Visualization with Ggplot in R

Dr. Namig Abbasov

2023-09-13

Getting Started

How to Install R and R Studio

We will use R statistical software within RStudio. You will need to install both software. R is an open-source language and environment for statistical computing and graphics. For additional details, please refer to the R project homepage. RStudio is an integrated development environment (IDE) specifically designed for the R programming language. It provides a user-friendly ***interface*** for writing, running, and managing R code. For additional details, please refer to the Posit's homepage. RStudio company rebranded itself as Posit. Both R and RStudio are free. RStudio is widely used in the R community for statistical modeling, data visualization, and scientific research. It is widely used for replication files of academic journal articles. For instance, you can download replication files in R file formats from https://dataverse.harvard.edu/.

Key features of RStudio:

1. **Script Editor/Source:** RStudio includes a script editor, making it easier to write, edit, and save R codes for future use.
2. **Console:** The R console allows you to interactively execute R code and see the results immediately. You can use it for testing code snippets and exploring data interactively.
3. **EnvironmentWorkspace/History:** In this window, you can manage your R workspace, enviroment, and command history within RStudio. It is a good practice to run rm(list = ls()) code to empty your environment before starting to work on your R file.
4. **File/Plot/Package Management:** In this window, you can work with R scripts, data files, and other resources directly. Plot pane allows to view and interact with plots and visualizations generated by R. From packages section, you can install, update, and loade packages. It also helps you keep track of which packages are currently active in your session.

To install R and RStudio and start using them, follow these steps:

Install R:

1. Go to the R Project website's download page.
   - For Macs, https://cran.r-project.org/bin/macosx/. Make sure whether your laptop is Apple silicon (M1/M2) Macs or Intel (chips) Macs. You can check it from "About This Mac" on your Mac.
   - For Windows, https://cran.r-project.org/bin/windows/base/ or https://cran.r-project.org/
   - For other operating systems, please, check out: https://cran.r-project.org/
2. Download the R installer appropriate for your operating system and run the installer, following the installation instructions.

Install R Studio:

1. Go to the RStudio download page
2. Scroll down and you will see "All Installers and Tarballs" Heading
3. Choose the RStudio Desktop edition that is suitable for your operating system (Windows, macOS, or Linux).
4. Download and run the RStudio installer and follow the installation instructions.
5. Start RStudio: you can open it by clicking on the RStudio icon in your applications.

Set Working Directory

You can set the working directory using the **`setwd()`** function. The working directory is the folder in your file system where R will look for and save files by default. Alternatively, you can set your directory either from RStudio menu or creating R projects.

1. **Using RStudio:**
   - Go to the "Session" menu at the top of the RStudio interface.
   - Select "Set Working Directory."
   - Choose "To Source File Location" to set the working directory to the location of the currently open R script.
   - Or choose "Choose Directory" to manually browse and select the desired directory.
   - Copy the working directory from R Consule and paste it to your R sript/source.
2. **Creating R Projects (Recommended):**

   It is a good practice to create R projects for your work. When you create an R project, the working directory is automatically set to the project's directory. To create an R project:
   - Go to the "File" menu in RStudio.
   - Select "New Project."
   - Choose a project type ("New Directory" or "Existing Directory") and specify the project directory.
   - Click "Create Project." RStudio will open the project and set the working directory accordingly.

Setting the working directory is essential for loading data files, saving output, and managing your R scripts effectively. It is important to set the working directory to ensure that R can access the necessary files and resources for your analysis. Below is the directory in my laptop. You will need to change it in your directory. If you want to work on this vis_1.qmd file in your RStudio, you can manually browse and select the desired directory (Using RStudio option above). The code with your selected working directory will appear in your consule. You can copy that code from R Consule and past to the chunk below, replacing my working directory with yours.

```
rm(list = ls())
setwd("/Users/namigabbasov/Library/Mobile Documents/com~apple~CloudDocs/Statistical training/R files/Rw
```

Importing Data to R

We will work with both built-in datasets and imported data. There are several built-in datasets that come with the base R installation. These datasets cover a wide range of topics and are often used in examples and tutorials. You can import your data either from your computer or an online link. The code you use to import data depends on the type of your data source and format. For instance, if your data is csv format, you will use read.csv() function. You can also import the data by clicking "import" and choosing the right data format in ***Environment*** section of RStudio.

In this tutorial, we will import our dataset from a link to GitHub repository. All RStudio, data, and rendered html files are stored in RWorkshop repository of our UnitforDataScience GitHub. The dataset is structured in a country-year format, covering comprehensive information about all countries spanning the years from 1946 to 2021. Most of variables have been merged from Quality of Government(QoG) Standard Dataset. You can access to the Codebook from QoG Institute.

```
total<-read.csv("https://raw.githubusercontent.com/UnitForDataScience/RWorkshop/main/RWorkshop_data.csv

#total<-read.csv("/Users/namigabbasov/Library/Mobile Documents/com~apple~CloudDocs/Statistical training
# This is from my laptop.
#It won't work in your machine because your directory is different.
#If you can't import the data from our GitHub repository,
#you can just download it and import it manually from your RStudio:
#Environment=>Import=>From Text(readr)=>Browse=>select where you have stored the data.
```

Install Required Packages

In R, packages are collections of functions, data sets, and documentation bundled together for specific purposes.

Installing Packages:

- Before you can use a package, you need to install it. You can install packages from CRAN (Comprehensive R Archive Network). Cran is the primary repository for R packages. You can also install from other sources like GitHub.
- To install a package from CRAN, use the **`install.packages()`** function, followed by the package name in quotes. For instance, **`install.packages(`**"tidyverse"**`)`**.
- To install a package from GitHub, you can use the **`devtools`** package and its **`install_github()`** function. But first, you will neeed to install and load the **`devtools`** package, and then install the package from GitHub.
- To see a list of all the packages installed, you can use the **`installed.packages()`** function.
- If you no longer need a package, you can remove it from your R installation using the **`remove.packages()`** function.

Loading Packages:

- Once a package is installed, you need to load it into your R session using the **`library()`** or **`require()`** function. Loading a package makes its functions available for use.
- You can also use **`require()`** to load a package, which returns a logical value (**`TRUE`** if the package is available, **`FALSE`** if it is not).
- You might need to update packages by running the **`update.packages()`** function. Over time, packages may receive updates to fix bugs or add new features. You might need to update packages if you get errors for no clear reason.

Most Common File Types in RStudio

1. **R Script (.R)**: most common file type used for writing R code. You can create, edit, and run R scripts in RStudio. You can create R Script from File=>New File=>R Script
2. **R Markdown (.Rmd)**: R Markdown files allow you to mix R code, text, and formatting in a single document. You can create from You can create R Script from File=>New File=>R Markdown.

3. **Quarto Document (qmd):** Quarto offers a comprehensive authoring framework tailored for data science. It is very effective in integrating your code, its outcomes, and your written explanations. Quarto documents ensure complete reproducibility and offer versatile output formats, including PDFs, Word documents, presentations, and more. Our R Workshop code files are prepared in this format.

   Quarto files are designed to serve three primary purposes:

   - **Effective Communication:** They enable concise communication with decision-makers who prioritize conclusions over the technical details of the analysis.
   - **Collaboration:** Quarto facilitates collaboration among data scientists, allowing them to delve into both the conclusions and the underlying code that led to those conclusions.
   - **Data Science Environment:** Quarto serves as a dynamic workspace for data science, functioning as a contemporary digital lab notebook. Here, you can not only document your actions but also capture your thoughts and insights. Please, refer to R for Data Science Book for more info.

4. **R Project (.Rproj)**: An R Project file is used to organize your work within RStudio. It helps manage your files, packages, and settings for a specific project.

5. **HTML Files (.html)**: You can create, edit, and view HTML files in RStudio, which is useful for generating web-based reports and documentation.

6. **PDF Files (.pdf)**: RStudio can generate PDF documents from R Markdown and Quarto files using packages like **`rmarkdown`** and **`knitr`**.

7. **Shiny Apps (.R, .Rmd)**: If you are developing Shiny web applications in RStudio, you'll typically work with .R and .Rmd files.

8. **Image Files**: **PNG, JPG, GIF (.png, .jpg, .gif)**: You can create and manipulate images using R packages like **`ggplot2`** and **`imager`**.

9. **Data Files**:

   - **CSV (.csv)**: Comma-separated values files are widely used for storing tabular data.
   - **Excel (.xlsx, .xls)**: RStudio can read and write Excel files using packages like **`readxl`** and **`writexl`**.
   - **R Data (.RData)**: You can save and load R objects, including data frames and variables, using .RData files.
   - **Other Data Formats**: RStudio supports various other data formats such as JSON, XML, HDF5, and more, often with the help of specific packages.

Make sure you have installed and loaded the following packages.

```
#install.packages("tidyverse")
#install.packages("stargazer")
#install.packages("grid")
#install.packages("fBasics")
#install.packages("extraGrid")
datasets::anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
```

```
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

```
library(grid)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(datasets)
library(stargazer)
```

```
##
## Please cite as:
##
##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
library(palmerpenguins)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggthemes)
library(fBasics)
library(grid)
library(gridExtra)
```

```
##
```

```
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following object is masked from 'package:fBasics':
##
##     tr
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

1. Why should we care about visualization?

- Caution Against Over-Reliance on Summary Statistics.
- Similar summary statistics can exhibit very different patterns when plotted visually.
- Detecting Outliers and Influential points

In this chunk, we will explore Anscombe's Quartet, which is a collection of four datasets that were created by the British statistician Francis Anscombe in 1973. These datasets have similar statistical properties, including means, variances, correlations, and linear regression parameters, but they exhibit dramatically different patterns when visualized. Anscombe created these datasets to emphasize the importance of graphical representation in exploring and understanding data.

Now let's look at some descriptive statistics. We will use anscombe dataset (Anscombe's Quartet), which is a built-in dataset from library(datasets) in R.

```
sapply(anscombe[, 1:4], mean) #Mean of Xs
```

```
## x1 x2 x3 x4
##  9  9  9  9
```

```
sapply(anscombe[, 1:4], sd)   #Standard Deviation of Xs
```

```
##       x1       x2       x3       x4
## 3.316625 3.316625 3.316625 3.316625
```

```
sapply(anscombe[, 1:4], var)  #Variance of Xs
```

```
## x1 x2 x3 x4
## 11 11 11 11
```

```r
sapply(anscombe[, 5:8], mean) #mean of Ys
```

```
##       y1       y2       y3       y4
## 7.500909 7.500909 7.500000 7.500909
```

```r
sapply(anscombe[, 5:8], sd)   #Standard Deviation of Ys
```

```
##       y1       y2       y3       y4
## 2.031568 2.031657 2.030424 2.030579
```

```r
summary(anscombe)             #Data summary
```

```
##        x1              x2              x3              x4           y1
##  Min.   : 4.0   Min.   : 4.0   Min.   : 4.0   Min.   : 8   Min.   : 4.260
##  1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 8   1st Qu.: 6.315
##  Median : 9.0   Median : 9.0   Median : 9.0   Median : 8   Median : 7.580
##  Mean   : 9.0   Mean   : 9.0   Mean   : 9.0   Mean   : 9   Mean   : 7.501
##  3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.: 8   3rd Qu.: 8.570
##  Max.   :14.0   Max.   :14.0   Max.   :14.0   Max.   :19   Max.   :10.840
##        y2              y3              y4
##  Min.   :3.100   Min.   : 5.39   Min.   : 5.250
##  1st Qu.:6.695   1st Qu.: 6.25   1st Qu.: 6.170
##  Median :8.140   Median : 7.11   Median : 7.040
##  Mean   :7.501   Mean   : 7.50   Mean   : 7.501
##  3rd Qu.:8.950   3rd Qu.: 7.98   3rd Qu.: 8.190
##  Max.   :9.260   Max.   :12.74   Max.   :12.500
```

```r
fBasics::basicStats(anscombe) #Basic Stats
```

```
##                     x1        x2        x3        x4        y1        y2
## nobs        11.000000 11.000000 11.000000 11.000000 11.000000 11.000000
## NAs          0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
## Minimum      4.000000  4.000000  4.000000  8.000000  4.260000  3.100000
## Maximum     14.000000 14.000000 14.000000 19.000000 10.840000  9.260000
## 1. Quartile  6.500000  6.500000  6.500000  8.000000  6.315000  6.695000
## 3. Quartile 11.500000 11.500000 11.500000  8.000000  8.570000  8.950000
## Mean         9.000000  9.000000  9.000000  9.000000  7.500909  7.500909
## Median       9.000000  9.000000  9.000000  8.000000  7.580000  8.140000
## Sum         99.000000 99.000000 99.000000 99.000000 82.510000 82.510000
## SE Mean      1.000000  1.000000  1.000000  1.000000  0.612541  0.612568
## LCL Mean     6.771861  6.771861  6.771861  6.771861  6.136083  6.136024
## UCL Mean    11.228139 11.228139 11.228139 11.228139  8.865735  8.865795
## Variance    11.000000 11.000000 11.000000 11.000000  4.127269  4.127629
## Stdev        3.316625  3.316625  3.316625  3.316625  2.031568  2.031657
## Skewness     0.000000  0.000000  0.000000  2.466911 -0.048374 -0.978693
## Kurtosis    -1.528926 -1.528926 -1.528926  4.520661 -1.199123 -0.514319
##                     y3        y4
## nobs        11.000000 11.000000
## NAs          0.000000  0.000000
## Minimum      5.390000  5.250000
## Maximum     12.740000 12.500000
```

```
## 1. Quartile  6.250000  6.170000
## 3. Quartile  7.980000  8.190000
## Mean         7.500000  7.500909
## Median       7.110000  7.040000
## Sum         82.500000 82.510000
## SE Mean      0.612196  0.612242
## LCL Mean     6.135943  6.136748
## UCL Mean     8.864057  8.865070
## Variance     4.122620  4.123249
## Stdev        2.030424  2.030579
## Skewness     1.380120  1.120774
## Kurtosis     1.240044  0.628751
```

```
cor(anscombe)                #Correlation
```

```
##            x1         x2         x3         x4         y1         y2         y3
## x1  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x2  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x3  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x4 -0.5000000 -0.5000000 -0.5000000  1.0000000 -0.5290927 -0.7184365 -0.3446610
## y1  0.8164205  0.8164205  0.8164205 -0.5290927  1.0000000  0.7500054  0.4687167
## y2  0.8162365  0.8162365  0.8162365 -0.7184365  0.7500054  1.0000000  0.5879193
## y3  0.8162867  0.8162867  0.8162867 -0.3446610  0.4687167  0.5879193  1.0000000
## y4 -0.3140467 -0.3140467 -0.3140467  0.8165214 -0.4891162 -0.4780949 -0.1554718
##            y4
## x1 -0.3140467
## x2 -0.3140467
## x3 -0.3140467
## x4  0.8165214
## y1 -0.4891162
## y2 -0.4780949
## y3 -0.1554718
## y4  1.0000000
```

```
cor(anscombe, method = c("pearson"), use = "pairwise.complete.obs")
```

```
##            x1         x2         x3         x4         y1         y2         y3
## x1  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x2  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x3  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x4 -0.5000000 -0.5000000 -0.5000000  1.0000000 -0.5290927 -0.7184365 -0.3446610
## y1  0.8164205  0.8164205  0.8164205 -0.5290927  1.0000000  0.7500054  0.4687167
## y2  0.8162365  0.8162365  0.8162365 -0.7184365  0.7500054  1.0000000  0.5879193
## y3  0.8162867  0.8162867  0.8162867 -0.3446610  0.4687167  0.5879193  1.0000000
## y4 -0.3140467 -0.3140467 -0.3140467  0.8165214 -0.4891162 -0.4780949 -0.1554718
##            y4
## x1 -0.3140467
## x2 -0.3140467
## x3 -0.3140467
## x4  0.8165214
## y1 -0.4891162
## y2 -0.4780949
## y3 -0.1554718
## y4  1.0000000
```

```r
cor(anscombe[, c('x1', 'x2', 'x3', 'x4', 'y1', 'y2','y3', 'y4')])
```

```
##            x1         x2         x3         x4         y1         y2         y3
## x1  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x2  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x3  1.0000000  1.0000000  1.0000000 -0.5000000  0.8164205  0.8162365  0.8162867
## x4 -0.5000000 -0.5000000 -0.5000000  1.0000000 -0.5290927 -0.7184365 -0.3446610
## y1  0.8164205  0.8164205  0.8164205 -0.5290927  1.0000000  0.7500054  0.4687167
## y2  0.8162365  0.8162365  0.8162365 -0.7184365  0.7500054  1.0000000  0.5879193
## y3  0.8162867  0.8162867  0.8162867 -0.3446610  0.4687167  0.5879193  1.0000000
## y4 -0.3140467 -0.3140467 -0.3140467  0.8165214 -0.4891162 -0.4780949 -0.1554718
##            y4
## x1 -0.3140467
## x2 -0.3140467
## x3 -0.3140467
## x4  0.8165214
## y1 -0.4891162
## y2 -0.4780949
## y3 -0.1554718
## y4  1.0000000
```

