Homework 5

The purpose of this homework is to better understand the theory of hypothesis testing, and to practice using `dplyr` to transform data and `ggplot2` to visualize data. Please fill in the appropriate code and write answers to all questions in the answer sections, then submit a compiled pdf with your answers through Gradescope by 11pm on Sunday October 8th.

As always, if you need help with the homework, please attend the TA office hours which are listed on Canvas and/or ask questions on Ed Discussions. Also, if you have completed the homework, please help others out by answering questions on Ed Discussions, which will count toward your class participation grade.

For written response, also please clearly report the value or make sure your codes properly display the output.

The team that made `dplyr` and `ggplot` has also created many learning resources. Some of them are listed below:

- Intro to dplyr vignette
- A conceptual overview from the dplyr home page
- R for Data Science is a book that discusses dplyr and ggplot

Part 1: Assessing the t-tests type I error rate

As discussed in class, if the Neyman-Pearson paradigm was followed perfectly, in a world where the null hypothesis was alwasy true, we would expect that only α proportion of hypothesis tests that were run would result in type I errors. We can use this fact to evaluate whether a given type of hypothesis test is "robust" to violations of the assumptions/conditions that are supposedly required to use a method.

In particular, we can use simulated data, where we know what the parameters values are, to assess whether a particular type of hypothesis test produces type I errors at the rate that is expected. We can do this by repeatedly generating random data, and then calculating the p-values from these samples. Once we have a large collection of p-values, we can assess the whether proportion of times that the test rejects the null hypothesis matches the specified significance level α. If it matches, then the method is working properly, otherwise, it is not reliable.

Let's briefly explore this below by assessing how robust the independent sample t-test is to violations of the condition that the data in each sample comes from a normal distribution.

Part 1.1 (3 points):

To make our lives easier, we will fuse the `t.test()` function to get p-values. When we run the `t.test()` function, the returned result is a data structure that contains many values related to the t-test (i.e., the

object that is returned from running the `t.test`function is an object similar to a list and it contains the t-statistic, the degrees of freedom, the p-value, etc.). If we assign the output of running the `t.test()` function to an object called `result`, we can use the syntax `result$p.value` to get the p-value from running the hypothesis test.

Let's start this analysis by reminding ourselves how the `t.test()` function works. In particular, use the `t.test()` function to run an **independent samples** t-test to test whether the mean initial weight of freshman is **different** than the mean final weight using the "Freshman 15" data from homework 4 (please refer back to this homework for a description of the data). Then extract the p-value and save it to an object called `pval`. Finally, print what is stored in your `pval` object to show you have extracted the correct p-value.

```
freshman <- read.table("freshman-15.txt", header = TRUE)

result <- t.test(freshman$Initial.Weight, freshman$Terminal.Weight)

(pval <- result$p.value)
```

```
## [1] 0.6497267
```

Part 1.2 (5 points):

One condition for running an independent samples t-test is that the data from the samples come from normal distributions. Let's start by assessing the type I error rate that occurs when this assumption is met. To do this we can create a for loop which continually add results to a vector called `rejections`. In each iteration of the for loop we will:

1. Create a first sample of 10 random values from a standard normal distribution and save this data to an object called `sample_1`.
2. Create a second sample of 10 random values from a standard normal distribution and save this data to an object called `sample_2`.
3. Use the `t.test()` method to get a p-value from running a two-tailed independent samples t-test between `sample_1` and `sample_2`.
4. Assess whether the p-value is less than a specified $\alpha = 0.05$ level and store the resulting Boolean in the `i`th place in our `rejection` object; i.e., a value of `TRUE` means the null hypothesis was rejected and a value of `FALSE` means it was not.

Once we have run the for loop for 10,000 iterations, we can assess the proportion of rejections and see whether it matches our $\alpha = 0.05$ level.

Please run the analysis below. Report the proportion of null hypotheses that were rejected and consequently if it appears that t-test is indeed giving the expected type I error rate.

```
alpha <- .05
rejections <- NULL

for(i in 1:1e4){
  sample_1 <- rnorm(10)
  sample_2 <- rnorm(10)
  pval <- t.test(sample_1, sample_2)$p.value
```

```
  rejections[i] <- pval < alpha
}

mean(rejections)
```

```
## [1] 0.0485
```

Answer

The null hypothesis was rejected 4.82% of the time. It does appear that the t-test is giving the expected type I error rate.

Part 1.3 (5 points):

Above we assessed with ideal conditions where the data comes from normal distributions. A more interesting question is what happens when some of the assumptions of the t-test are violated. Let's explore this by recreating the code above, but rather than sampling 10 points for each sample from a normal distribution, instead sample the points from a standard exponential distribution (i.e., an exponential distribution with a rate parameter of 1), which is a distribution that has long tails.

Please estimate and report the type I error rate using the code chunk below, and answer whether the t-test appear to be robust to the assumption that the data comes from normal distributions.

