Data Visualization using ggplot2

Srikar Vinayak Mulgund

2022-07-23

Matriculation Number: 400308183 (mulgund.srikar@stud.hs-fresenius.de) Sose22, Data Science for business, Prof. Huber, MIEIM 2

Word count: 3049

Abstract

One of the best ways to learn R programming is through data visualization, as the elegant and informative plots can help users quickly analyze and understand the data frame. Compile the data visualization process in a swirl package, and users have a very interactive and practical experience learning R programming. Thanks to the swirl package that converts the R console into an interactive learning environment. Therefore, we have developed a swirl course that teaches users data visualization through the ggplot2 package. The course is divided into three modules, each dealing with different data visualization methods: scatter plots, facets and geometric objects, and bar graphs. Each module assists the users step-by-step in understanding each aspect of the data visualization process at their own pace. Hence, at the end of these modules, users can write appropriate code, understand the process, implement it, and comprehend how data visualization can make strategic decision-making easy and fast. Basic knowledge of R and Rstudio is expected to understand this course thoroughly.

Introduction to data Visualization

Why visualize data?

Studies of research methods in graduate school are more focused on essential skills such as probability and statistical theory, regression, analysis, maximum likelihood estimation (MLE), etc. While these are fundamental approaches for analyzing data and evaluating research problems, drawing an image (also known as visualization) can occasionally be a more accurate method of statistical computing.(soltoff, 2022)

Visualization helps to perceive and appreciate some broad features of data. It enables one to look beyond those general features and explore what else the data has to offer. Often, statistical calculations are carried out based on a set of assumptions about the behavior of the data. And there are chances that those assumptions might be wrong; thus, the calculations become misleading. Therefore, this process is time-consuming and sometimes unfruitful. In such a scenario, visualization can help to check where exactly the assumptions went wrong and what can be done to pivot back to the right track. (Anscombe 1973)

A typical data analysis process is as follows:

Import: The first step in data analysis is to get the data right. Data is generally stored in flat files(e.g., spreadsheets). However, data can also be present in a database, web API, or books. Therefore, extracting and converting data into the correct format is essential to make it usable.

Tidy: Tidying the data means making it standardized to use the standard library of functions to analyze it. Once the data is tidy, each column becomes a variable and each row an observation. Furthermore, tidying data in such a format makes it easy to concentrate on the questions about the data rather than the format of the data. Due to these reasons, tidying is considered a crucial step in data analysis.

Transform: Once the data is tidy, it's time to explore it. But to fully understand the data, sometimes it's necessary to create new data frames based on new variables that are functions of the existing variables. This process is called data transformation.

Visualize: After successfully transforming the data, it's important to understand its reliability. This is where the visualization process comes in handy. Data visualization validates the credibility of data. Based on the data type, there are numerous ways to visualize it appropriately.

Model: Like any other process, visualization also comes up with a set of limitations. As the data gets more and more complex due to the increasing number of variables, it becomes difficult to represent it on graphs. As a result, it is not helpful to make use of visualization henceforth. Therefore, it's necessary to develop

complex mathematical models to analyze such data frames. It is effortless to scale such models to handle many variables. However, it is important to know that all models are based on assumptions about the relationships between variables so that the model can generate irrelevant results with an inappropriate link.

Communicate: After the data analysis, it's time to communicate the results with the concerned audience. The results are significant as they can educate the community and even change their perception of the problem.

Swirl is a software package that teaches users R programming and data science by converting the R console into an interactive learning environment at their own pace. The swirl package allows users to load learning modules in Rstudio. The modules are designed in such a way to give the users immediate feedback as they go through the self-paced lessons.

Swirl is a powerful learning platform as it promotes learning by doing it operates from within the R it is a free and open source

The biggest advantage of swirl is its practical approach, and users learn a particular code by writing it into the console. Mistakes are rectified immediately with the help of pre-coded instructions or by the trial and error method. This way, the users are always guided in the right direction throughout the lesson. To run any swirl course, it's necessary to install the swirl package using the install.package ("swirl"). This is just one side of the swirl. Apart from learning, swirl can also be used to create interactive lessons for teaching R programming and data science. The swirlify R package provides a comprehensive toolbox for swirl instructors. Instructors can make use of this tool to create interactive learning modules swiftly and efficiently.