Let's build linear regression models

```r
m1<-lm(y1~x1, data=anscombe)
m2<-lm(y2~x2, data=anscombe)
m3<-lm(y3~x3, data=anscombe)
m4<-lm(y4~x4, data=anscombe)
stargazer(m1, m2, m3, m4, type="text")
```

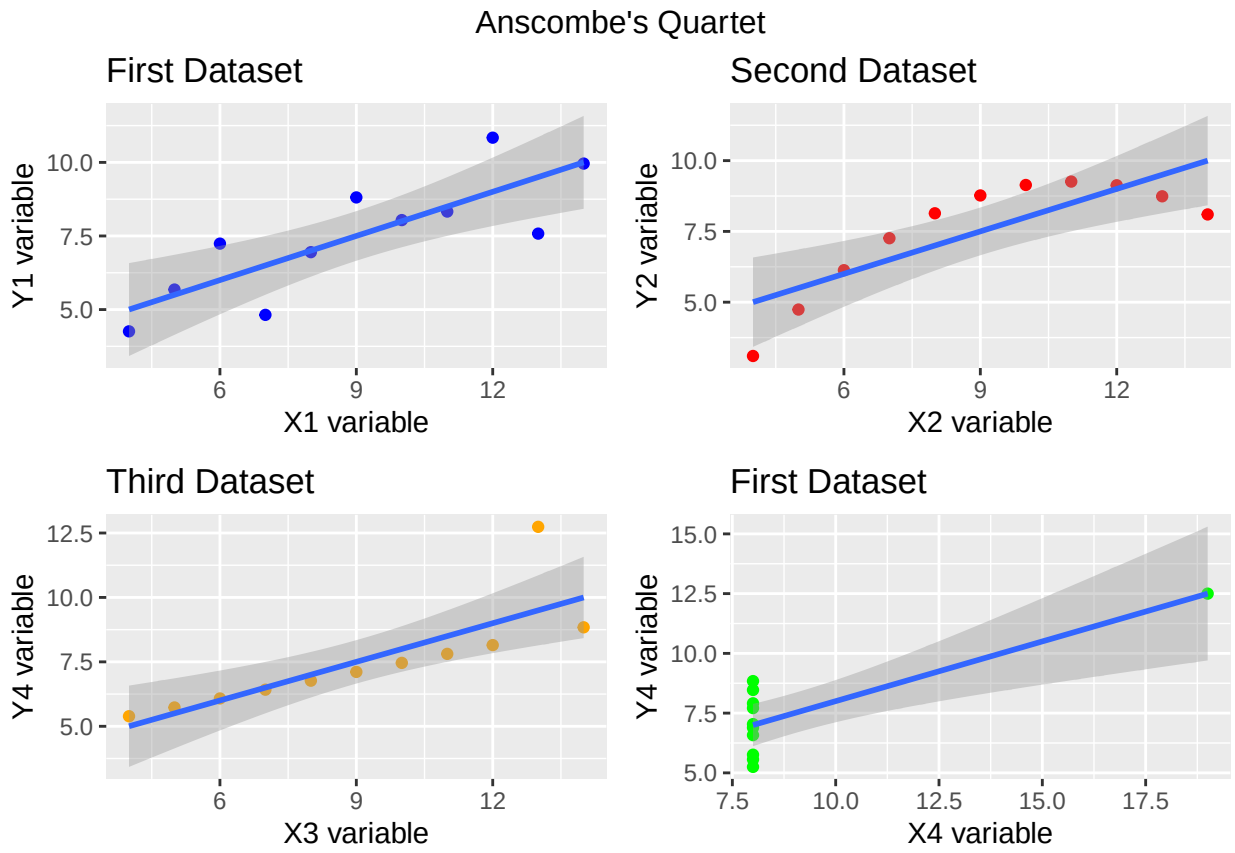
```
##
## ===============================================================
##                                   Dependent variable:
##                               -----------------------------------
##                                  y1       y2       y3       y4
##                                  (1)      (2)      (3)      (4)
## ---------------------------------------------------------------
## x1                            0.500***
##                               (0.118)
##
## x2                                     0.500***
##                                        (0.118)
##
## x3                                              0.500***
##                                                 (0.118)
##
## x4                                                       0.500***
##                                                          (0.118)
##
## Constant                      3.000**  3.001**  3.002**  3.002**
##                               (1.125)  (1.125)  (1.124)  (1.124)
##
## ---------------------------------------------------------------
```

```
## Observations                    11        11        11        11
## R2                             0.667     0.666     0.666     0.667
## Adjusted R2                    0.629     0.629     0.629     0.630
## Residual Std. Error (df = 9)   1.237     1.237     1.236     1.236
## F Statistic (df = 1; 9)      17.990*** 17.966*** 17.972*** 18.003***
## ============================================================================
## Note:                                        *p<0.1; **p<0.05; ***p<0.01
```

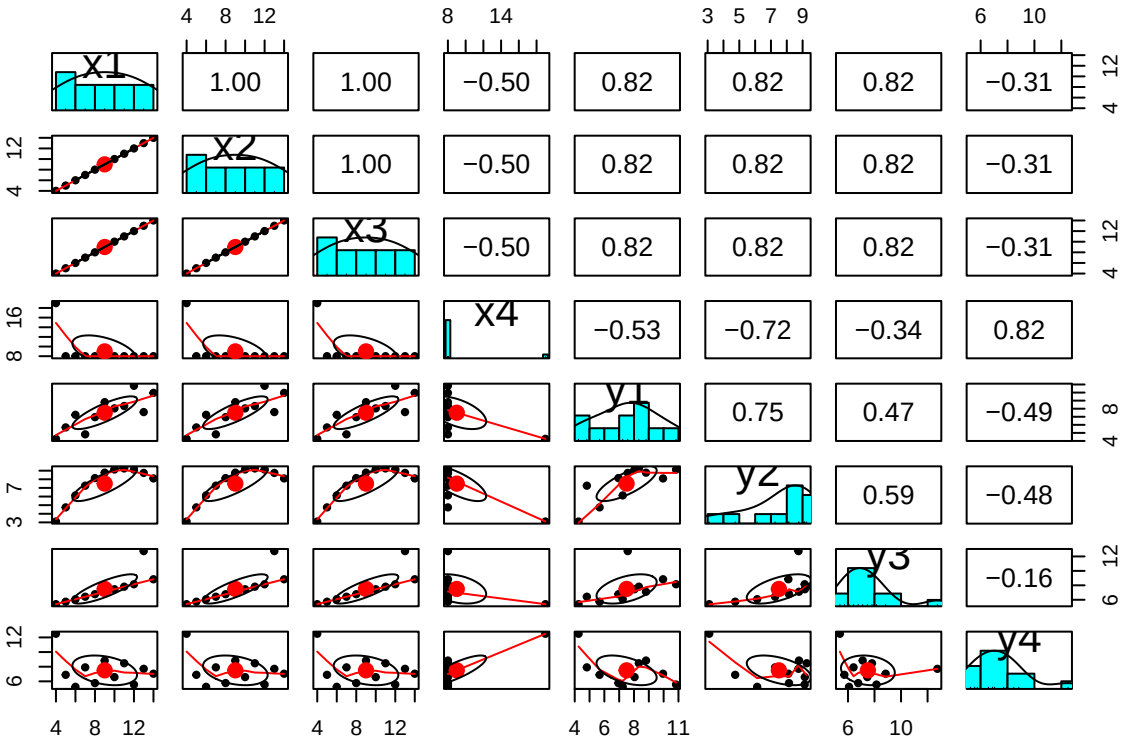
Now Let's plot and explore the relationship between these variables visually.

```
p1<-ggplot(data=anscombe, aes(x=x1, y=y1))+
  geom_point(color="blue")+
  labs(x = "X1 variable", y = "Y1 variable",
       title = "First Dataset" ) +
  geom_smooth(method = "lm")

p2<-ggplot(data=anscombe, aes(x=x2, y=y2))+
  geom_point(color="red")+
  labs(x = "X2 variable", y = "Y2 variable",
       title = "Second Dataset" ) +
  geom_smooth(method = "lm")

p3<-ggplot(data=anscombe, aes(x=x3, y=y3))+
  geom_point(color="orange")+
  labs(x = "X3 variable", y = "Y4 variable",
       title = "Third Dataset" ) +
  geom_smooth(method = "lm")

p4<-ggplot(data=anscombe, aes(x=x4, y=y4))+
  geom_point(color="green")+
  labs(x = "X4 variable", y = "Y4 variable",
       title = "First Dataset" ) +
  geom_smooth(method = "lm")

grid.arrange(grobs = list(p1, p2, p3, p4),
             ncol = 2,
             top = "Anscombe's Quartet", padding = unit(0.5, "line"))
```

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




```
pairs.panels(anscombe)
```



2. What is ggplot: Basic Syntax

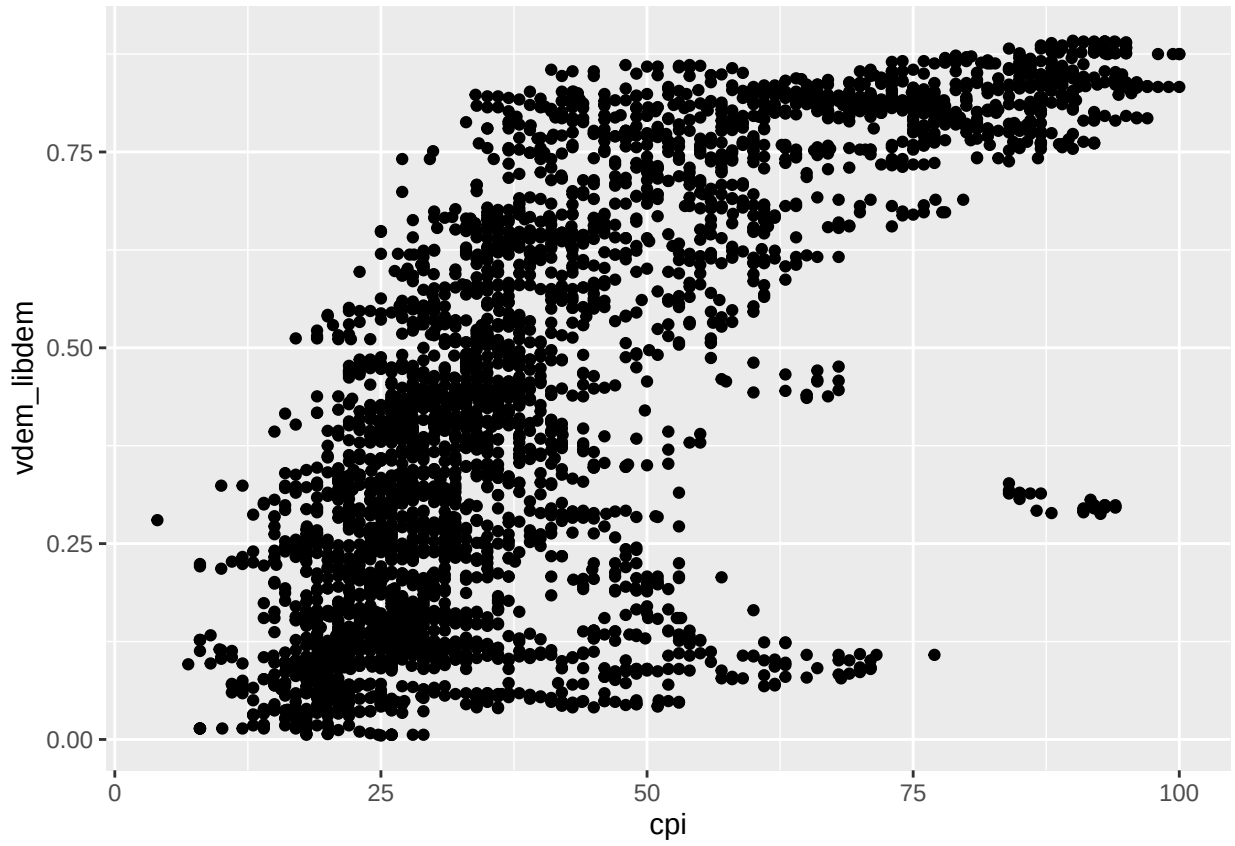
ggplot2 is a powerful data visualization package in R to create high-quality and customization visualizations. The "gg" in ggplot2 means ***"Grammar of Graphics".***

Key aspects of ggplot2

1. ggplot2 follows a layered grammer approach to building visualizations. Each layer represents a different aspect of the plot.
2. Data Mapping aspect maps variables to aesthetics of the plot, such as x and y positions, color, shape, size, and more. It allows to establish complex visualizations.
3. Geometric objects (aka geoms) represent the visual elements of plots. Geoms enable to make various forms of visualizations such scatterplot, lines, bars etc.
4. Aesthetics and themes customize the aesthetics of plot elements, such as colors, shapes, and sizes, to convey information effectively.
5. Ggplot2 supports faceting. Faceting means splitting the data into subsets and creating separate plots for each subset. Faceting is very useful to visualize data across different categories.
6. The ggplot2 package has inspired the development of various extensions and packages that build upon its framework to create specialized and complex visualizations. Examples include **gganimate, ggmap, ggplotly, ggdist.**