Hint: You can use the `rexp(k)` function to generate `k` random points from a standard exponential distribution.

```
exp_rejections <- NULL

for(i in 1:1e4){
  sample_1 <- rexp(10)
  sample_2 <- rexp(10)
  pval <- t.test(sample_1, sample_2)$p.value
  exp_rejections[i] <- pval < alpha
}

mean(exp_rejections)
```

```
## [1] 0.0378
```

Answer

The type I error rate is 3.78%. The t-test does appear to be robust to the assumption that the data comes from normal distributions.

Note: this simulation approach can be used to assess the robustness of many properties and methods including how outliers impact particular methods, and also to assess whether confidence intervals are capturing the parameters the correct proportion of the time (e.g., this method could be used to assess whether bootstrap confidence intervals are working correctly, etc.).

Part 2: Data transformations with dplyr

Travel by airplane can be convenient because airplanes fly very fast. However, even though the airplanes themselves are fast, their scheduled departure times are often delayed, which can significantly add to ones travel time, and can be frustrating.

Let's analyze data on flights to gain insight into how best to avoid flight delays. In particular, we will look at airplane flights that left the airports in New York City, since these airports are some of the closest major airports to New Haven, and we will use dplyr to do some quick explorations of the data to see if there are some ways to potentially avoid flight delays.

To begin, let's load the data for flights leaving New York City in 2013 using the code below. To get more information on this data set, use *? flights* (you don't need to modify anything on the code below).

```
#install.packages("nycflights13")

# get the flight delays data and load dplyr
require("nycflights13")
data(flights)
data(airlines)   # the names of the airline carriers
```

Part 2.1 (5 points): One way to avoid being delayed would be to avoid the worst airlines. Which airline had the longest arrival delays on average, and how long was this average delay? Use the `airlines` data frame to figure out which airline each carrier code corresponds to.

Note: For the analysis on this homework, just ignore missing values in any summary statistics you create.

```
library(dplyr)

# get the average delay for each airline

all_delays <- select(flights, arr_delay, carrier) %>%
  group_by(carrier)

avg_delay <- summarize(all_delays, delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(delay)

longest <- slice_max(avg_delay, delay)

name <- filter(airlines, carrier == longest$carrier)

combine(name, longest)
```

```
## # A tibble: 2 x 3
##   carrier name                  delay
##   <chr>   <chr>                 <dbl>
## 1 F9      Frontier Airlines Inc.  NA
## 2 F9      <NA>                    21.9
```

Answers:

Frontier Airlines Inc. had the longest average delay, 21.9207 minutes.

Part 2.2 (5 points): Flights that start off with a delay might end up making up some time during the course of the flight. Examine whether this is true on average by reporting relevant descriptive statistics.

Hint: we only use flights that have positive departure delay, since a flight needs to be delayed in order to "make up time".

```
# Filter for flights with a departure delay
# Select the departure, arrival, and carrier columns
# Create a new column with the difference between arrival and departure delays
  # If this difference is less than zero, the flight made up some time
(delay_diff <- filter(flights, dep_delay > 0) %>%
    select(dep_delay, arr_delay, carrier) %>%
    mutate(diff = arr_delay - dep_delay))
```

```
## # A tibble: 128,432 x 4
##    dep_delay arr_delay carrier  diff
##        <dbl>     <dbl> <chr>   <dbl>
##  1         2        11 UA          9
##  2         4        20 UA         16
##  3         2        33 AA         31
##  4         1        -6 B6         -7
##  5         8        32 MQ         24
##  6        11        14 UA          3
##  7         3         4 B6          1
##  8        13         5 AA         -8
##  9        24        12 EV        -12
## 10         8        -9 UA        -17
## # ... with 128,422 more rows
```

```
# Find the proportion of flights that started off with a delay but made up
  # some time during the course of the flight
mean(delay_diff$diff < 0, na.rm = TRUE)
```

```
## [1] 0.6554307
```

Answers:

On average, most flights that start off with a delay make up some time over the course of the flight.

Part 2.3 (5 points): Another way to avoid flight delays would be to avoid particularly bad times to fly. Which month of the year had the longest departure delays? Also report which hour of the day had the longest departure delays. Finally, report how many flights left at the hour of the day with the longest delay and what the average delay was at that hour.