Swirl Course summary

Our swirl course will teach students how to visualize the data frame in rstudio using the ggplot2 package. The data frame is obtained from the World Bank using the WDI package. The data we have chosen is transformed from two different WDI data frames - forest area% and urban population. The data contains 5 variables and 217 observations. This transformed data frame is already available to download from a github link within the course. Our course visualizes the data frame in three different ways using different layers of different functions. Apart from just data visualization, this course also teaches students how to create aesthetically pleasing graphs. Naturally, to help students to learn the ggplot2 package and not get confused between different functions, the course is subdivided in three different modules, each created by a single

member in the group. The data frame used in each module is the same i.e the master data. Module 1 This is related to the scatter plots, which is important in the statistics department. Provides a visual explanation of how the data works. Each point in the graph represents one unit of the values of two variables. One variable is placed on the horizontal axis and the other on the vertical axis. Module 2 focuses on Facets and Geometric objects. This module will help you learn how to create facets, called grid graphs, and provide in-depth knowledge about geometric objects and their functions in creating data visualization projects. Module 3 deals with bar graph representation. This module teaches students to plot bar graphs based on the data frame, make scale changes depending upon different variables, and how to make a bar graph aesthetic. Students can skip any module and jump to the next depending upon their learning goal. Upon completion of our course students will successfully be able to understand the entire process of data visualization and learn the difference between different functions.

Swirl Module Summary

This swirl module aims to deliver to the learner with in-depth knowledge about two topics under data visualization with ggplot2, that are Facets and Geometric objects. Facets can be described as an alternate way to split your plots into sub-plots to enhance readability and create individual sub-plots each displaying a sub-set of your data. This chapter will also highlight the differences between using color aesthetics to that of using facets.

Whereas, Geometric objects defines the core of how different kinds of plots are categorized, with the function geom, that uses the plot to represent its data. Geom, a ggplot syntax of Geometrical object is often used to represent the type of plot, for example a bar chart uses a *geom bar* and a line plot uses a *geom line* function. The interesting part of this chapter will explain you how you can use geom to create plot the same data.

Found the chapter's interesting and want to learn how to use facets and geometric objects? Jump right into the next parts!

Base Knowledge

This course module contains an overview of the basic knowledge required about Data visualization using ggplot, and that should be enough for you to jump further into the course. But, if you are interested in establishing a better base before jumping into this module, I would recommend you to read through the following : **Link - Data visualization with ggplot**

How to access and install module

The contents of this module can be found on this Github repository. Please make sure you are well connected to the Internet to download all the necessary files. The following files can be found and are necessary to run the entire module: 1. lesson.yaml 2. initLesson.R 3. dependson.txt 4. customTests.R

Once the user has downloaded the .swc file from the repository, They can simply use the command

```
library(swirl)
swirl::install_course_directory(filepath)
```

The user has now successfully installed the course!Now, to begin the course, the user can simply run

```
swirl()
```

Running the above code will provide the user with the course menu.

NOTE: The course module "facets and Geometric objects" requires installed R packages before you could install the course. An R package is a group of additional functions, information, and documentation for the R programming language.

ggplot is one of the core members of the *tidyverse* universe and hence requires you to install and load and run the package. *tidyverse* hosts various packages like ggplot2, tibble, tidyr, readr, purrr

To Install the *tidyverse* package, simply enter the following code into your R script and run it.

```
install.packages("tidyverse")
```

Once your package is successfully installed, Do not forget to load the package by using the following code

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
```

```
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Data Set preperation

One of the most important tasks in creating this learning module was selecting the right kind of data which can be easily understood by the learners but also be diverse enough to explain the course modules in an efficient manner.

After assessing the data provided by world data bank, the team collectively came up with the following idea of merging two relevant data sets to form a master data, which consisted of more variables, and also could be assessed together.