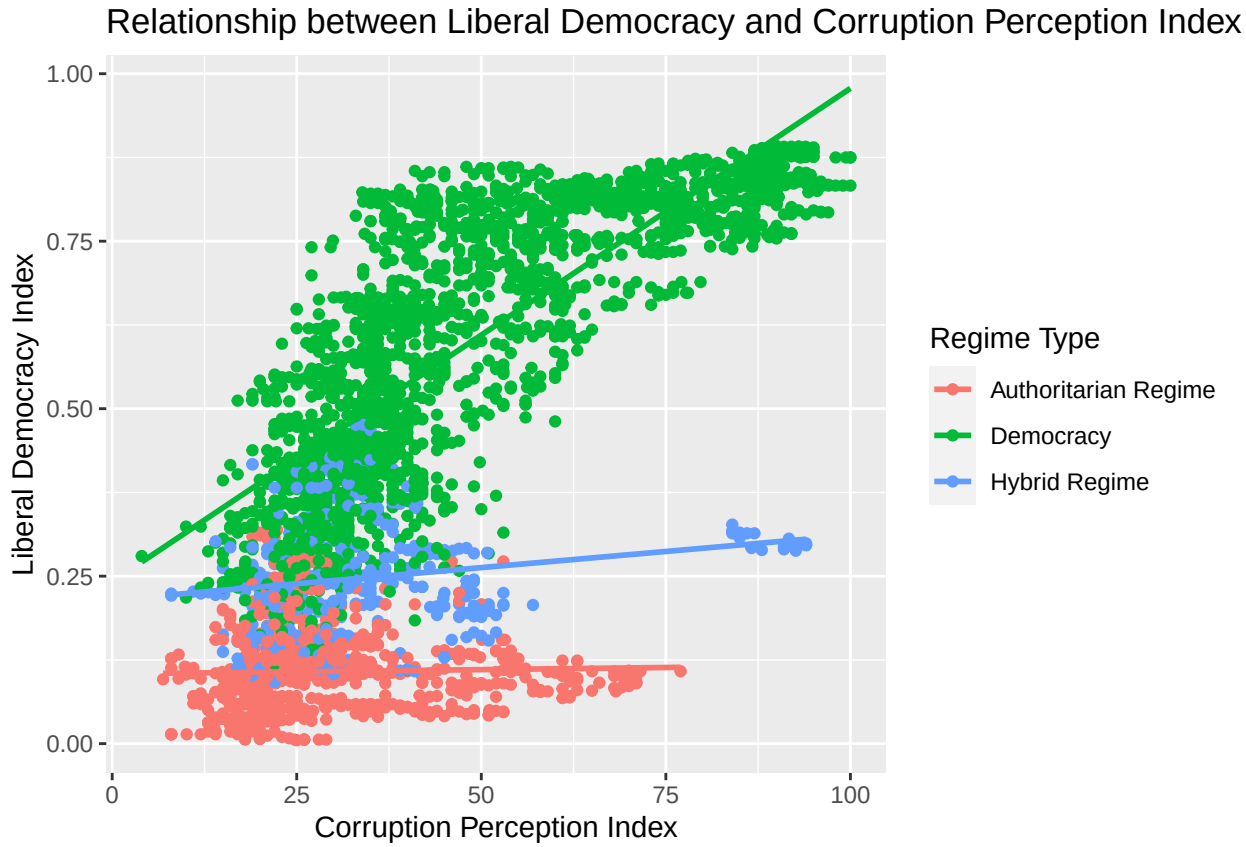
```
#basic syntax to get a plot
basic_plot<-total[c("cpi", "vdem_libdem")]          #first subset needed columns
basic_plot<-na.omit(basic_plot)                     #delete missing observations
```

```
b1<-ggplot(                                          #calling ggplot
  data = basic_plot,                                 #define data
       mapping = aes(x = cpi, y = vdem_libdem))+     #identify x-axis and y-axis
  geom_point()                                       #choose geom type
b1
```



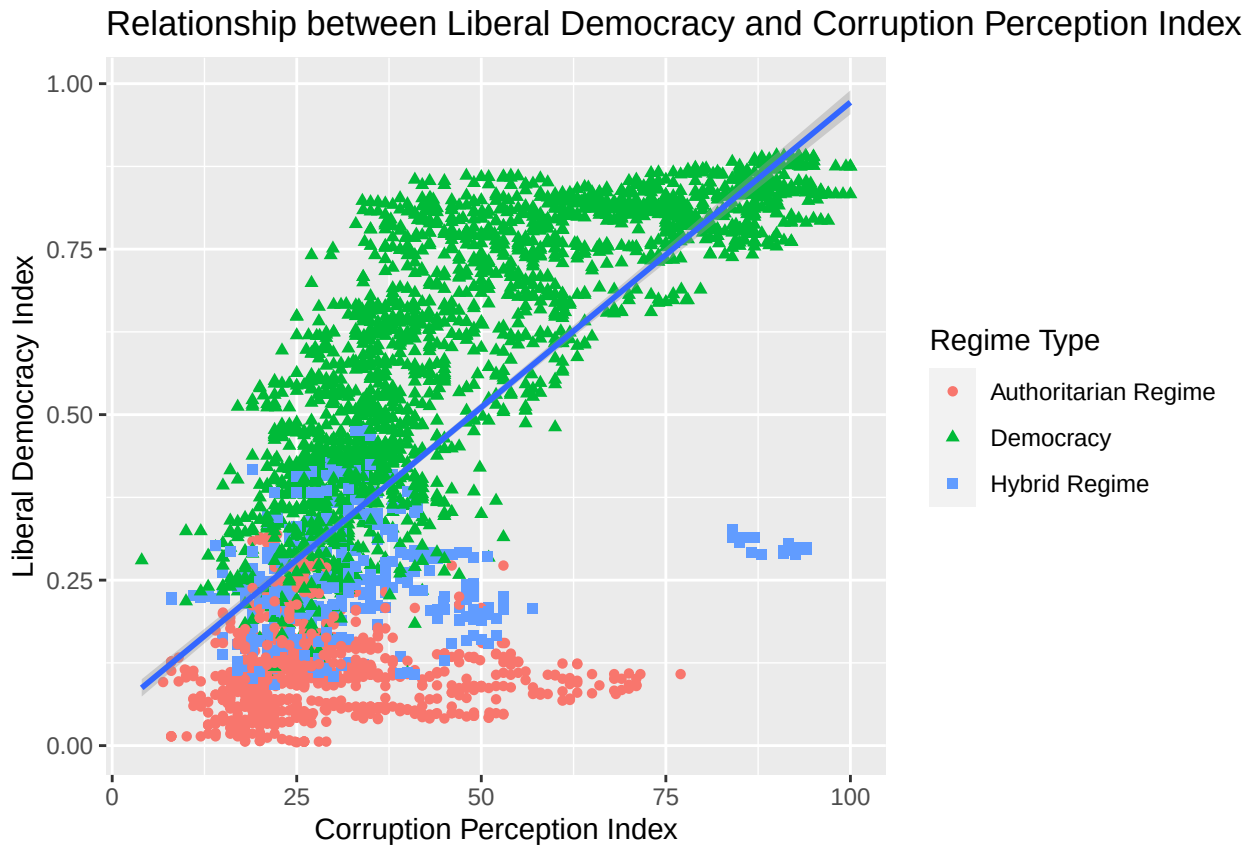

```
#Adding more features: Global Environment
b2<-total |>
  filter(!is.na(cpi),
         !is.na(vdem_libdem),
         !is.na(regime_status_name)) |>
  ggplot(aes(x = cpi, y = vdem_libdem, color =regime_status_name)) +
  geom_point() +
  geom_smooth(method = "lm", se=F) +
  labs(
    title = "Relationship between Liberal Democracy and Corruption Perception Index",
    x = "Corruption Perception Index",
    y = "Liberal Democracy Index",
    color = "Regime Type"
  )
b2
```

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



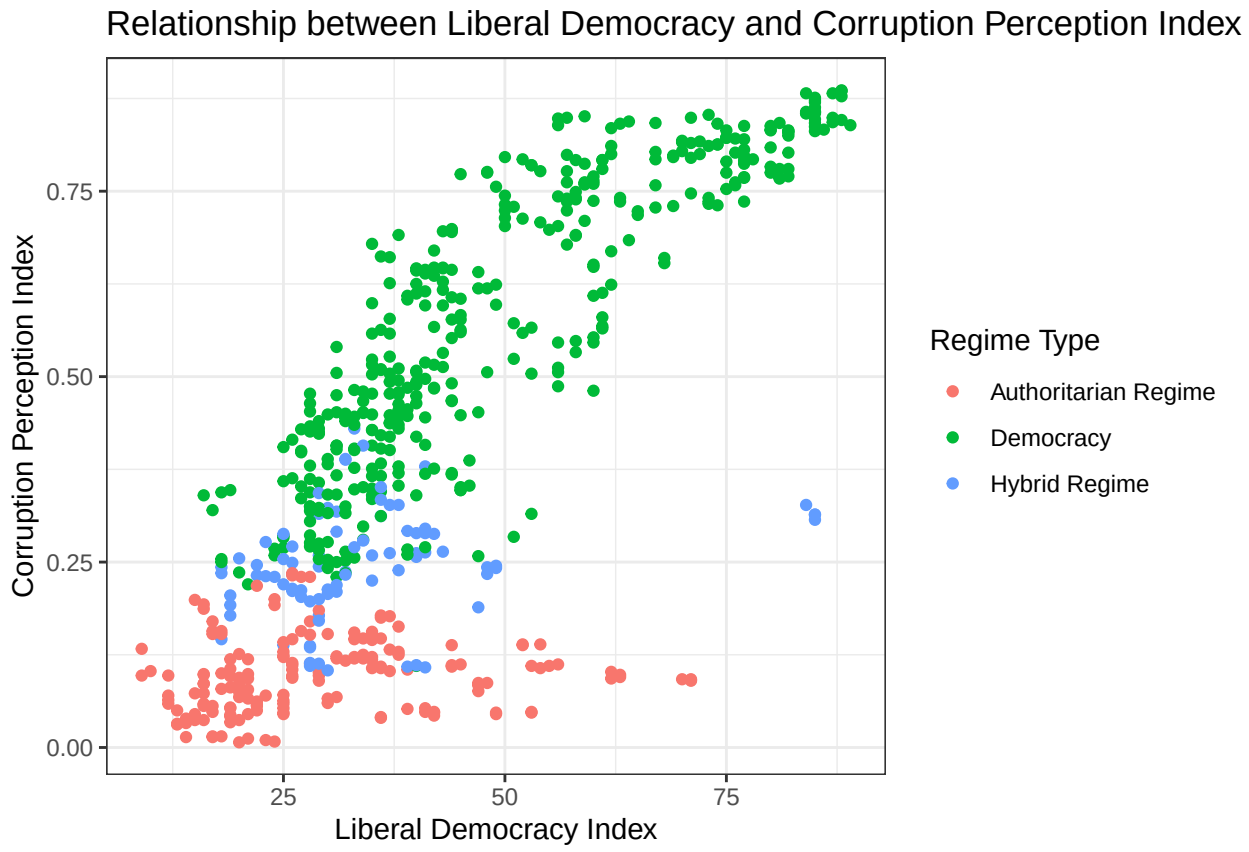

```
#Local Environment: One line
b3<-total |>
  filter(!is.na(cpi),
         !is.na(vdem_libdem),
         !is.na(regime_status_name)) |> #choosing not missing observations
  ggplot(aes(x = cpi, y = vdem_libdem)) +
  geom_point(aes(color =regime_status_name, shape=regime_status_name)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Relationship between Liberal Democracy and Corruption Perception Index",
    x = "Corruption Perception Index",
    y = "Liberal Democracy Index",
    color = "Regime Type",
    shape = "Regime Type"
  )
b3
```

