SPEC REU R Resources: Line Graphs with ggplot2

Alix Ziff, Gaea Morales, Zachary Johnson, Jasmine Chu, Claudia Salas Gimenez and Ben Graham

Summer 2024

In this second walkthrough, we'll focus on creating and refining line plots using `ggplot2`. Line plots are invaluable for tracking changes over time, making them ideal for exploring trends in economic growth, demographic shifts, or any time-based data.

This assignment will get you comfortable with building and customizing line plots, especially in handling data across multiple groups. Let's dive in!

Initial Setup

To start, set up your environment by defining your working directory, loading necessary libraries, and importing the dataset.

```r
# Set working directory
#setwd("YourFolderPath")

# Load required libraries
library(ggplot2)
library(dplyr)
library(tidyr)
library(directlabels)

# Load the data
dat <- read.csv("wdi_cleaned_part2.csv")
```

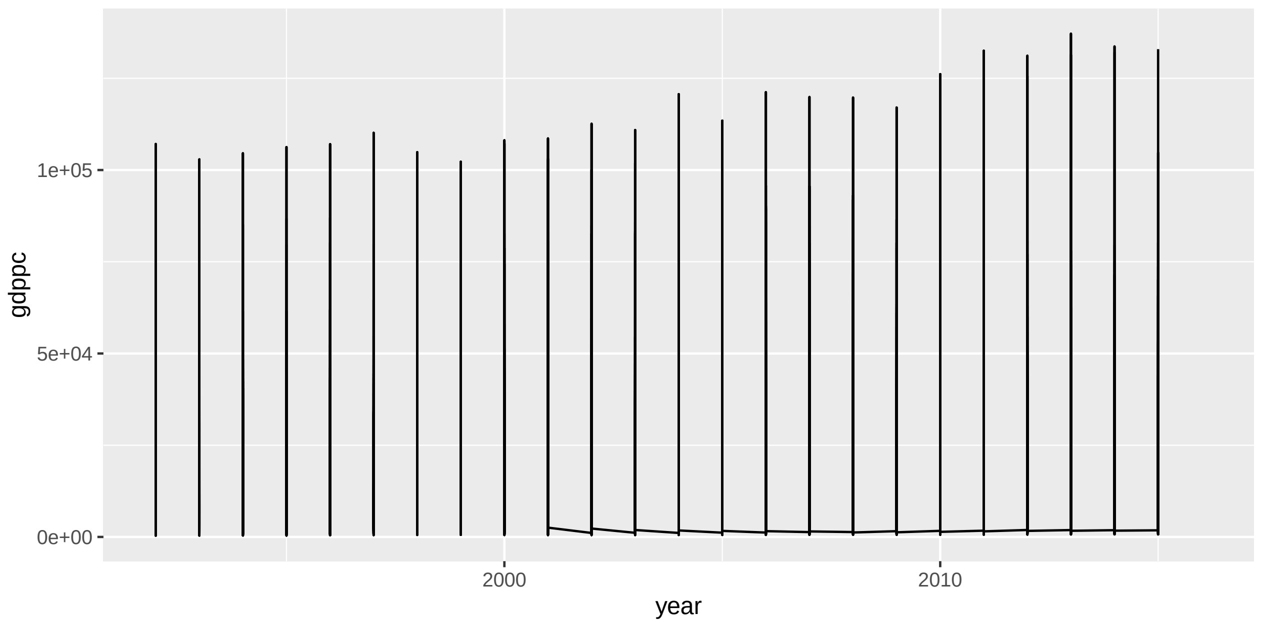
We'll be working with a familiar dataset, `wdi_cleaned_part2.csv`. As before, it's helpful to keep track of what each variable represents. For a refresher on this dataset, refer to the World Development Indicators data catalog. Note that some variable names may vary slightly in this modified version.

Creating Basic Line Graphs

Line graphs are excellent for visualizing trends over time, like changes in GDP per capita. Using `geom_line()` in `ggplot2`, we can connect data points across years to create a clear time-series plot.

