Introduction to Business Analysis

Lecture 6: Visualization in R

Iegor Vysnevskyi

Woosong University

April 4/10, 2023

Agenda

1. Benefits of using R for Visualization
2. Basic Visualization Techniques in R
3. Intro to ggplot2 package in R
4. ggplot2 Practical Use
5. In-class Assignment

1. Benefits of using R for Visualization

Key Benefits of using R for Visualization:

- ***Integration with Data Analysis:*** R has native support for working with data frames and matrices, allowing for seamless integration between analysis and visualization.
- ***Rich and Extensive Visualization Capabilities:*** R has a vast library of visualization packages, providing a wide range of chart types and customization options for static and interactive visualizations.
- ***Open-Source and Free:*** R is an open-source language that is free to download and use, making it a cost-effective solution for data visualization and analysis.

Key Benefits of using R for Visualization:

- ***Reproducibility and Reusability:*** R scripts can be used to create and save visualizations, allowing for easy reproduction and sharing of results. R code and packages are also widely available online, allowing for easy reuse and customization of existing visualization templates.
- ***Integration with Other Tools:*** R can be integrated with other tools and languages commonly used in data analysis, such as Python and SQL, allowing for a seamless workflow across different stages of data analysis and visualization.

R vs Tableau. When R is more reasonable for data visualization

- ***Complex Data Manipulation:*** R provides greater control over data cleaning and transformation, making it better suited for large or messy datasets.
- ***Customization and Control:*** R provides more customization and control over visualizations, allowing for highly customized or specialized visualizations.
- ***Statistical Analysis:*** R is better suited for advanced statistical analysis and modeling, making it useful for visualizing and communicating results to stakeholders.
- ***Programming Flexibility:*** R provides more flexibility and customization options than Tableau's point-and-click interface, making it easier to create complex or customized visualizations.
- ***Cost and Licensing:*** R is an open-source language that is free to use, making it a more cost-effective solution for data visualization and analysis.

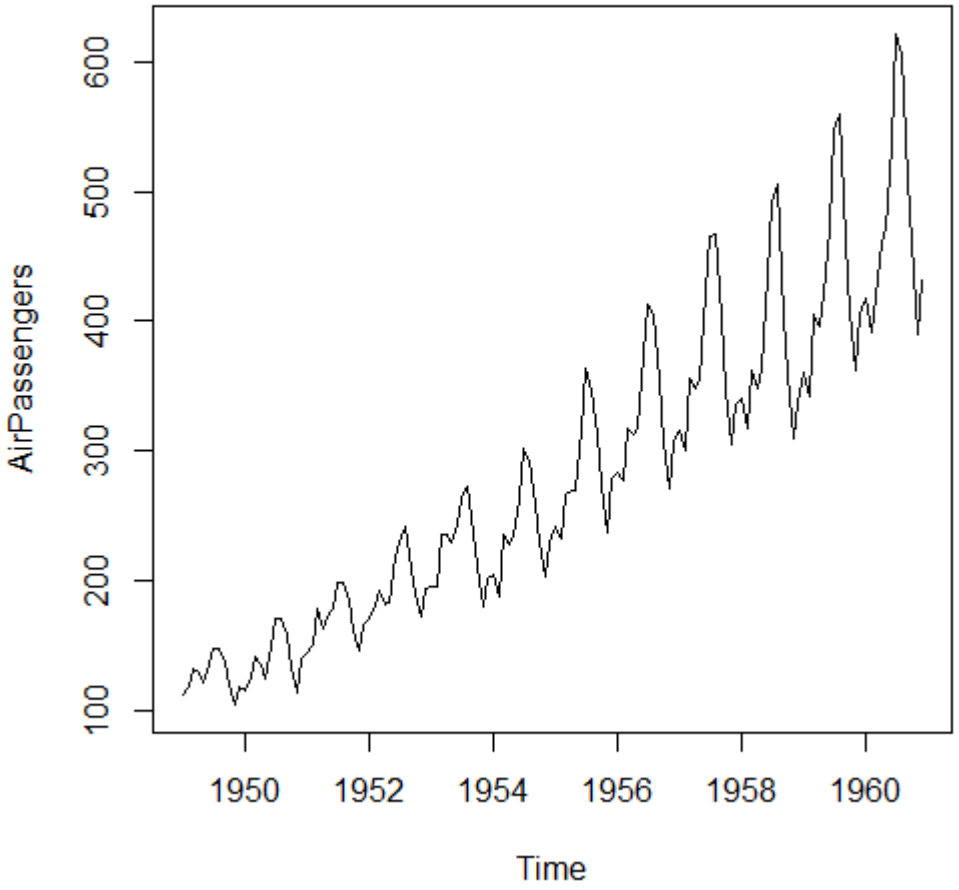
2. Basic Visualization Techniques in R

The simplest line plot

Let’s look at dataset of monthly totals of international airline passengers, 1949 to 1960. This is a time series built-in in R for practice.

```
> print(AirPassengers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```r
plot(AirPassengers)
```

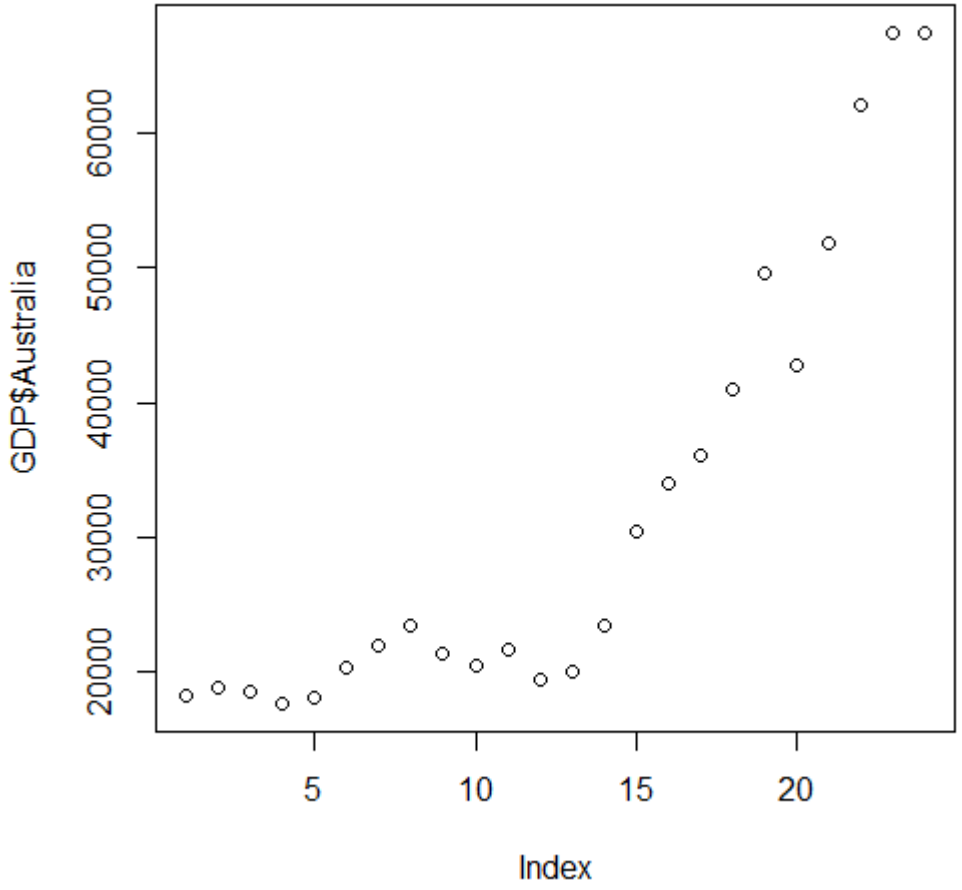


More complex line plot

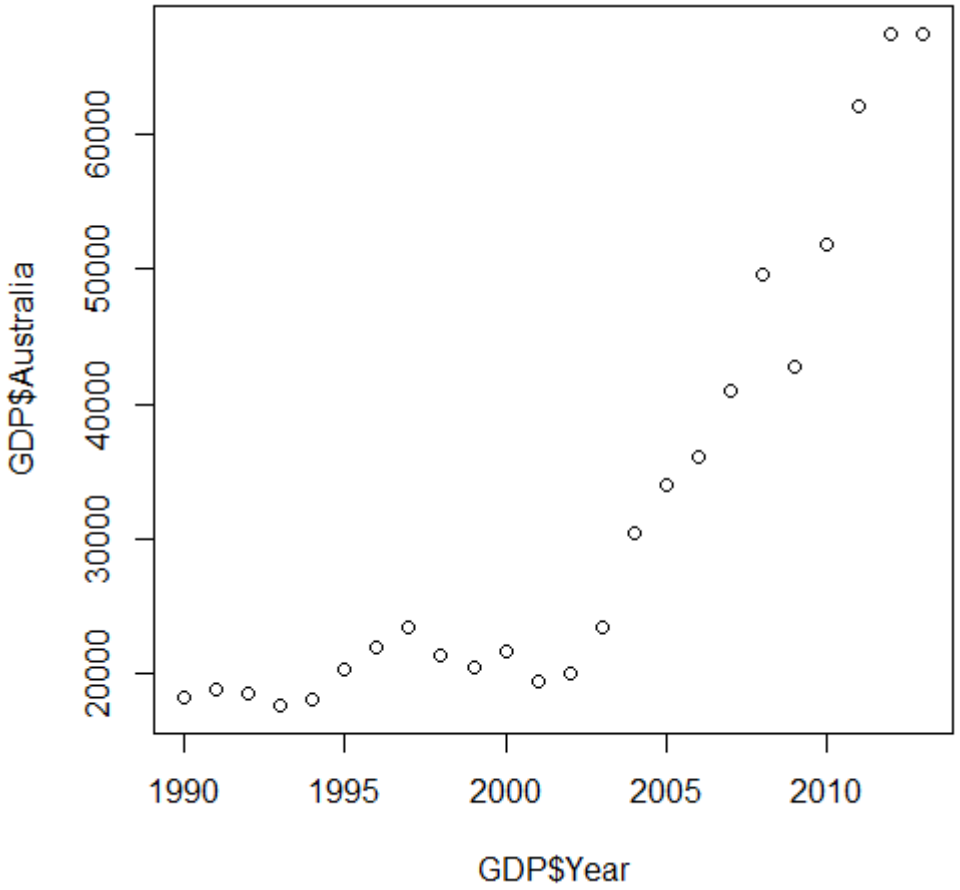
Open GDP_Yearly in R, which contains world data on GDP per Capita from 1990 – 2013

```
> head(GDP)
  Year Australia    China  Germany   France       UK    India    Japan   Russia Singapore      USA    World
1 1990  18247.39 314.4310 21583.84 21300.80 17805.25 375.8908 25123.63 3485.112  12766.19 23954.52 4220.646
2 1991  18837.19 329.7491 22603.62 21268.23 18571.36 310.0838 28540.77 3427.318  14504.52 24404.99 4357.310
3 1992  18599.00 362.8081 25604.73 23330.26 19211.86 324.4951 31013.65 3095.087  16144.33 25492.96 4591.093
4 1993  17658.08 373.8003 24735.62 21944.03 17270.12 308.5348 35451.30 2929.303  18302.37 26464.78 4604.253
5 1994  18080.70 469.2128 26375.85 23059.23 18664.39 354.8549 38814.89 2663.457  21578.14 27776.43 4882.079
6 1995  20375.30 604.2283 30887.87 26403.11 20349.96 383.5509 42522.07 2669.946  24937.31 28781.95 5323.422
```

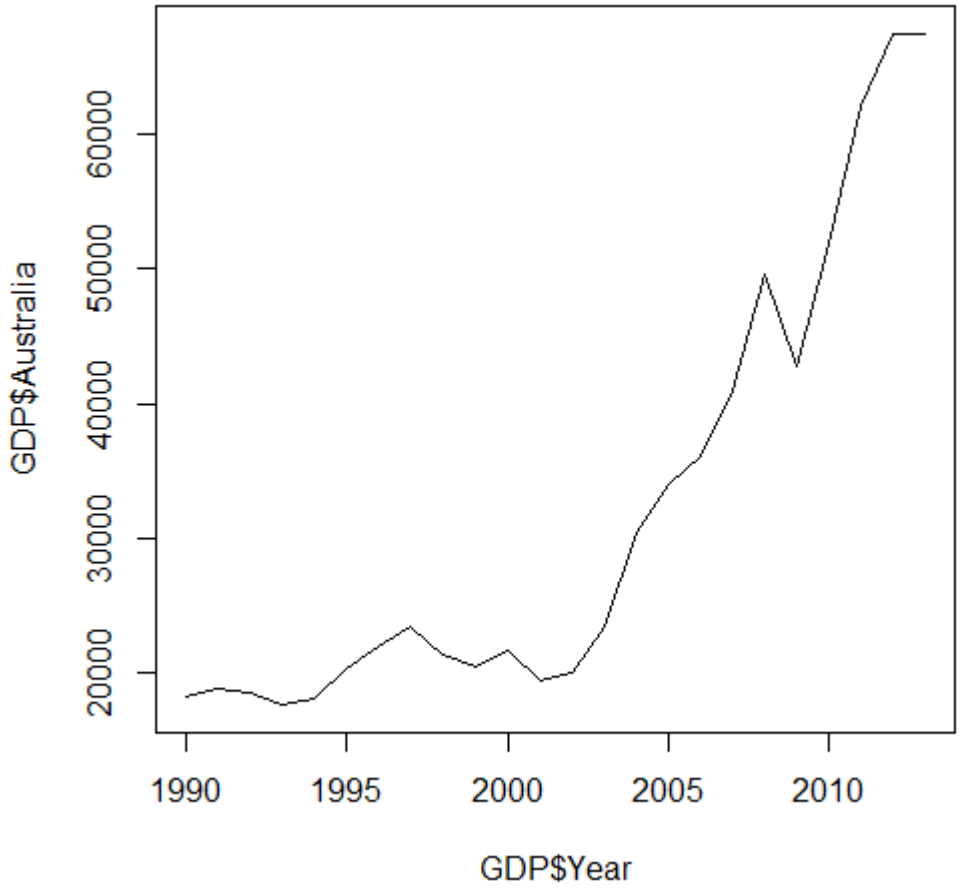
```r
plot(GDP$Australia)
```



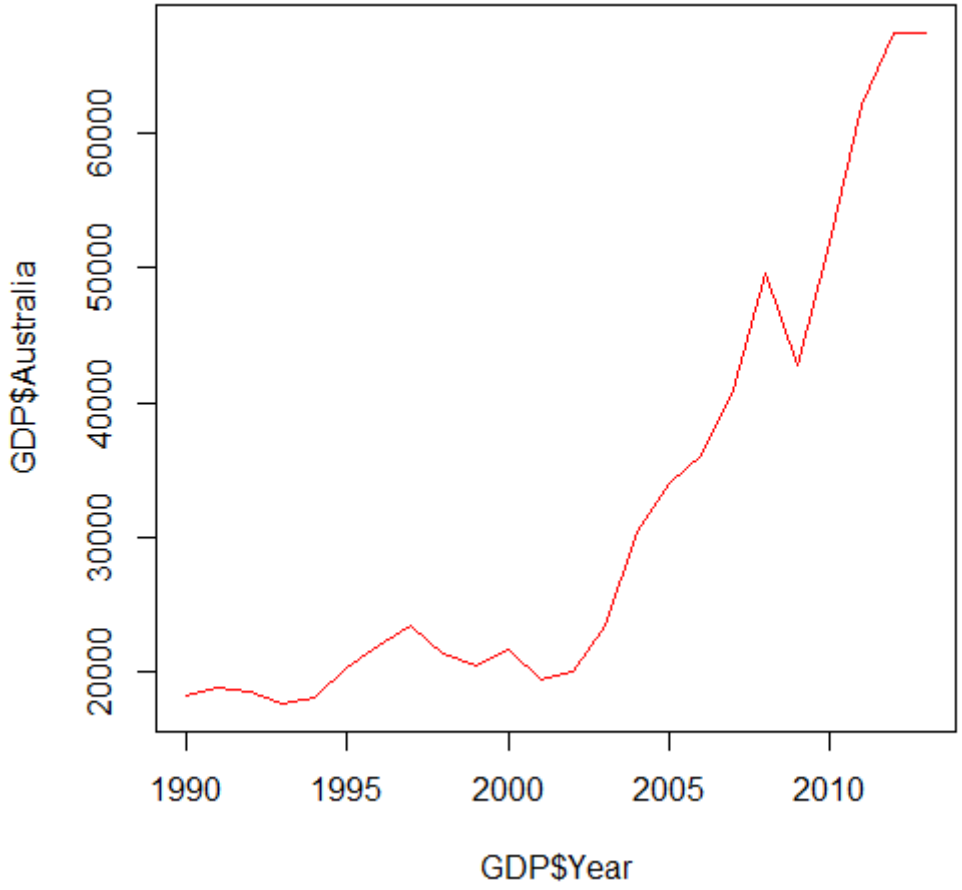
```r
#specify values for x-axis and y-axis
plot(x = GDP$Year,
     y = GDP$Australia)
```