To understand the process better, we can divide the steps as:

A) What : The first data set selected from the world bank data was Urban population growth in the world. Since the data set consisted of all the countries in the world, we wanted to narrow it down to one country from each part of the world, which resulted in the data of 7 countries.

The second data set chosen was the Forest land (in sq.km) of the same countries as the first data set. The reason behind choosing this data set was that we wanted to highlight the correlation between growing population and declining forest area as we see it.

How:

1. Instead of directly searching the data through R studio using WDI search function, we directly inserted the ID of the data set found in the 'details' section of the desired data set on This link

The following function was used to import both the data sets into the R studio environment:

```
install.packages("WDI")
```

```
## Installing package into '/home/proton-exe/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
install.packages("pacman")
```

```
## Installing package into '/home/proton-exe/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(WDI)
library(pacman)
```

```
urbanpopulation= WDI(indicator= "SP.URB.TOTL", country = c('AU','BR','CN','DE','IN','NG','US'),
                     start =1990, end = 2020 )
```

Once the code is run, we can see the data set in our environment. A extract of the data set is shown below in Table 1.

Table 1: Urban population

iso2c	country	SP.URB.TOTL	year
AU	Australia	22158130	2020
AU	Australia	21845994	2019
AU	Australia	21488110	2018
AU	Australia	21133982	2017

Similarly, the second data set needed can be obtained by changing the Indicator ID to the required data set.

```
forestarea= WDI(indicator= "AG.LND.FRST.K2", country = c('AU','BR','CN','DE','IN','NG','US'),
                     start =1990, end = 2020 )
```

Table 2: Forest Area

iso2c	country	AG.LND.FRST.K2	year
AU	Australia	1340051	2020
AU	Australia	1340051	2019
AU	Australia	1340051	2018

iso2c	country	AG.LND.FRST.K2	year
AU	Australia	1340174	2017

Subsequently, the usage of merger function by entering the following code will provide us with the master data required to go ahead with this course module.

```
master_data = merge(urbanpopulation, forestarea, by = c("iso2c","year", "country"))
```

Table 3: Master data

iso2c	year	country	SP.URB.TOTL	AG.LND.FRST.K2
AU	1990	Australia	14579227	1338822
AU	1991	Australia	14761055	1336754
AU	1992	Australia	14920611	1334686
AU	1993	Australia	15044687	1332618
AU	1994	Australia	15181749	1330550

You can now observe both the required floating variables are now merged together.

Changing column names

Column names can be changed according to desire, thus making the data frame more readable.This can be done using the *colnames* function.

```
colnames(dataframe)[which(names(dataframe) == "columnName")] <- "newColumnName"
```

After Entering the required data,, we can see the output with desired column names below:

Table 4: Master data

iso2c	year	country	urban_population	forest_area
AU	1990	Australia	14579227	1338822
AU	1991	Australia	14761055	1336754

iso2c	year	country	urban_population	forest_area
AU	1992	Australia	14920611	1334686
AU	1993	Australia	15044687	1332618
AU	1994	Australia	15181749	1330550

The table above depicts the final dataframe required to go ahead with the course. However, the final data frame will be provided to the learner and the data preparation steps are not in scope in the learning module.

Note: Before we begin working examples on Facets, Please ensure the required package of *tidyverse* which hosts *ggplot* is loaded. if not, you can load it again using the *library()* function.

#Foundation to Facets and Geometric objects

If you can recall on how to create a basic scatter plot, it will act as a foundation towards learning facets and geometric objects. Nonetheless, even if you dont, you can access section 3.2.2 from this link to learn more: link

Chapter 1 - Facets

Chapter 1 of this course module focuses on Facets. Let us do a comprehensive overview to understand what the course module cover about facets.

