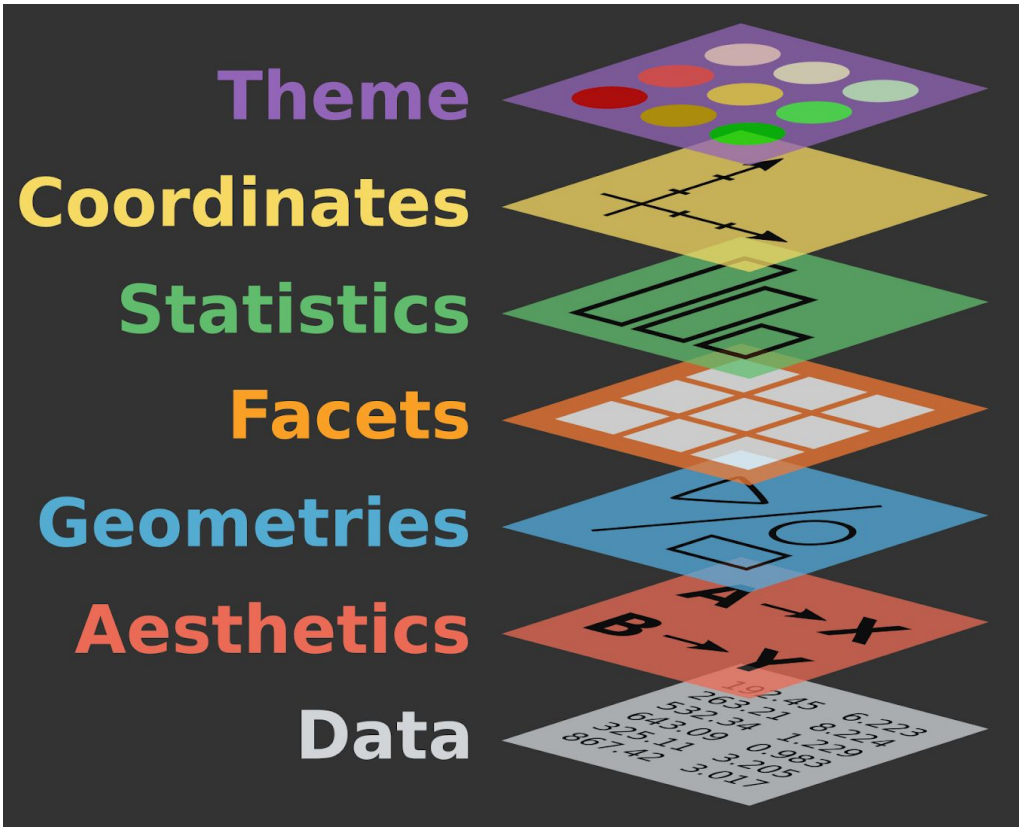
Geometric and graph grammar



The ggplot2 package is based on graph grammar (Wilkinson, 2005) and this has several advantages over other graphs as a very good aesthetic or simplification in complex graphics commands. Although it also has some limitations since you cannot make three-dimensional or interactive graphics.

The basic idea: independently specify building blocks and combine them to create virtually any type of graphic display you want.

The building blocks of a graph include:

- Data.
- Aesthetics (Aesthetic mapping).
- Geometries.
- Facets.
- Statistics
- Coordinates.
- Theme.

Data.

The data we want to visualize and a set of aesthetic elements that describe how the data variables will be mapped to aesthetic attributes that we can perceive.
Data that is commonly handled when working with R.

- Data in text (or tabular) format
    - CSV: .csv (comma separated values or, in Spanish, data separated by commas)
    - other data in text format
- Formats of other programs (proprietary software)
    - EXCEL: .xls and .xlsx
    - SPSS: sav and .for
    - STATA: .dta
    - SAS: .sas
- own formats R
    - R objects: .RData or .rda
    - RSerialized objects: .rds
- Other Formats
    - JSON
    - XML

```
movies <- read.csv ("Project-Data.csv")
```

Aesthetics (aes).

The main feature of the ggplot2 structure is the way graphics are plotted by adding “layers”. Let's see the process step by step.