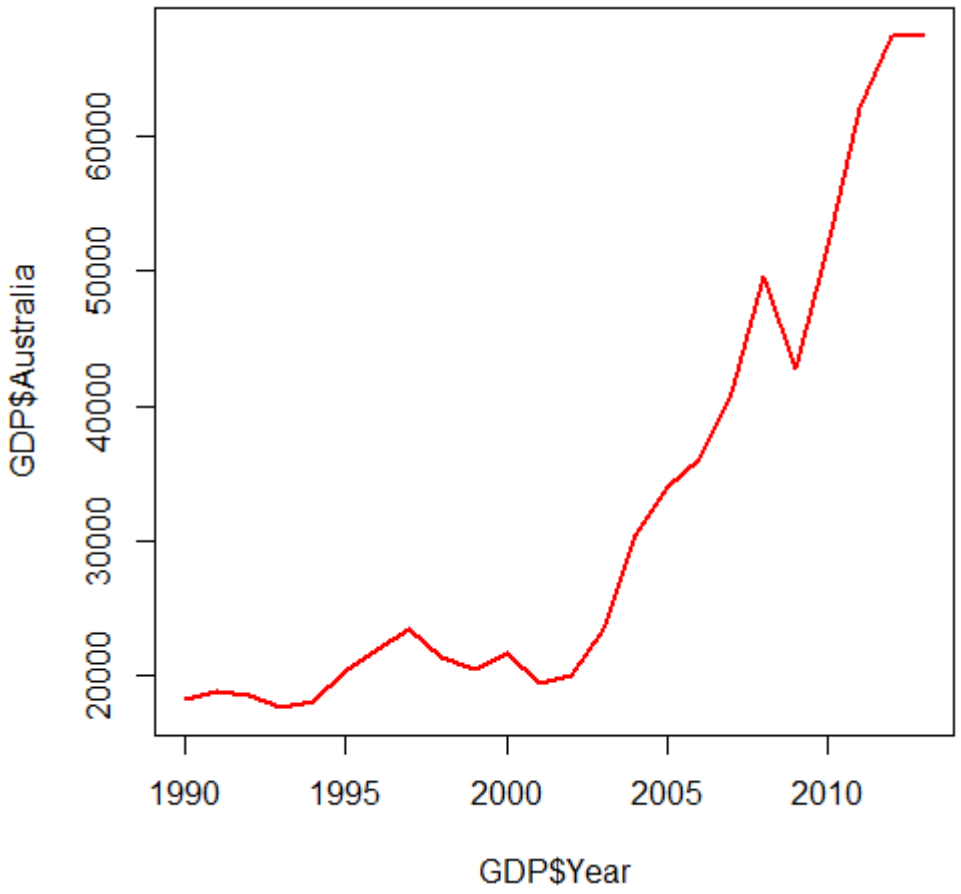
```r
plot(x = GDP$Year,
     y = GDP$Australia,
     #specify the type of the line
     type = "l")
```



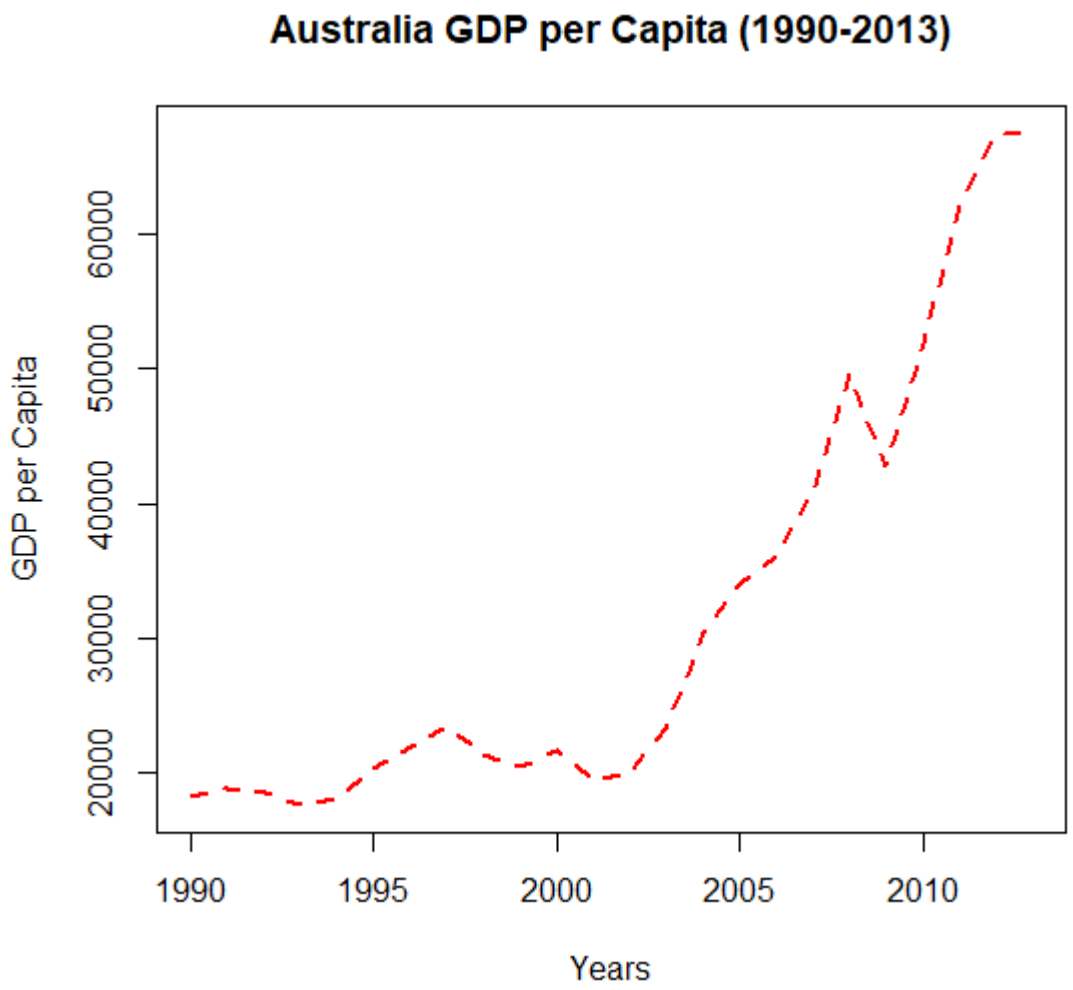
```r
plot(x = GDP$Year,
     y = GDP$Australia,
     type = "l",
     #change the color of the line
     col = "red")
```

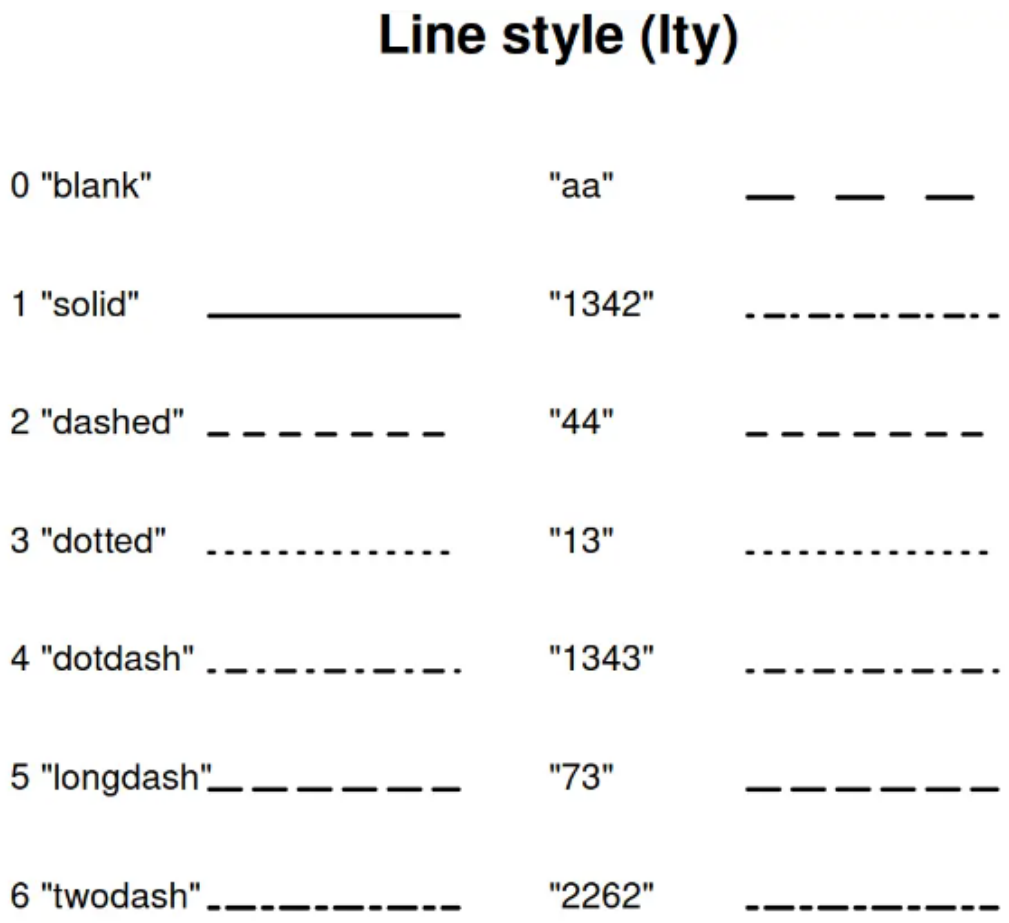


```r
plot(x = GDP$Year,
     y = GDP$Australia,
     type = "l",
     col = "red",
     #change the line thickness
     lwd = 2)
```



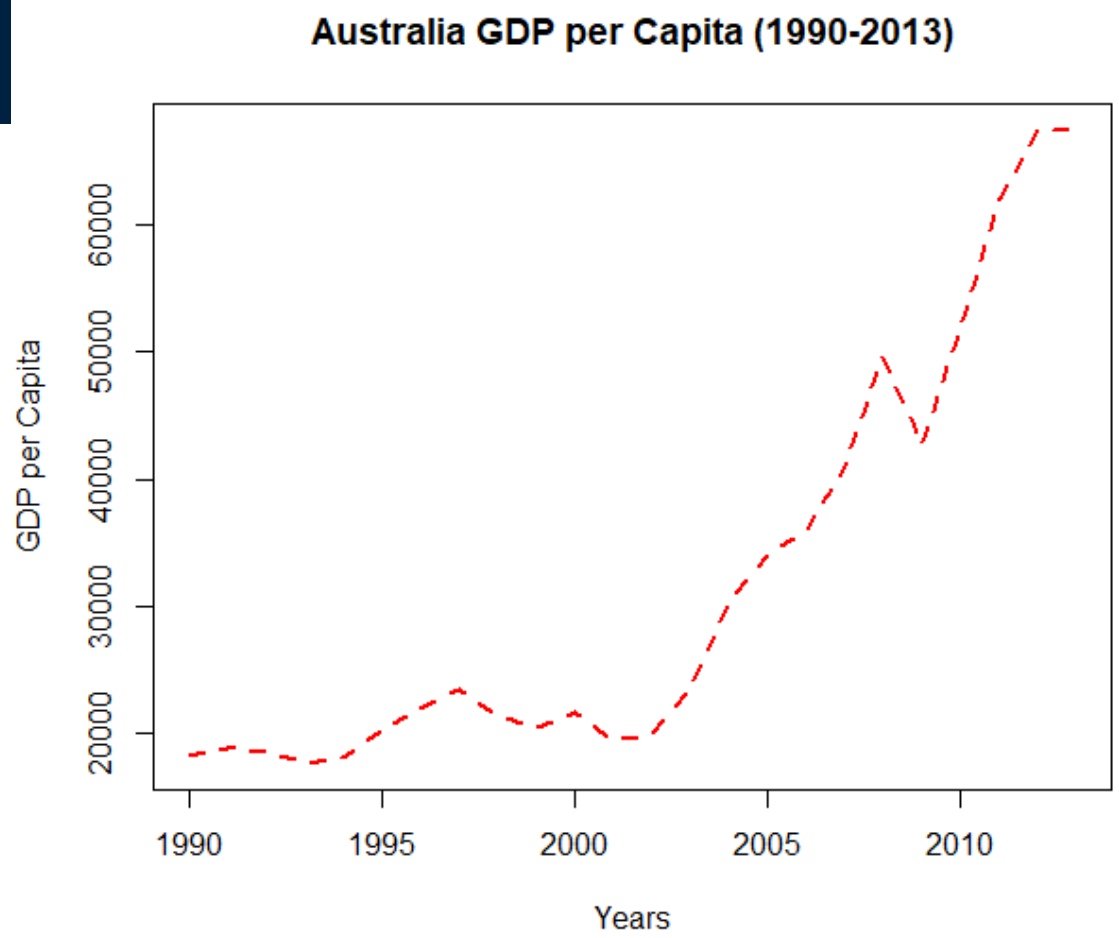
```r
plot(x = GDP$Year,
     y = GDP$Australia,
     type = "l",
     col = "red",
     lwd = 2,
     #specify the line type of the plot
     lty = 2)
```