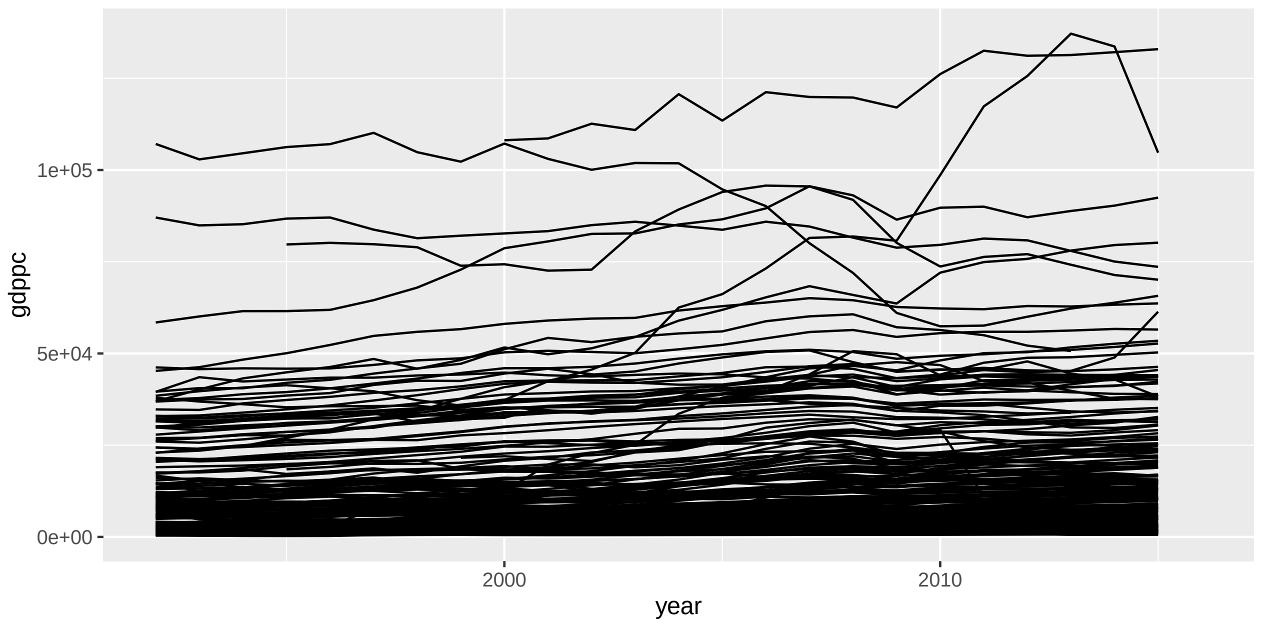
In this walkthrough, we're be working with panel data, which includes multiple observations over time for each country. This structure allows us to see variations both within individual countries and across different countries over time. Let's start by plotting GDP per capita as a line graph to explore these trends.

```r
# Basic line plot
ggplot(dat, aes(x = year, y = gdppc)) +
  geom_line()
```



This plot connects all available GDP per capita data points across years and countries, but it can be overwhelming without properly grouping or subsetting the data. To plot a separate line for each country, specify the `group` parameter within the `aes()` function.

```r
# Line plot grouping by country
ggplot(dat, aes(x = year, y = gdppc, group = country)) +
  geom_line()
```



Grouping by country results in numerous lines, which can still be difficult to interpret. To focus on a specific subset, we can narrow down the data to just a few countries, such as NAFTA-nations (Canada, Mexico, and the United States).