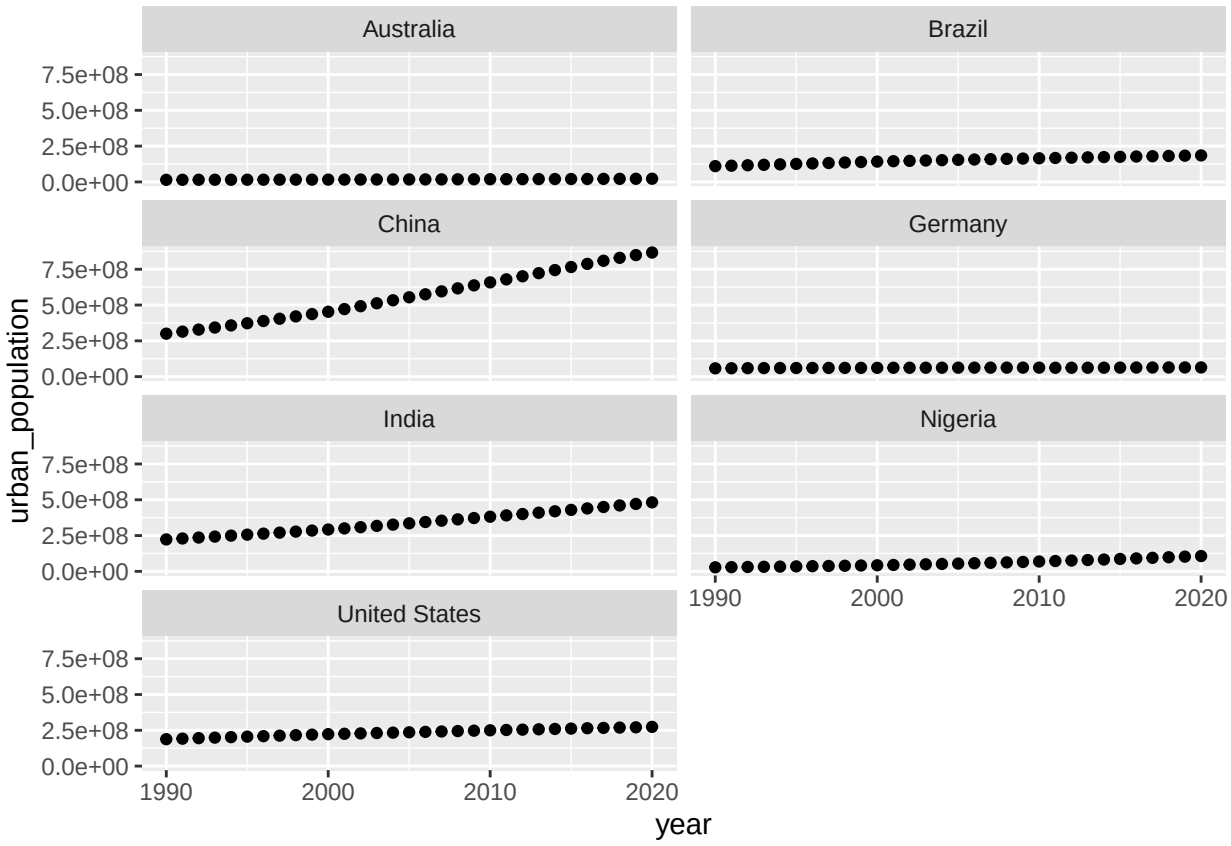
What are Facets?

Previously, you have learned about introducing additional variables into a plot by using Aesthetic mappings. Similarly, Facets is another way of adding additional variables in the plot. Importantly for categorical variables, using facets will split your data into sub-plots of individual sub-sets.

Syntax for facets can be divided into two types,

a. To facet your plot by individual variables, *facet_wrap()* argument is used. The first part is the formula towards creating your ggplot, followed by a '~' operator to fix the facet argument.
   For example, we can apply the following function on the master_data set we have obtained earlier.