Hint: You can use `n()` within summarize to get the size of each group specified by the call to `group_by()`.

```
# Select month, hour, and departure delay columns
filtered_flights <- select(flights, month, hour, dep_delay)
```

```r
# Group by month
month_group <- group_by(filtered_flights, month)
# Group by hour
hour_group <- group_by(filtered_flights, hour)

# Average delay by month
month_avg <- summarize(month_group, avg_delay = mean(dep_delay, na.rm = TRUE))
# Average delay by hour
hour_avg <- summarize(hour_group, avg_delay = mean(dep_delay, na.rm = TRUE))

# Month with the longest average departure delays
(longest_month <- slice_max(month_avg, month))
```

```
## # A tibble: 1 x 2
##   month avg_delay
##   <int>     <dbl>
## 1    12      16.6
```

```r
# Hour with the longest average departure delays
(longest_hour <- slice_max(hour_avg, hour))
```

```
## # A tibble: 1 x 2
##    hour avg_delay
##   <dbl>     <dbl>
## 1    23      14.0
```

```r
# Filter for flights that left at longest_hour
hour_flights <- filter(filtered_flights, hour == longest_hour$hour)

# Number of flights that left at longest_hour
dim(hour_flights)
```

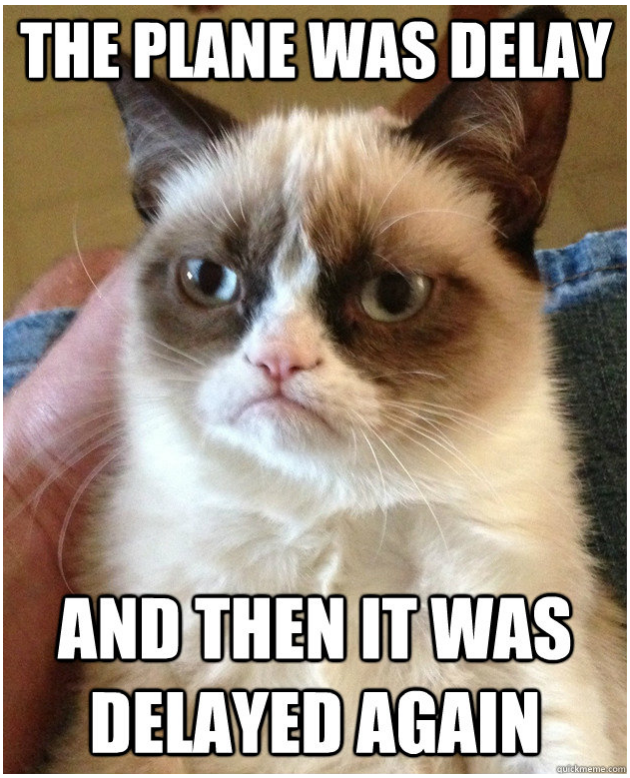
```
## [1] 1061    3
```

```r
# Average delay at longest_hour
mean(hour_flights$dep_delay, na.rm = TRUE)
```

```
## [1] 14.01718
```

Answers:

December had the longest departure delays by month. 11:00pm had the longest departure delays by hour. 1061 flights left at 11:00pm. The average delay at 11:00pm was 14.01718 minutes.



Part 3: Data visualization

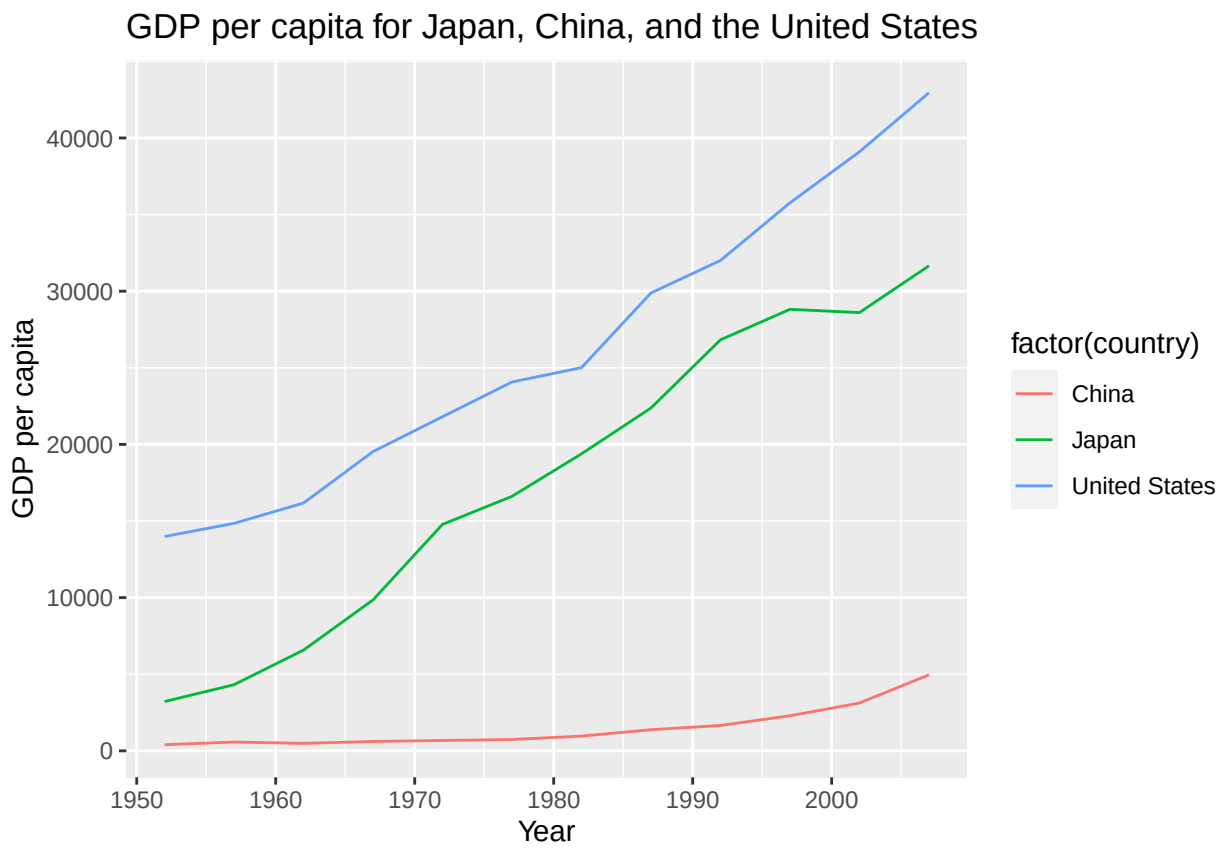
In the next set of exercises you will use `ggplot2` to compare different visualizations and see which gives the clearest insights. As mentioned above, a useful resource for ggplot and other tidyverse code is the book R for Data Science.