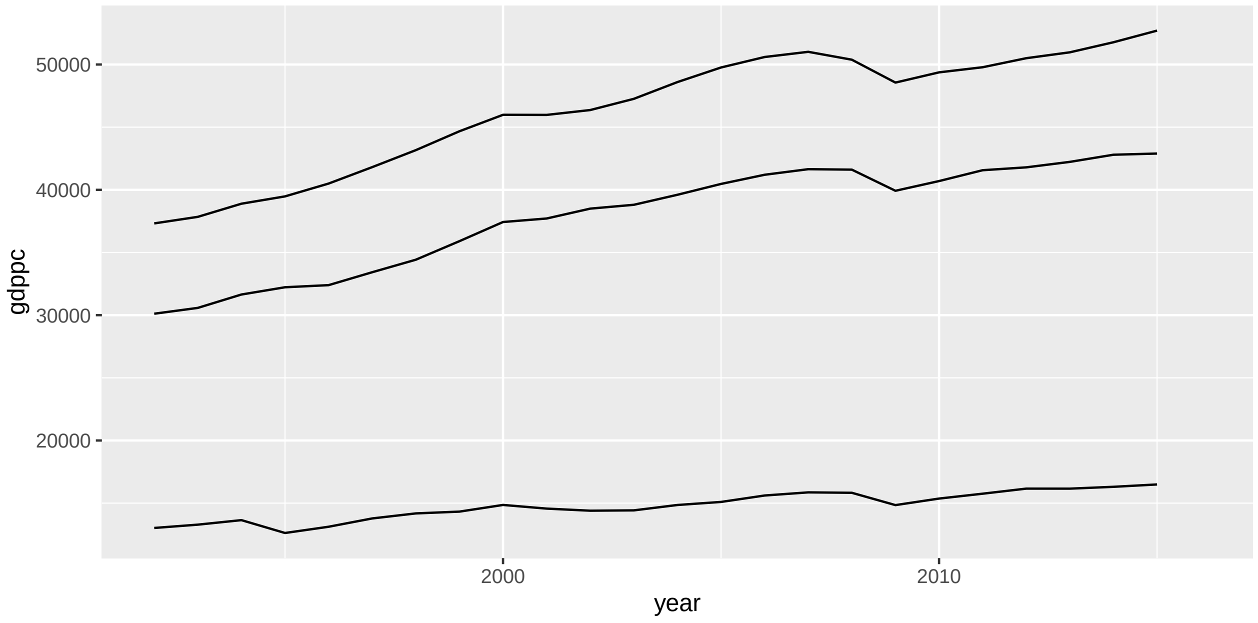
Subsetting Data for Clearer Line Plots

There are a couple of ways to subset the data for a more focused analysis:

First, we can subset the data using the `subset()` function directly within our `ggplot()` command to isolate specific countries.

Helpful hint: This approach uses the `%in%` operator combined with the `c()` function to select the countries we want to analyze.

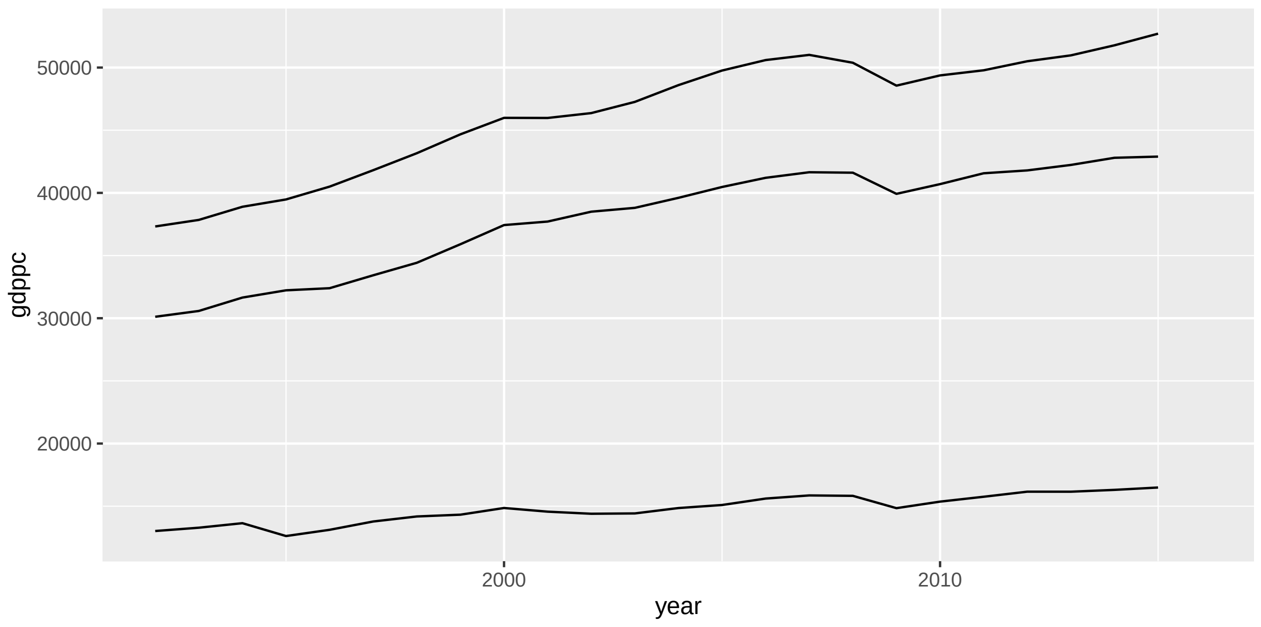
```r
# Line plot subsetting for Canada, Mexico, and the United States
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line()
```