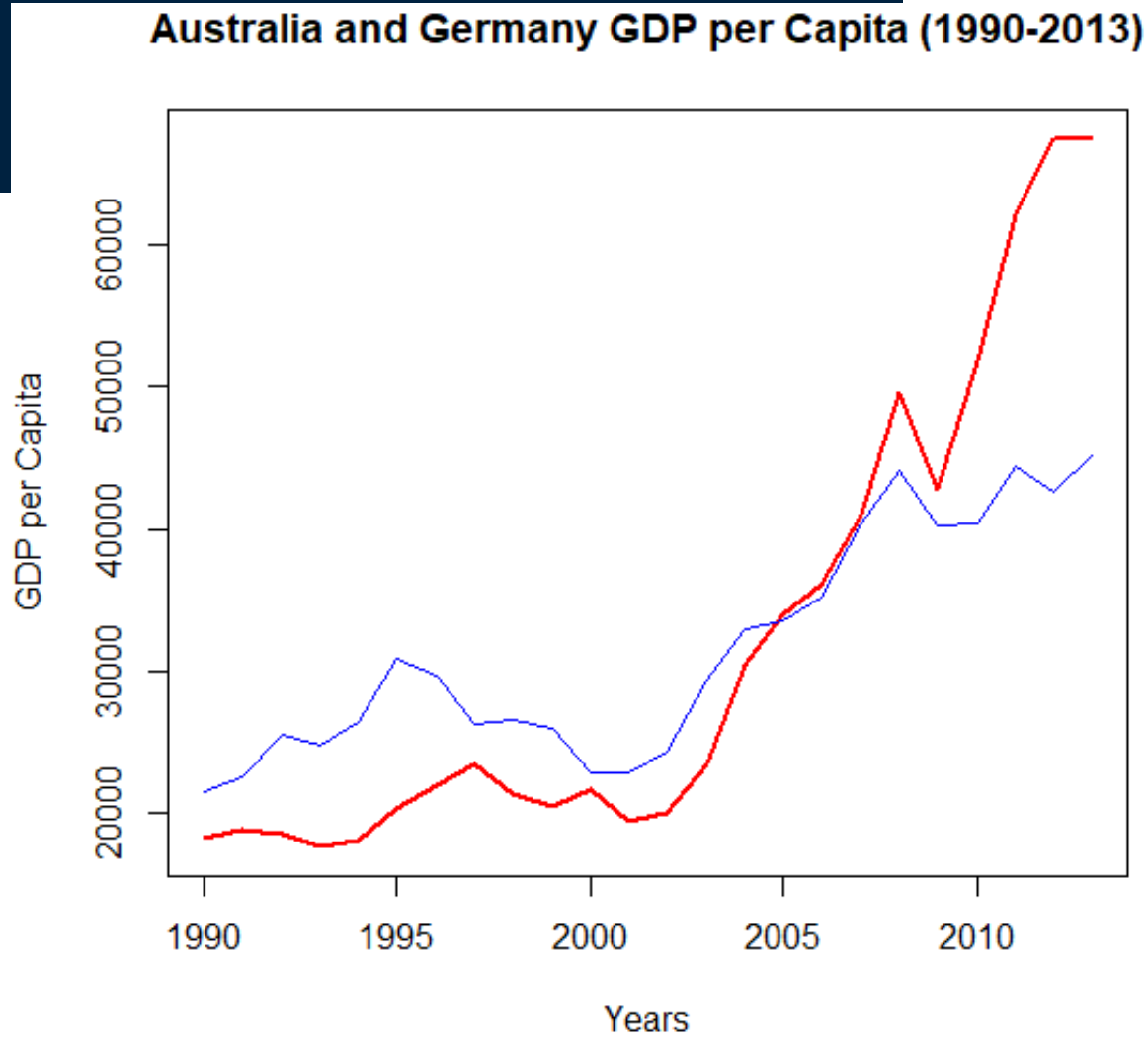
Resource: https://r-charts.com/base-r/line-types/

```r
plot(x = GDP$Year,
     y = GDP$Australia,
     type = "l",
     col = "red",
     lwd = 2,
     lty = 2,
     #add title
     main = "Australia GDP per Capita (1990-2013)",
     #add axis labels
     xlab = "Years",
     ylab = "GDP per Capita")
```



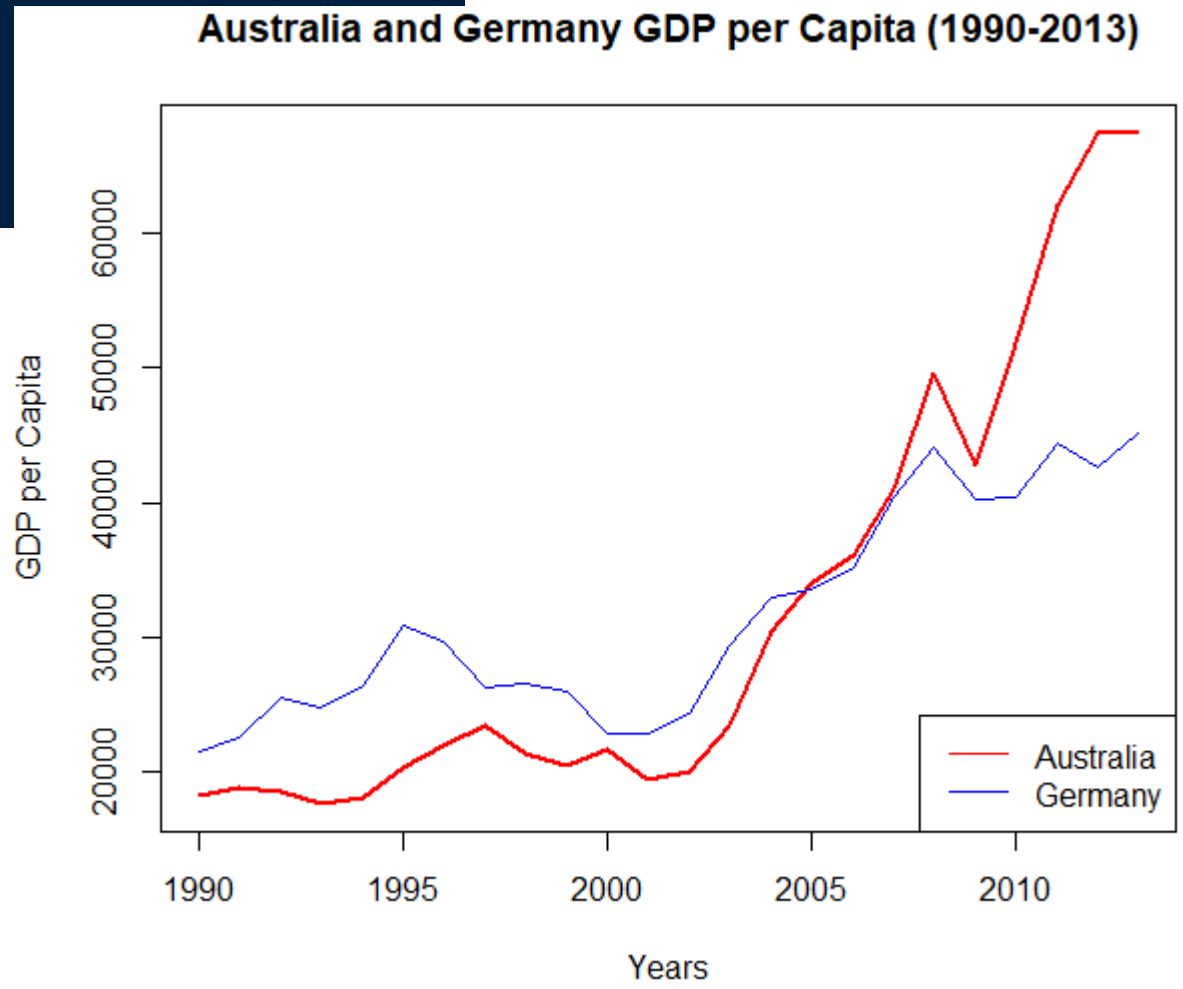
```r
plot(x = GDP$Year, y = GDP$Australia,
     type = "l", col = "red", lwd = 2,
     main = "Australia and Germany GDP per Capita (1990-2013)",
     xlab = "Years", ylab = "GDP per Capita")

#add the new line to the plot
lines(x = GDP$Year,
      y = GDP$Germany,
      col = "blue")
```



```r
plot(x = GDP$Year, y = GDP$Australia,
     type = "l", col = "red", lwd = 2,
     main = "Australia and Germany GDP per Capita (1990-2013)",
     xlab = "Years", ylab = "GDP per Capita")
lines(x = GDP$Year, y = GDP$Germany, col = "blue")

#add legend box to the plot
legend(x = "bottomright",
       legend = c("Australia", "Germany"),
       col = c("red", "blue"),
       lty = c(1, 1))
```

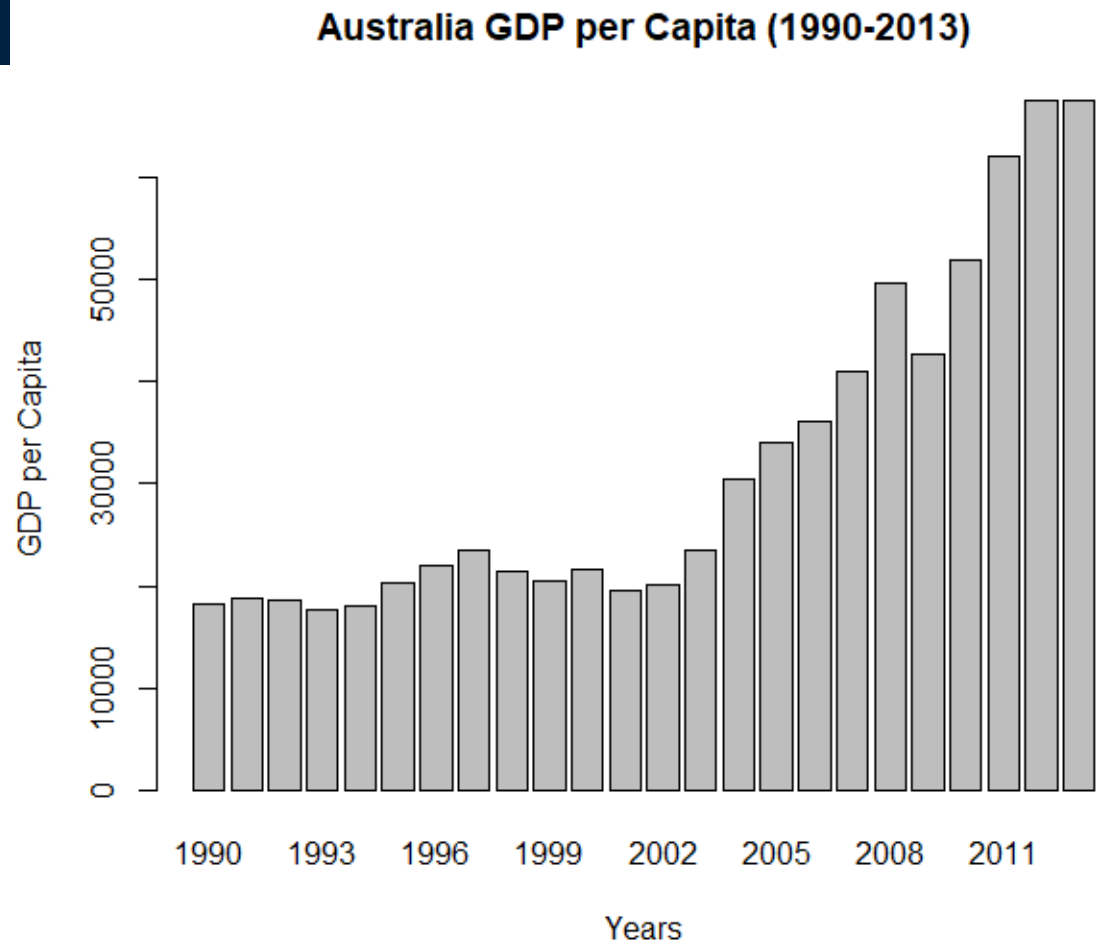


Possible type of positions:

"topleft", "top", "topright", "left", "center", "right", "bottomleft", "bottom", "bottomright“.

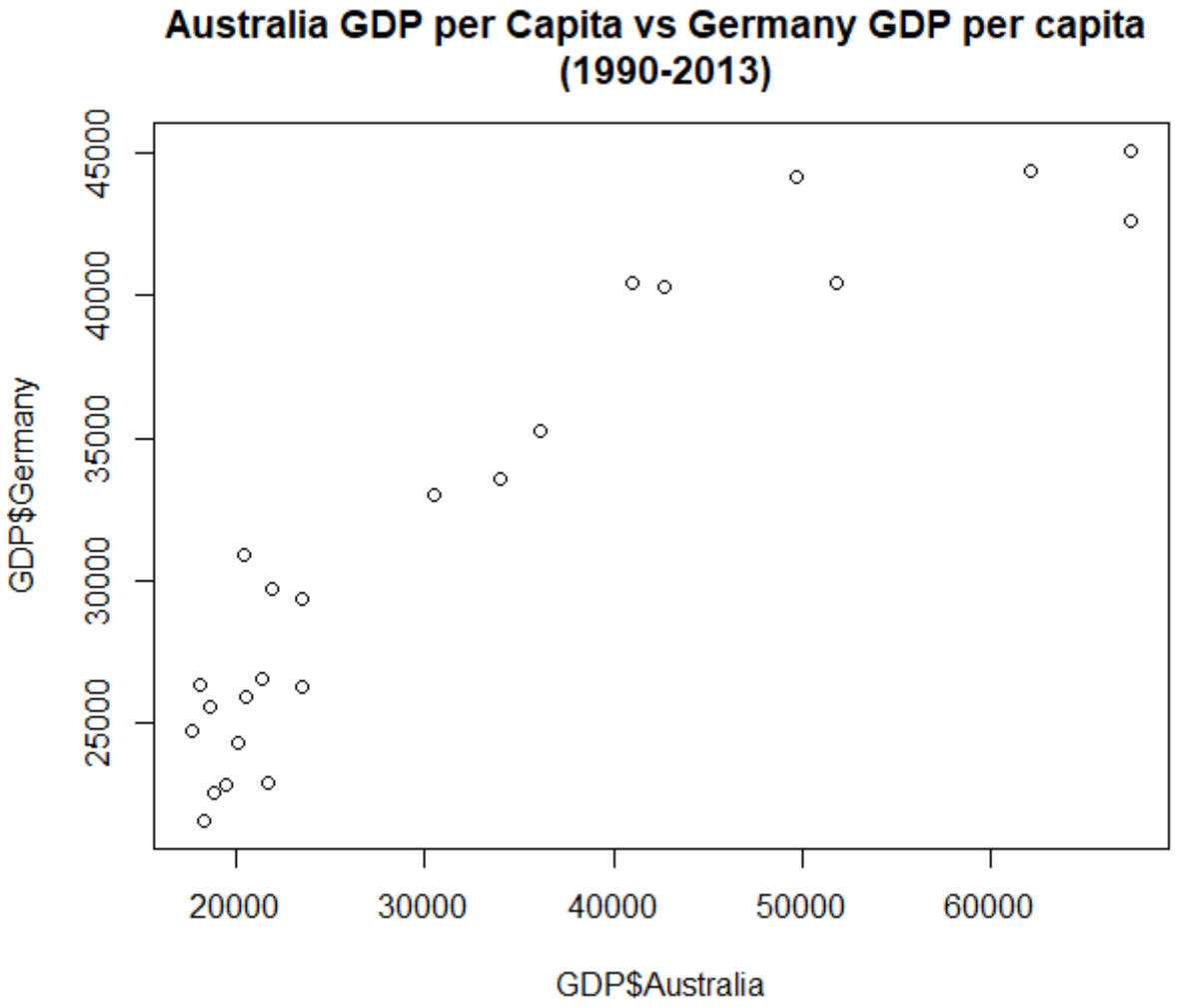
Bar plot

```
barplot(height = GDP$Australia,
        names.arg = GDP$Year,
        main = "Australia GDP per Capita (1990-2013)",
        xlab = "Years",
        ylab = "GDP per Capita")
```



Scatter plot

```
plot(GDP$Australia, GDP$Germany,
     main = "Australia GDP per Capita vs Germany GDP per capita \n (1990-2013)")
```



Pie Chart

Before building pie chart, GDP table requires some transformation.

```
#extract 2013 data
GDP_2013 <- GDP[nrow(GDP),]
```

```
> GDP_2013
   Year Australia    China  Germany   France       UK    India    Japan  Russia Singapore      USA    World
24 2013  67468.07 6807.431 45084.87 41420.76 39336.91 1498.872 38492.09 14611.7  55182.48 53142.89 10514.33
```

```
#delete the first column
GDP_2013 <- GDP_2013[, -1]
```

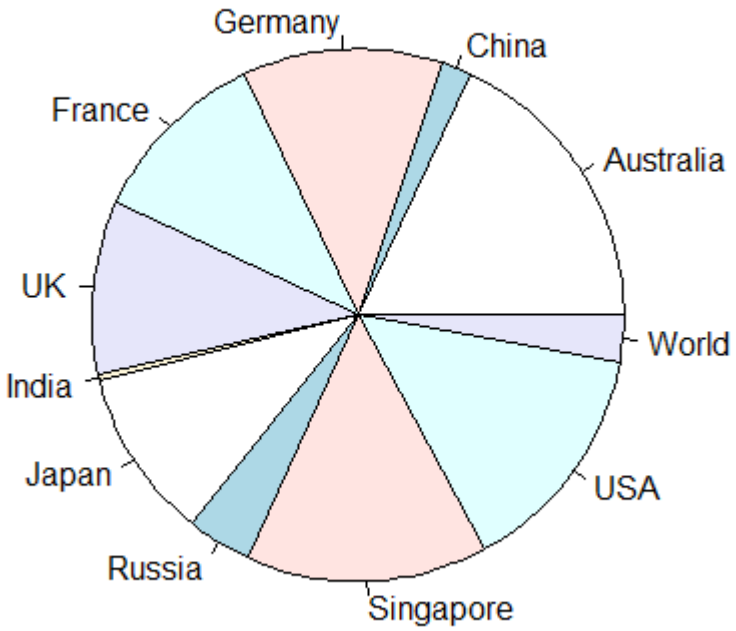
```
> GDP_2013
   Australia    China  Germany   France       UK    India    Japan  Russia Singapore      USA    World
24  67468.07 6807.431 45084.87 41420.76 39336.91 1498.872 38492.09 14611.7  55182.48 53142.89 10514.33
```

```
#change the class from data.frame to matrix
GDP_2013 <- as.matrix(GDP_2013)
```

```
> class(GDP_2013)
[1] "matrix" "array"
```

Pie Chart

```
pie(GDP_2013, labels = colnames(GDP_2013))
```



3. Intro to ggplot2 package in R

ggplot2 is an R package which provides a large variety of plotting functionality to enable better and highly customizable graphs.