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

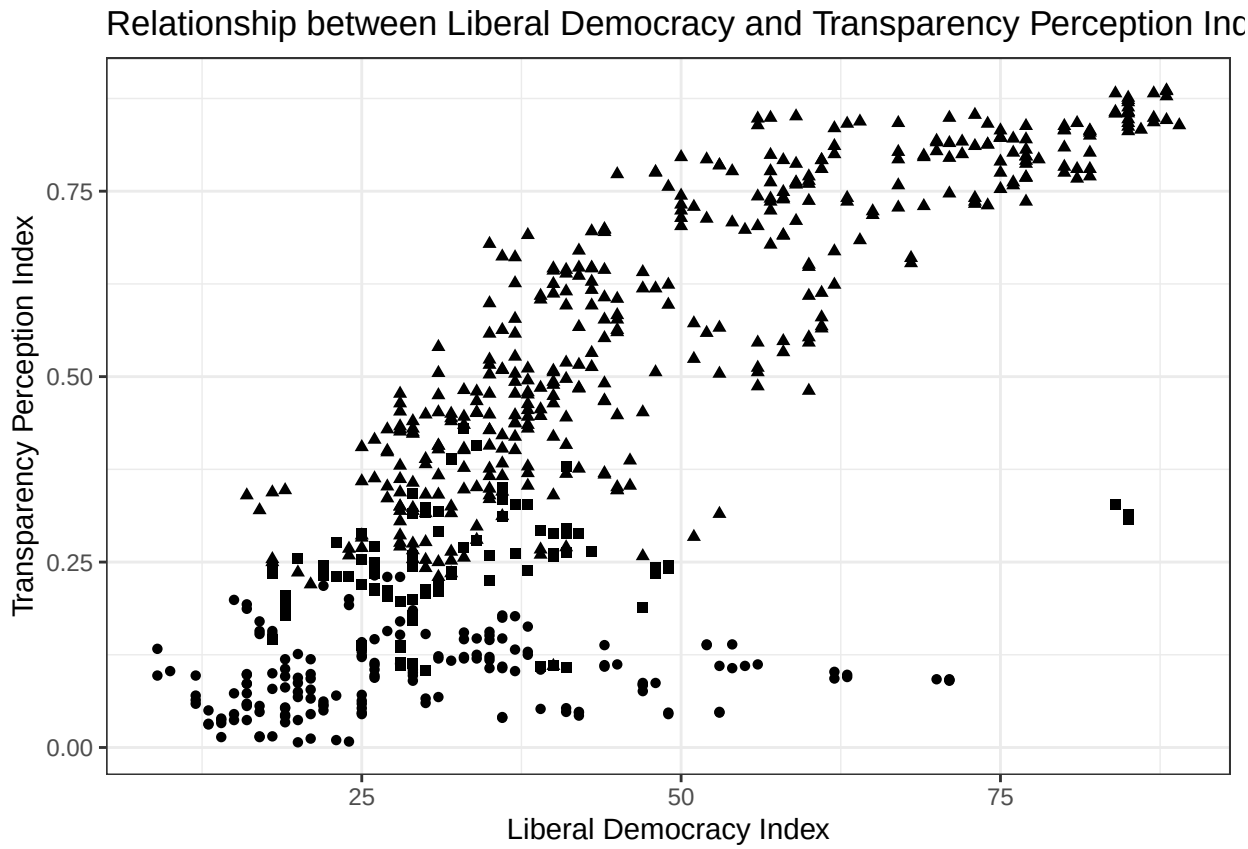


3. Aesthetic mappings: Adding a Third Variable

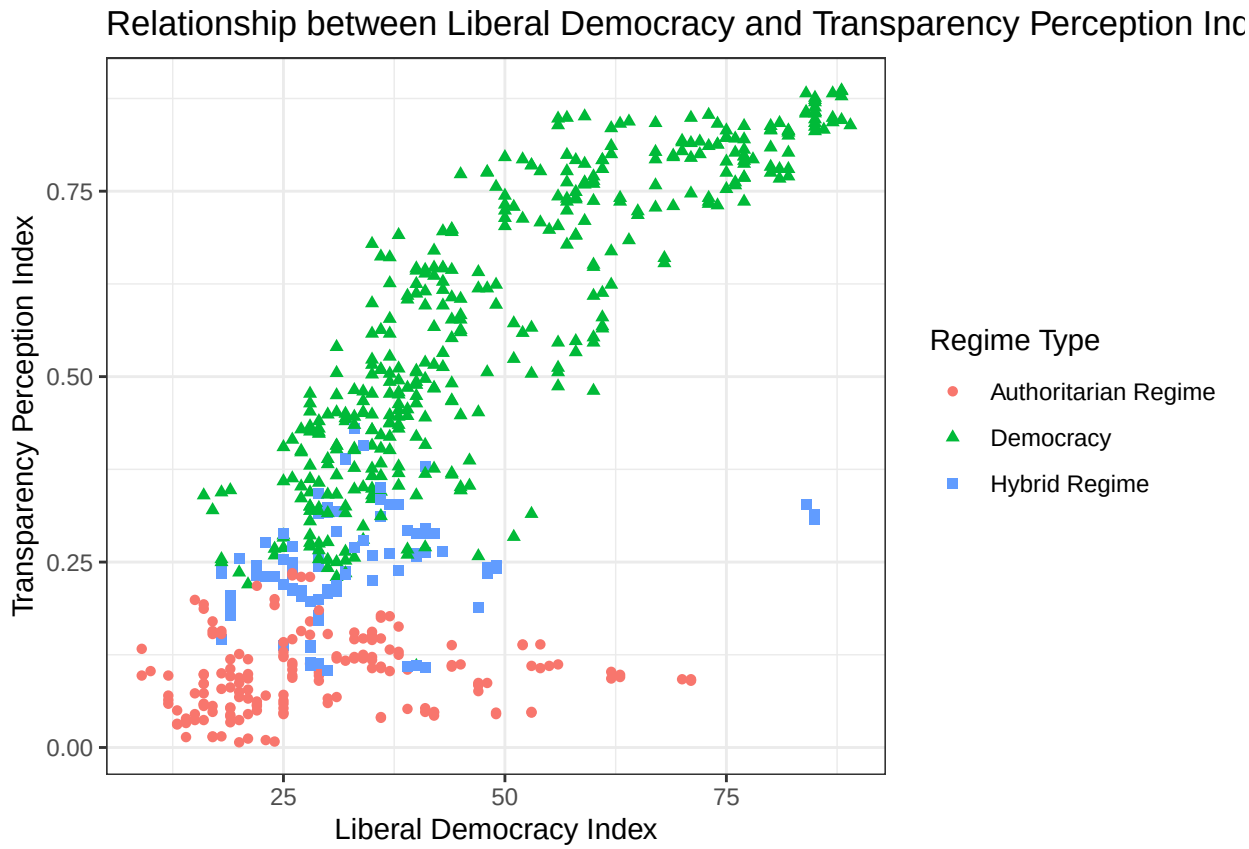
```
total20 <- total |> #subsetting data
  filter(year %in% c(2017:2021))


#color aesthetic
p1<-ggplot(data=total20 |>
             filter(!is.na(cpi),
                    !is.na(vdem_libdem),
                    !is.na(regime_status_name)),
           mapping = aes(x=cpi, y=vdem_libdem, color=regime_status_name))+
  geom_point()+
  labs(title = "Relationship between Liberal Democracy and Corruption Perception Index",
       x = "Liberal Democracy Index",
       y = "Corruption Perception Index",
       color="Regime Type")+
  theme_bw()
p1
```



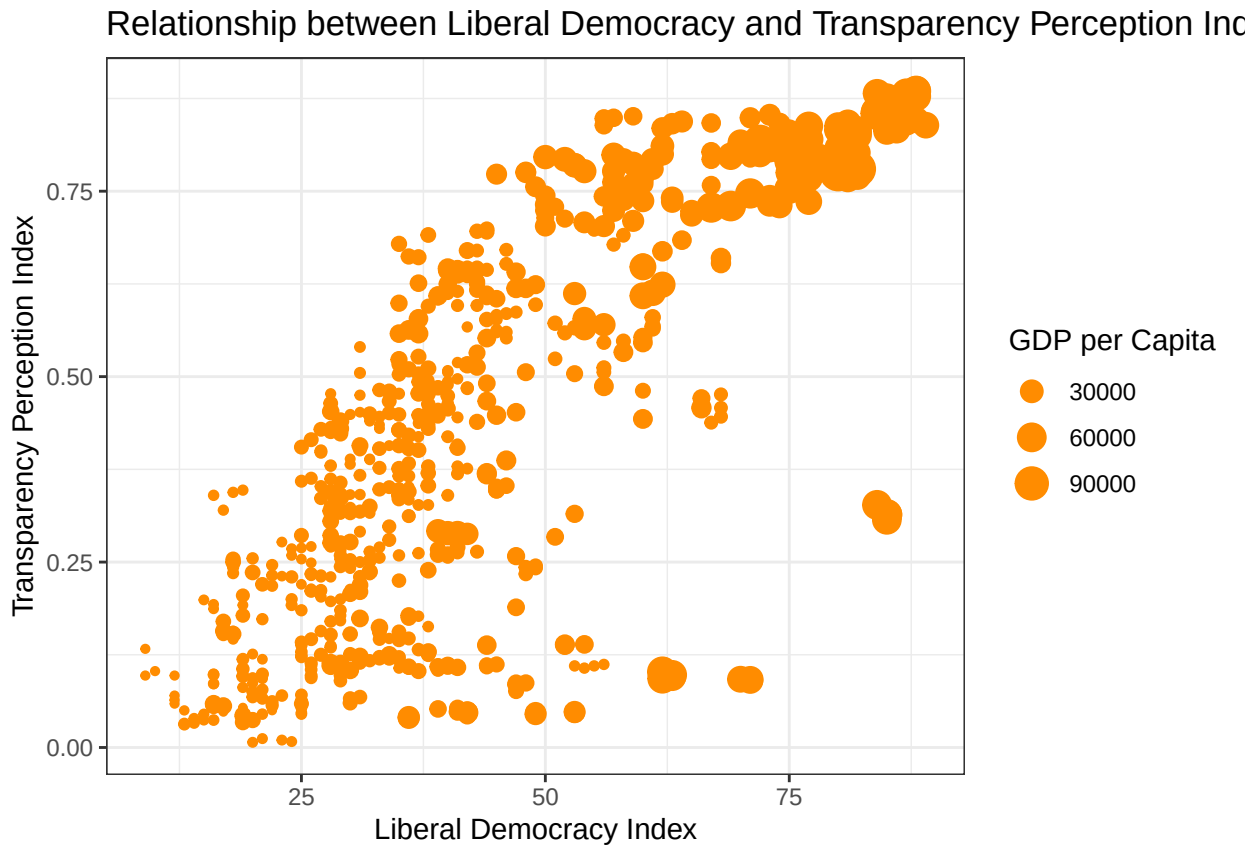

```
#class aesthetic
p2<-ggplot(data=total20 |>
             filter(!is.na(cpi),
                    !is.na(vdem_libdem),
                    !is.na(regime_status_name)),
           mapping = aes(x=cpi, y=vdem_libdem, shape=regime_status_name))+
  geom_point(color="black")+
  labs(title = "Relationship between Liberal Democracy and Transparency Perception Index",
       x = "Liberal Democracy Index",
       y = "Transparency Perception Index",
       shape="Regime Type")+
  guides(shape="none")+
  theme_bw()
p2
```




```
#class and color aesthetic
p3<-ggplot(data=total20 |>
             filter(!is.na(cpi),
                    !is.na(vdem_libdem),
                    !is.na(regime_status_name)),
           mapping = aes(x=cpi, y=vdem_libdem, shape=regime_status_name, color=regime_status_name))+
  geom_point()+
  labs(title = "Relationship between Liberal Democracy and Transparency Perception Index",
       x = "Liberal Democracy Index",
       y = "Transparency Perception Index",
       shape="Regime Type",
       color="Regime Type")+
  theme_bw()
p3
```

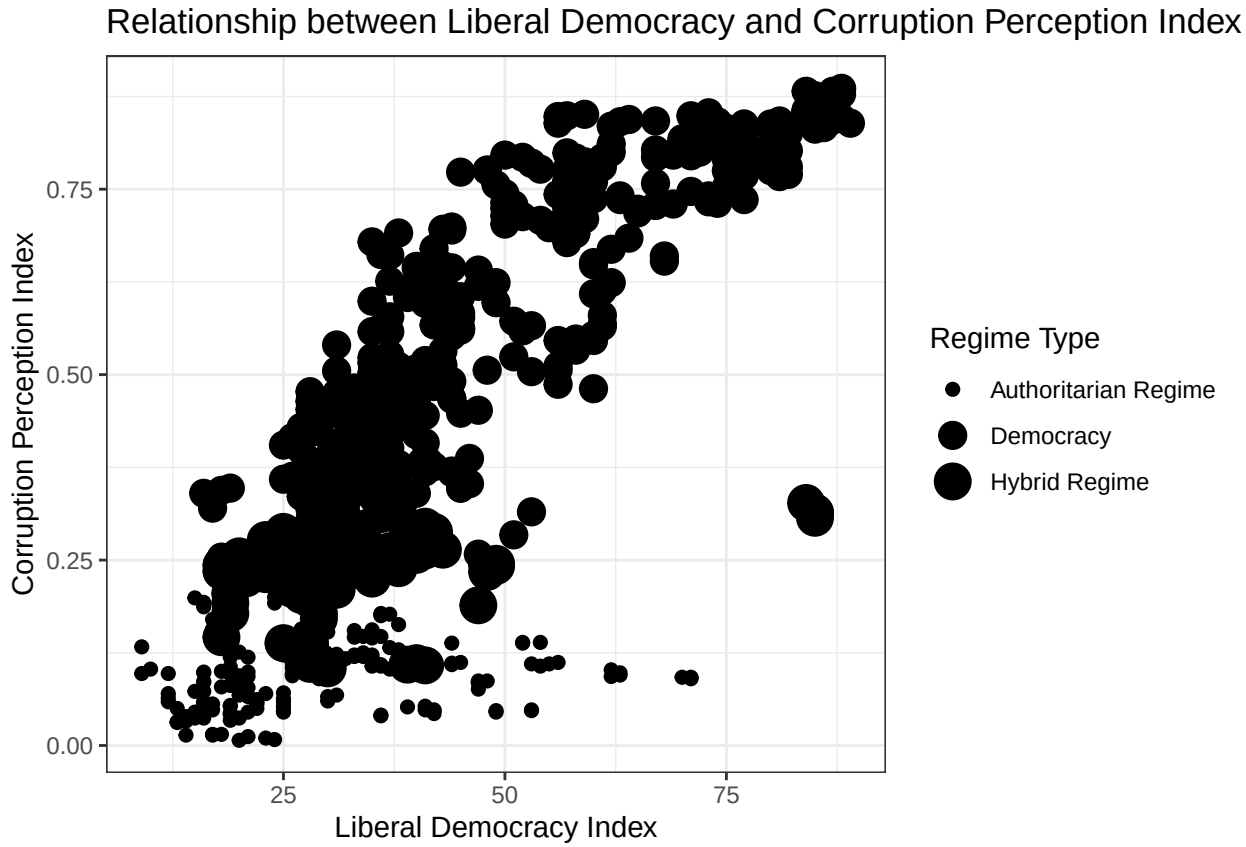



```
#size aesthetic
p4<-ggplot(data=total20 |>
             filter(!is.na(cpi),
                    !is.na(vdem_libdem),
                    !is.na(gdp)),
           mapping = aes(x=cpi, y=vdem_libdem, size=gdp))+
  geom_point(color="darkorange")+
  labs(title = "Relationship between Liberal Democracy and Transparency Perception Index",
       x = "Liberal Democracy Index",
       y = "Transparency Perception Index",
       size="GDP per Capita")+
  theme_bw()
p4
```




```
#size aesthetic with a categorical variable(not recommended)- looks terrible?
p5<-ggplot(data=total20 |>
             filter(!is.na(cpi),
                    !is.na(vdem_libdem),
                    !is.na(regime_status_name)),
           mapping = aes(x=cpi, y=vdem_libdem, size=regime_status_name))+
  geom_point()+
  labs(title = "Relationship between Liberal Democracy and Corruption Perception Index",
       x = "Liberal Democracy Index",
       y = "Corruption Perception Index",
       size="Regime Type")+
  theme_bw()
p5
```

```
## Warning: Using size for a discrete variable is not advised.
```



4. Geom Objects: Different Types of Graphical Presentation

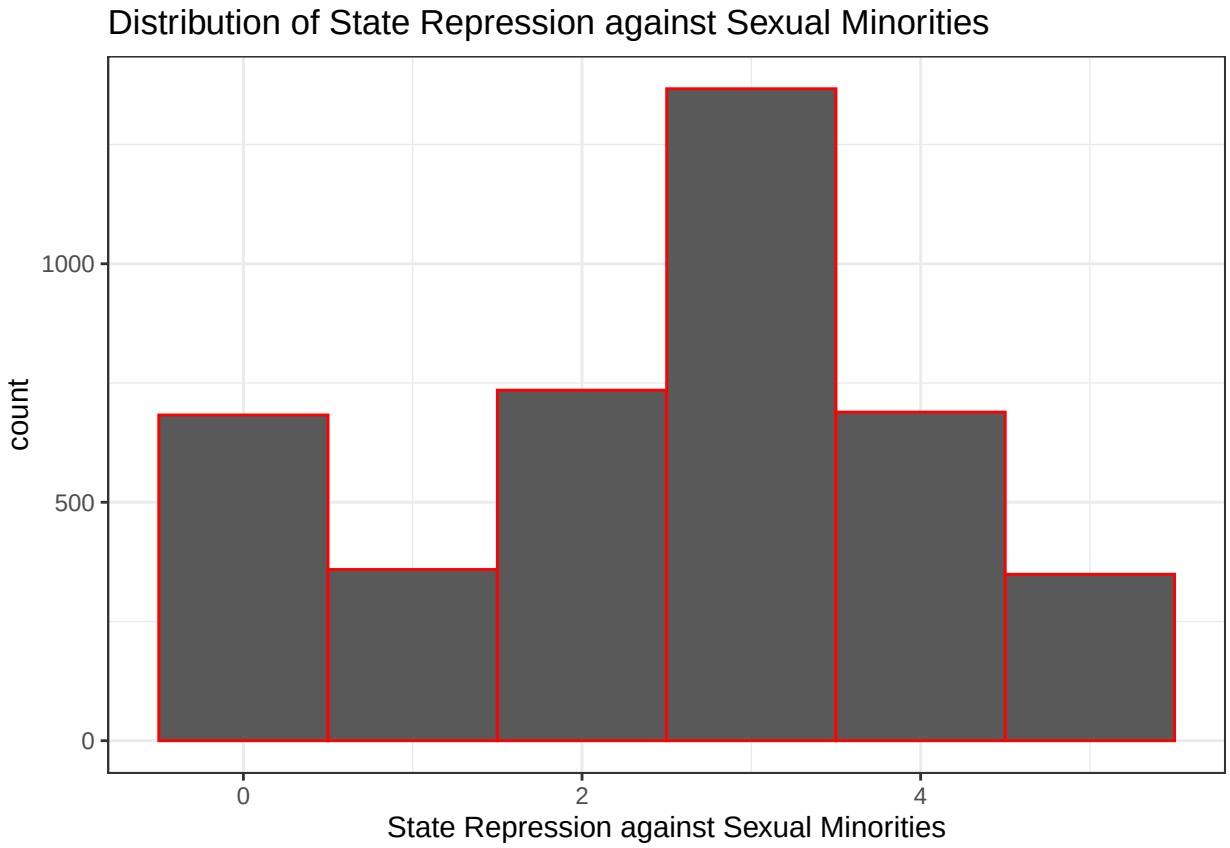
"Geom" (short for geometry) objects represent different types of graphical elements in a plot. They define how data points are displayed visually. The types of geoms include points, lines, bars, and more. Geom objects allow you to customize the visual representation of your data in a highly flexible and customizable manner. For example, geom_point() adds points to the plot(scatterplot), geom_line() adds lines. Here are some common Geom objects in ggplot2:

1) Geom_Point: Displays individual data points (Scatterplot). Good for two continuous variables
2) Geom_Line: Connects data points with lines, suitable for visualizing trends (Line graph).
3) Geom_Bar: Creates bar charts by displaying data as vertical or horizontal bars (Bar plot).
4) Geom_Histogram: Creates histograms to visualize the distribution of continuous data(Histogram).
5) Geom_Boxplot: Represents the distribution of data through quartiles, medians, and outliers (Boxplot).
6) Geom_Text: Adds text labels to specific data points on a plot.
7) Geom_Abline: Adds reference lines to a plot.
8) Geom_Ribbon: Creates ribbons that fill the area between two lines, used for visualizing confidence intervals.