Alternatively, you can use `filter()` for a similar output using methods aligned with tidy data principles.

```r
# Line plot filtering for Canada, Mexico, and the United States
mylineplot <- dat %>%
  filter(country %in% c("Canada", "Mexico", "United States")) %>%
  ggplot(aes(x = year, y = gdppc, group = country)) +
    geom_line()

mylineplot
```



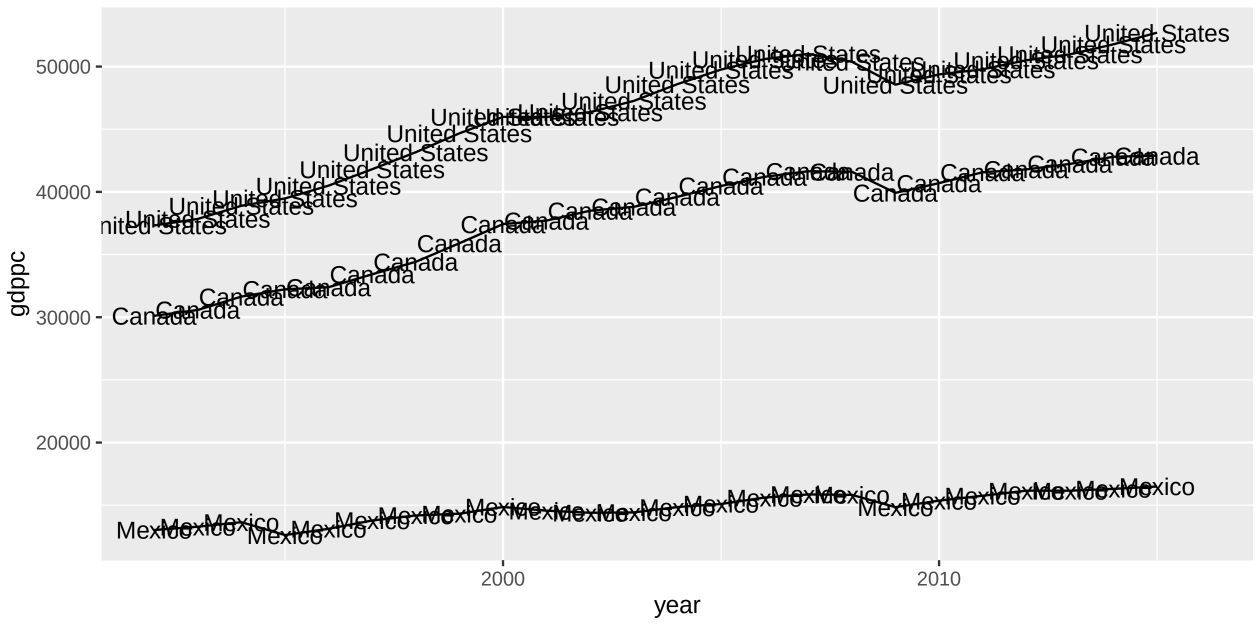
Enhancing Line Plot Readability

Now that we've isolated specific countries, let's add elements to make the plot more readable.