Part 3.1 (8 points): Let's start by comparing some visualizations on the gapminder data which contains information about different countries in the world over time. Use `ggplot` and the `gapminder` data to compare the GDP per capita of Japan, the United States and China. Plot a line graph of GDP per capita as a function of the year, with each country in a different color. Also, create a plot that compares these countries' GDP per capita as a function of the year using facets, where the data from each country is on a separate subplot. As always, make sure to label your axes in this plot and in all other plots in this homework. Do you think one type of plot is better than another in comparing these countries? Explain why.

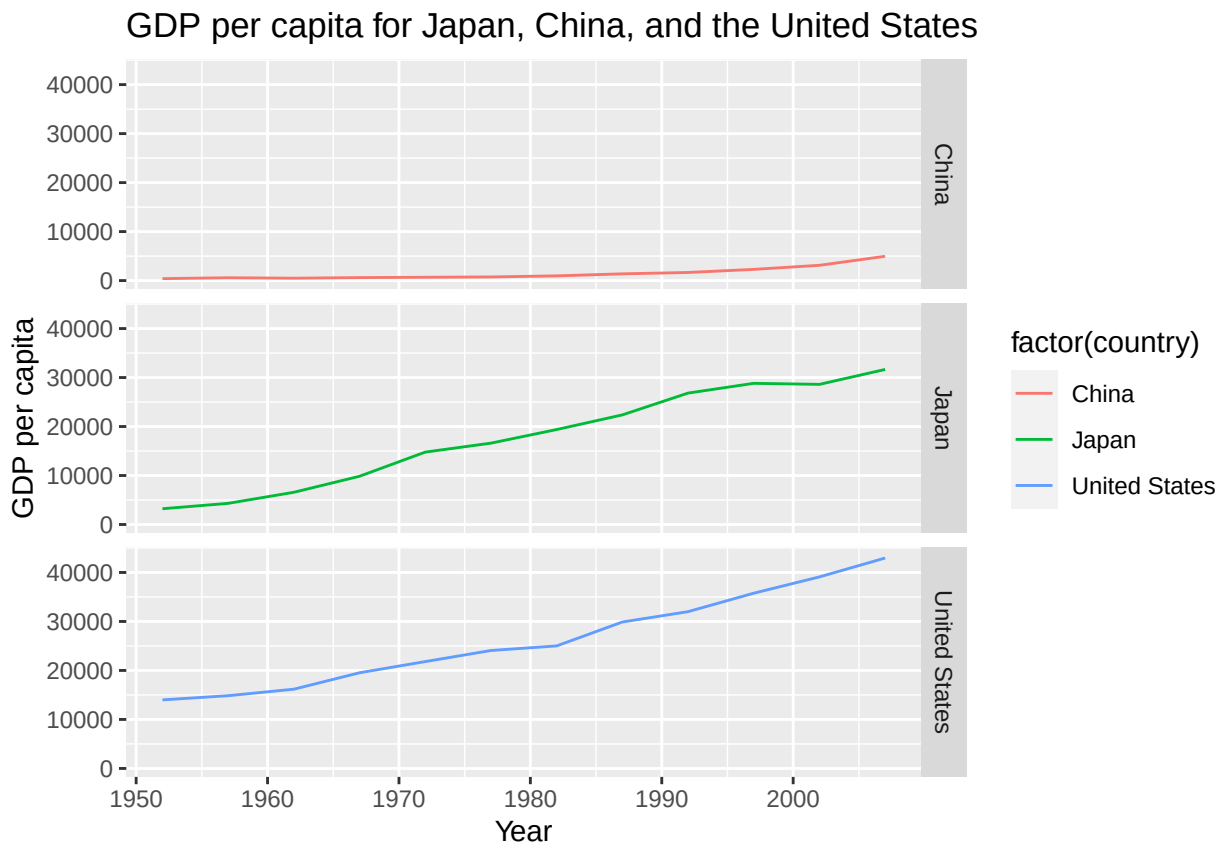
Hint: first use the dplyr `filter()` function to get the subset of data you need, then plot it.

```
# compare the GDP per capita of Japan, the United States and China
# Plot a line graph of GDP per capita as a function of the year, with each country in a different color
# create a plot that compares these countries' GDP per capita as a function of the year using facets, w
library(gapminder)

gapminder %>%
  filter(country %in% c("Japan", "China", "United States")) %>%
  ggplot(aes(x = year, y = gdpPercap, col = factor(country))) +
```

```
  geom_line() +
  labs(x = "Year",
       y = "GDP per capita",
       title = "GDP per capita for Japan, China, and the United States")
```




```
gapminder %>%
  filter(country %in% c("Japan", "China", "United States")) %>%
  ggplot(aes(x = year, y = gdpPercap, col = factor(country))) +
  facet_grid("country") +
  geom_line() +
    labs(x = "Year",
         y = "GDP per capita",
         title = "GDP per capita for Japan, China, and the United States")
```



