SPEC Lab REU R Resources: Data visualization with `ggplot2`: Line Graphs

Alix Ziff, Gaea Morales, Zachary Johnson, Benjamin Graham, and Jasmine Chu
based on earlier materials by Therese Anders

Summer 2022

ggplot2 continued

In Part 2 of the Data Visualization II walkthrough, we will cover line plots.

We will continue working with data from the World Development Indicators, using `wdi_cleaned_part2_csv`.

Load ggplot2 as well as the necessary data if you are starting from a new R session.

```
dat <- read.csv("wdi_cleaned_part2.csv")

library(ggplot2) # load the necessary libraries
library(tidyr)
library(dplyr)
library(tidyverse)
```

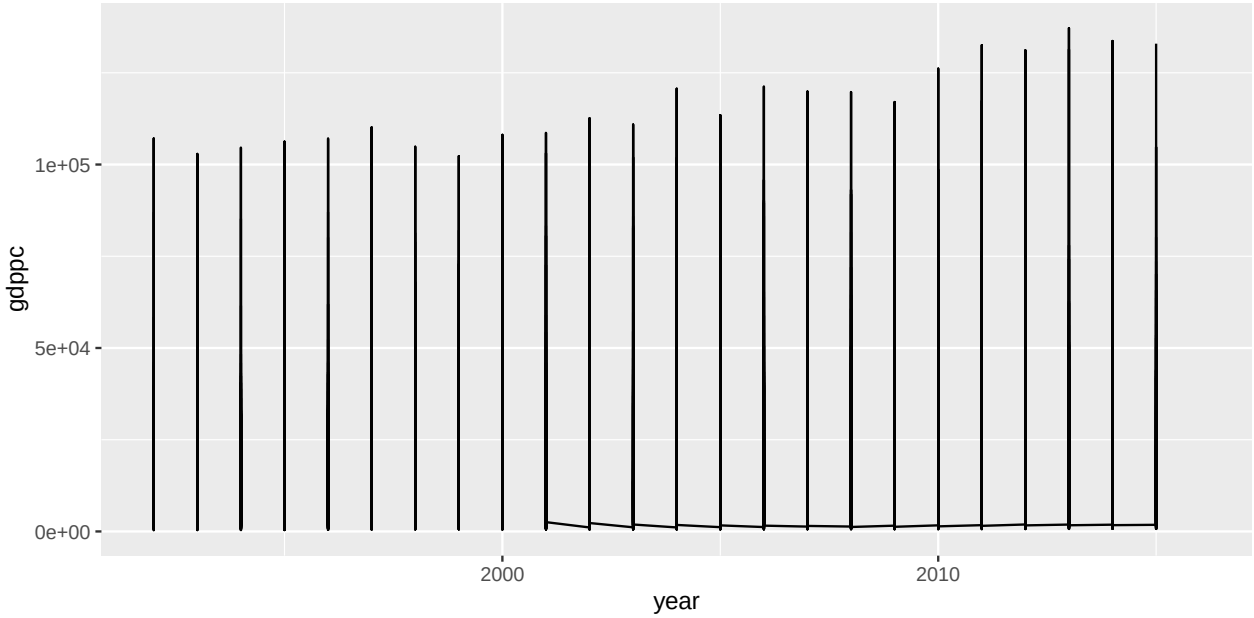
Line graphs

One of the most common uses for line graphs is the display of data over time. For example, we could be plotting the evolution of GDP per capita over time. Within the `aes()` command, we specify which variable is to be plotted on the x- and y-axes. The geometric object we use for line graphs is `geom_line()`.

Here we are working with a panel data set. This means that we have multiple observations over time for each country, allowing us to observe variation across country units, as well as variation within and across these units over time.

Displaying all this information without grouping or subsetting does not result in a plot that is accessible for interpretation. In the plot below, `geom_line()` is trying to draw a line *connecting* all non-missing country-year observations for the variable `gdppc`.