Adding Labels

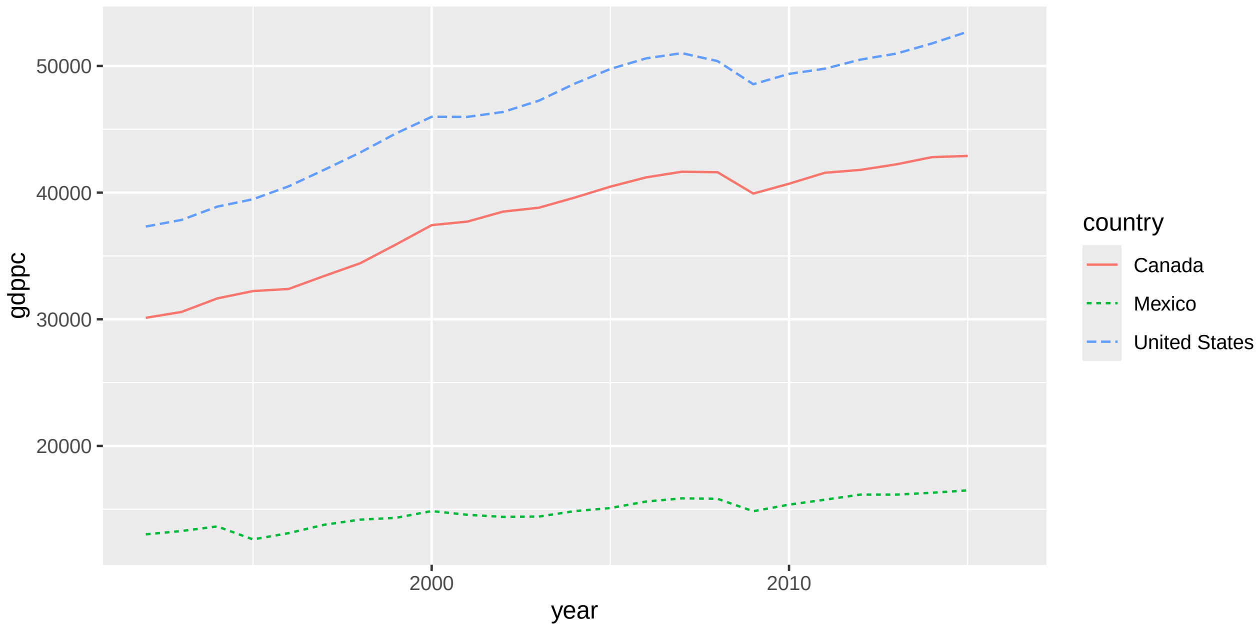
In the graphs above, it's not immediately clear which line corresponds to the per capita GDP evolution of each country. To clarify this, we can use labels. One option is using the `geom_text()` function, although this approach would place a label on every data point for each year, which could clutter the graph.

```r
# Line plot labeling axis and lines by country
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country)) +
  geom_line() +
  geom_text()
```



Alternatively, we can add labels selectively or color-code the lines to enhance the clarity of the line plot. This can be achieved by adding `color` and `linetype` parameters within the `aes()` function of the `ggplot()` command.