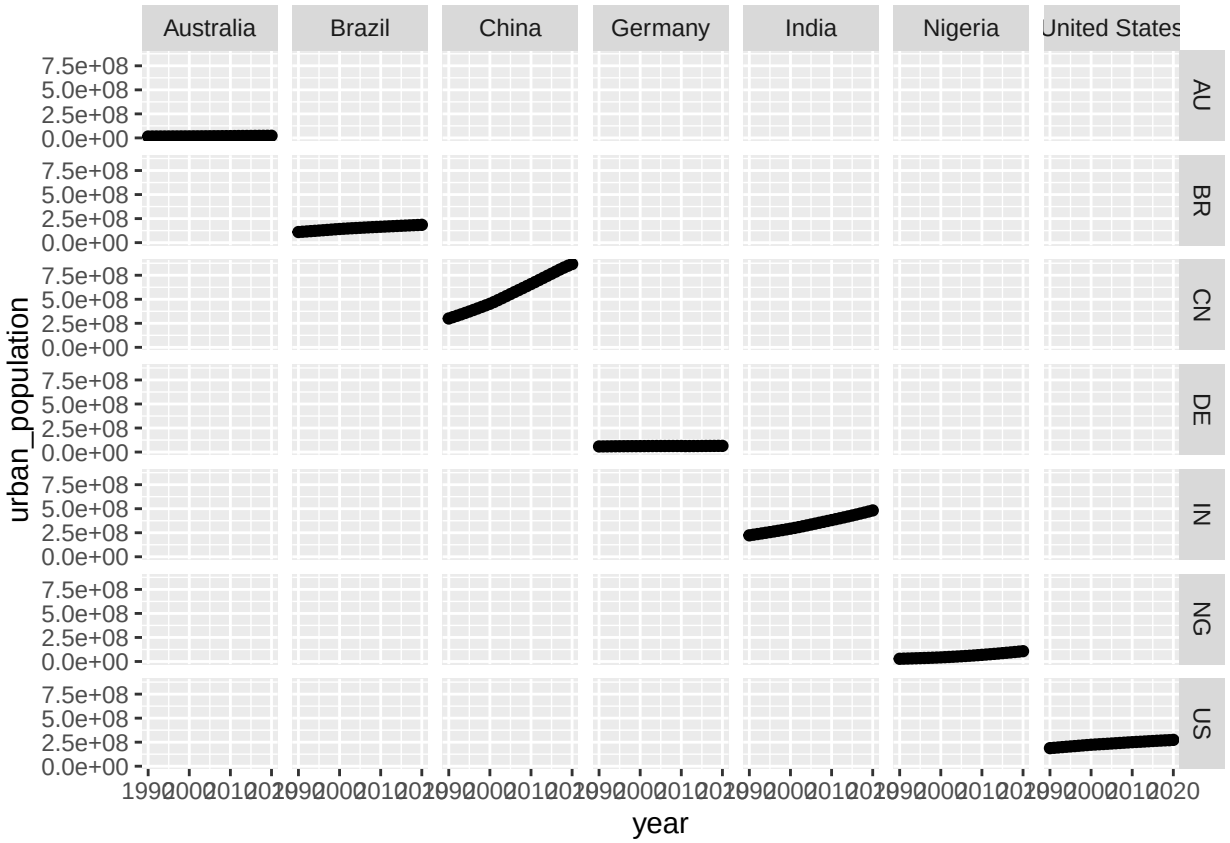
```
library(tidyverse)
ggplot(data = master_data) +
geom_point(mapping = aes(x = year, y = urban_population)) +
facet_wrap(~ country, nrow = 4)
```



The results obtained are now in the form of sub-sets for individual countries. Facets work really well when the data set is categorical, if not, the sub-plots become messy and cannot be analysed properly. In the case above, A observation was made that opting for color aesthetics is a better chose when you length of data is large.

b. To facet your plot by a combination of two variables, you can use facet_grid() instead of facet_wrap. The two variable names are sperate dby a ~ operator.

```
ggplot(data = master_data) +
geom_point(mapping = aes(x = year, y = urban_population)) +
facet_grid(iso2c ~ country)
```