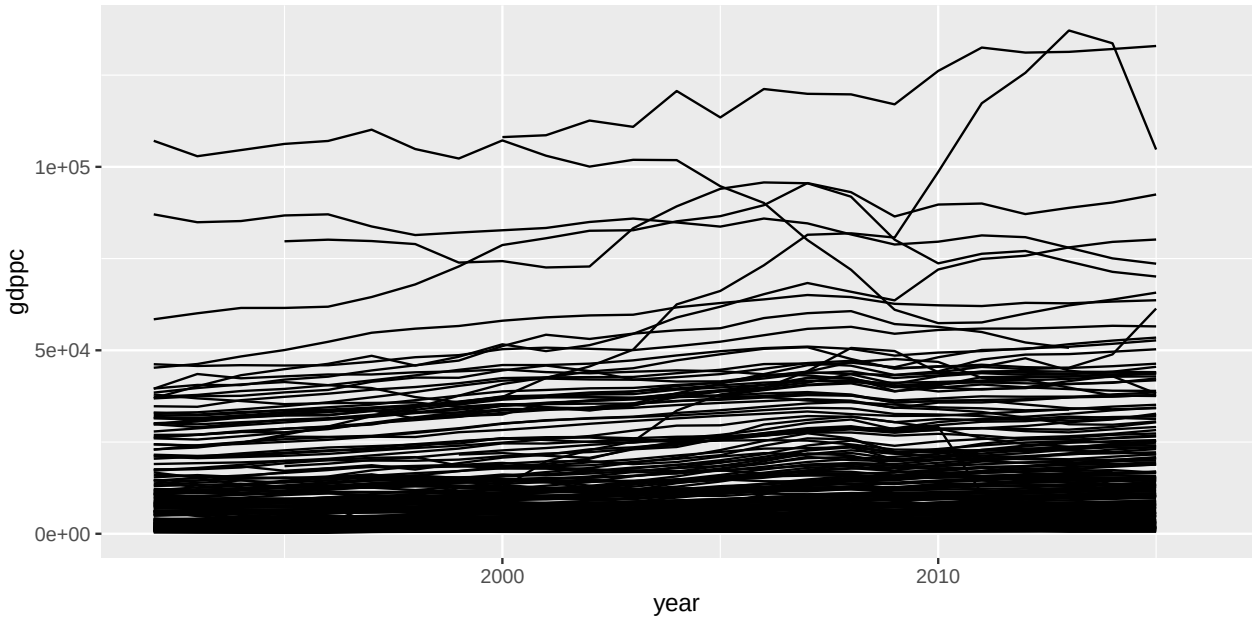
```
ggplot(dat, aes(x = year, y = gdppc)) +
  geom_line() #connecting points on our graph
```



Helpful Hint: Here we have created a line graph which *connects points*. This is a different from a scatterplot and shouldn't be confused with trend lines which outlines the pattern of existing points. You can see the connections more clearly when you zoom in your plot (e.g., using the `Zoom` icon in the Plots window on R Studio).

In order to plot one line per country, we can specify the `group` parameter inside the aesthetics argument. However, in this data set, even if we grouped the data by country, there are still too many groups (countries) to be displayed in one graph.

```
ggplot(dat, aes(x = year, y = gdppc, group = country)) +
  geom_line() #now we have what appears to be a horizantal line for each country
```