The first thing you should do is tell the ggplot function which dataset (or dataset) to use. This is done by writing ggplot (df), where df is a dataframe that contains all the necessary features to make the plot. Unlike base graphics, ggplot does not take vectors as arguments. This is the first step.

The aes () argument is synonymous with aesthetics, ggplot2 considers the graph's X and Y axis to be aesthetic, along with color, size, shape, fill, etc. Any aesthetic you want can be added within the aes () argument, such as indicating the X and Y axes, specifying the respective variables of the data set. The variable based on which the color, size, shape and stroke should change can also be specified right here. Please note that the aesthetics specified here will be inherited by all subsequent added geom layers unless otherwise noted on each layer.

```
gg <- ggplot (movies, aes (x = Genre, y = GrossPor))
```

Geometries (geom).

The ggplot2 layers are also called geom. Once the base configuration is done, you can add the geoms on top of each other.

Every graphic has at least one geometry. Geometry determines the type of graph:

- geom_point (for points)
- geom_lines (for lines)
- geom_histogram (for histogram)

- geom_boxplot (for boxplot)
- geom_bar (for bars)
- geom_smooth (smooth lines)
- geom_polygons (for polygons on a map)
- etc. (if you run the command help.search (“geom_”, package = “ggplot2”) you can see the list of geometric objects)

```
gg + geom_jitter (aes (color = Studio, size = Budget))
```

```
gg + geom_jitter (aes (color = Studio, size = Budget)) + geom_boxplot
(aes (color = Budget, alpha =1.5))
```

Facets.

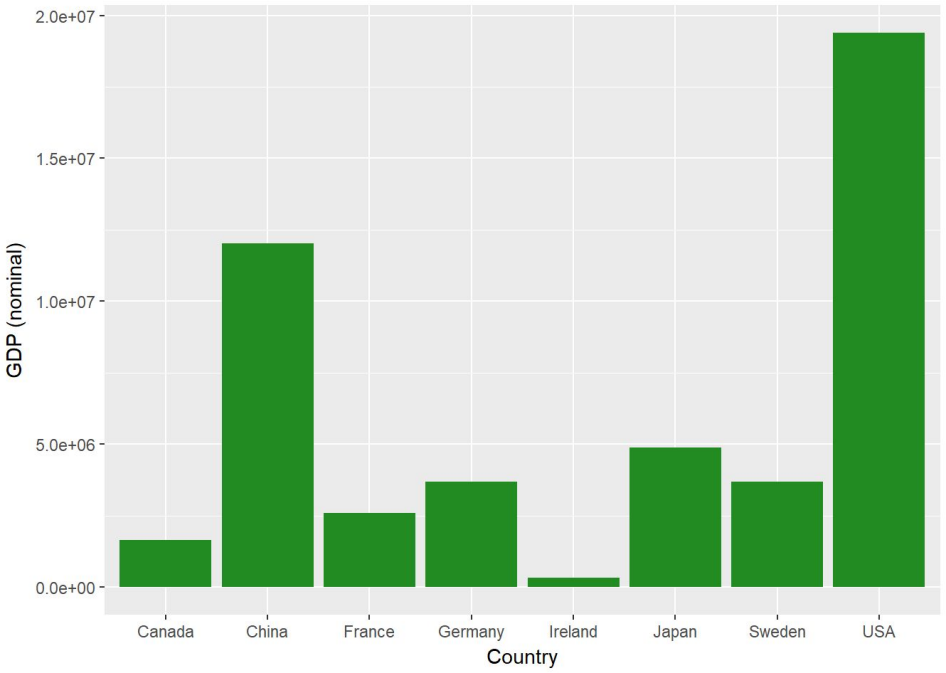
When you call ggplot it provides a data source, usually a data frame, then asks ggplot to assign different variables in our data source to a different aesthetic, such as the position of the x or y axes the color of our points or bars. With facets, you get an additional way to map variables. To demonstrate this, you will use the following data set, which includes a series of economic indicators for a selection of countries. Most of these are variants of the GDP, the Gross Domestic Product of each country.

```
##   Country  GDP_nom  GDP_PPP GDP_nom_per_capita GDP_PPP_per_capita
## 1     USA 19390600 19390600              59501              59495
## 2  Canada  1652412  1769270              45077              48141
## 3   China 12014610 23159107               8643              16807
## 4   Japan  4872135  5428813              38440              42659
## 5  France  2583560  2835746              39869              43550
## 6 Germany  3684816  4170790              44550              50206
## 7  Sweden  3684816   520937              53218              51264
## 8 Ireland   333994   343682              70638              72632
##   GNI_per_capita        Region
## 1          58270 North America
## 2          42870 North America
## 3           8690          Asia
## 4          38550          Asia
## 5          37970        Europe
## 6          43490        Europe
## 7          52590        Europe
## 8          55290        Europe
```