Histograms: Visualizing Distributions

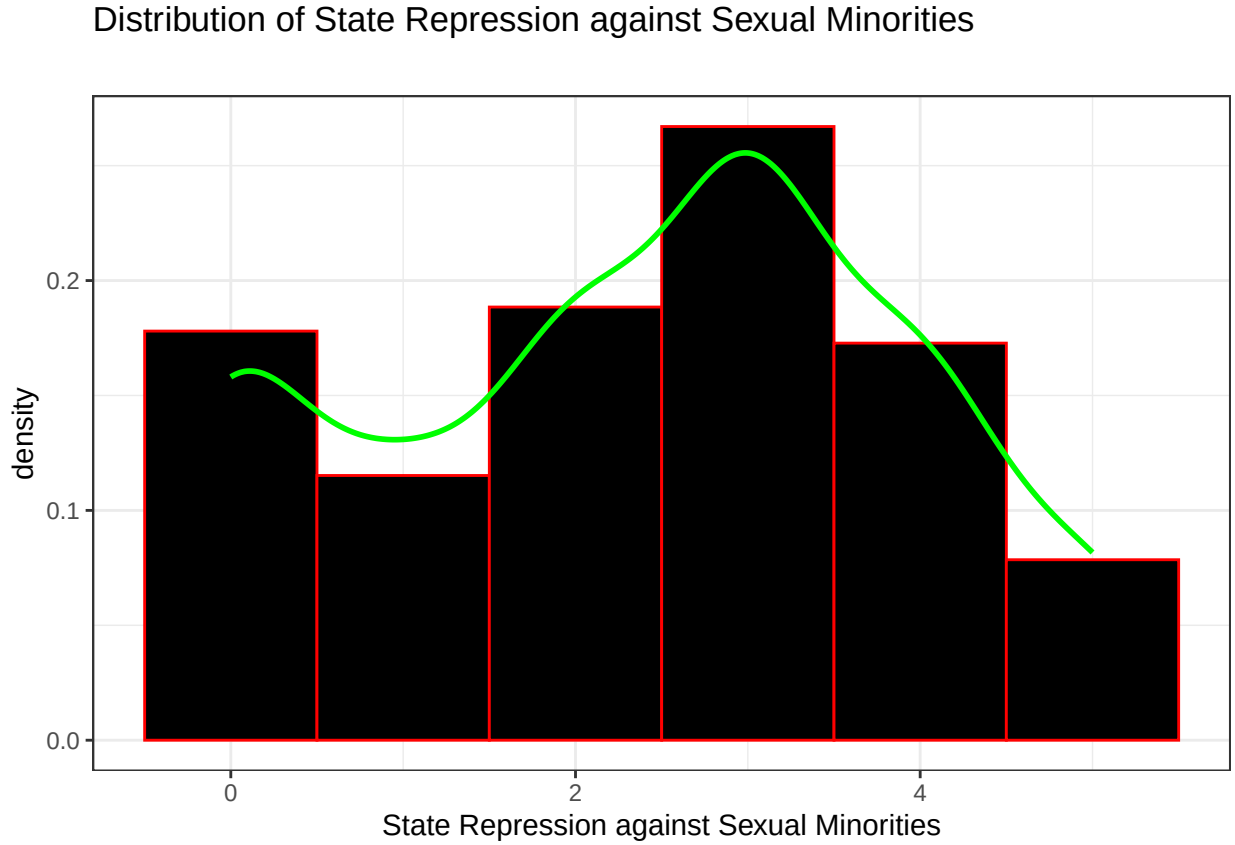
A histogram is a graphical representation of the distribution of data. It provides insights into the frequency/count of data points falling within specific intervals or "bins" along the range of the data. You can add geom_histogram() layer to create histograms. You can also create histograms using the hist() function.

```r
total2018 <- total |> #subseting the data
  filter(year==2018)


#Let's explore distribution of state repression against sexual minorities
h1<-ggplot(total, aes(x =dv_pts, )) +
  geom_histogram(binwidth =1, na.rm = T, color="red")+
  labs(title="Distribution of State Repression against Sexual Minorities",
       x="State Repression against Sexual Minorities")+
  theme_bw()
h1
```



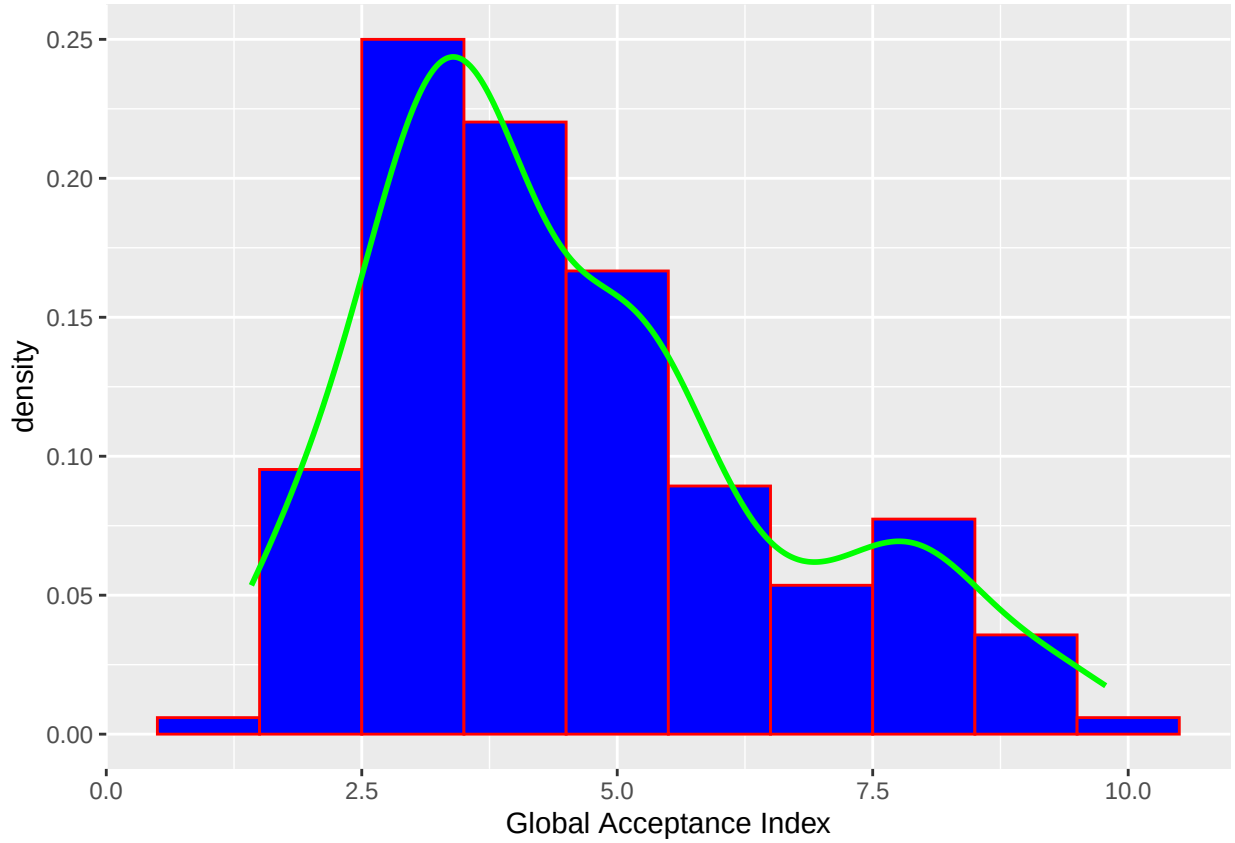

```r
#add probability density
h2<-ggplot(total2018, aes(x =dv_pts, after_stat(density))) +
  geom_histogram(binwidth =1, na.rm = T, color="red", fill="black")+
  labs(title="Distribution of State Repression against Sexual Minorities",
       x="State Repression against Sexual Minorities", subtitle = "")+
  theme_bw()+
  geom_density(color = "green", linewidth = 1)
h2
```

```
## Warning: Removed 7 rows containing non-finite values (‘stat_density()‘).
```



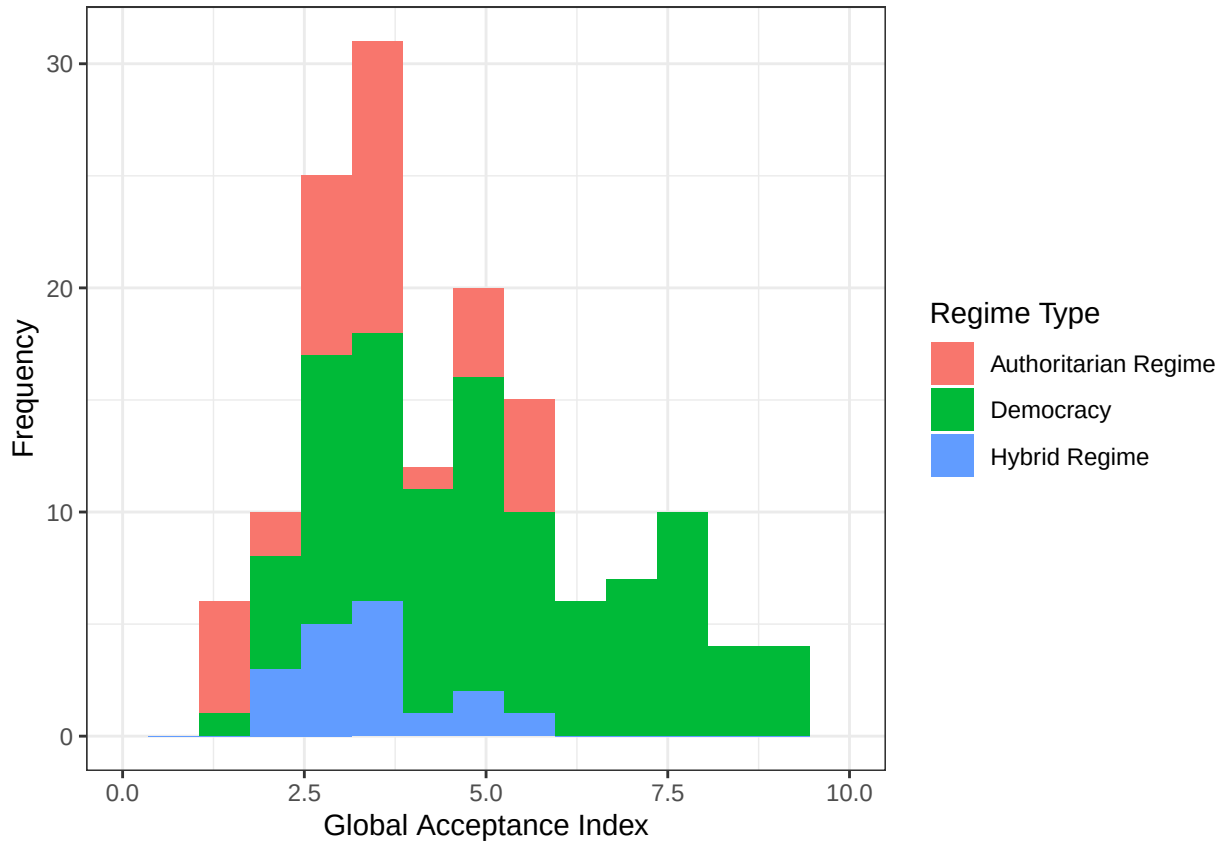

```
#Let's explore distribution of Global Acceptance Index in 2018
h3<-total2018|>
ggplot(aes(x = gaiscore, after_stat(density))) +
  geom_histogram(binwidth =1, na.rm = T, color="red", fill="blue")+
  labs(x="Global Acceptance Index")+
  geom_density(color = "green", linewidth = 1)
h3
```

```
## Warning: Removed 30 rows containing non-finite values (‘stat_density()‘).
```

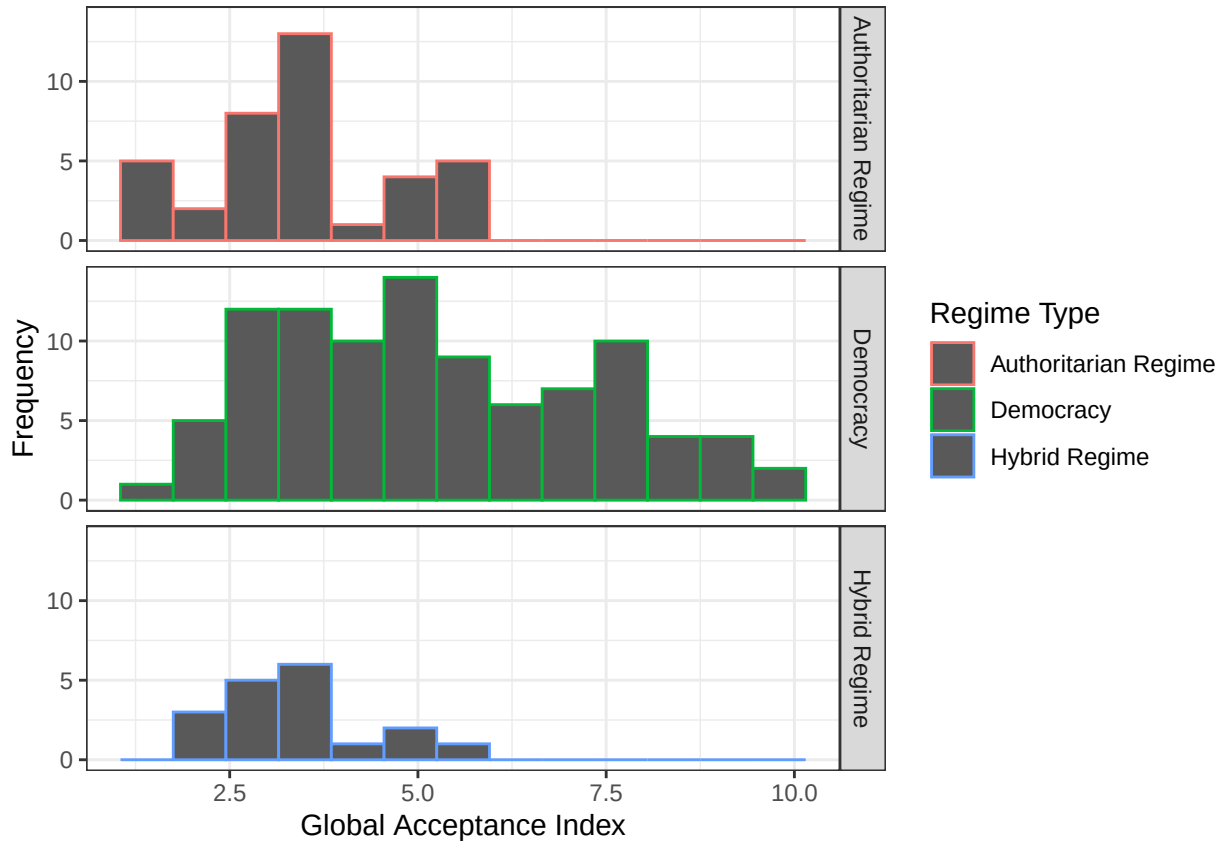



```
#Add a third variable using fill argument
h4<-ggplot(total2018|>
             filter(!is.na(gaiscore),
                    !is.na(regime_status_name)),
           aes(x = gaiscore, fill=regime_status_name, na.rm = T)) +
  geom_histogram(binwidth =.7)+
  labs(x="Global Acceptance Index", fill="Regime Type", y="Frequency")+
  xlim(0, 10)+
  theme_bw()
h4
```