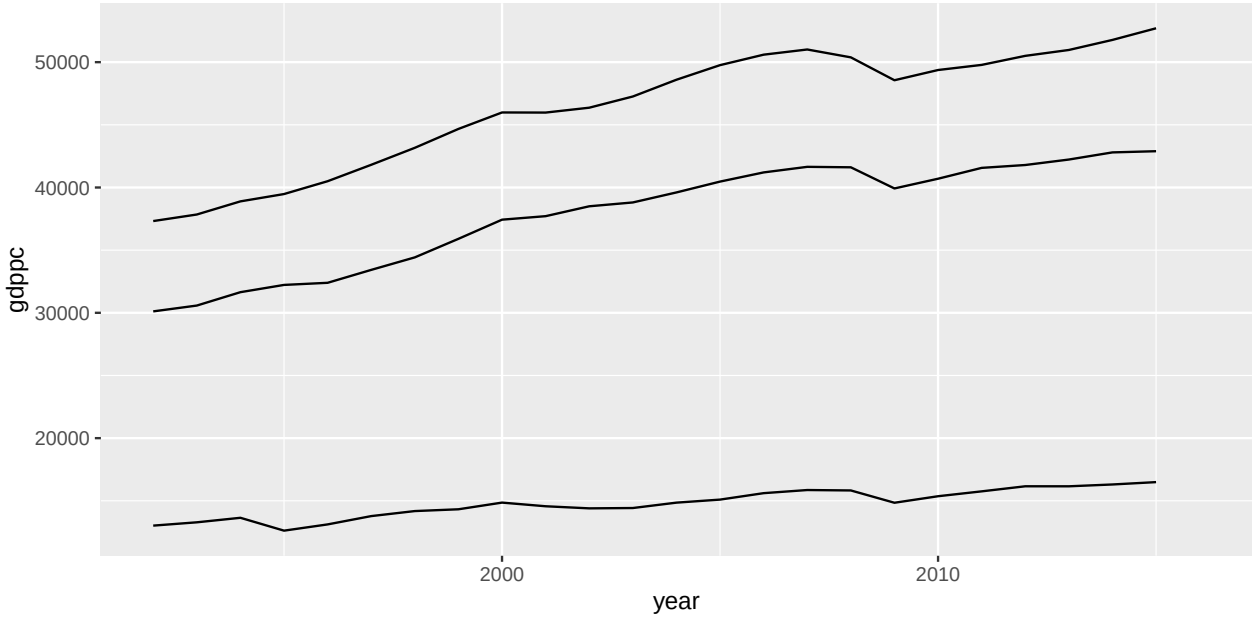
What we can do is extract a subset of countries and plot the evolution of their per capita GDP over time. Suppose we want to see how the per capita GDP has changed over time in member countries of the North American Free Trade Agreement (NAFTA), that is Canada, Mexico, and the US.

Subsetting for line plots

We can do this a few different ways. One way is to use the base R command `subset()` inside of our ggplot command, as we saw in part 1 of the walkthrough.

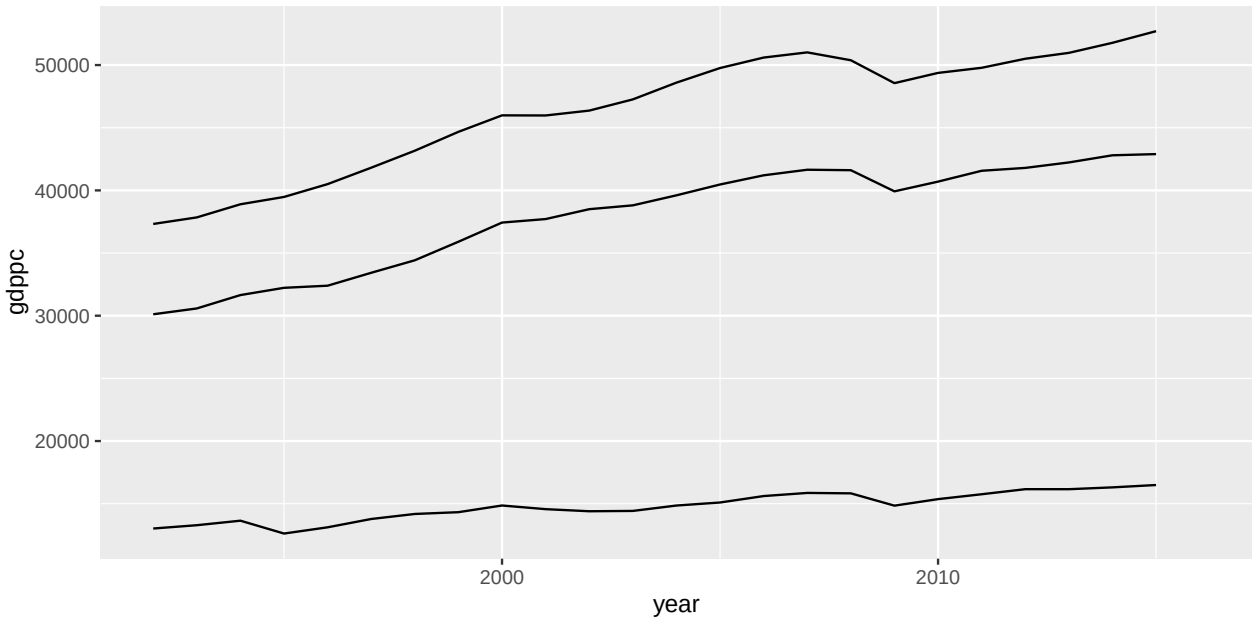
We can use `subset()` inside the `ggplot()` function to plot a separate line for each of the countries. Remember that we can use the concatenate (`c()`) function with the `%in%` helper function to specify the list of units that we are interested in.

```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line()
```