To start, let's make a simple bar chart of each country's nominal GDP.

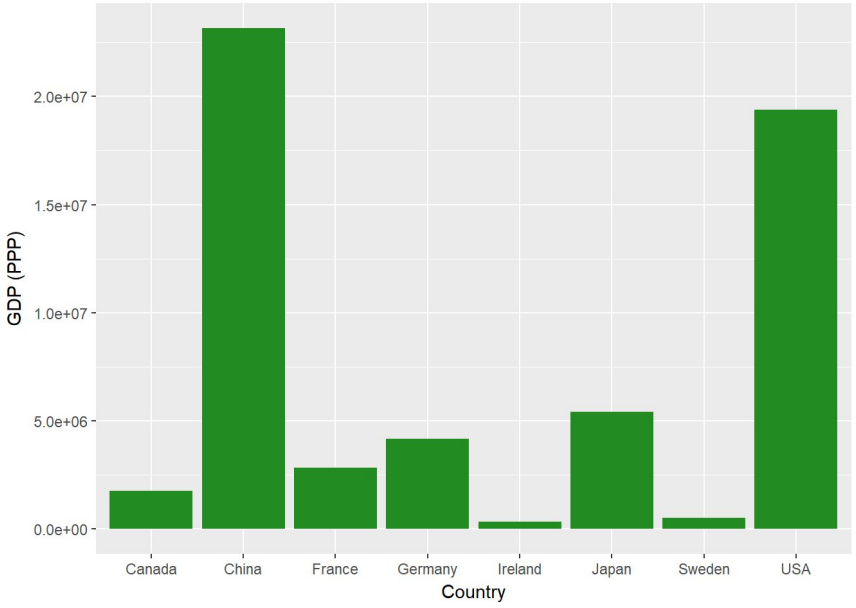
```
ggplot (econdata, aes (x = Country, y = GDP_nom)) +
```

```
  geom_bar (stat ='identity', fill ="forest green") +
  ylab ("GDP (nominal)")
```



You can also plot another variable, the adjusted GDP by PPP.

```
ggplot (econdata, aes (x = Country, y = GDP_PPP)) +
  geom_bar (stat ='identity', fill ="forest green") +
  ylab ("GDP (PPP)")
```