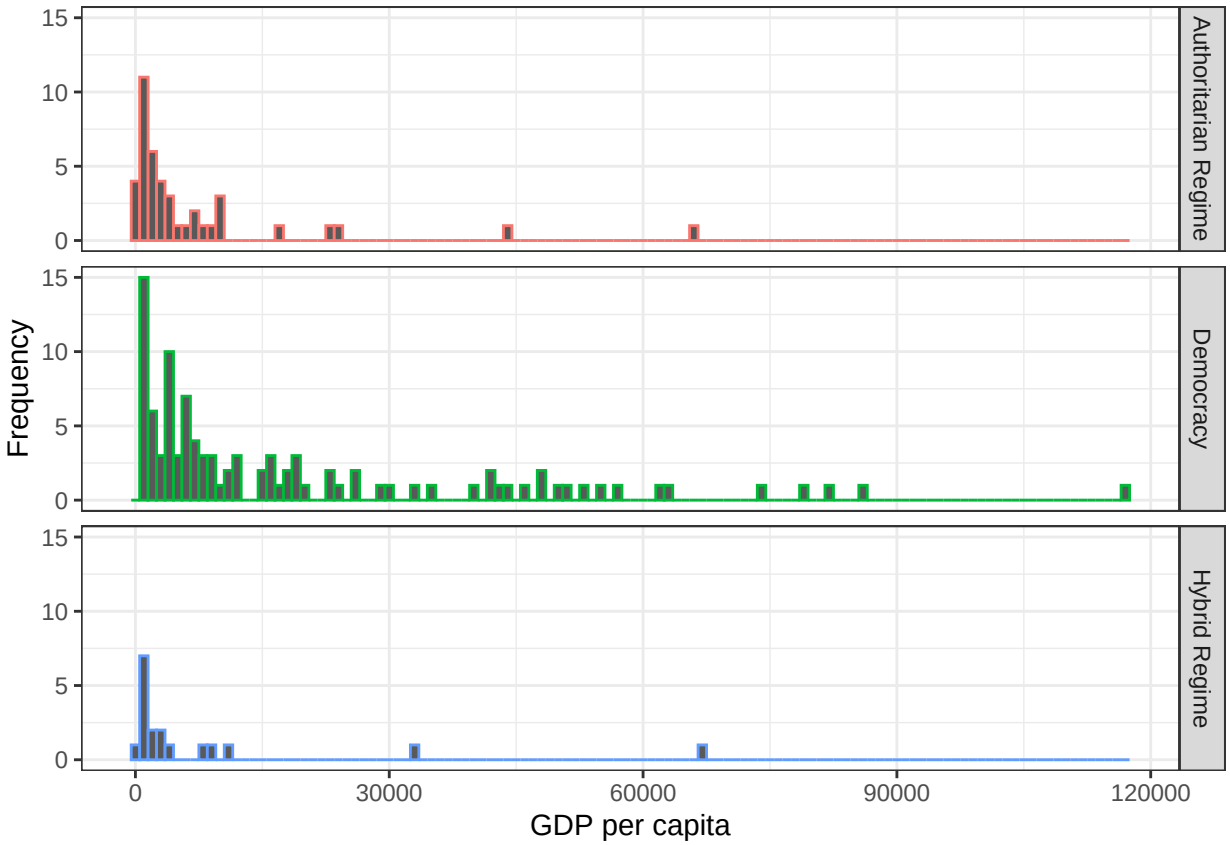
```
## Warning: Removed 6 rows containing missing values (‘geom_bar()‘).
```




```
#Using facets
h5<-ggplot(total2018|>
             filter(!is.na(gaiscore),
                    !is.na(regime_status_name)),
           aes(x = gaiscore, na.rm = T, color=regime_status_name)) +
  geom_histogram(binwidth =.7)+
  labs(x="Global Acceptance Index", y="Frequency", color="Regime Type")+
  facet_grid(vars(regime_status_name))+
  theme_bw()
h5
```



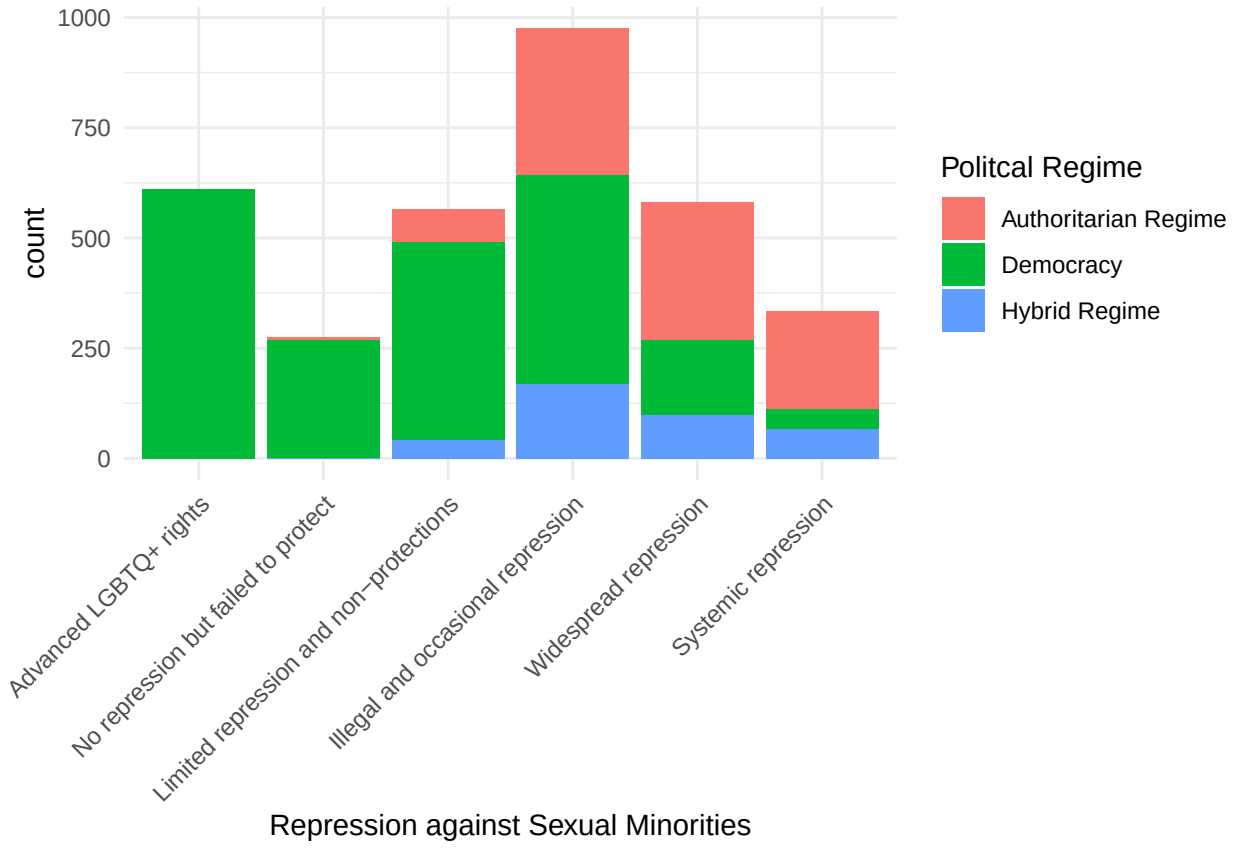

```
#Using facets
h6<-ggplot(total2018|>
             filter(!is.na(gdp),
                    !is.na(regime_status_name)),
           aes(x =gdp, na.rm = T, color=regime_status_name, fil=regime_status_name )) +
  geom_histogram(binwidth =1000)+
  labs(x="GDP per capita",
       y="Frequency",
       color="Regime Type")+
  facet_grid(vars(regime_status_name))+
  theme_bw()+
  guides(color = "none")+
  theme(legend.position = "bottom")
h6
```



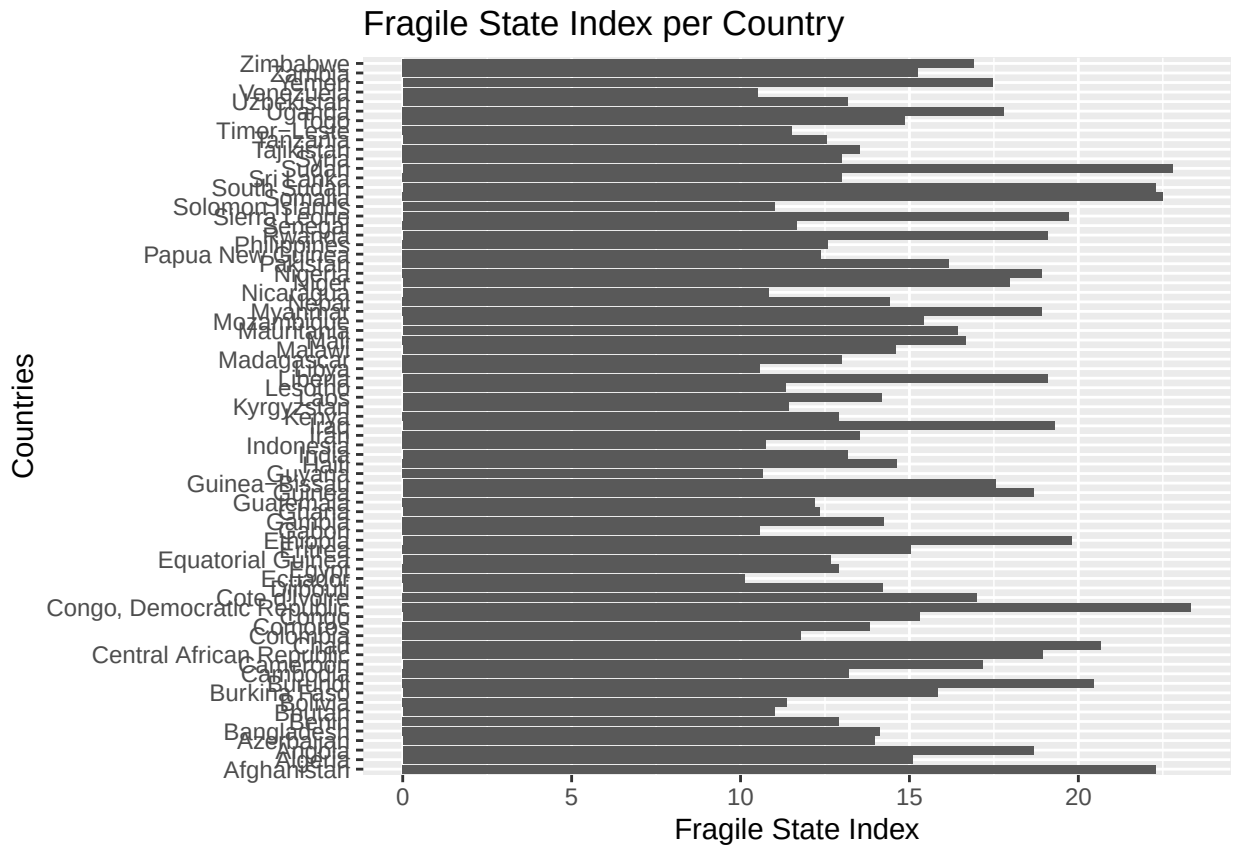
Bar Plot

A bar plot (aka a bar chart or bar graph) is a common type of data visualization used to represent categorical data. It displays the frequency, count, or proportion of data points within different categories. Each category corresponds to a bar and the length of the bar is proportional to the value it represents.

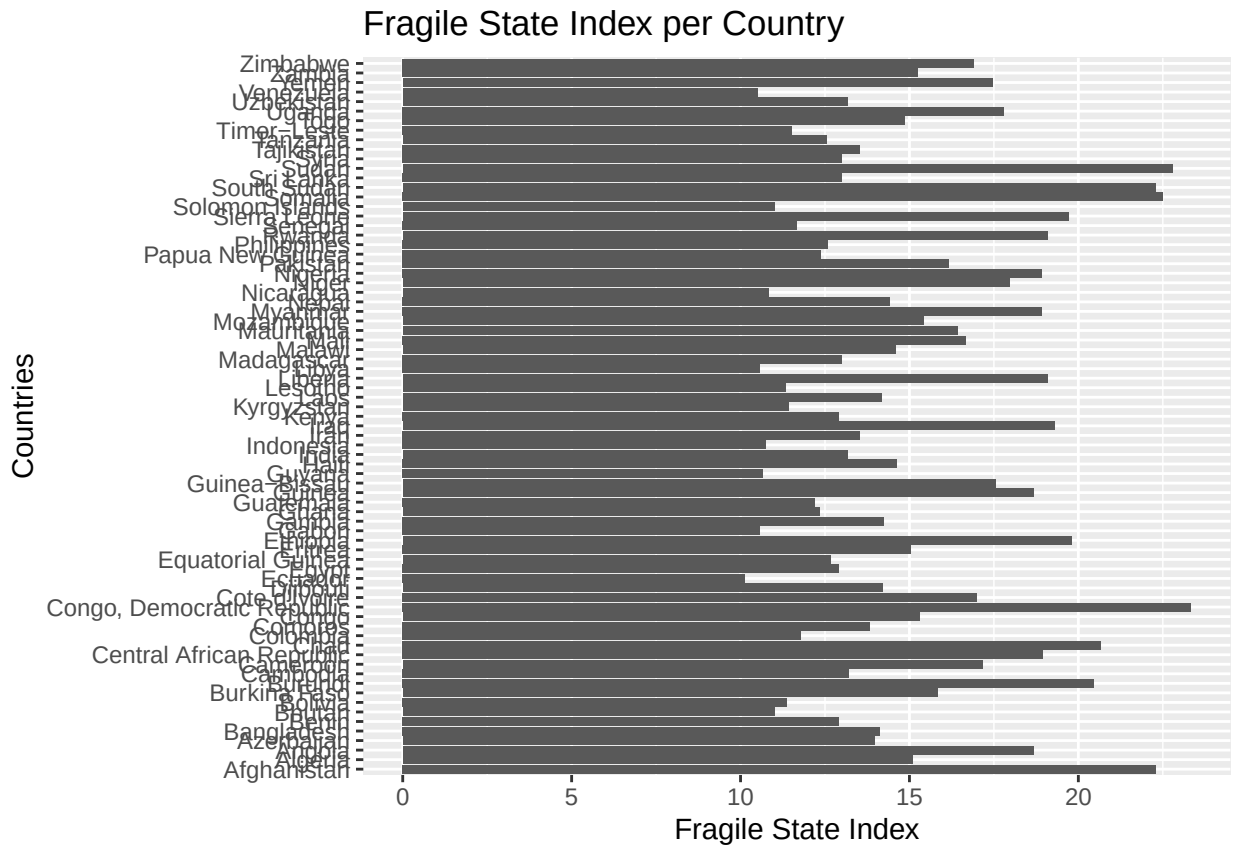
```
# The state repression against sexual minorities
total$rep_text<-recode_factor(total$dv_pts, #first create new variable
                              '0'="Advanced LGBTQ+ rights",
                              '1'="No repression but failed to protect",
                              '2'="Limited repression and non-protections",
                              '3'="Illegal and occasional repression",
                              '4'="Widespread repression",
                              '5'="Systemic repression")

b1<-ggplot(total |>
             filter(!is.na(rep_text),
                    !is.na(regime_status_name))
         , aes(x = rep_text, fill=regime_status_name)) +
  geom_bar() +
  labs(fill = "Politcal Regime",
       x="Repression against Sexual Minorities") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
b1
```