Alternatively, we can use the `filter()` command from the tidyverse that we learned in Data Management I. It will produce the same figure.

```
mylineplot <- dat %>%
  filter(country %in% c("Canada", "Mexico", "United States")) %>%
  ggplot(aes(x = year, y = gdppc, group = country)) +
    geom_line()
mylineplot # since we saved the plot in an object, we need a separate
```

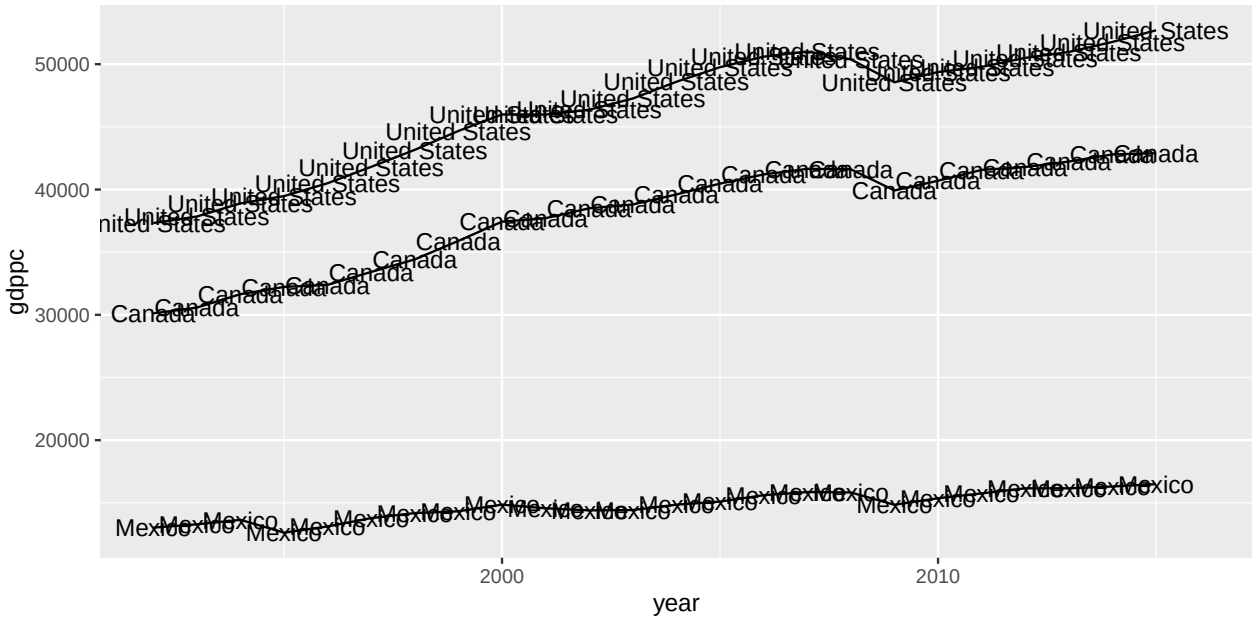



```
  # line to View the object
```

Adding labels

However, this graph does not tell us which line represents the evolution of per capita GDP in which country. We could add labels to each line to denote the country it represents. We could do this with the `geom_text()` function, but this will add a label for each country-year observation.