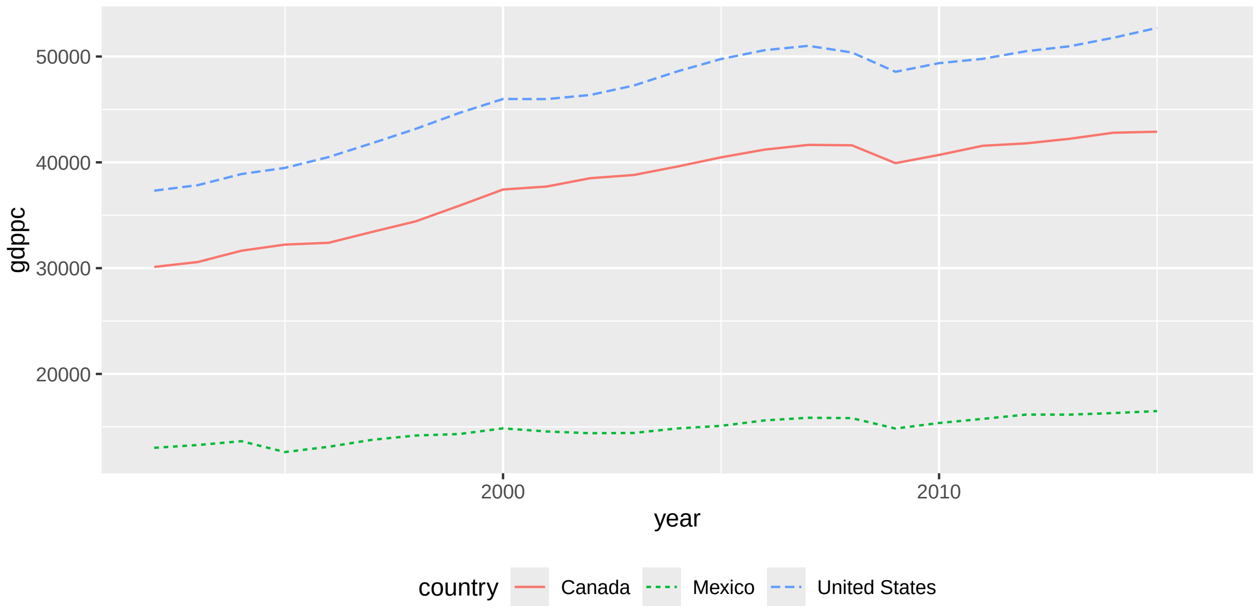
```
# Line plot adding color and changing the linetype by country
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country, color = country,
           linetype = country)) +
  geom_line()
```



Adding a Legend

Also, incorporating a legend into the line plot can further assist in identifying each country's line. By default, `ggplot2` places the legend on the right side of the plot, but you can customize its position to improve layout and readability. To do this, simply add `theme(legend.position = "desired_position")` to the plot command, ensuring to include the `+` sign for proper syntax.

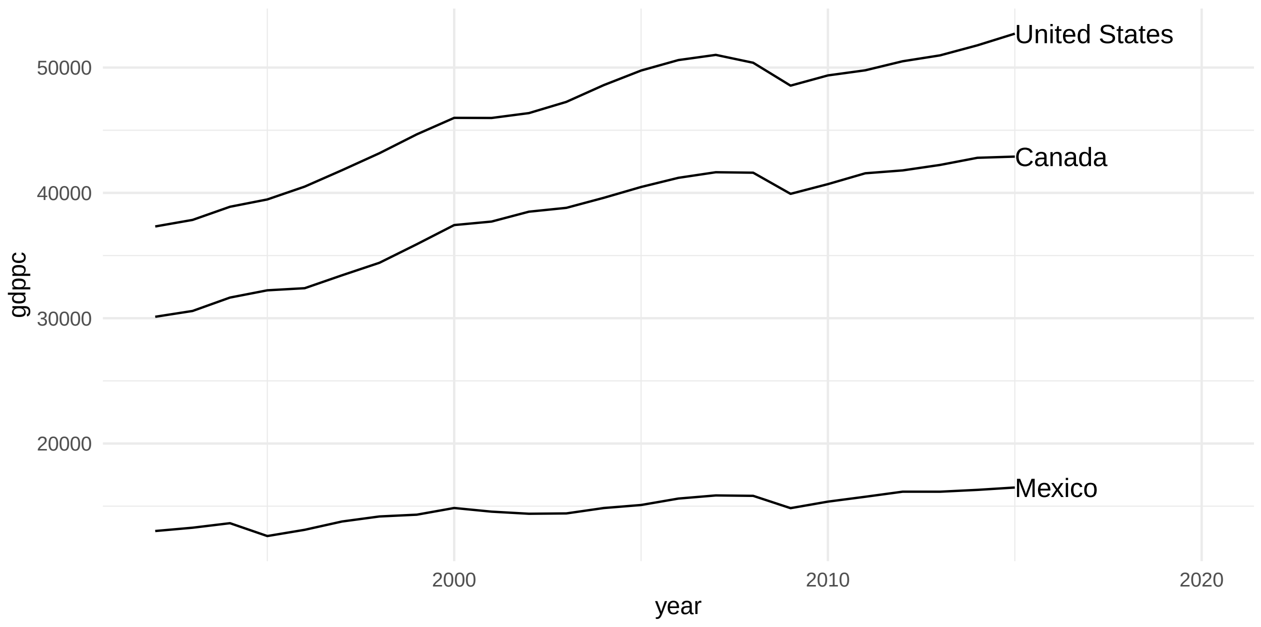
```r
# Line plot with a legend
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country, label = country, color = country,
           linetype = country)) +
  geom_line() +
  theme(legend.position = "bottom")  # Sets the legend at the bottom of the plot
```