The basic structure of a ***ggplot2*** code involves creating a plot object and adding layers to it, such as data points, lines, labels, and axes.

ggplot2 allows for a high degree of customization, allowing you to control almost every aspect of the plot.

When it is better to use basic plot functions

- For quick and simple visualizations that don't require a lot of customization.
- When working with small datasets that don't require advanced customization or layering.
- If you're already familiar with basic plot functions and want to quickly create a visualization without learning a new syntax or package.
- For creating simple charts such as histograms or bar charts.
- For scatter plots with large datasets as ggplot may slow down.

When it is better to use ggplot2

- For creating complex visualizations with multiple layers and aesthetics.
- When you want to customize the plot in detail.
- For creating more specialized plot types.
- When working with large datasets and need to use facets to split the plot.
- For adding advanced statistical methods, such as smoothing lines or correlation coefficients.
- For creating high-quality graphics suitable for publication or presentation purposes.

4. ggplot2 Practical Use

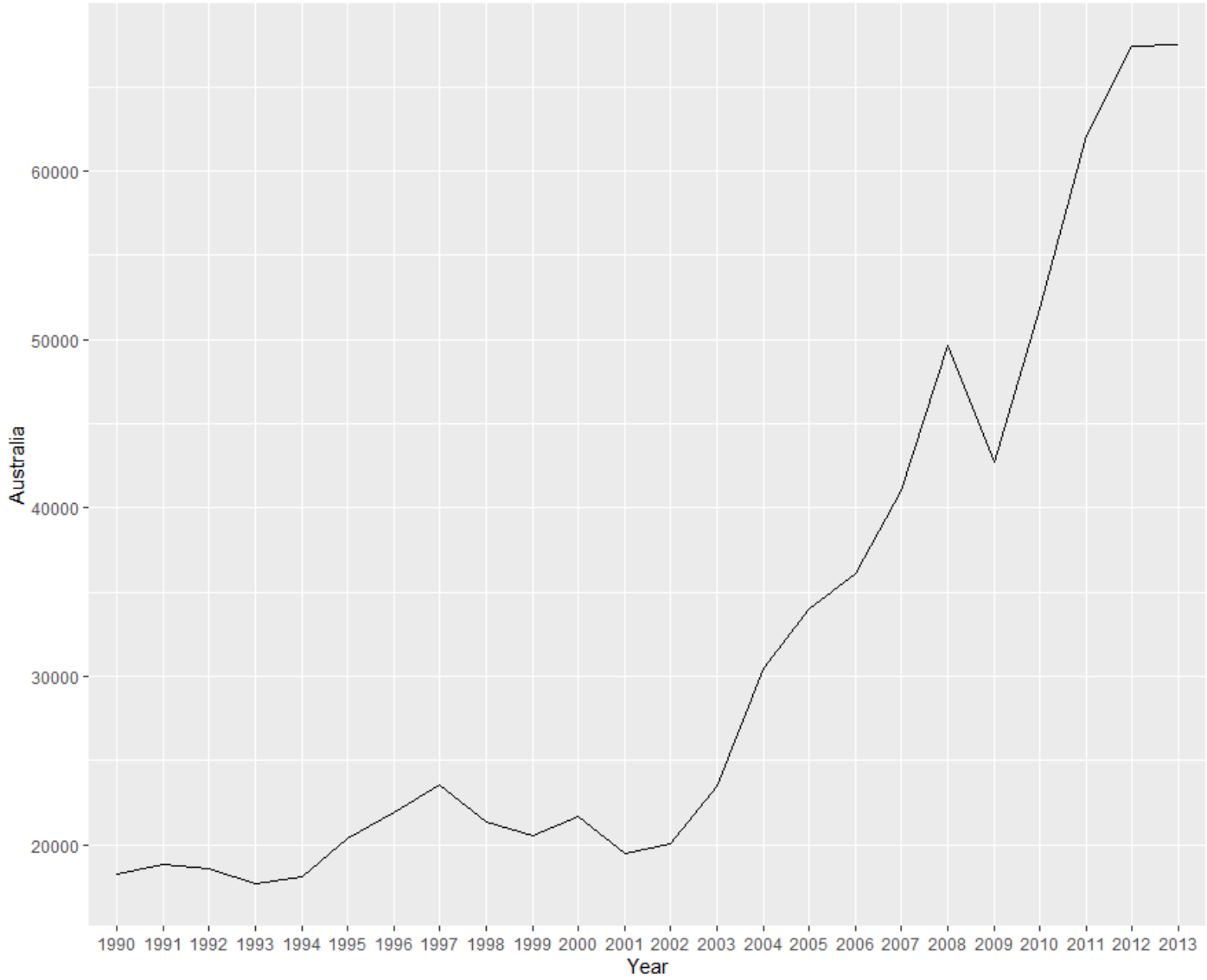
Install and loan the package

```
install.packages("ggplot2")
library(ggplot2)
```

Line plot

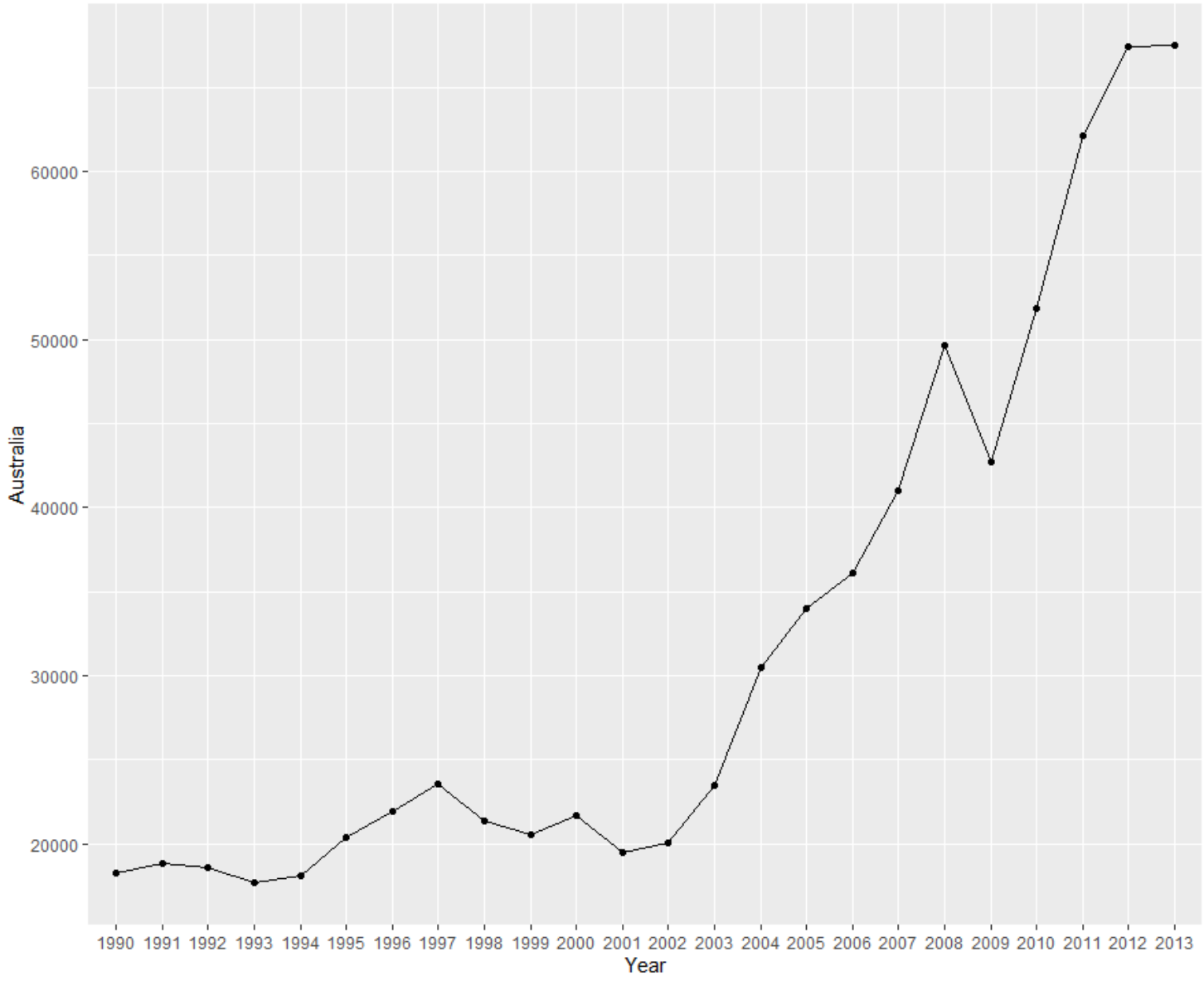
```
ggplot(GDP,
       aes(x = Year,
           y = Australia,
           group = 1)) +
  geom_line()
```

The parameter group = 1 is used to group all the data points together into a single group.



Line plot

```
ggplot(GDP,
       aes(x = Year,
           y = Australia,
           group = 1)) +
  geom_line()+
  #add points to the line
  geom_point()
```



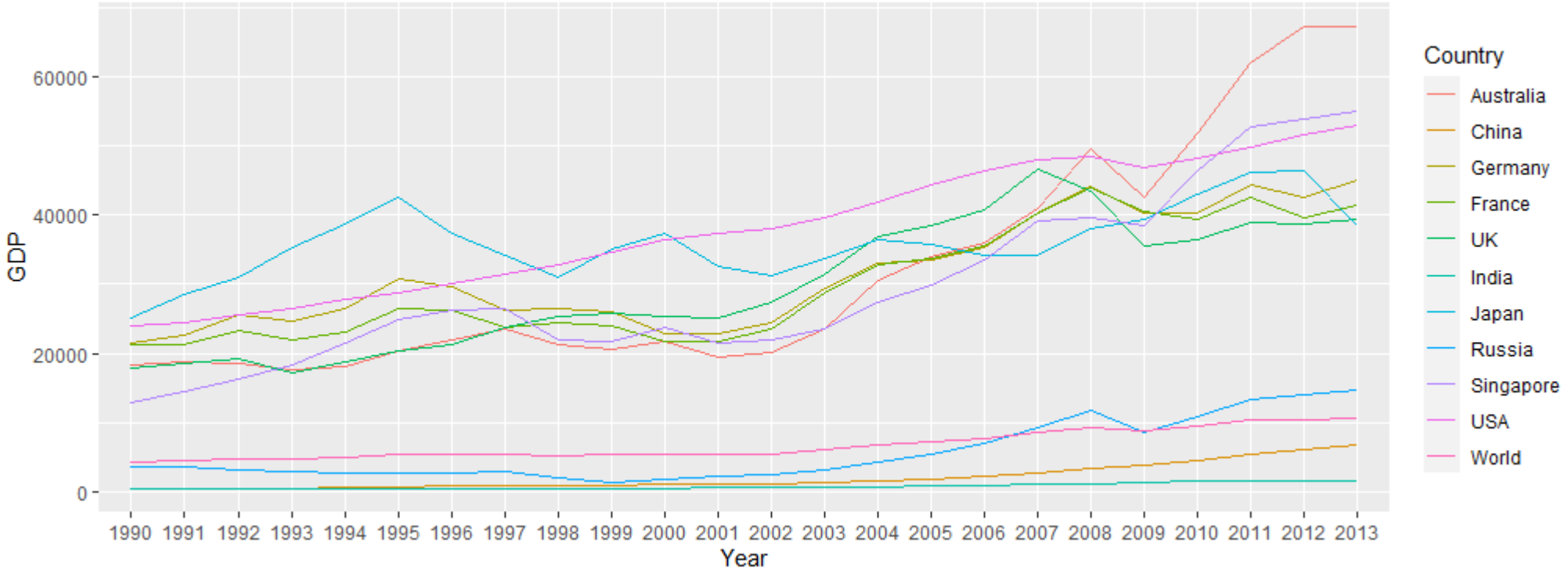
For further practice, GDP table requires to be transformed from wide format to long format.

```
> head(GDP)
  Year Australia    China  Germany   France       UK    India    Japan   Russia Singapore      USA    World
1 1990  18247.39 314.4310 21583.84 21300.80 17805.25 375.8908 25123.63 3485.112  12766.19 23954.52 4220.646
2 1991  18837.19 329.7491 22603.62 21268.23 18571.36 310.0838 28540.77 3427.318  14504.52 24404.99 4357.310
3 1992  18599.00 362.8081 25604.73 23330.26 19211.86 324.4951 31013.65 3095.087  16144.33 25492.96 4591.093
4 1993  17658.08 373.8003 24735.62 21944.03 17270.12 308.5348 35451.30 2929.303  18302.37 26464.78 4604.253
5 1994  18080.70 469.2128 26375.85 23059.23 18664.39 354.8549 38814.89 2663.457  21578.14 27776.43 4882.079
6 1995  20375.30 604.2283 30887.87 26403.11 20349.96 383.5509 42522.07 2669.946  24937.31 28781.95 5323.422
```

```
> head(GDP_l)
  Year   Country      GDP
1 1990 Australia 18247.39
2 1991 Australia 18837.19
3 1992 Australia 18599.00
4 1993 Australia 17658.08
5 1994 Australia 18080.70
6 1995 Australia 20375.30
```