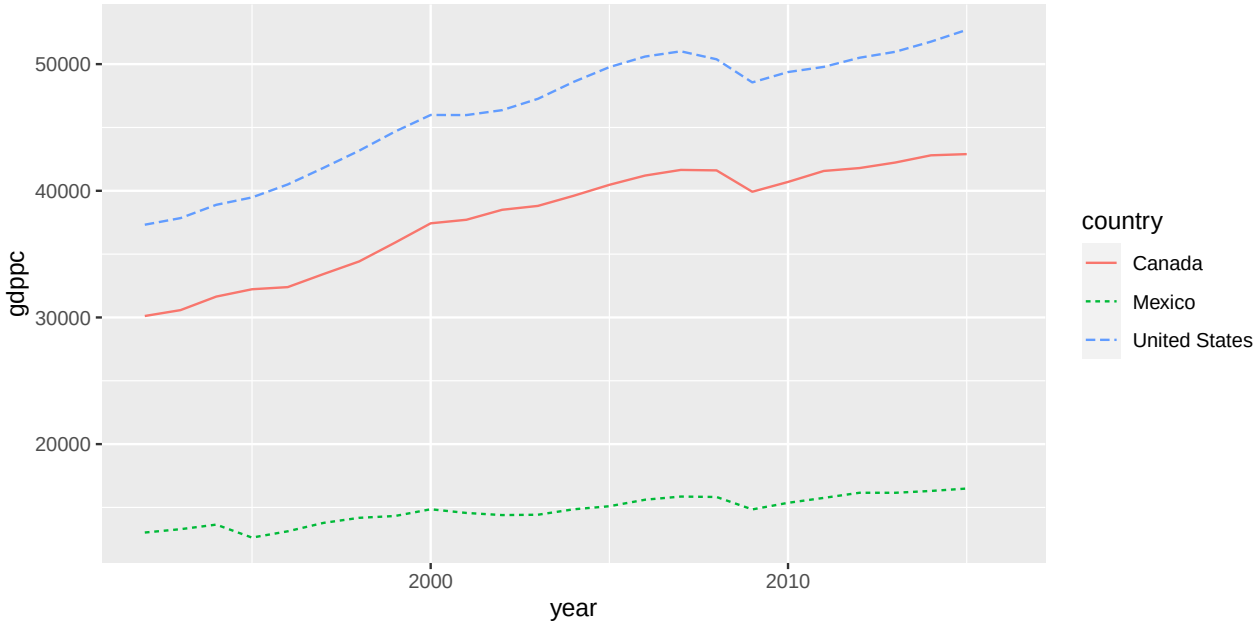
```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country)) +
  geom_line() +
  geom_text()#label each country
```



Another way we could label each line to denote the country it represents is by adding a legend to the