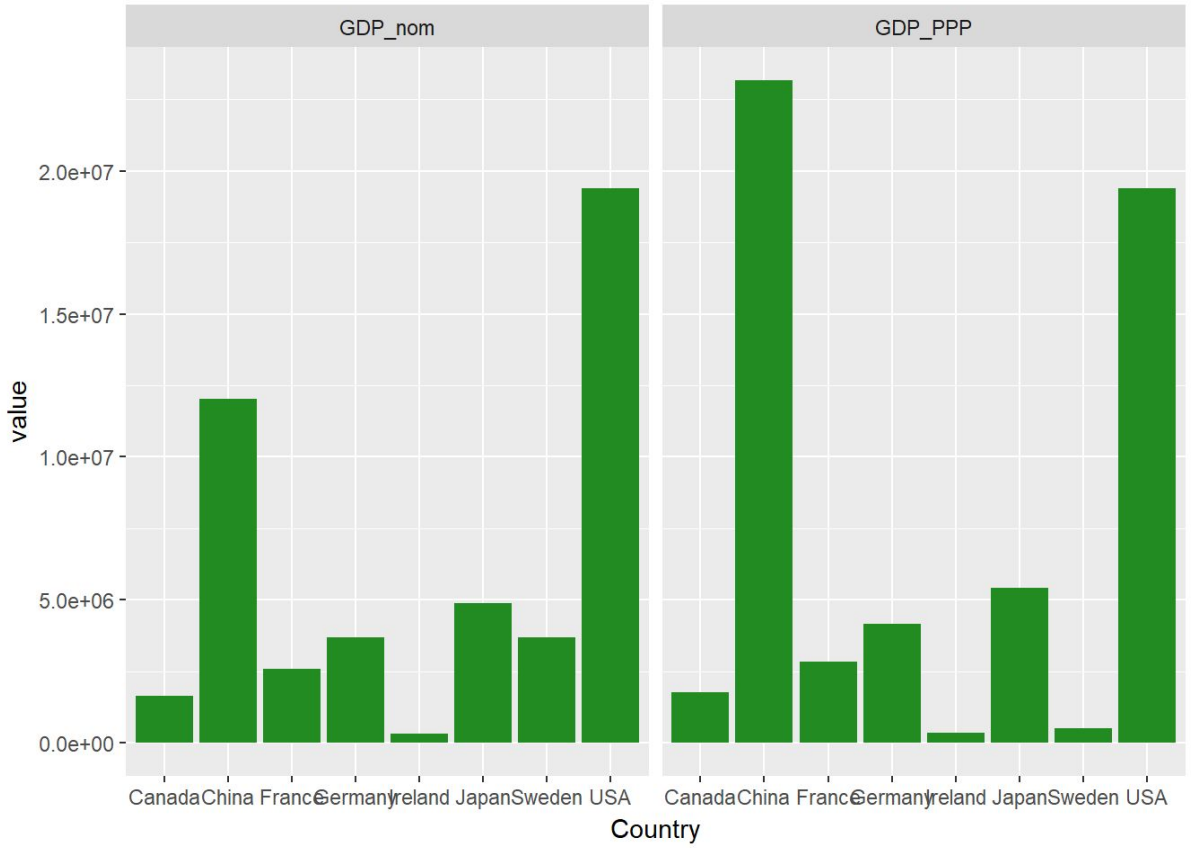
This gives you a separate second graph, similar to the last one but using a different variable. Let's say you want to plot both GDP (nominal) and GDP (PPP) together. You will use facets

to do it. First, you will need to reformat your data, changing it from a "wide" format with each variable in its own column to a "long" format, where you use one column for your measurements and another for a key variable that tells us what measure we use in each row.

```
econdatalong <- gather (econdata, key ="measure", value ="value", c ("GDP_nom",
"GDP_PPP"))
```

Once you have the data in that format, you can use our key variable to trace with facets. Let's build a simple graph, showing both nominal GDP (from our first graph) and GDP (PPP) (from our second graph). To do so, simply modify your code to add it + facet_wrap () and specify that ~ measure, our key variable, should be used for faceting.

```
ggplot (econdatalong, aes (x = Country, y = value)) +
  geom_bar (stat ='identity', fill ="forest green") +
  facet_wrap (~ measure)
```

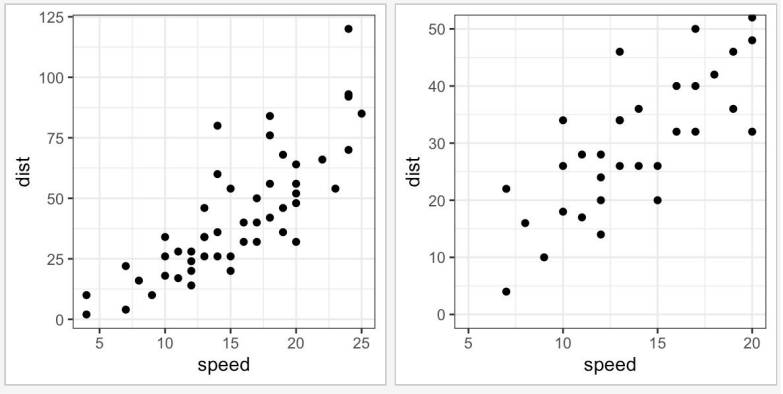


Coordinates.