Multiple line Plot

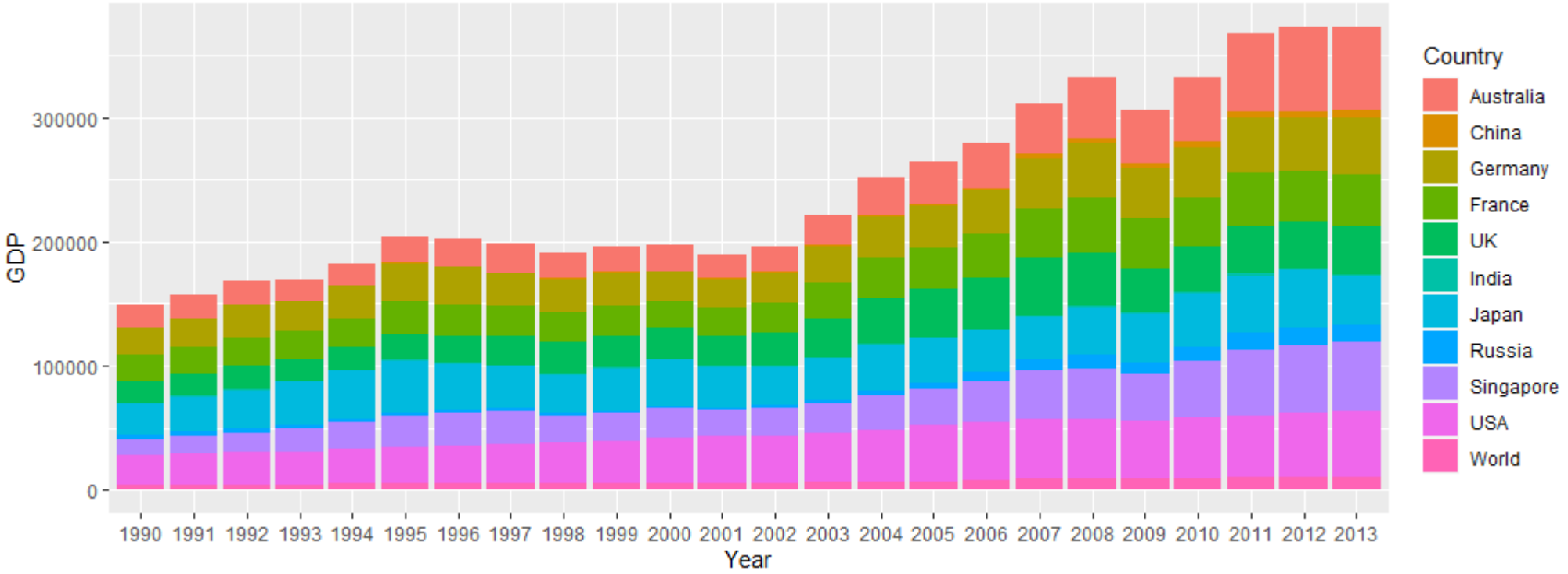
```
ggplot(GDP_l,
       aes(Year, GDP,
           group = Country,
           colour = Country))+
geom_line()
```



Bar Plot with geom_bar()

```
ggplot(GDP_l,
       aes(x = Year, y = GDP))+
  geom_bar(aes(fill = Country), stat = "identity")
```

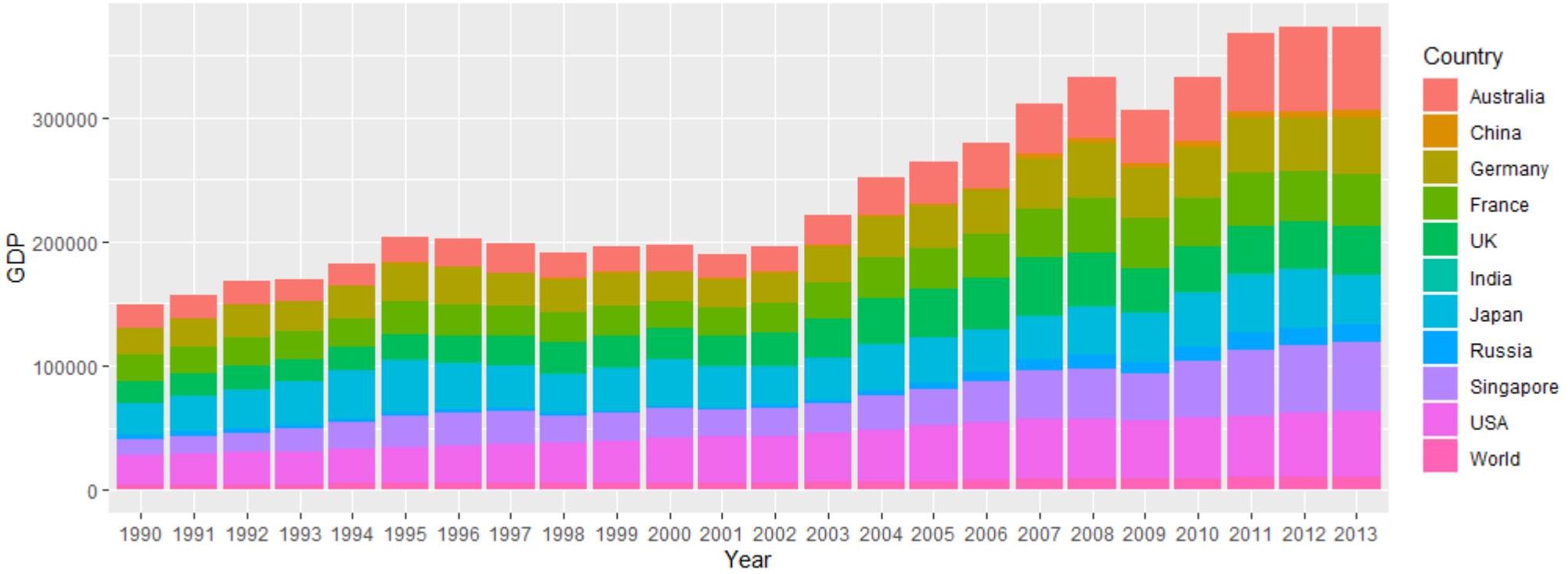
The argument stat = "identity" tells ggplot2 to use the actual data values as the heights of the bars, rather than calculating a summary statistic such as count or proportion.



Bar Plot with geom_col()

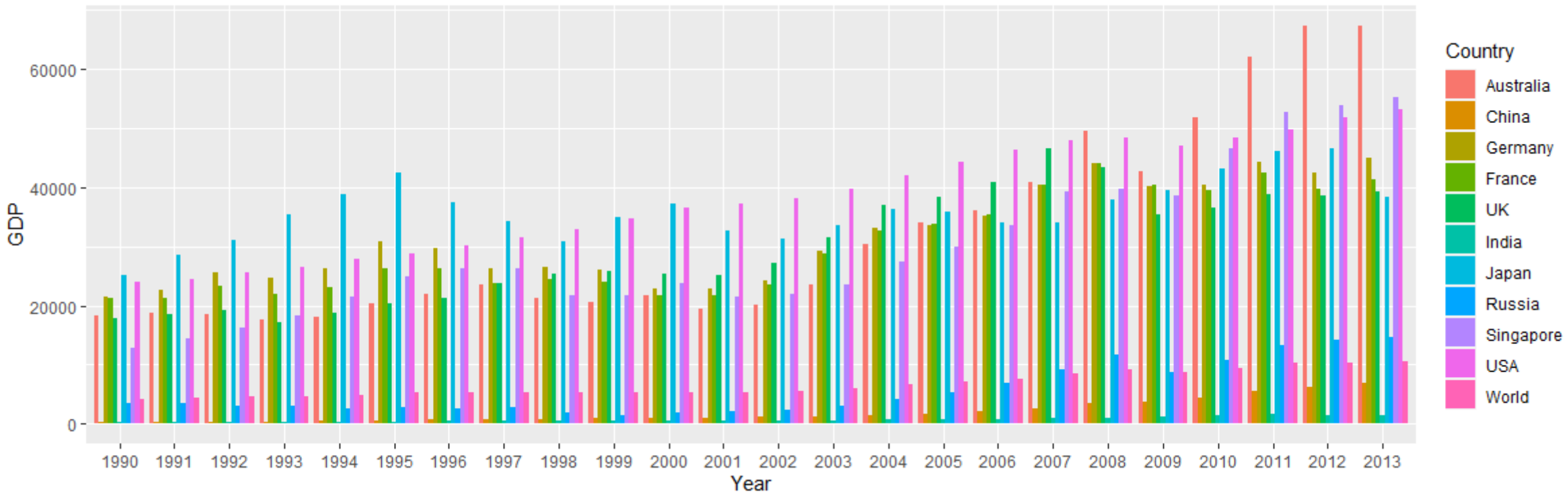
You will have the same chart using the below code as well.

```
ggplot(GDP_l,
       aes(x = Year,y = GDP))+
  geom_col(aes(fill = Country))
```



Grouped Bar Chart

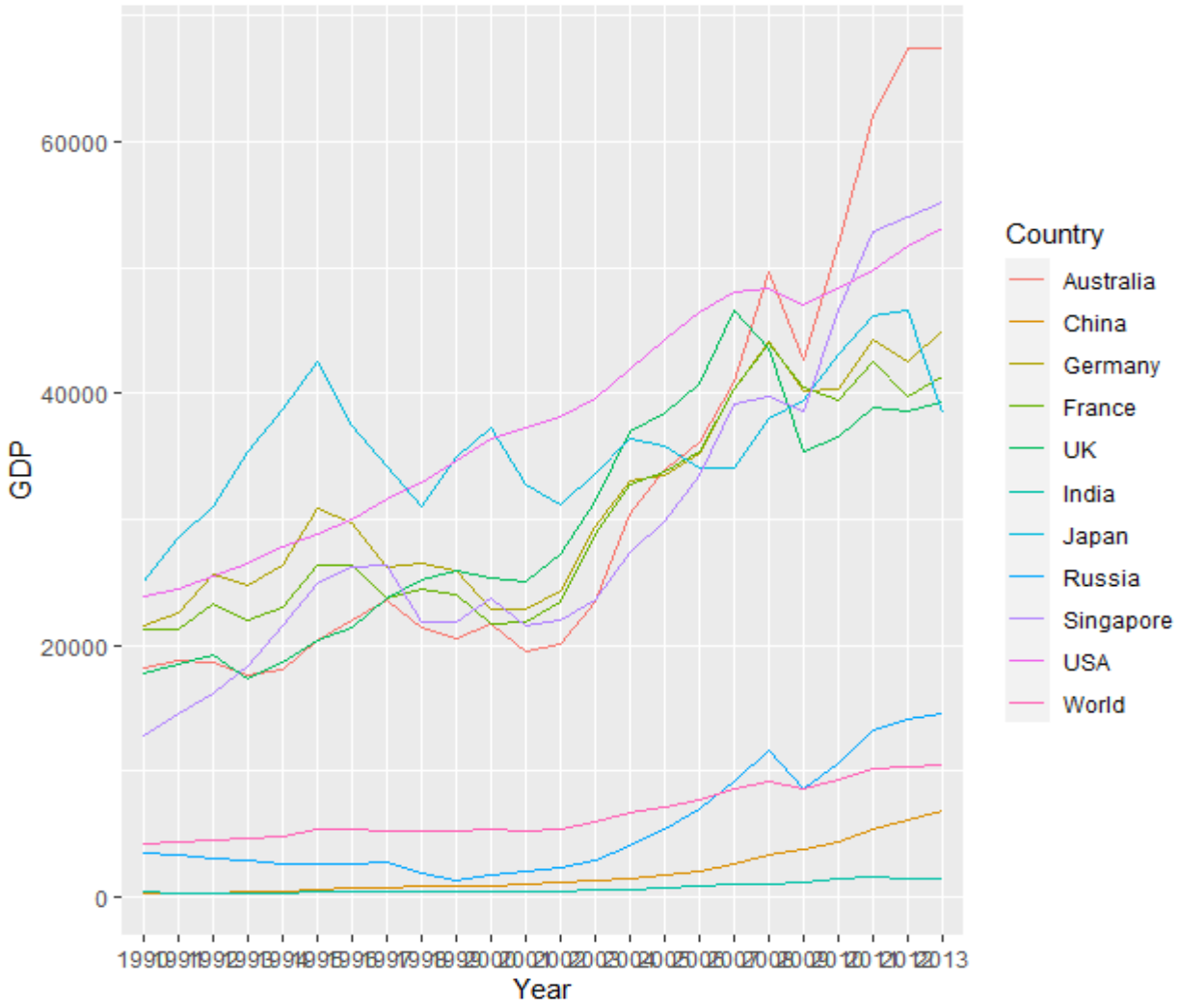
```
ggplot(GDP_l,
       aes(x = Year, y = GDP))+
  geom_bar(aes(fill = Country), stat = "identity", position = "dodge")
```



Plot Customization

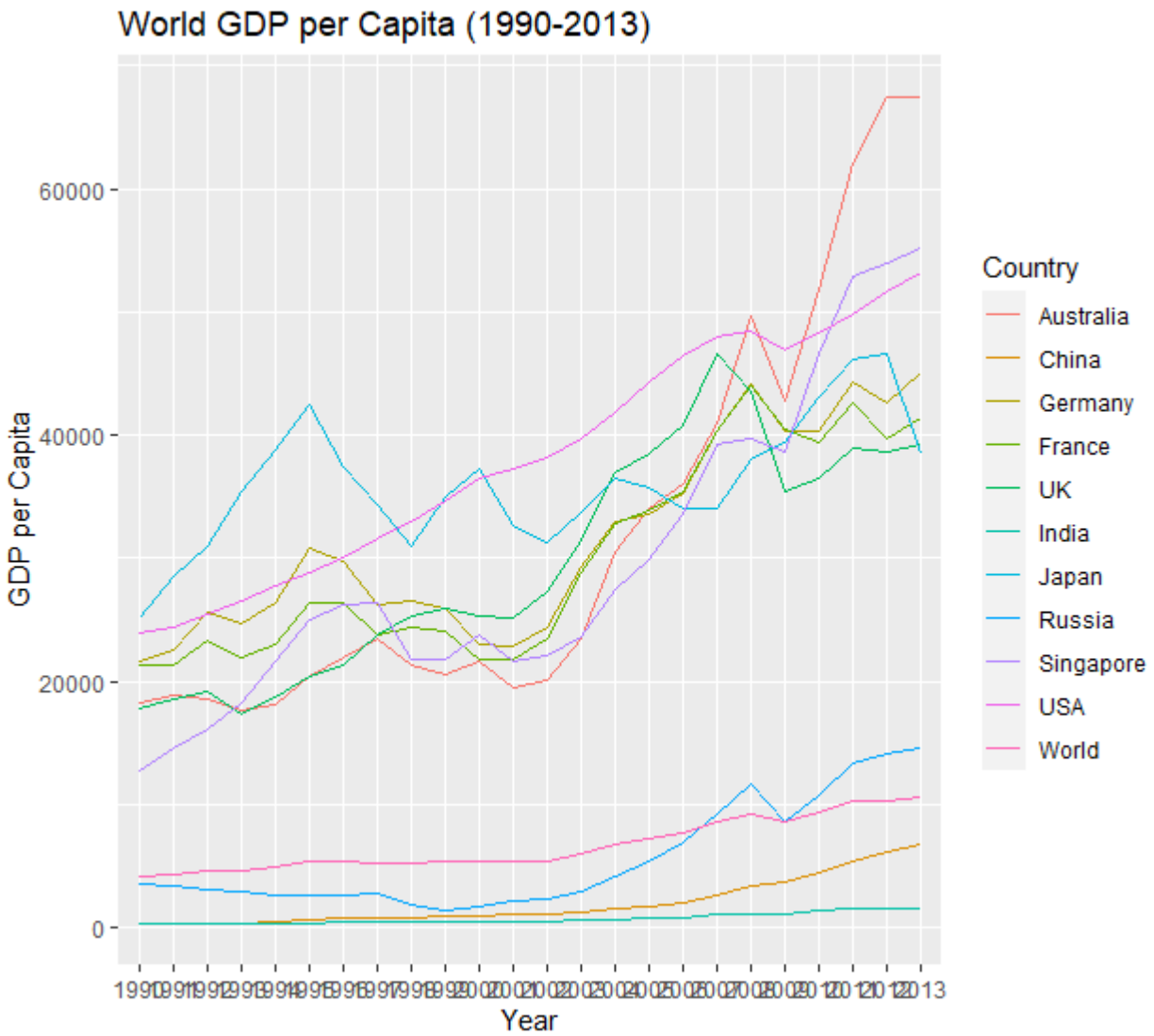
Recall the multiple line plot we made earlier today