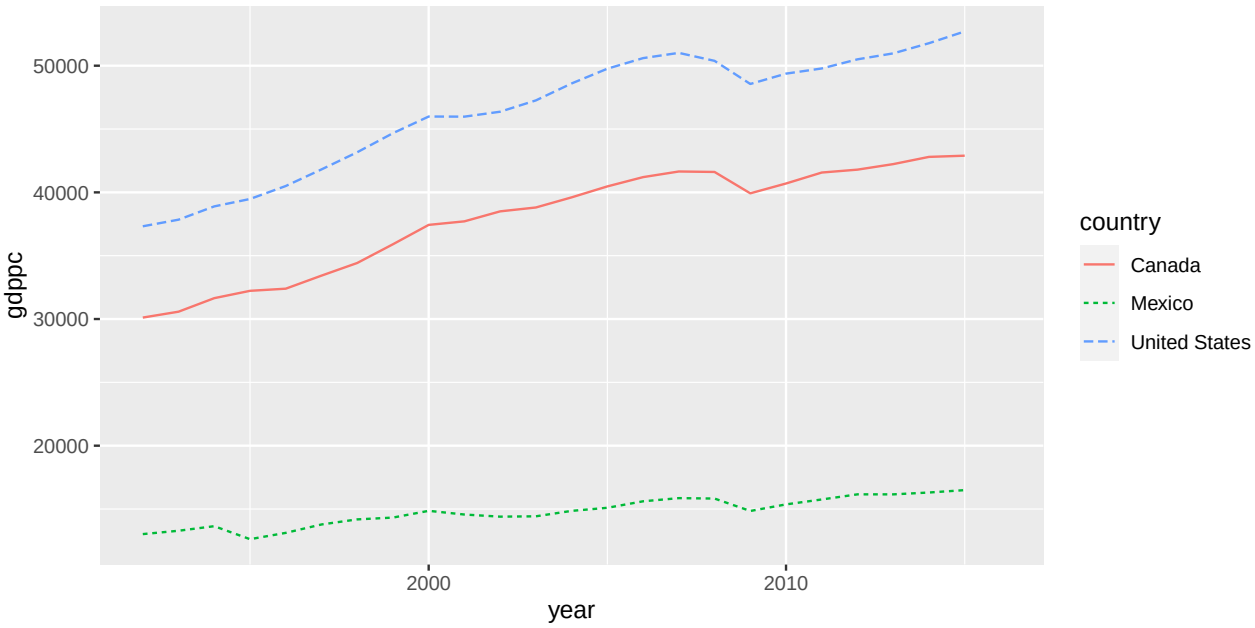
line-plot, with each country representing a different color and/or line type. We could do this by adding another input for color and linetype within aes().

```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country, color=country,
           linetype = country)) +
  geom_line()
```



Another way we could label each line to denote the country it represents is by adding a legend to the right side of the line-plot.

```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country, color=country,
           linetype = country)) +
  geom_line()
```