Labeling Only the Last Data Point

Lastly, a better way to label each line is using the `geom_dl()` function in the `directlabels` package in `ggplot()`. Specify `method = ("last.points")` in the `geom_dl()` function to add labels only to the last point of each line, representing each country. We can also adjust the text size using the `cex` argument and fine-tune the x-axis range to ensure clear labeling with the `coord_cartesian()` function.

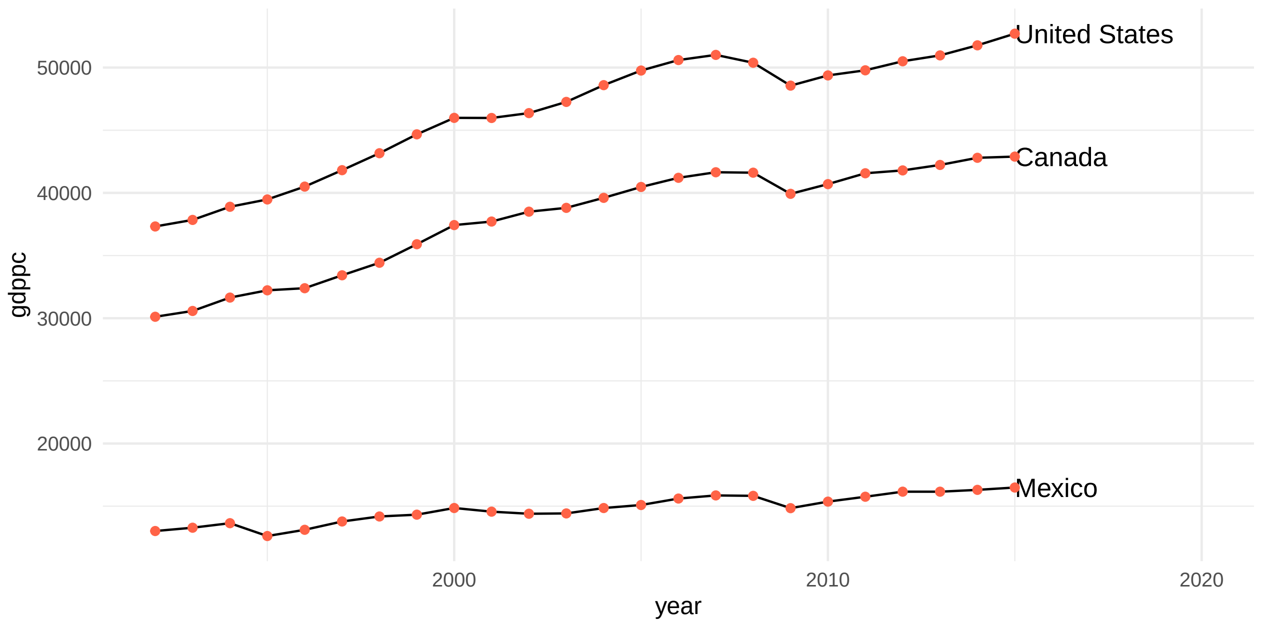
```r
# Line plot labeling only to the last point of each line
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line() +
  geom_dl(aes(label = country), method = ("last.points"), cex = 0.5) +
  coord_cartesian(xlim = c(1992, 2020)) +
  theme_minimal()
```



This visualization reveals closely linked economic trends among the NAFTA countries. The US, Canada, and Mexico share similar patterns of economic crises and growth, although the rise in per capita GDP is noticeably sharper in Canada and the US compared to Mexico.