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()
```



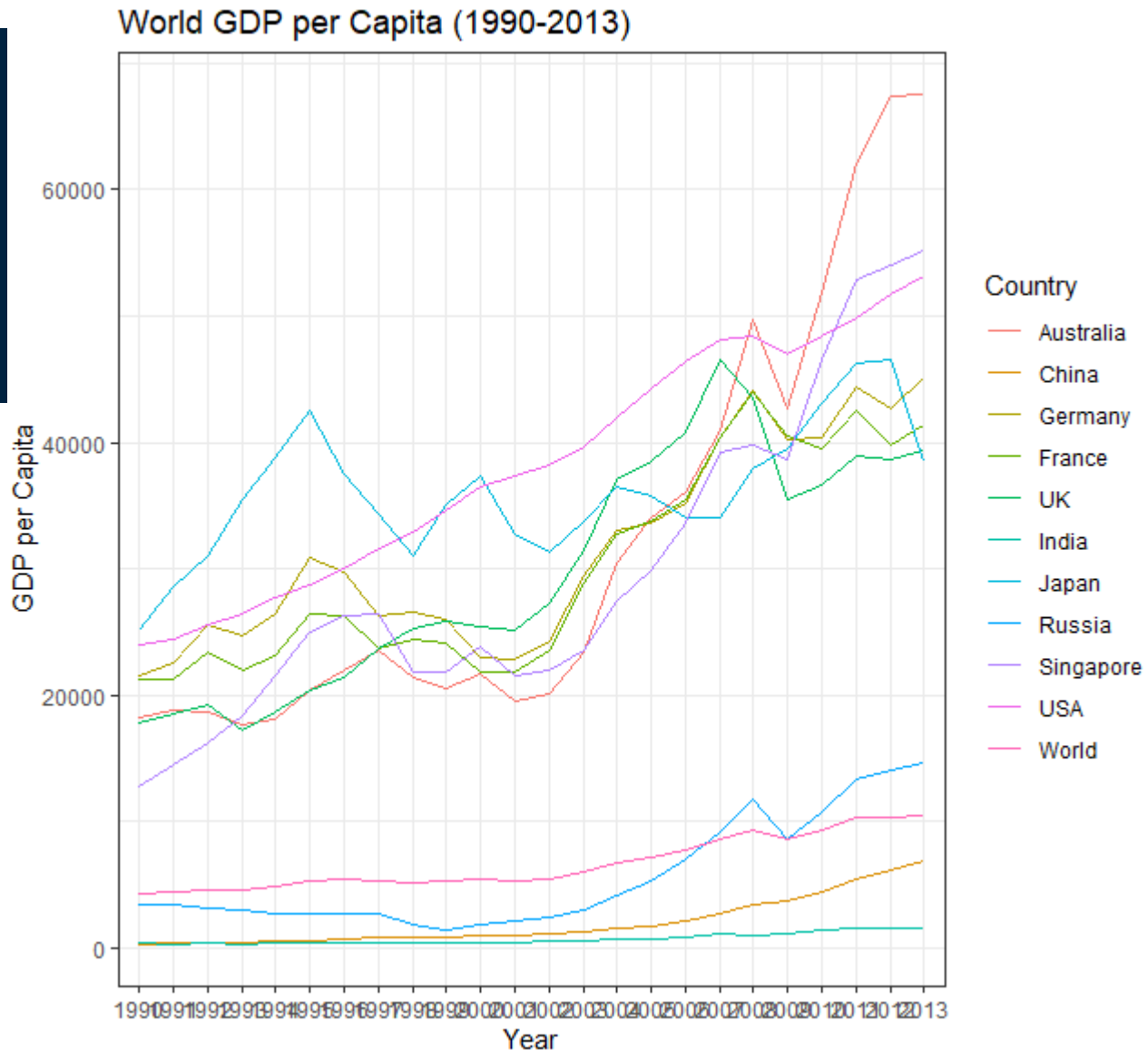
- ***Add title and labels for x-axis and y-axis***

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  #add labels
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")
```



• *Customize the background*

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  #customize the background
  theme_bw()
```



Other background theme examples:

theme_gray(), theme_minimal(), theme_classic(), theme_light(),

theme_dark(), theme_void(),

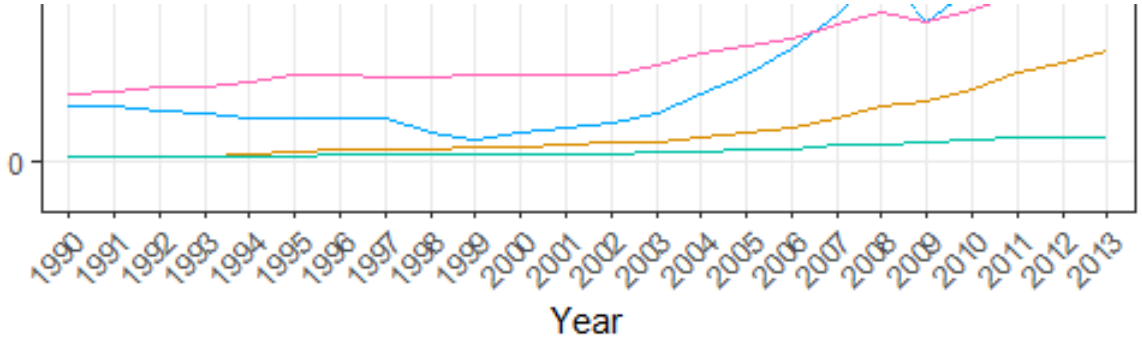
theme_linedraw(),

theme_dark_minimal(),

theme_light_minimal() etc.

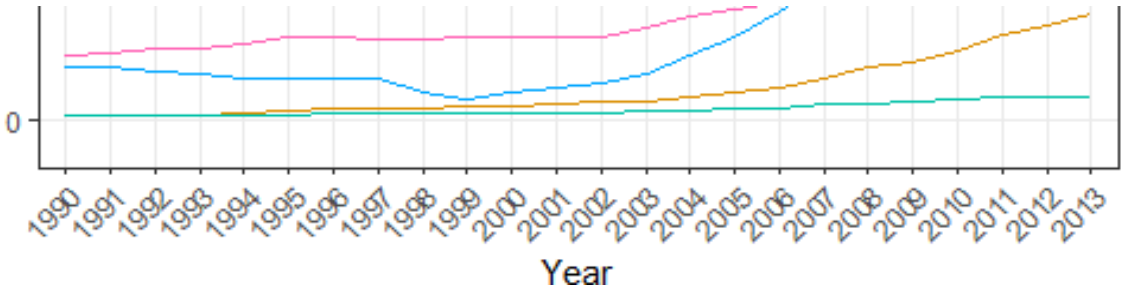
- ***Modify the appearance of the x-axis labels***

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  #modify the appearance of the x-axis labels
  #rotates the labels by 45 degrees
  theme(axis.text.x=element_text(angle = 45,
                                 #adjusts the vertical justification of the labels
                                 vjust = 1,
                                 #adjusts the horizontal justification of the labels
                                 hjust = 1))
```

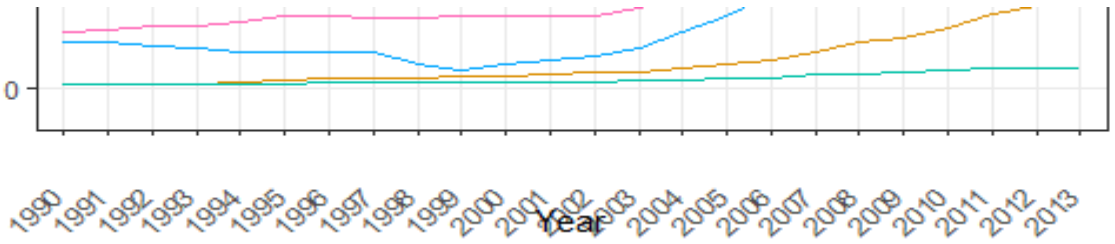


Vertical justification of the labels

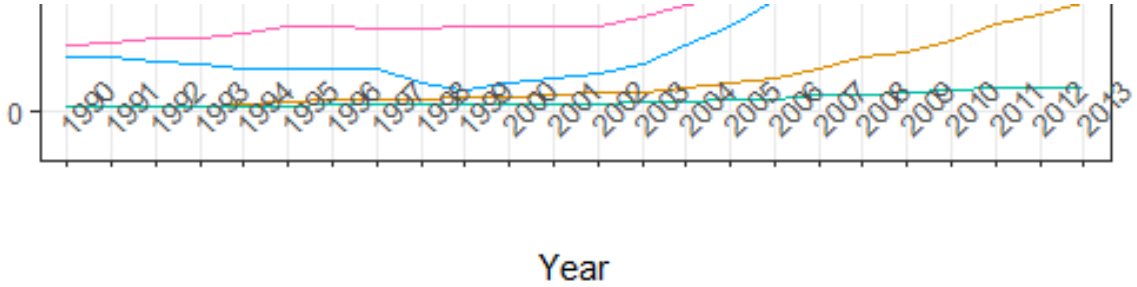
```
ent_text(angle = 45,
         vjust = 1,
         hjust = 1))
```




```
ent_text(angle = 45,
         vjust = 0,
         hjust = 1))
```

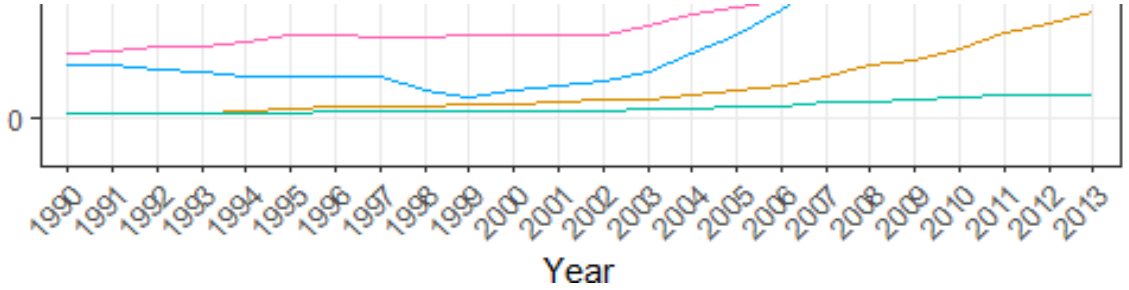



```
ment_text(angle = 45,
          vjust = 3,
          hjust = 1))
```

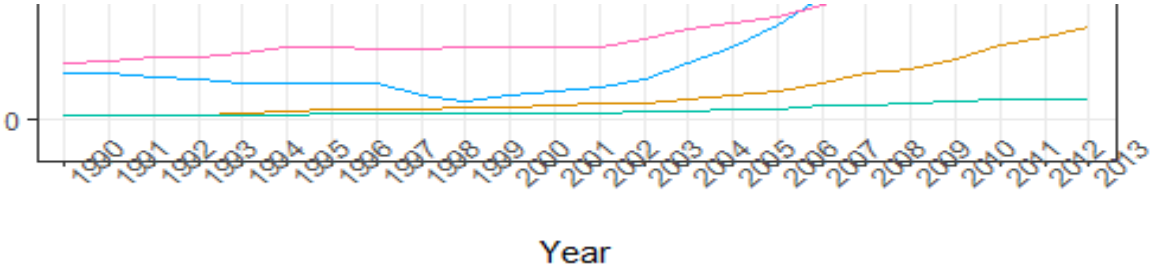


Horizontal justification of the labels

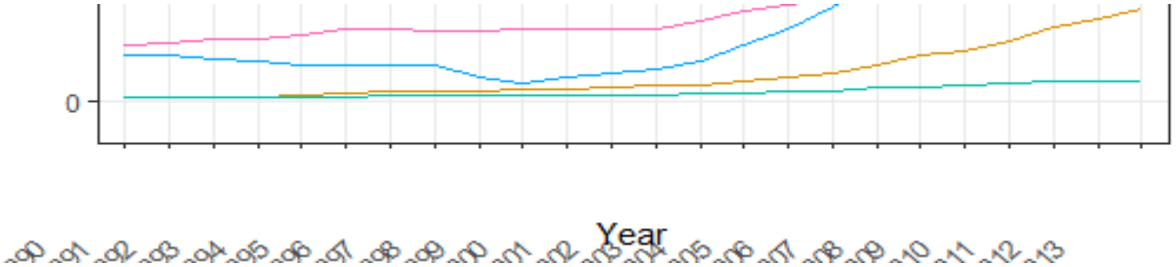
```
ent_text(angle = 45,
         vjust = 1,
         hjust = 1))
```




```
nt_text(angle = 45,
        vjust = 1,
        hjust = 0))
```



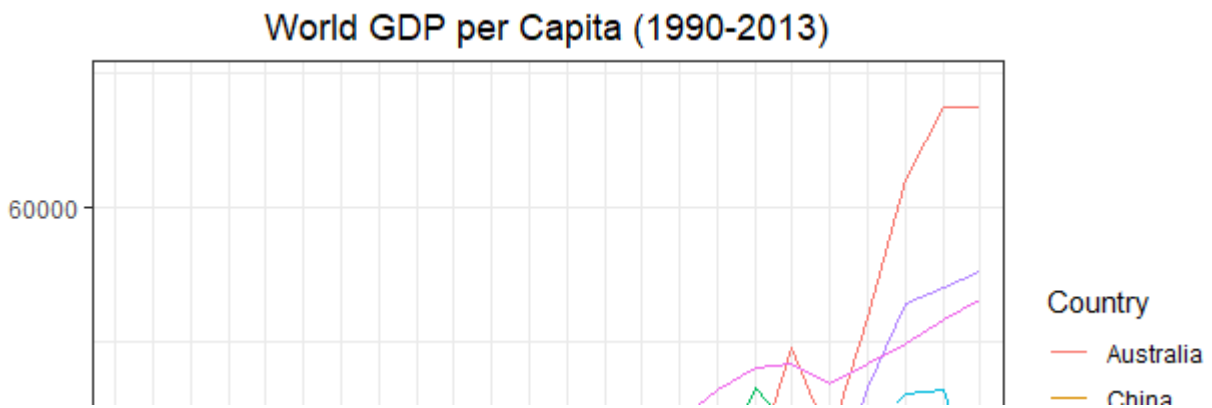

```
nt_text(angle = 45,
        vjust = 1,
        hjust = 3))
```



Note that the label moves horizontally with respect to the direction of the text, but no to the direction of general plot area