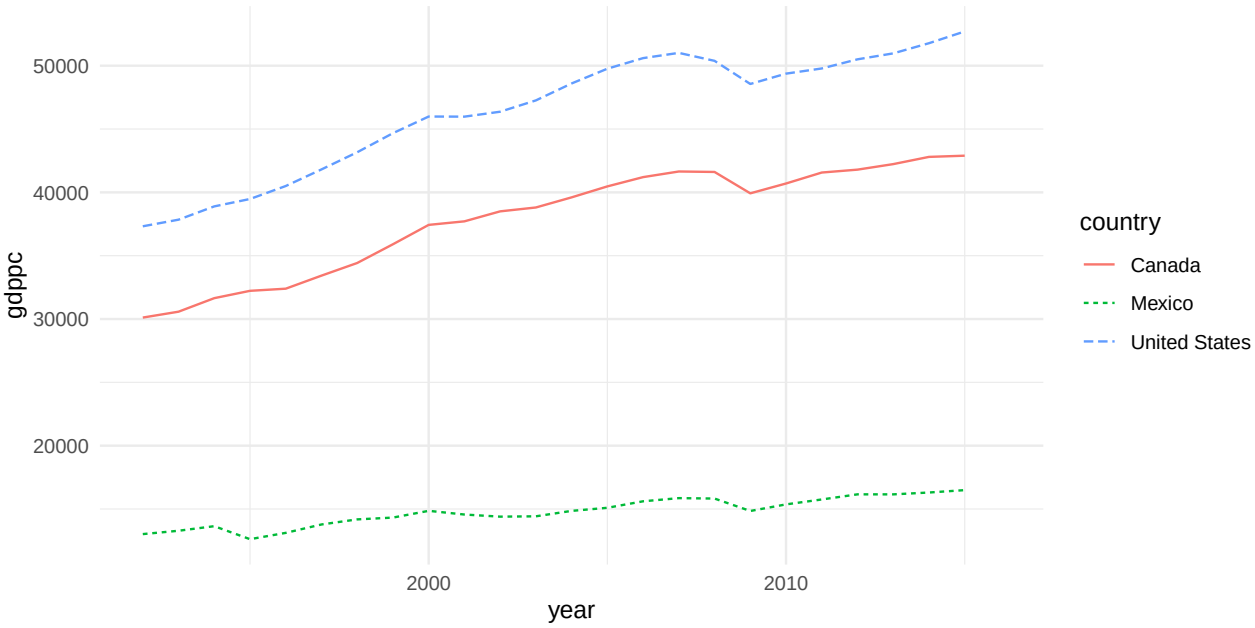


Helpful hint: We can adjust the position of the legend by adding the line (don't forget the +!): theme(legend.position="bottom") or theme(legend.position="top"), etc.

Cleaning up the plot background

As in Part 1 of the walkthrough, we can once again use theme_minimal().theme_minimal() is a theme that creates a white background for your line plot, which helps to better distinguish the colored lines from one another.

```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country, color=country,
           linetype = country)) +
  geom_line() +
  theme_minimal() #this sets our background to white
```

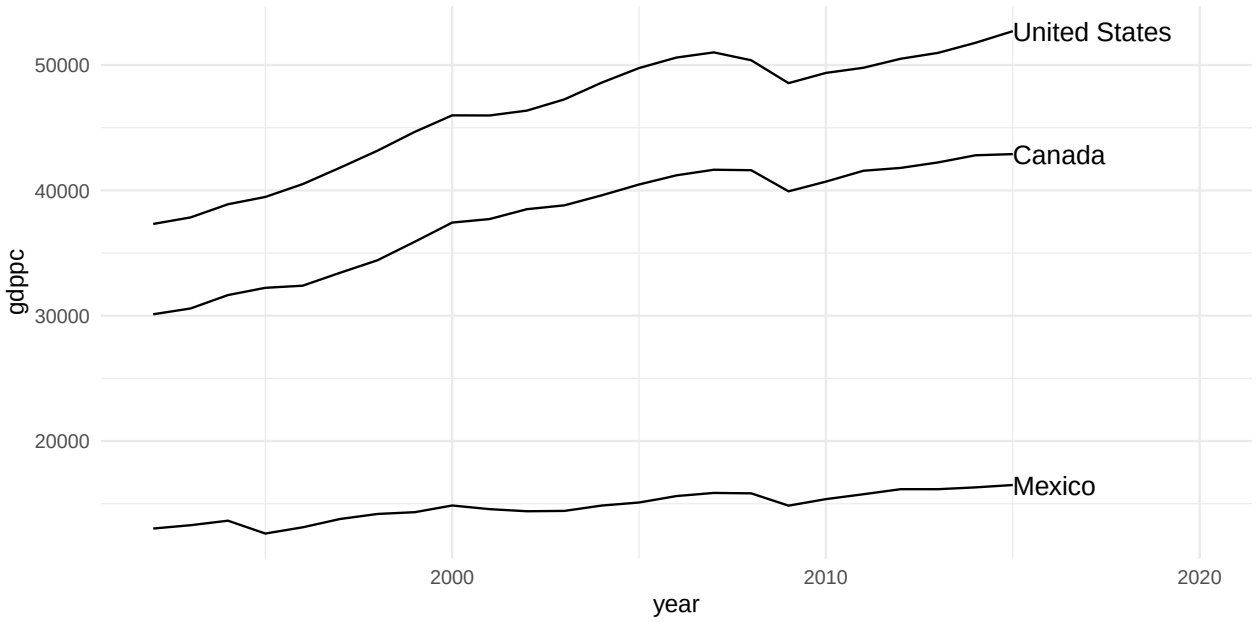


A better way to label each line (and not each country-year observation) is using the `directlabels` package and the `geom_dl()` geometric object in `ggplot()`. You have to specify a method for the `geom_dl()` function to work (here: `"last.points"`, which specifies that we only want labels for the last point for each line (e.g., country unit)). We can use the `cex` argument to control the size of the text labels. Here, we also adjust the range of the x-axis to ensure that all lines are properly labeled.