Additionally, while not essential for this example, adding points to the plot with `geom_point()` can illustrate the specific yearly data contributing to each trend line. This technique uses `ggplot`'s layering capability to graph multiple geometric objects in one plot.

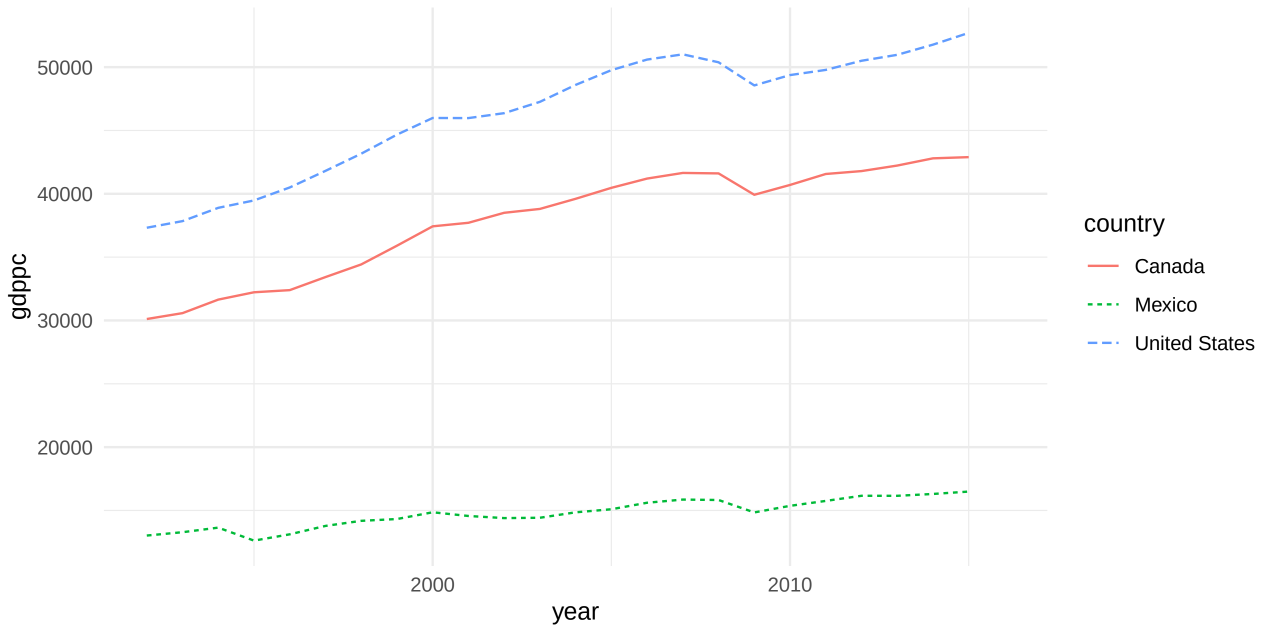
```r
# Line plot adding red points to the plot
ggplot(subset(dat, country %in% c("Canada", "Mexico", "United States")),
       aes(x = year, y = gdppc, group = country)) +
  geom_line() +
  geom_dl(aes(label = country), method = ("last.points"), cex = 0.5) +
  geom_point(color = "tomato") +
  coord_cartesian(xlim = c(1992, 2020)) +
  theme_minimal()
```



Improving Aesthetics

Just like we did in Data Visualization II Walkthrough 1, we can apply `theme_minimal()` again for a clean, black-and-white background, which streamlines the plot and enhances the contrast between line colors.

```
# Line plot with black and white background
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()
```



By following this walkthrough, you've practiced creating informative and visually clear line plots. Stay tuned for the next part, where we'll dive into integrating geographic data into our visualizations!