Answers: [Explain whether you think of of these plots is more informative than the other].

I think the first graph, with all three countries on the same graph, is more informative than the other because it is easier to compare the relative GDP's.

Part 3.2 (8 points): DataExpo is a Statistics event at the Joint Statistical Meetings where different researchers compare data analysis methods applied to a common data set. In 2018, the data set consisted of weather predictions made between 2014 and 2017. In this exercise, let's look at the data from this event and try to visualize the prediction accuracies for predictions made 0 to 6 days in the future.

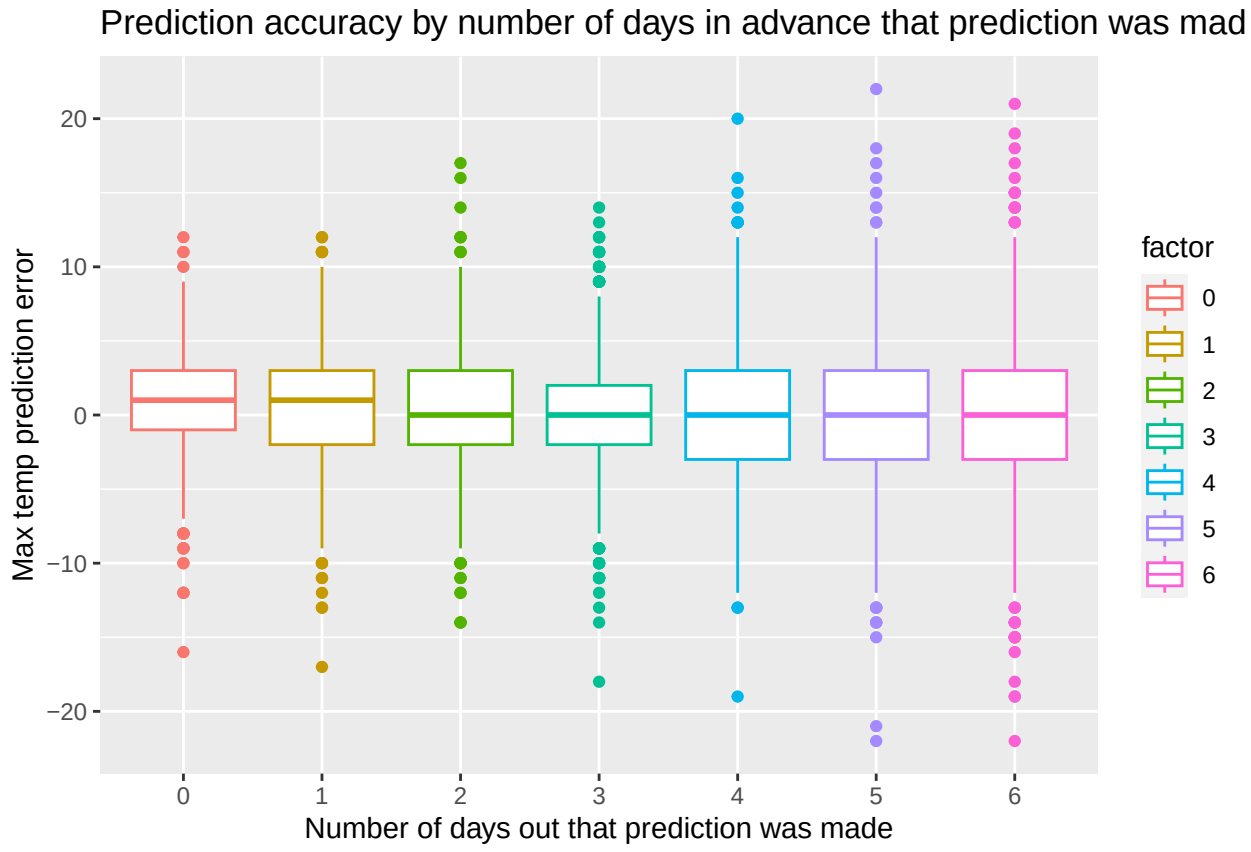
The code below loads a data frame called `forecast_ne_joined` that has the prediction errors for the maximum temperature for the 9 cities in New England, along with several other variables. First, create a new data frame called `new_haven_preds` that has only the predictions from New Haven, and has only the variables `cityID`, `city`, `num_days_out_prediction_made` and `max_temp_prediction_error`. Also, convert the variable `num_days_out_prediction_made` to a factor using the `as.factor()` function inside of the `mutate()` function. Then use ggplot to create plots that compare the prediction accuracy as a function of the number of days in advance that a prediction was made using the following geoms:

1. Create a box plot using `geom_boxplot()`
2. Create a violin plot using `geom_violin()`
3. Create a joy plot using `geom_density_ridges()`

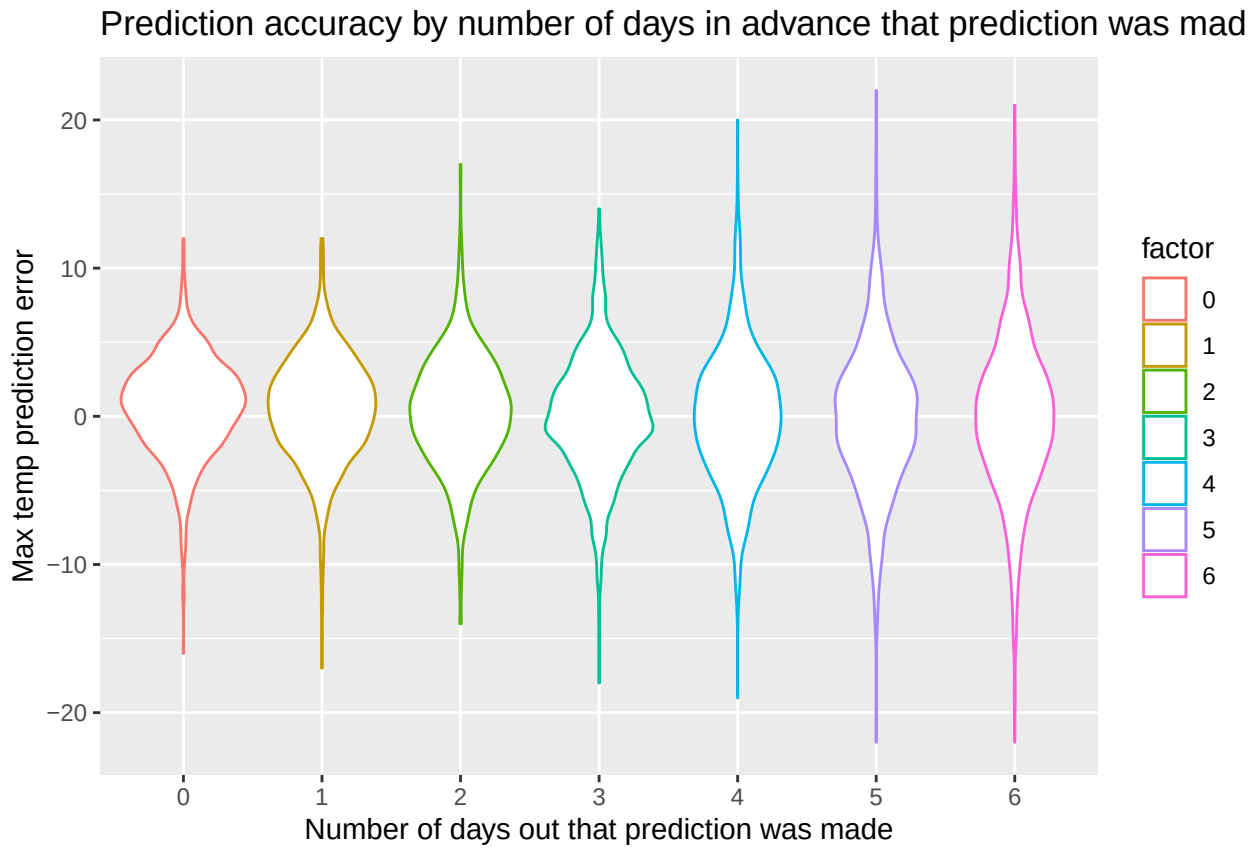
Note that the geom `geom_density_ridges()` comes from the ggridges packages that was loaded at the top of the worksheet, and that the x and y aesthetic mapping needs to be in the opposite order as the mapping used for the `geom_boxplot()` and `geom_violin()` geoms.