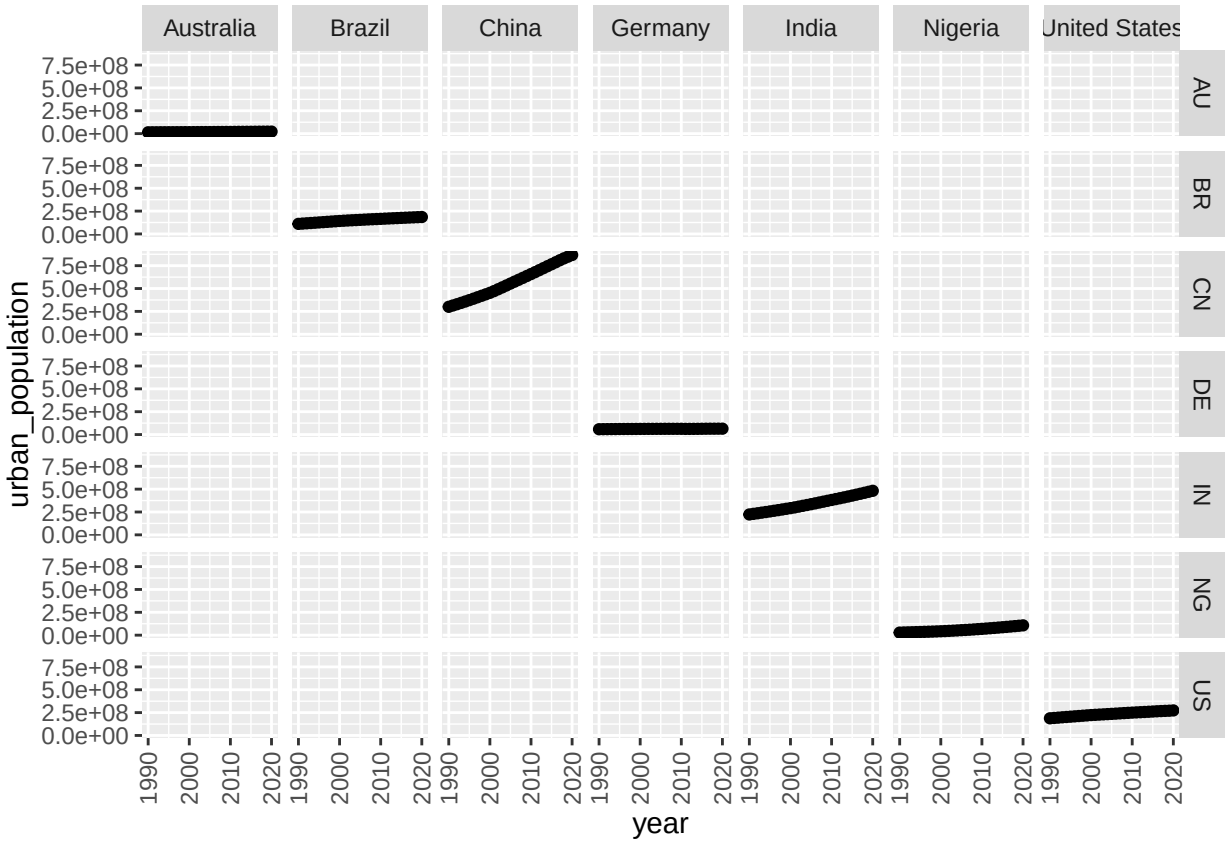
As you can observe, the values at X-axis are clustered together, this was tackled using the theme function.

```
ggplot(data = master_data) +

geom_point(mapping = aes(x = year, y = urban_population)) +

facet_grid(iso2c ~ country) +

theme(axis.text.x=element_text(angle = 90, vjust = 0.5))
```



That plot concluded the first chapter of Facets.