- ***Change the location of the title***

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  #change the location of the title
  theme(plot.title = element_text(hjust = 0.5))
```

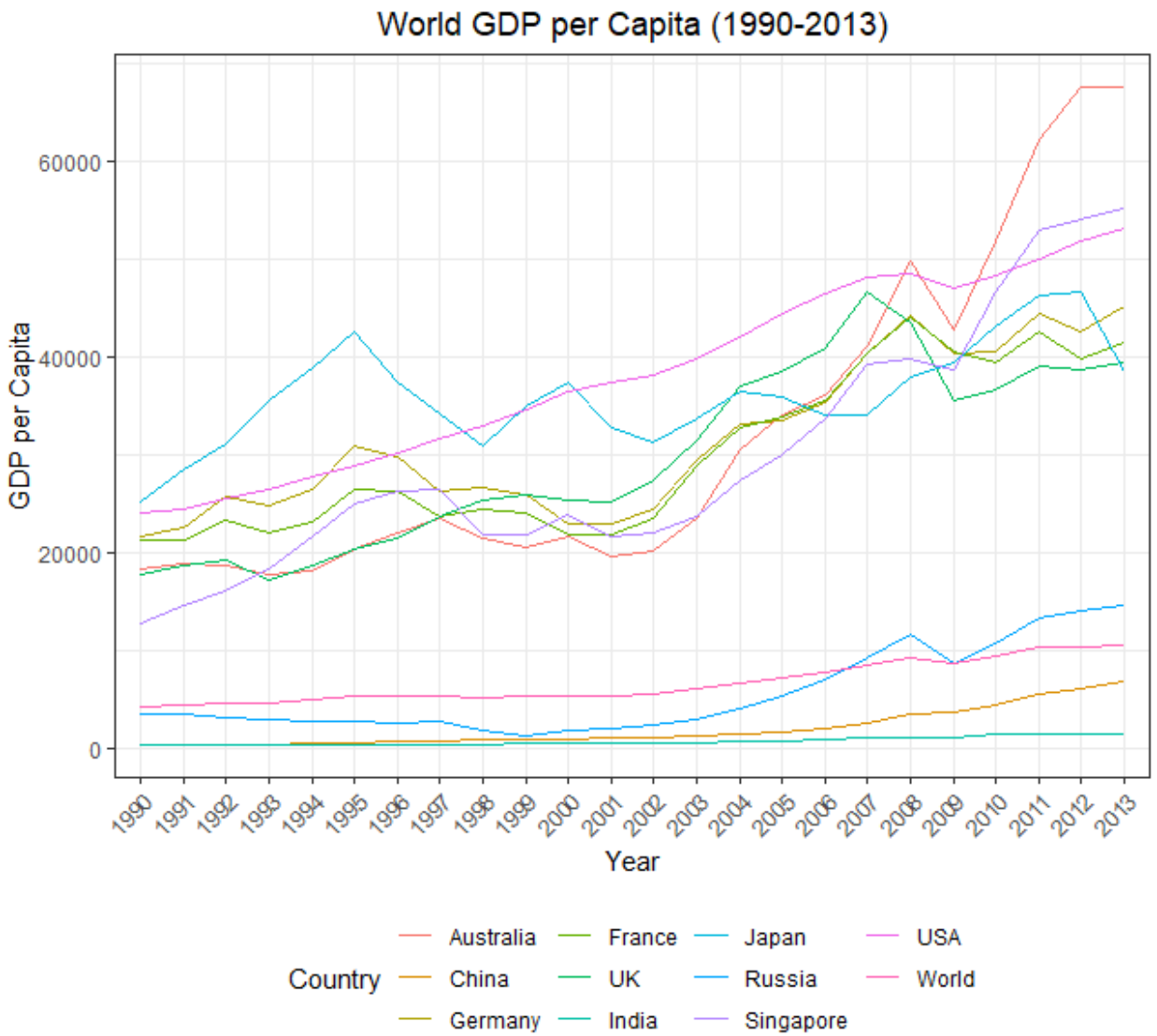


```
theme(plot.title = element_text(hjust = 0))
```


```
theme(plot.title = element_text(hjust = 1))
```

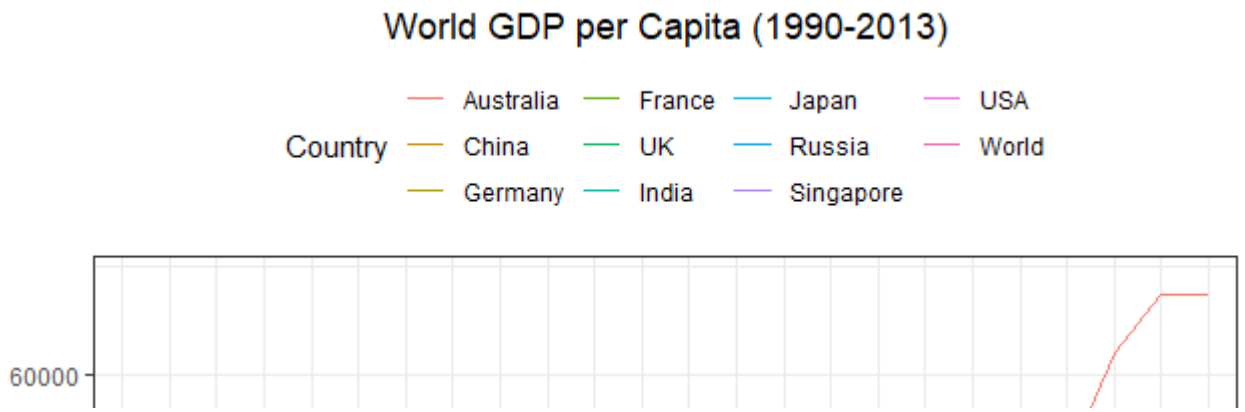
- ***Change the position of the legend***

```r
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  #change the location of the legend
  theme(legend.position = "bottom")
```

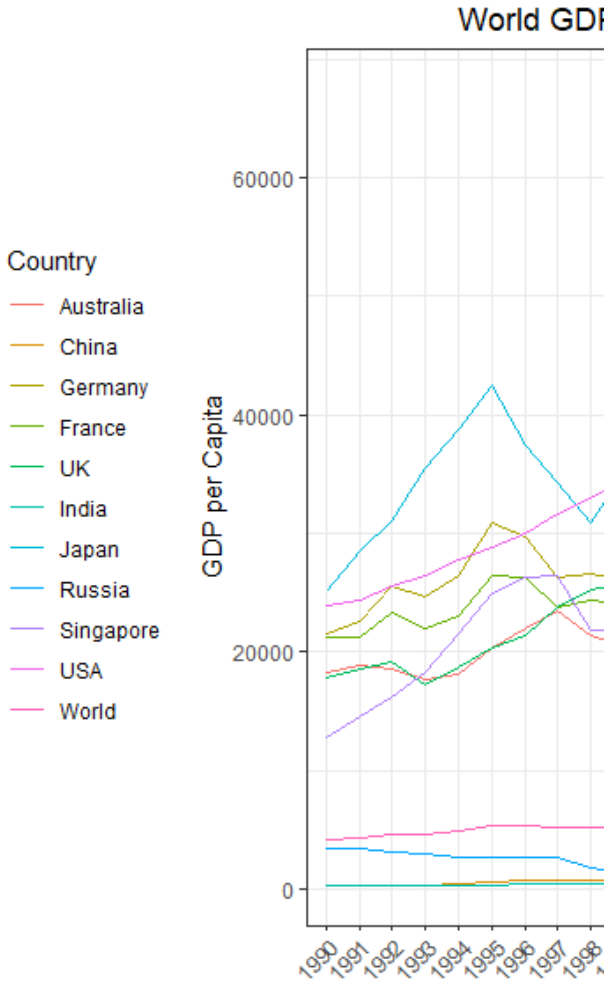


Other possible options

```
theme(legend.position = "top")
```




```
theme(legend.position = "left")
```



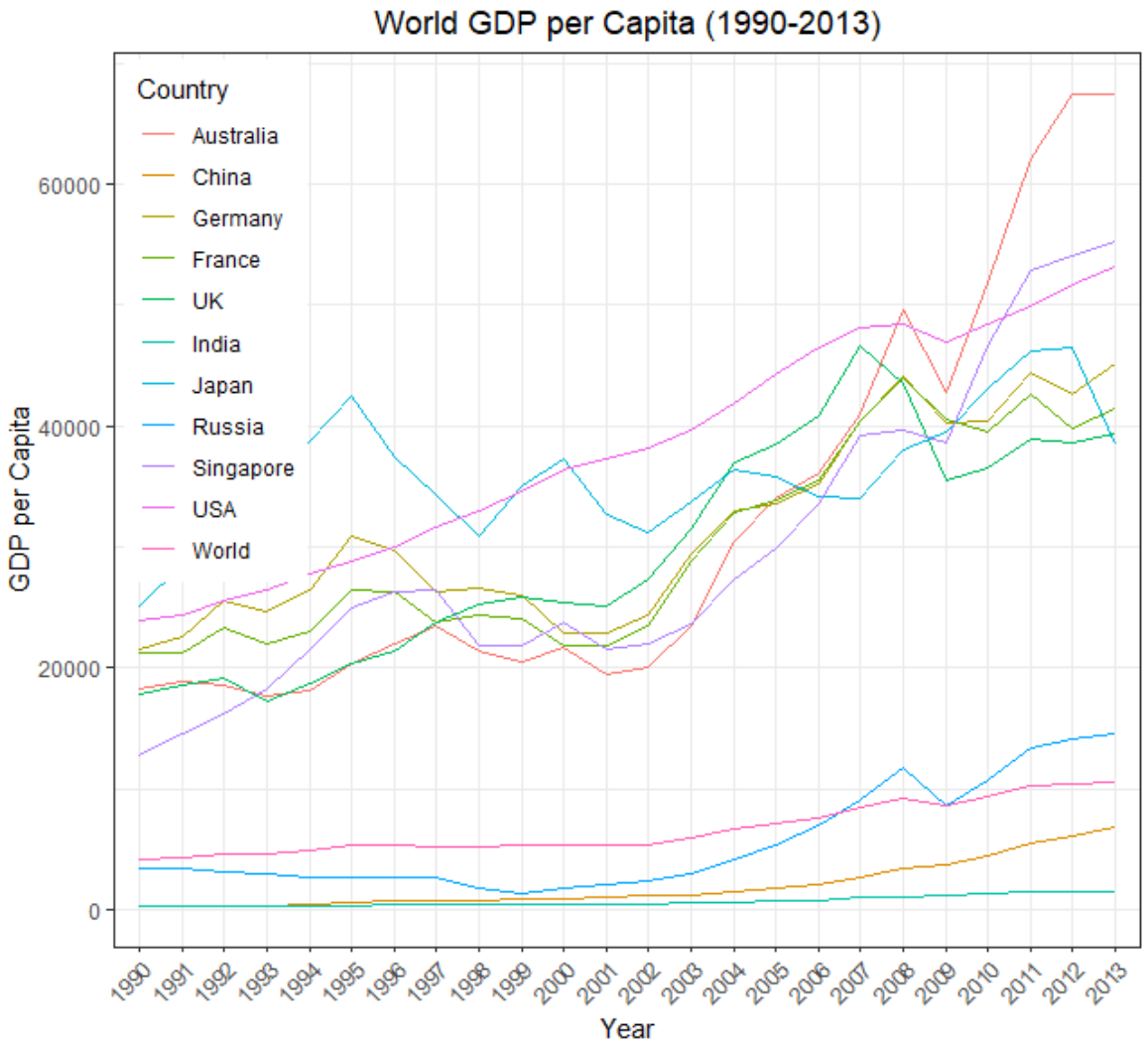

```
theme(legend.position = "none")
```

- ***Place the legend inside the box***

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  #change the location of the legend
  theme(legend.position = c(0.1, 0.7))
```

Horizontal position

Vertical position

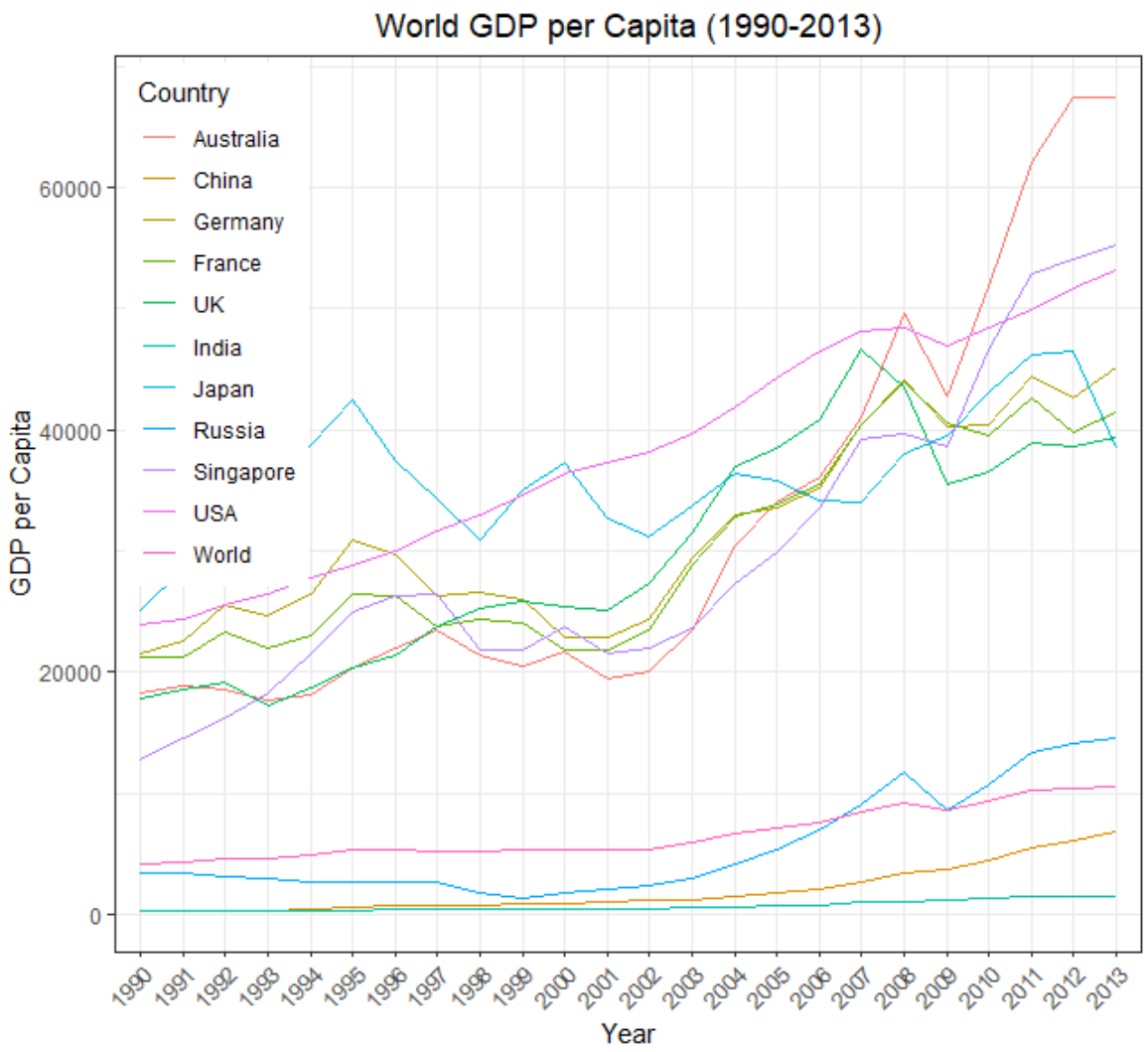


```
theme(legend.position = c(0.1, 0.7))
```

