pop_emp_stl with the original columns from pop_emp_recent, and the additional columns Data, Trend and Seasonal12 corresponding to the what Cairo calls "observed", "trend", and "seasonal".

```
library(forecast)
pop_emp_stl <- mstl(ts(pop_emp_recent$percent_emp, freq = 12)) %>%
  as_tibble() %>%
  bind_cols(pop_emp_recent)
```

You'll need to create the residuals yourself, then reshape the data so you can plot each component in a panel with faceting.