After you created these plots, briefly discuss which plot you believe most clearly shows how the prediction accuracy decreases as a function of days in the future. Also, don't forget to label your axes using the `xlab()` and `ylab()` functions.

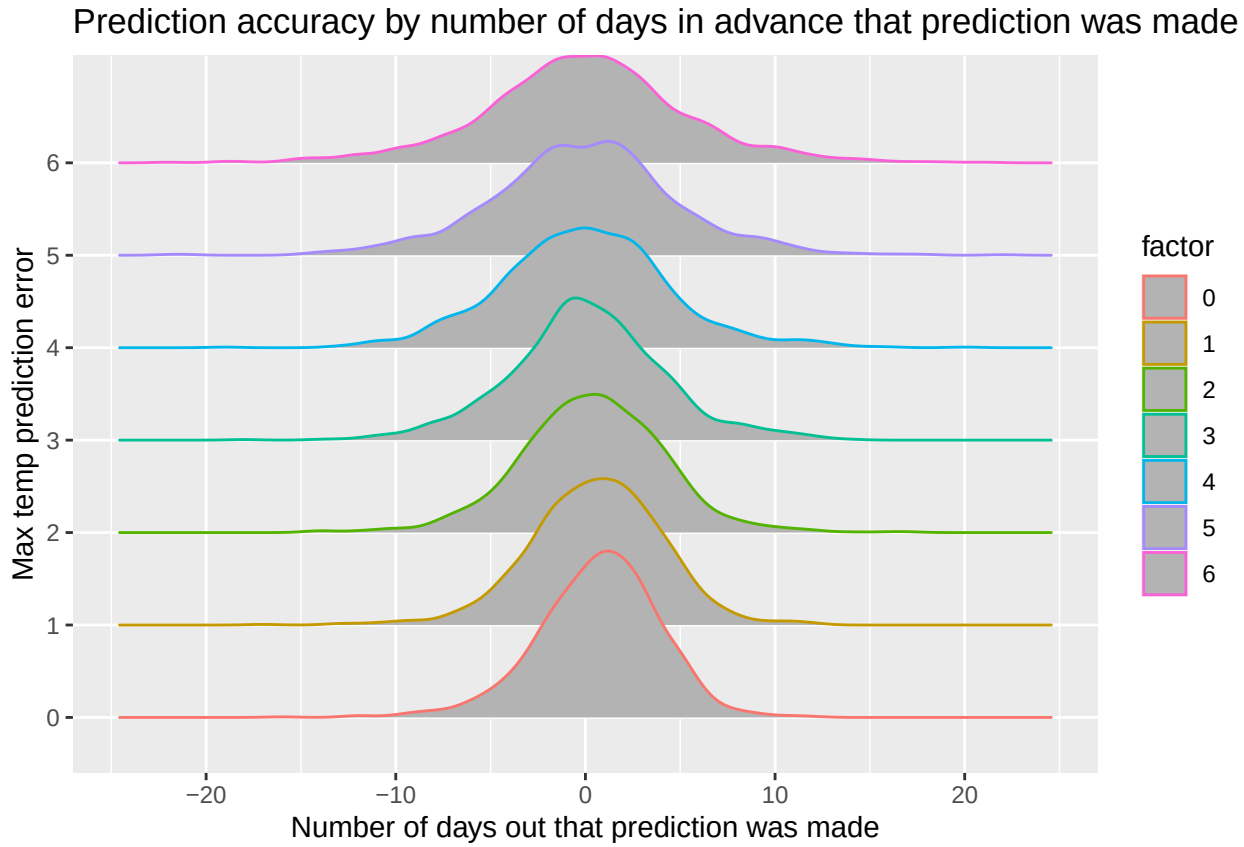
```
# load the data that has the weather prediction errors
load('forecast_ne_joined.rda')

new_haven_preds <- forecast_ne_joined %>%
  filter(city == "New Haven") %>%
  select(cityID, city, num_days_out_prediction_made, max_temp_prediction_error) %>%
  mutate(factor = as.factor(num_days_out_prediction_made))

new_haven_preds %>% ggplot(aes(x = factor, y = max_temp_prediction_error, col = factor)) +
  geom_boxplot() +
  labs(x = "Number of days out that prediction was made",
       y = "Max temp prediction error",
       title = "Prediction accuracy by number of days in advance that prediction was made")
```




```
new_haven_preds %>% ggplot(aes(x = factor, y = max_temp_prediction_error, col = factor)) +
  geom_violin() +
  labs(x = "Number of days out that prediction was made",
       y = "Max temp prediction error",
       title = "Prediction accuracy by number of days in advance that prediction was made")
```




```
new_haven_preds %>% ggplot(aes(x = max_temp_prediction_error, y = factor, col = factor)) +
  geom_density_ridges() +
  labs(x = "Number of days out that prediction was made",
       y = "Max temp prediction error",
       title = "Prediction accuracy by number of days in advance that prediction was made")
```