Change axis limits

Use coord_cartesian

```
p + coord_cartesian (xlim = c (5, 20), ylim = c (0, 50))
```



Use xlim and ylim

- p + xlim (min, max): change the x axis limits
- p + ylim (min, max): change the axis limits and

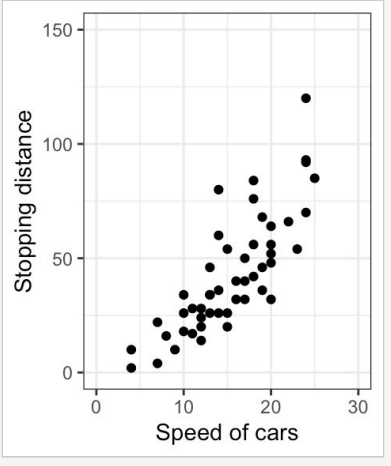
Any value outside the limits will be replaced by NA and discarded.

```
p + xlim (5, 20) + ylim (0, 50)
```

Use scale_x_continuous and scale_y_continuous

Can be used to change axis scales and labels at the same time, respectively:

```
p + scale_x_continuous (name = "Speed of cars", limits = c (0, 30)) +
  scale_y_continuous (name = "Stopping distance", limits = c (0, 150))
```



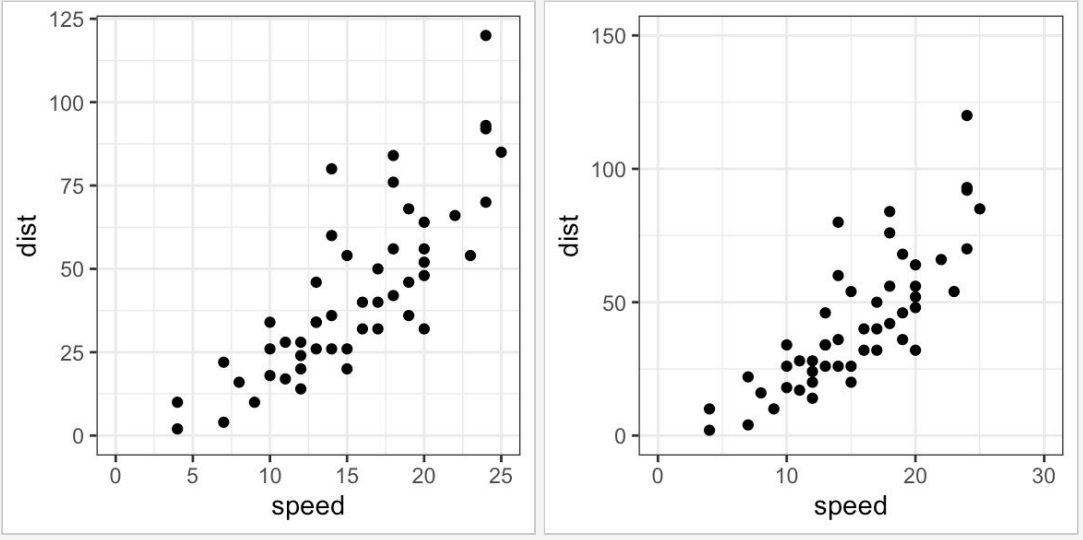
Expands the limits of the frame

Key function expand_limits (). It can be used to:

- quickly set the intersection of the x and y axes to (0,0)
- expand the limits of the x and y axes

```
p + expand_limits (x = 0, y = 0)
```

```
p + expand_limits (x = c (0,30) , y = c (0, 150))
```



Theme.

The only thing left to specify is the size of the labels and the title of the legend. Adjusting the size of the labels can be done using the theme () function by configuring plot.title, axis.text.x and axis.text.y, which must be specified inside the element_text (). Adjusting the title of the legend is a bit tricky. If the legend is that of a color attribute and varies based on a factor, you must set the name using scale_color_discrete (), where the color part belongs to the color attribute and is discrete because the legend is based on a factor variable.

```
gg1 <- gg + theme (plot.title = element_text (size =30, face ="bold"),
axis.text.x = element_text (size =15), axis.text.y = element_text (size
=15), axis.title.x = element_text (size =20), axis.title.y =
element_text (size =20)) + scale_color_discrete (name ="Diamond Cut")
```

Diagrama de puntos