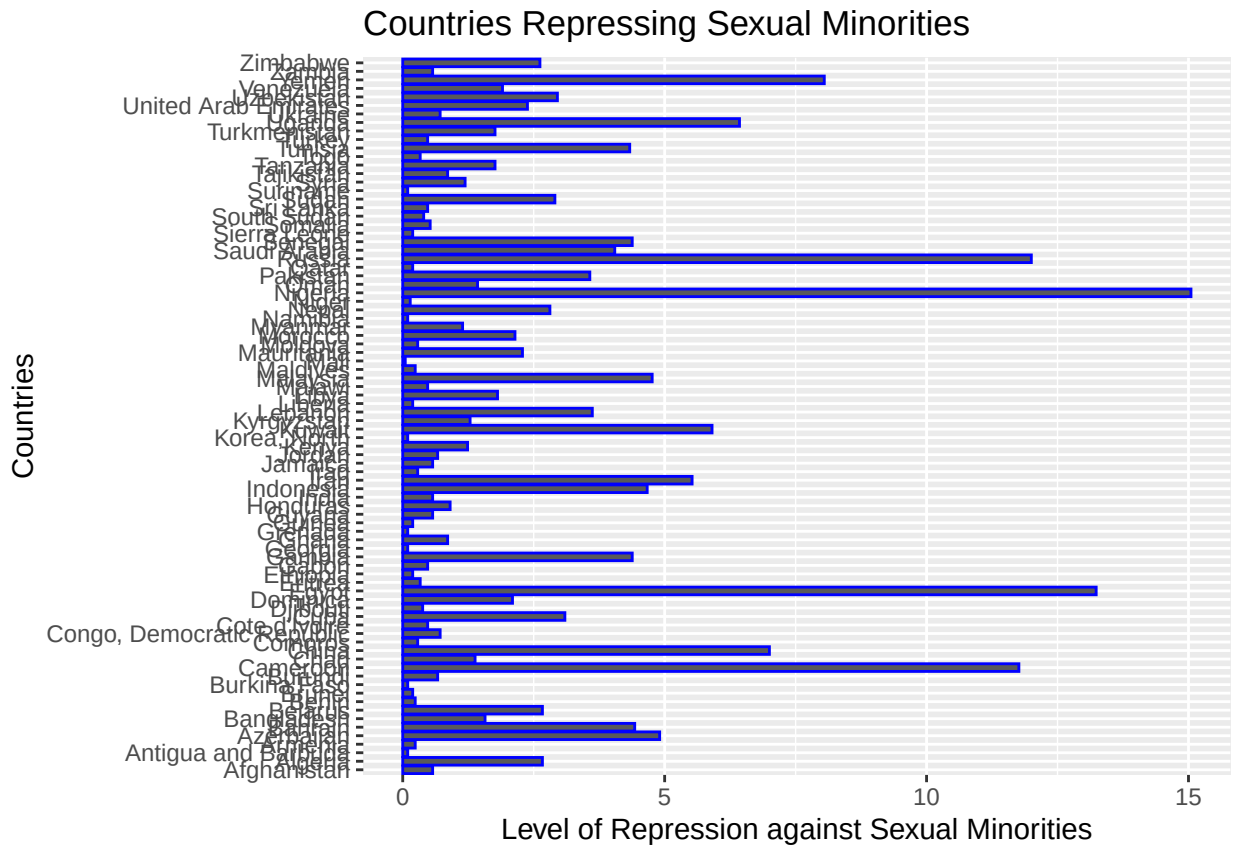

```
#Fragile State Index(FSI)
sfi<-total|> #Data transformation
  group_by(country) |>  #grouping the data by country
  summarise(m_sfi=mean(sfi, na.rm = T), n=n()) |> #the mean of FSI
  arrange(desc(m_sfi)) |> #arranging in descending order
  na.omit() |> #deleting missing observations
  filter(m_sfi >10) #choosing rows with a higher fragile  index(>10).

b2<-sfi |> #visualize it using geom_bar
ggplot(aes(y=m_sfi, x=country))+
  geom_bar(stat = "identity")+
  labs(title="Fragile State Index per Country",
       x="Countries",
       y = "Fragile State Index", width = 0.1)+
  coord_flip()
b2
```




```
b3<-ggplot()+ #you can alternatively use geom_col
  geom_col(mapping =aes(y=m_sfi, x=country), data = sfi, position = "stack")+
  labs(title="Fragile State Index per Country",
       x="Countries",
       y = "Fragile State Index", width = 0.1)+
  coord_flip()
b3
```




```r
#Which countries repress sexual minorities?
total$rep_sexmin<-total$arrest+
  total$imprison+
  total$torture+
  total$execution+
  total$disappear+
  total$extra_kill                                    #create a new variable

LGBT_rep <- total |>                                  # data transformation
  group_by(country) |>                                #grouping by country
  summarise(rep_m=mean(rep_sexmin, na.rm = T), n=n()) |> #mean repression
  na.omit()|>                                         #deleting NAs
  filter(rep_m > 0)                                   #rows with > 0

b4<-ggplot(LGBT_rep, aes(x =country, y =rep_m)) +
  geom_bar(stat = "identity", color="blue") +
  coord_flip()+
  labs(title="Countries Repressing Sexual Minorities",
       x="Countries",
       y = "Level of Repression against Sexual Minorities", width = 0.1)
b4
```



Scatter Plot

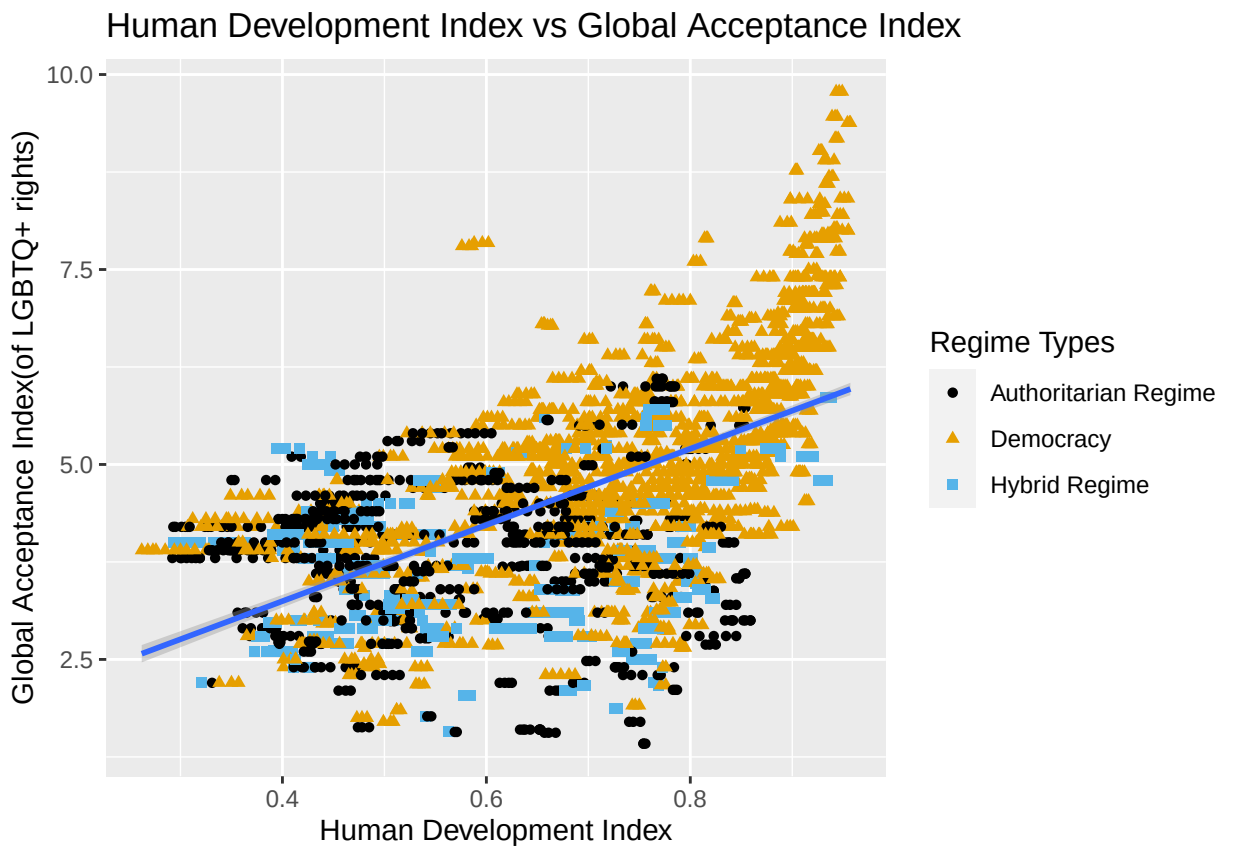
A scatter plot is created to display individual data points as dots on a two-dimensional coordinate system. Each dot represents a single observation in the dataset. The position of a dot is determined by the values of two variables: the x-axis variable and the y-axis variable. Scatter plots are used to visualize the relationship/correlation between two continuous variables. They are useful to identify patterns, trends, and outliers in the data.Scatter plots can help answer these questions:

- Is there a linear relationship between the two variables?
- Are the variables positively or negatively correlated?
- Are there any outliers that deviate from the general pattern?
- Are there any clusters or patterns in the data points?

```
#Human Development Index(HDI) vs Liberal Democracy Index
s1<-ggplot(
  data = total |>
    filter(!is.na(undp_hdi),
           !is.na(gaiscore),
           !is.na(regime_status_name)),
  mapping = aes(x =undp_hdi, y =gaiscore)
) +
  geom_point(aes(color = regime_status_name, shape = regime_status_name)) +
  geom_smooth(method="lm") +
```

```
  labs(
    title = "Human Development Index vs Global Acceptance Index",
    x = "Human Development Index",
    y = "Global Acceptance Index(of LGBTQ+ rights)",
    color = "Regime Types",
    shape = "Regime Types"
  ) +
  scale_color_colorblind()
s1
```

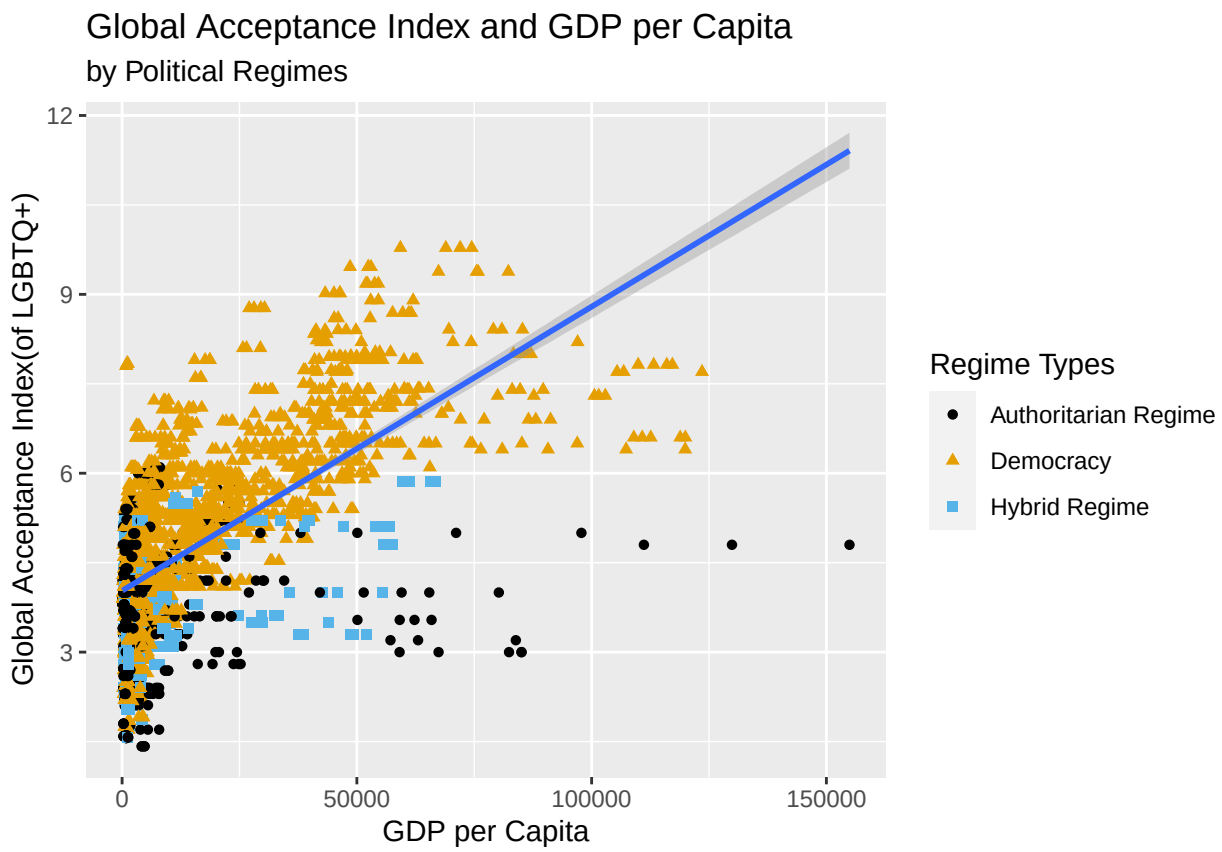
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#How about relationship between GDP per Capita and Global Acceptance Index.
#Is GDP per capita a good measure of development?
#Which authoritarian countries have higher GDP per capita
#but relatively lower acceptance index?
s2<-ggplot(
  data =total |>
    filter(!is.na(gaiscore),
           !is.na(gdp),
           !is.na(regime_status_name)),
  mapping = aes(x = gdp, y = gaiscore)
) +
  geom_point(aes(color = regime_status_name, shape = regime_status_name)) +
  geom_smooth(method="lm") +
```