Chapter 2 - Geometric Objects

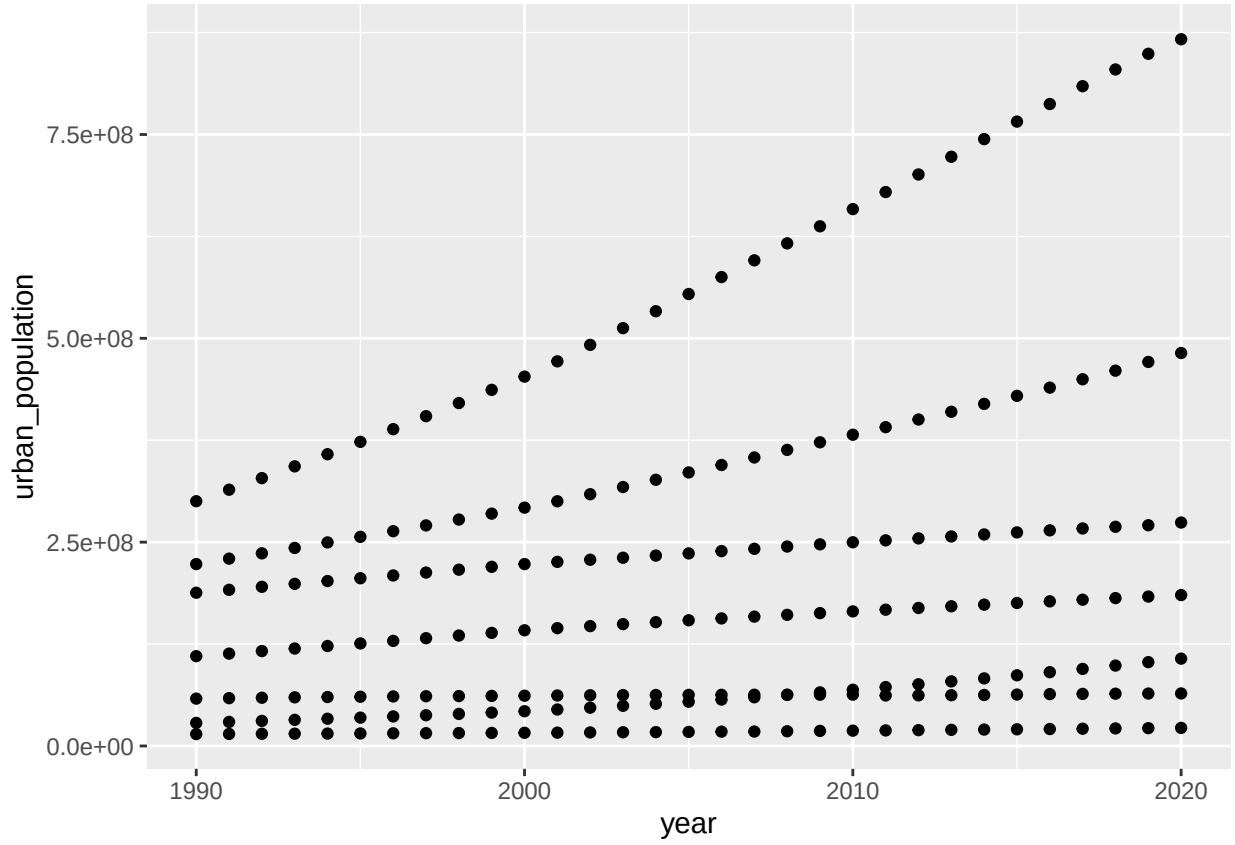
Welcome to chapter 2 of this course module. This chapter focuses on the importance of geometric points, or else known as geom in ggplot syntax. If you recall the summary of this course module, a geom is a geometrical object that a plot uses to represent data.

ggplot hosts around 30 geoms, with additional ones from external packages. The best way to get a comprehensive overview is the ggplot2 cheatsheet, which you can find at http://rstudio.com/cheatsheets. To learn more about any single geom, use help: ?geom_smooth .

A simple overview of what the chapter covers, can be established by creating two plots of the same data, using different geom functions.

Now, if we run a geom_point function for our data frame:

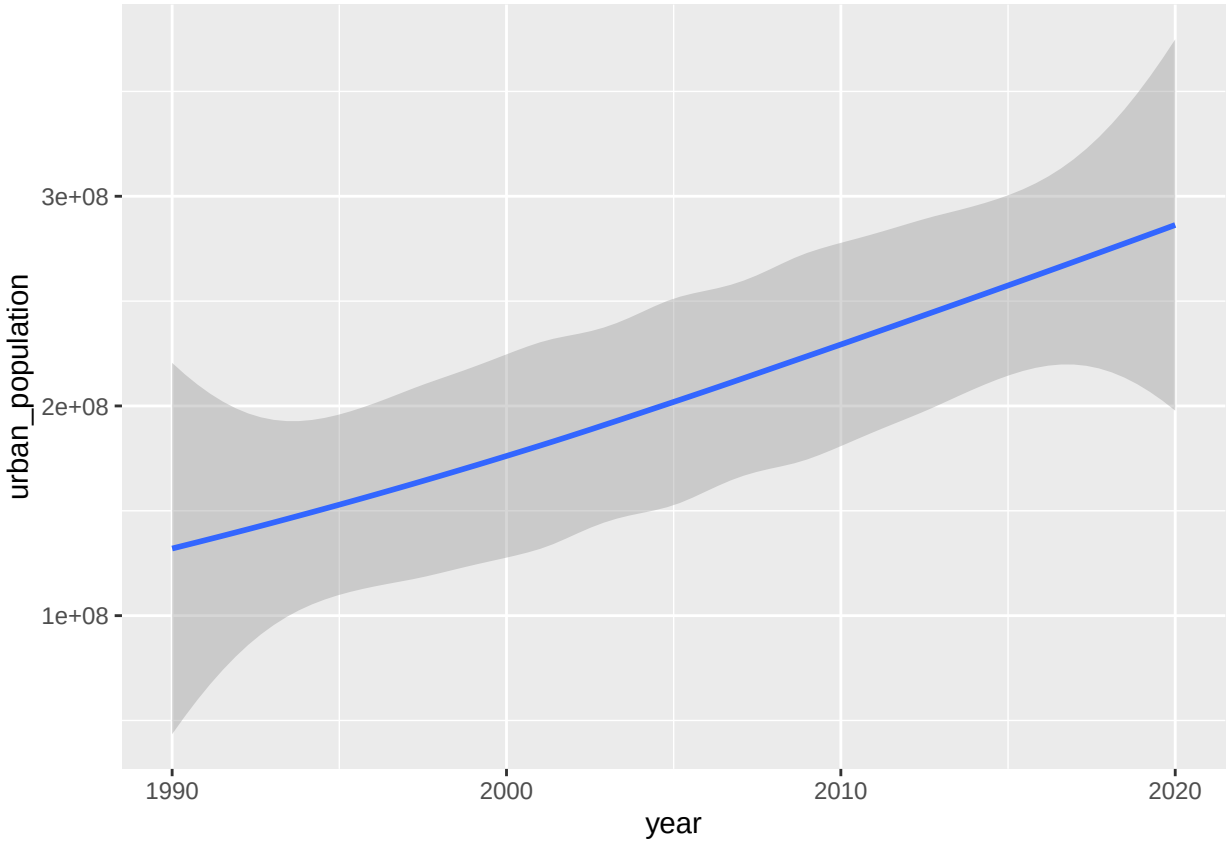
```
ggplot(data = master_data) +
geom_point(mapping = aes(x = year, y = urban_population))
```



Whereas, running the same data with a geom_smooth function will give us:

```
ggplot(data = master_data) +
geom_smooth(mapping = aes(x = year, y = urban_population))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



It is very evident as how two similar data frames can result in different plots due to the geom function. This chapter explores more of geoms and it's influence over plots.

Challenges faced

Creating the module was nothing but full of learning and solving hurdles. Some of the challenges I faced while creating the course module were

1. During the initial phase, It was challenging to integrate the desired data directly in the course module. Assessing the problem made me to believe that the step is not of utmost importance, rather, it was easier to direct the learner to install the data set directly from Github.

2. While creating facets, I faced a hurdle in choosing the appropriate variables to make the visualization of sense. This was tackled by experimenting and playing around with different functions.

3. Since creating .pdf files with R Markdown was a new topic, the formatting of the report was quite exploratory.

4.With very little information available about *swirly* over the internet, a excessive amount had to be dedicated to experimenting with different swirly functions.

What would I have done better with more time and Resources?

Evidently, additional amount of time would have provided me with an opportunity to decide for a data set that suits all the course modules. But this disadvantage provided me with a learning outcome, that not all forms of data visualization suit all types of data sets. Additional time would also provided all three authors to coordinate better and create a master pdf comprising of all three modules.