The graph shows that the evolution of wealth in the three countries is intimately connected. There are similar patterns in terms of crises and upswings for the US, Canada, and Mexico. However, the increase of per capita GDP appears to be a lot steeper in Canada and the US than in Mexico, despite similar trends.

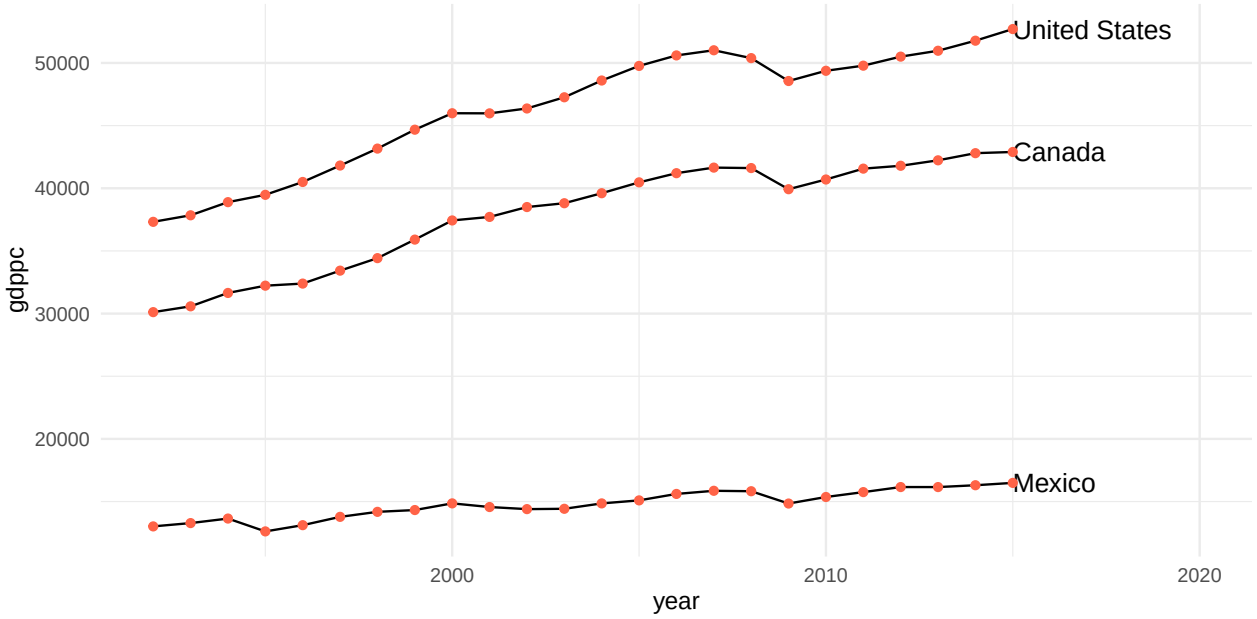
```
#Note: Install these libraries if you have not downloaded these before
library(directlabels)
library(gridtext)
library(quadprog)
library(proto)
attach(dat)

ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line() +
  geom_dl(aes(label = country), method = ("last.points"), cex = 0.5) +
  coord_cartesian(xlim = c(1992, 2020)) +
  theme_minimal()
```



It is not necessary in this example, but sometimes we would like to show which observations (e.g., which year values) go into the computation of the line by adding points. Remember, that ggplot uses layers to graph multiple geometric objects in one plot. We can therefore just overlay the line plot with a `geom_point()` layer.

```
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line() +
  geom_dl(aes(label = country), method = ("last.points"), cex = 0.5) +
  geom_point(color = "tomato") +
  coord_cartesian(xlim = c(1992, 2020)) + #  minimum and maximum values of the x axis
  theme_minimal()
```



This walkthrough provided basics for creating lineplots. In the next part, we will go through another form of visualization that integrates geographic data, namely maps.