0 1

1

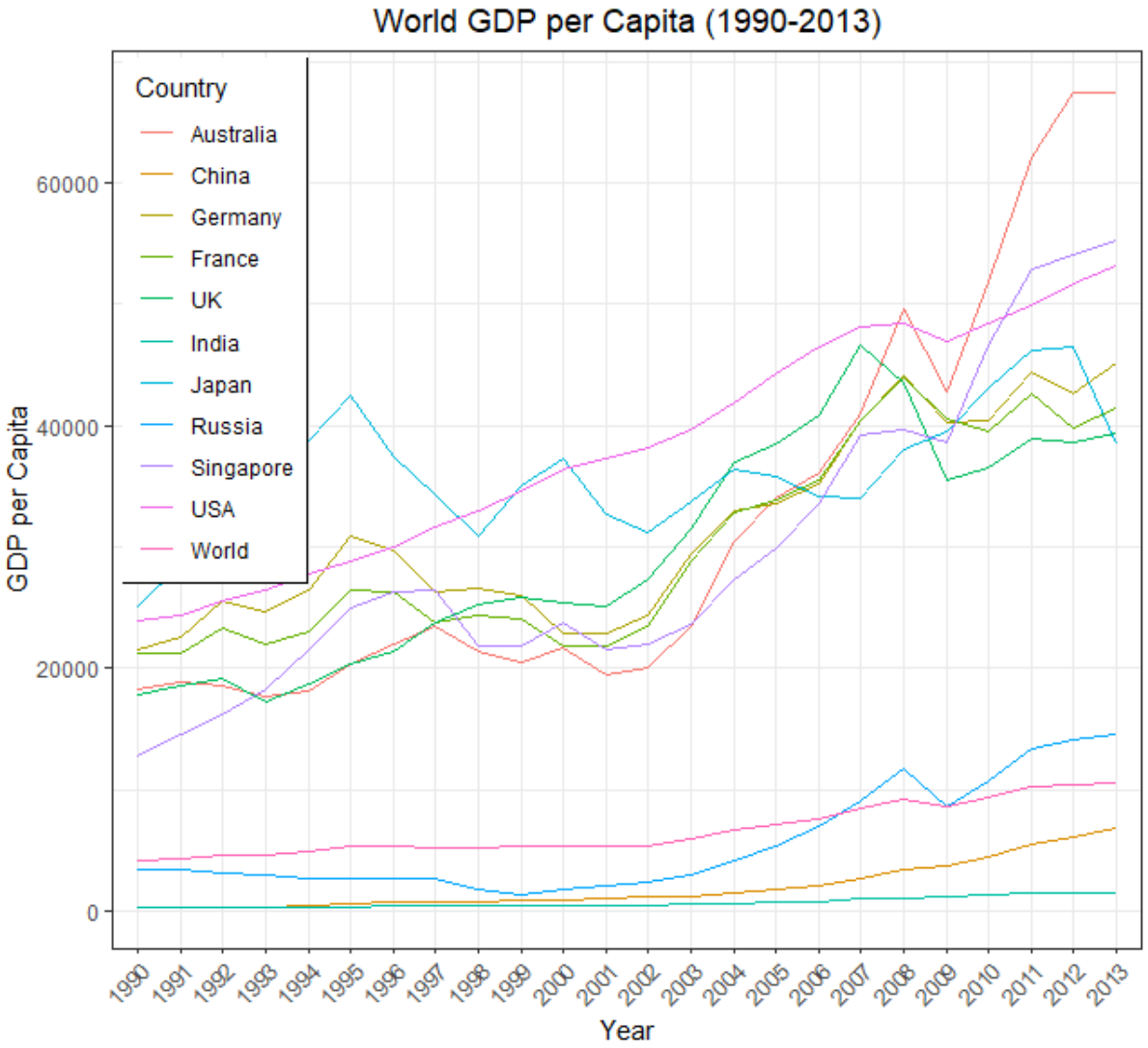
0




```
theme(legend.position = c(0.1, 0.7))
```

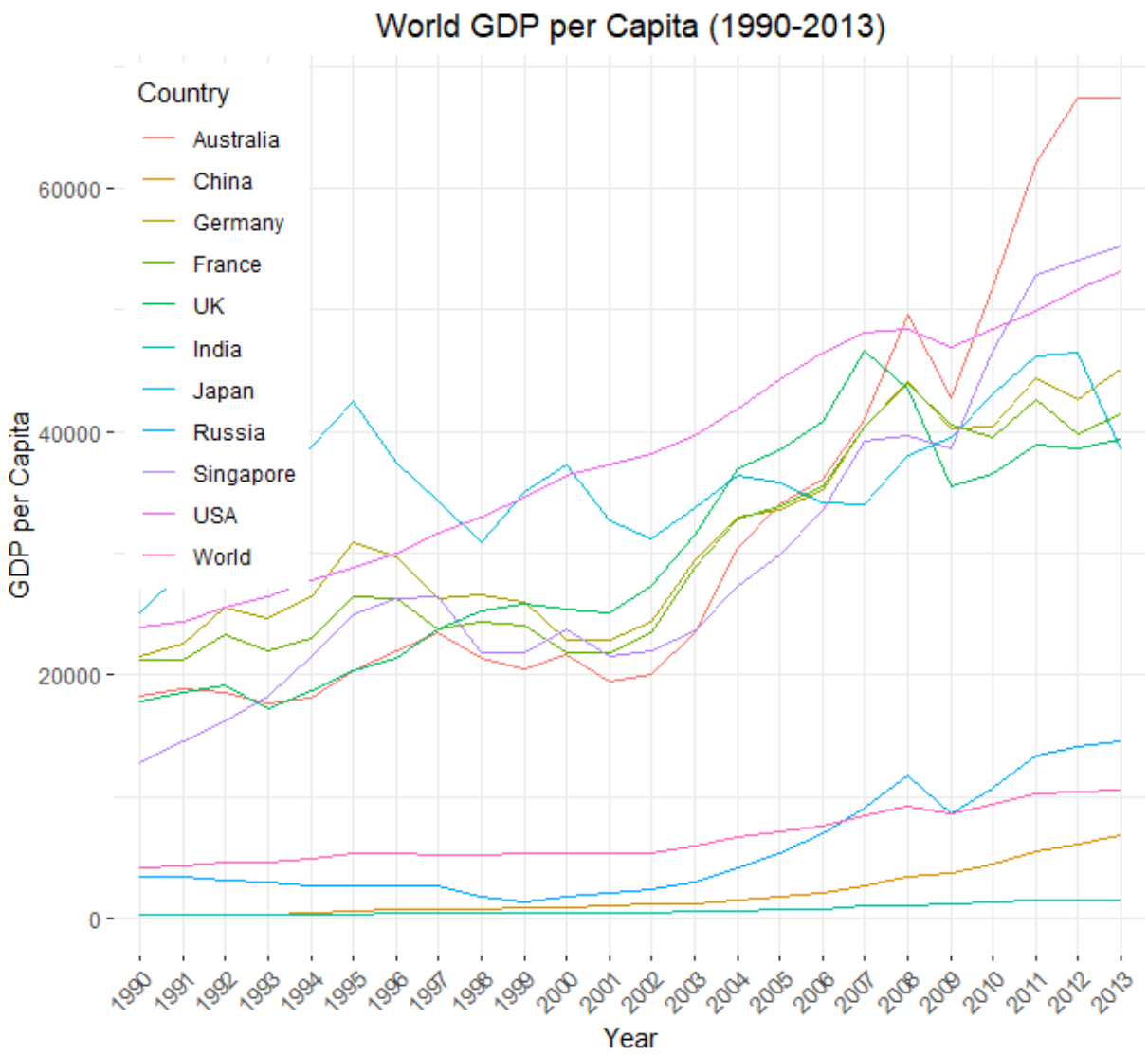
• *Modify the background of the legend box*

```r
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = c(0.1, 0.7),
        #modify the background of the legend box
        legend.box.background = element_rect())
```



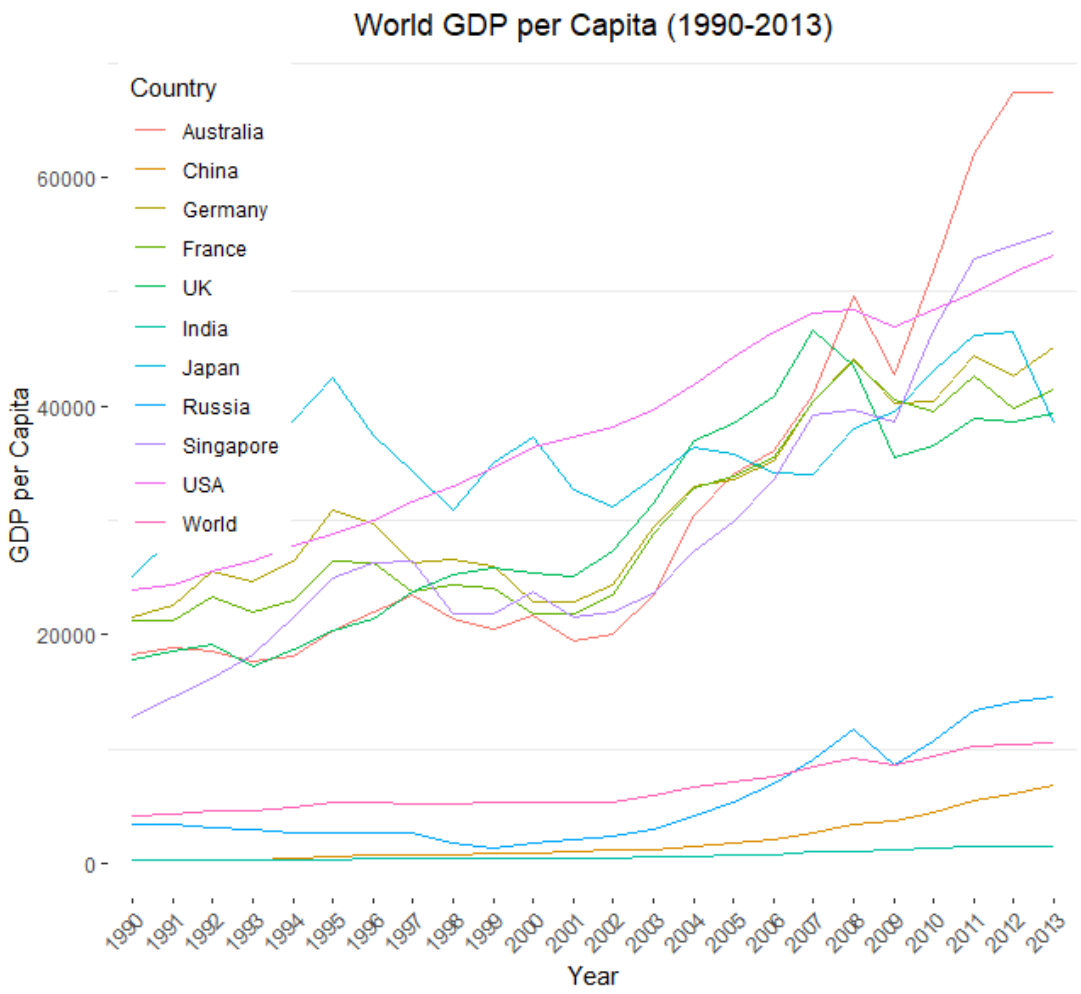
- ***Remove the borders of the plot***

```r
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = c(0.1, 0.7),
        #legend.box.background = element_rect(),
        #remove the borders of the plot
        panel.border = element_blank())
```



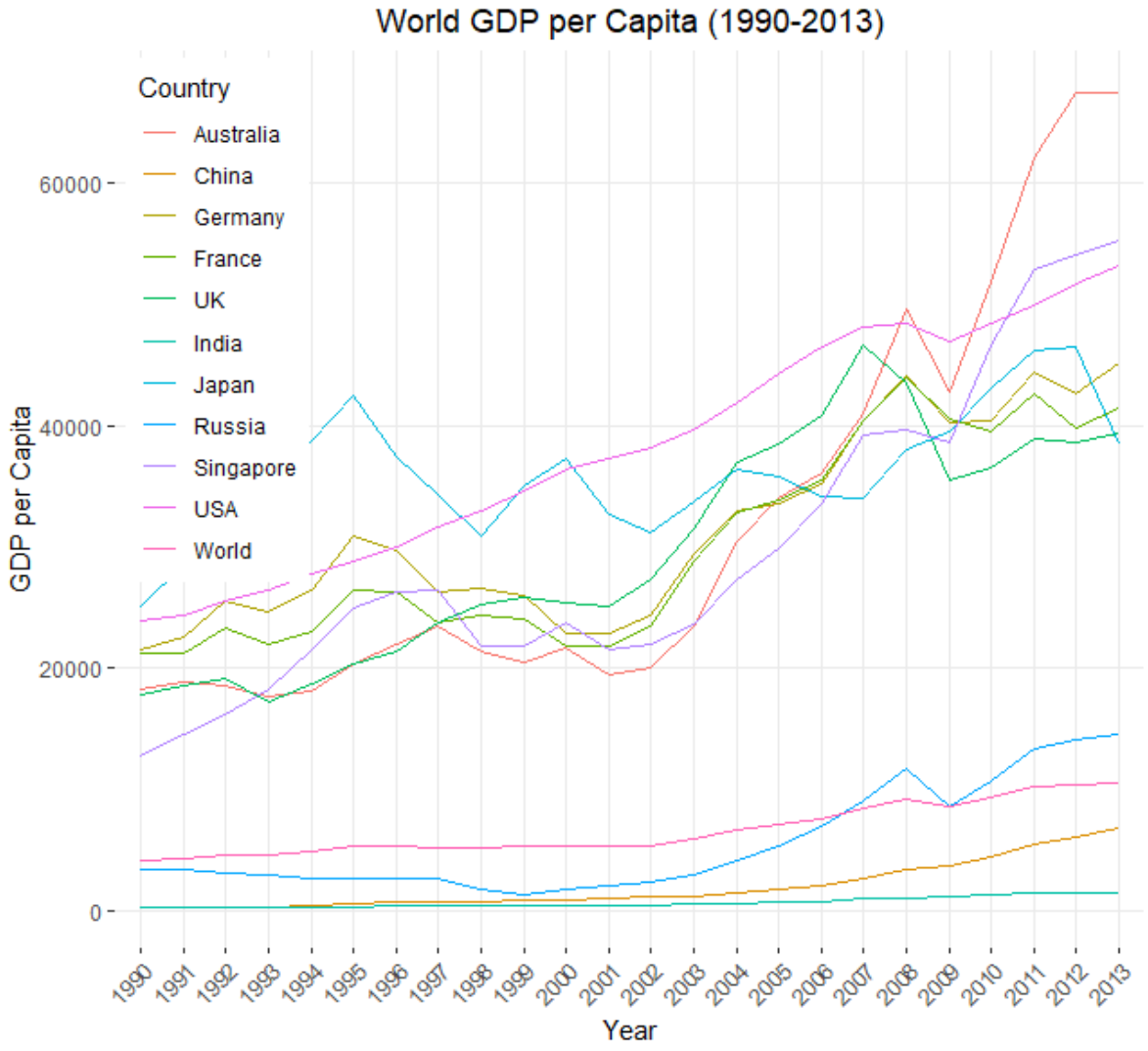
- ***Remove the <u>major</u> grid lines from the plot***

```r
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = c(0.1, 0.7),
        #legend.box.background = element_rect(),
        panel.border = element_blank(),
        #remove the major grid lines from the plot panel
        panel.grid.major = element_blank())
```



- ***Remove the minor grid lines from the plot***

```
ggplot(GDP_l,
       aes(Year, GDP,
           colour = Country,
           group = Country))+
  geom_line()+
  labs(title = "World GDP per Capita (1990-2013)",
       x = "Year", y = "GDP per Capita")+
  theme_bw()+
  theme(axis.text.x=element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = c(0.1, 0.7),
        #legend.box.background = element_rect(),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        #remove the minor grid lines from the plot panel
        panel.grid.minor = element_blank())
```



5. In-class Assignment

- The data set you'll be working with contains daily stock prices of four major European stock indices: DAX (Germany), SMI (Switzerland), CAC (France), and FTSE (UK) from January 1, 1991 to December 31, 1998.
- *stocks* file contains data in a wide format
- *stocks_l* file contains the same data but in a long format

```
> head(stocks)
# A tibble: 6 × 5
  Date         DAX   SMI   CAC  FTSE
  <date>     <dbl> <dbl> <dbl> <dbl>
1 1991-07-01 1629. 1678. 1773. 2444.
2 1991-07-02 1614. 1688. 1750. 2460.
3 1991-07-03 1607. 1679. 1718  2448.
4 1991-07-05 1621. 1684. 1708. 2470.
5 1991-07-06 1618. 1687. 1723. 2485.
6 1991-07-08 1611. 1672. 1714. 2467.
```

```
> head(stocks_l)
# A tibble: 6 × 3
  Date       Stock Price
  <date>     <chr> <dbl>
1 1991-07-01 DAX   1629.
2 1991-07-02 DAX   1614.
3 1991-07-03 DAX   1607.
4 1991-07-05 DAX   1621.
5 1991-07-06 DAX   1618.
6 1991-07-08 DAX   1611.
```