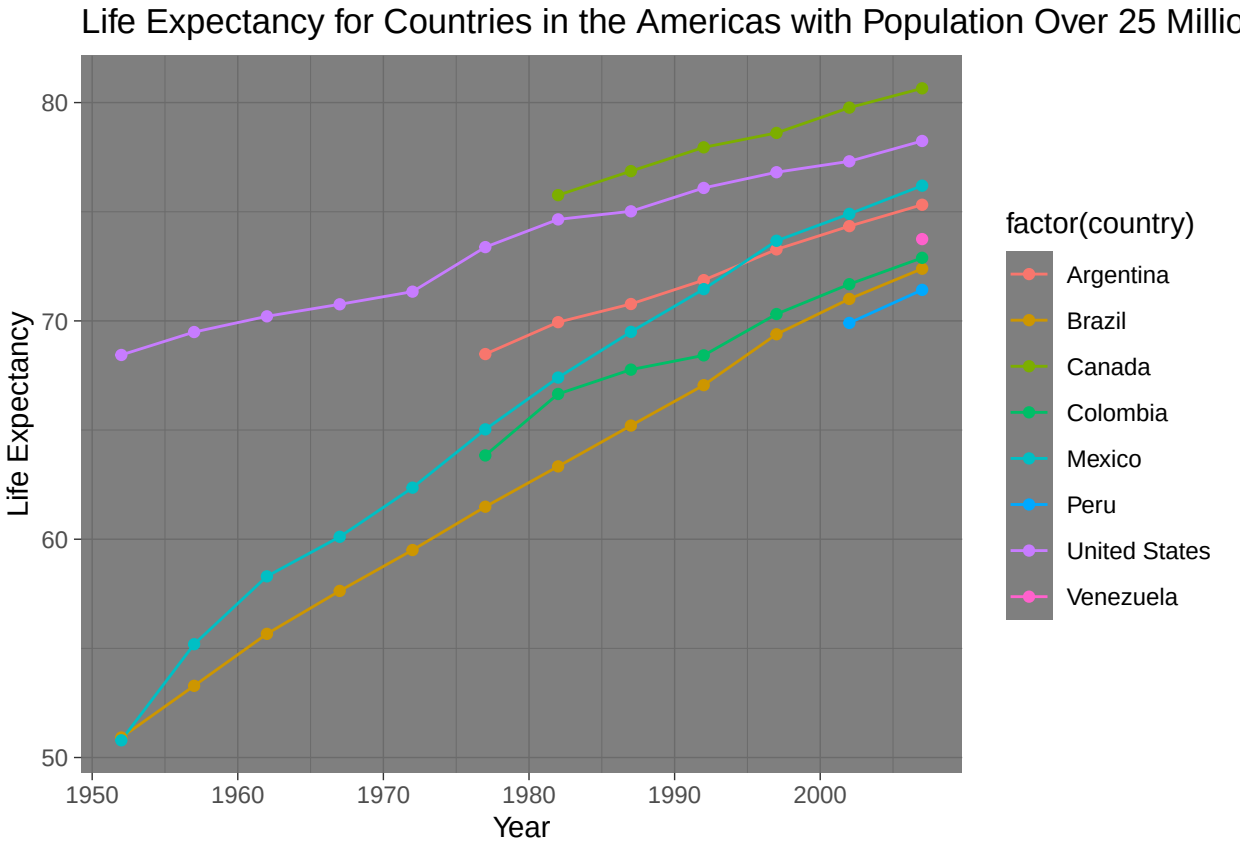


Answers:

I believe the violin plot most clearly shows how the prediction accuracy decreases as a function of days in the future. It shows the spread of the data decreasing and becoming concentrated around zero as the number of days decreases.

Part 3.3 (5 points): Create an *interesting* plot using one of the data sets we have discussed in class or another data set you can find. Try exploring other features of ggplot we have not discussed in class using the ggplot cheat sheet. See if you can find something interesting in the data and explain why you find it interesting.

```
gapminder %>%
  filter(continent == "Americas") %>% filter(pop > 25000000) %>%
  ggplot(aes(x = year, y = lifeExp, col = factor(country))) +
  geom_line() +
  geom_point() +
  theme_dark() +
    labs(x = "Year",
         y = "Life Expectancy",
         title = "Life Expectancy for Countries in the Americas with Population Over 25 Million")
```



Answers:

I explored life expectancies in the Americas according to the Gapminder data, specifically for countries with populations over 25 million. I found it interesting that the data indicates that high population is not necessarily correlated with higher life expectancy, as the two countries with highest and lowest life expectancy (Canada and Peru, respectively) do not reach a popuation of 25 million until after 1980. It is also interesting to see the rapid increase in life expectancy for Mexico and Brazil, especially compared to the US, the only other country to have a population over 25 million in 1950.

Reflection (2 points)

Please reflect on how the homework went by going to Canvas, going to the Quizzes link, and clicking on Reflection on homework 5