```
  labs(
    title = "Global Acceptance Index and GDP per Capita",
    subtitle = "by Political Regimes",
    x = "GDP per Capita",
    y = "Global Acceptance Index(of LGBTQ+)",
    color = "Regime Types",
    shape = "Regime Types"
  ) +
  scale_color_colorblind()
s2
```

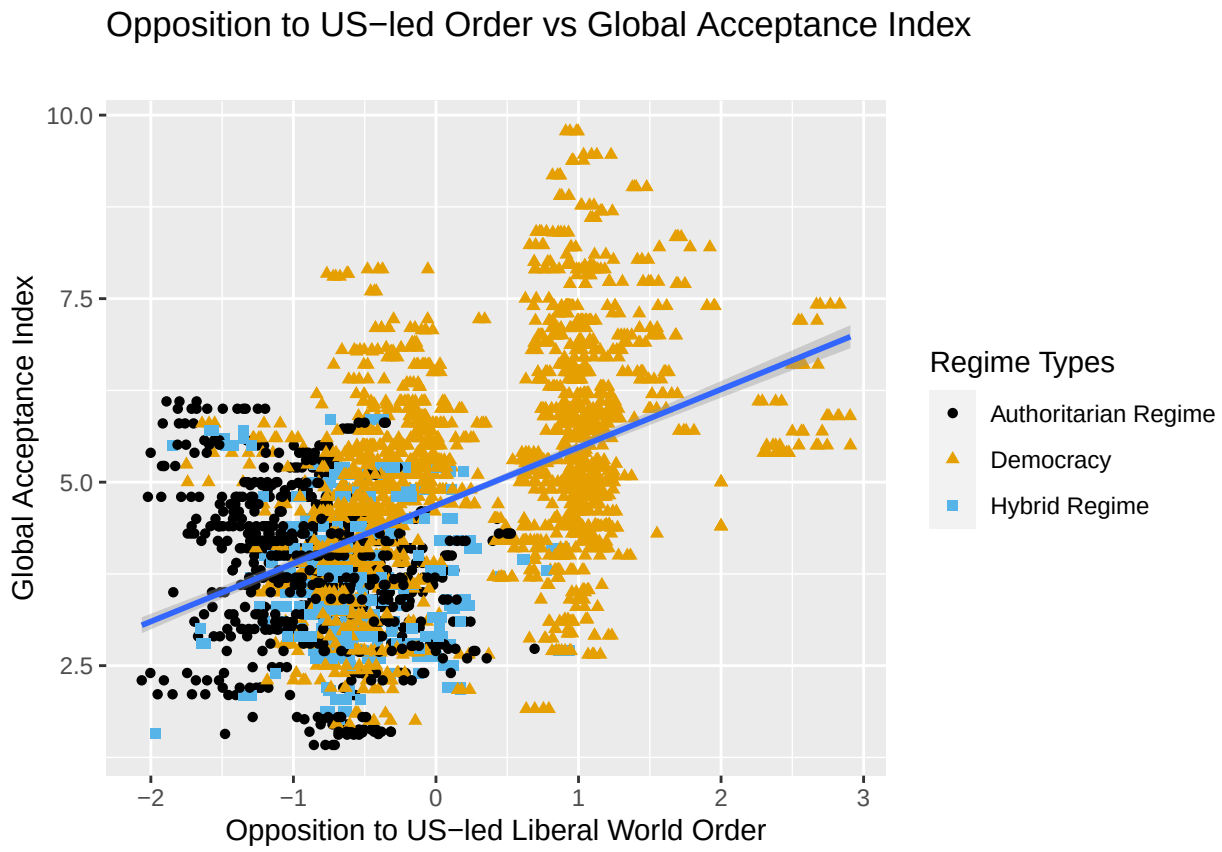
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#How about the Opposition to US-led Liberal World Order? Russia?
s3<-ggplot(
  data = total |>
    filter(!is.na(IdealPointAll),
           !is.na(gaiscore),
           !is.na(regime_status_name)),
  mapping = aes(x =IdealPointAll, y =gaiscore)
) +
  geom_point(aes(color = regime_status_name, shape =regime_status_name)) +
  geom_smooth(method="lm") +
  labs(
    title = "Opposition to US-led Order vs Global Acceptance Index",
```

```
    subtitle = "",
    x = "Opposition to US-led Liberal World Order",
    y = "Global Acceptance Index",
    color = "Regime Types",
    shape = "Regime Types"
  ) +
  scale_color_colorblind()
s3
```

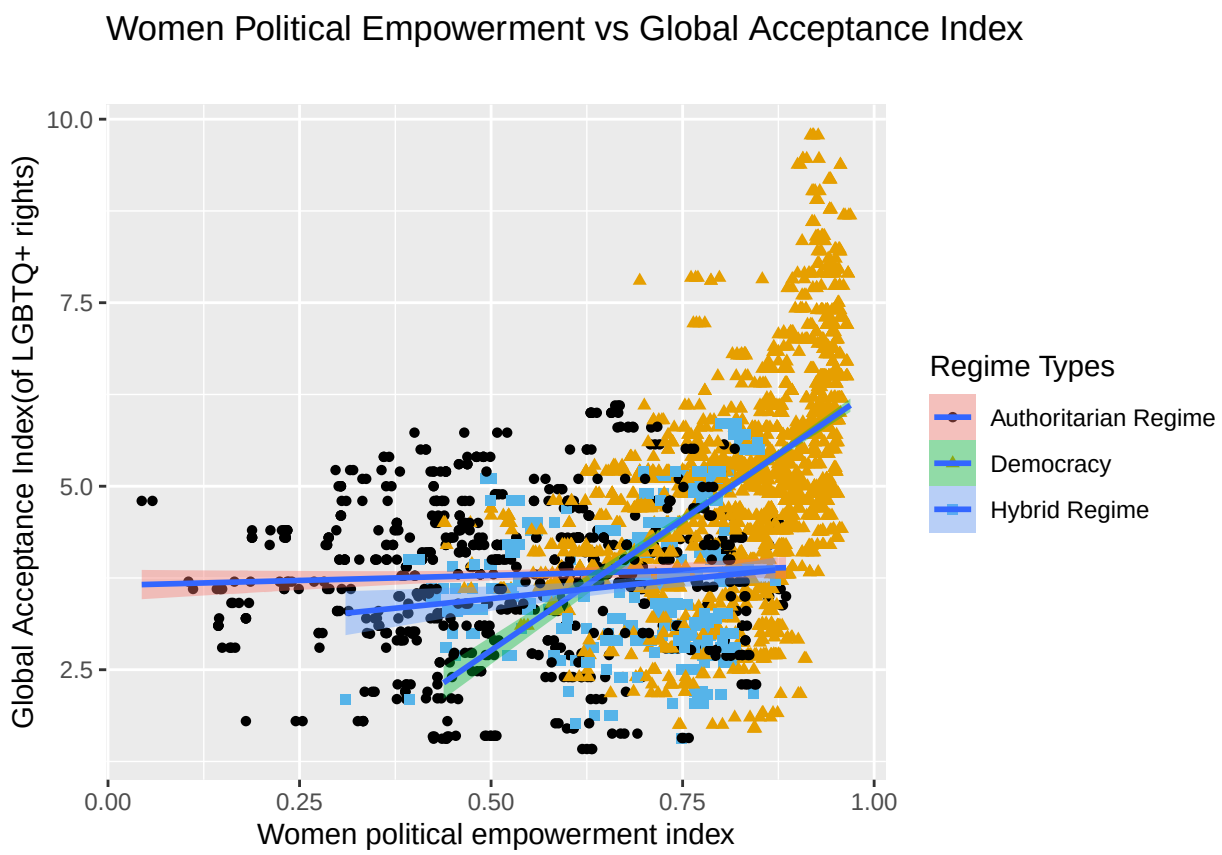
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#Any parallels between Global Acceptance Index and Women Empowerment Index?
gen_par<-total[c("gaiscore", "vdem_gender","regime_status_name" )] |>
  na.omit(gen_par)

s4<-ggplot(
  data = gen_par,
  mapping = aes(x =vdem_gender, y = gaiscore, fill=regime_status_name)
) +
  geom_point(aes(color =regime_status_name, shape =regime_status_name)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Women Political Empowerment vs Global Acceptance Index",
    subtitle = "",
    x = "Women political empowerment index",
```

```
    y = "Global Acceptance Index(of LGBTQ+ rights)",
    color = "Regime Types",
    shape = "Regime Types",
    fill="Regime Types"
  ) +
  scale_color_colorblind()
s4 #Why do we get three lines?
```

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



Line Graph

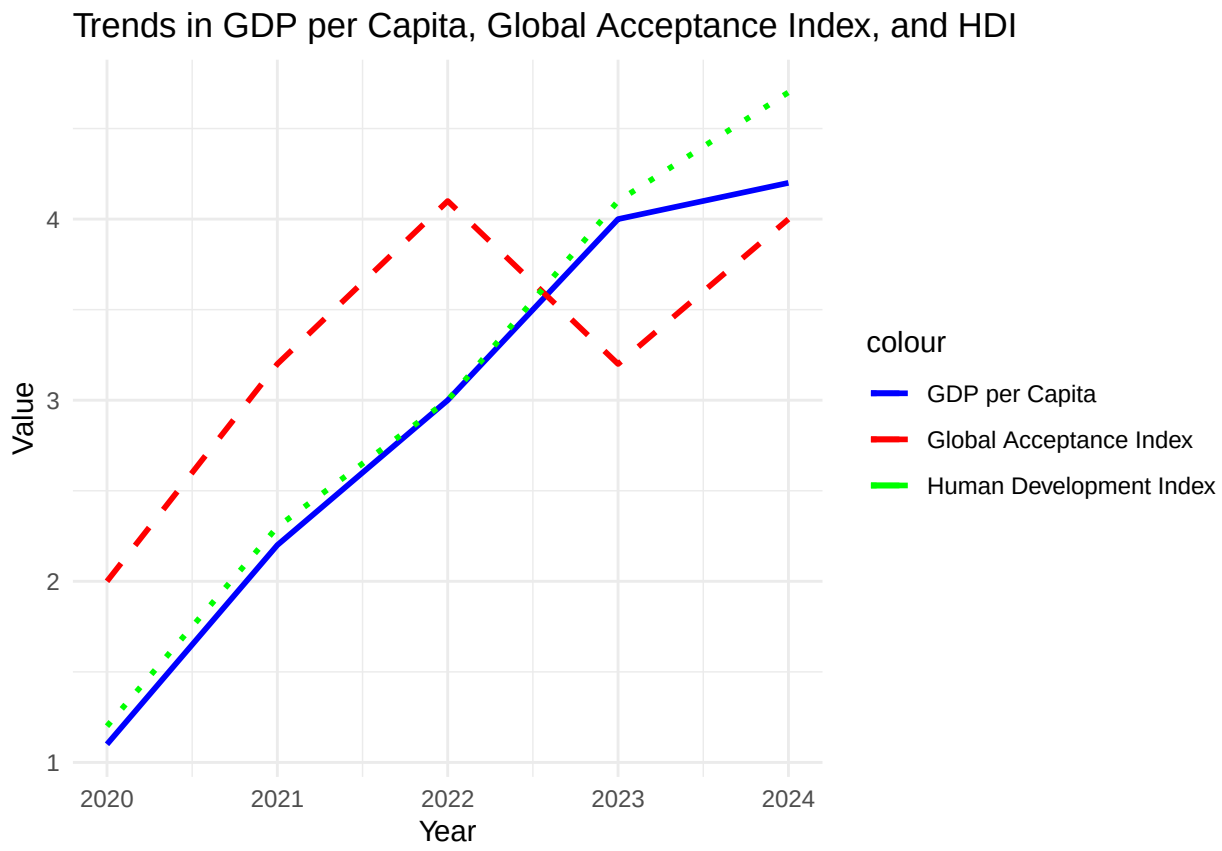
A line graph, also known as a line plot or line chart, is a type of data visualization that displays data points as individual markers connected by straight lines. Line graphs are commonly used to represent trends and patterns over continuous or sequential data, such as time series data. In ggplot2, you can create line graphs using the geom_line() function.

```
# A line graph based on fake data
year <- c(2020, 2021, 2022, 2023, 2024)
m_gdp_scaled <- c(1.1, 2.2, 3, 4, 4.2)
m_gai <- c(2, 3.2, 4.1, 3.2, 4)
m_hdi <- c(1.2, 2.3, 3, 4.1, 4.7)

# Create the line graph
```

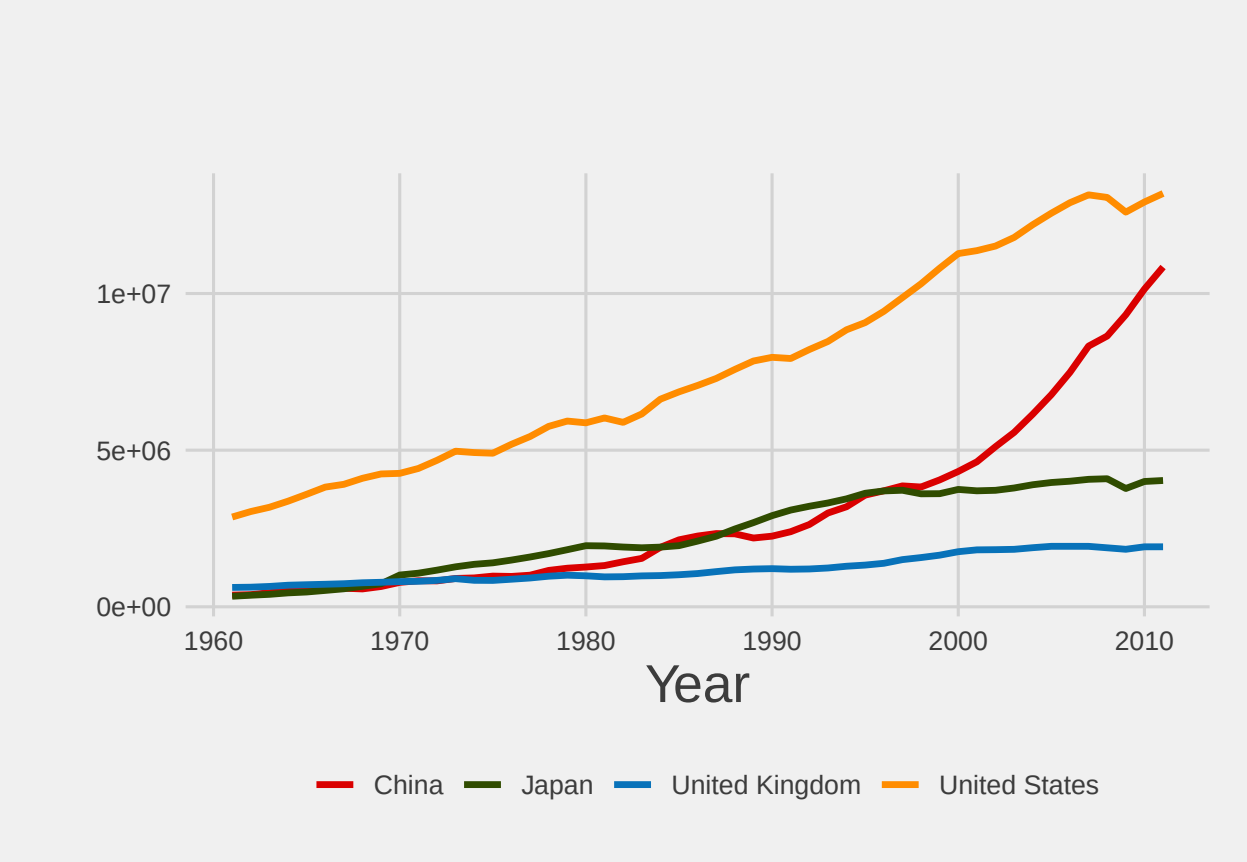
```r
mydata <- data.frame(year, m_gdp_scaled, m_gai, m_hdi)
l1<- ggplot(mydata, aes(x = year)) +
  geom_line(aes(y = m_gdp_scaled,
                color = "GDP per Capita"),
            linetype = "solid", size = 1) +
  geom_line(aes(y = m_gai,
                color = "Global Acceptance Index"),
            linetype = "dashed", size = 1) +
  geom_line(aes(y = m_hdi,
                color = "Human Development Index"),
            linetype = "dotted", size = 1) +
  labs(x = "Year",
       y = "Value",
       title = "Trends in GDP per Capita, Global Acceptance Index, and HDI") +
  scale_linetype_manual(values = c("solid", "dashed", "dotted")) +
  scale_color_manual(values = c("GDP per Capita" = "blue",
                                "Global Acceptance Index" = "red",
                                "Human Development Index" = "green")) +
  theme_minimal() # Add labels to the lines manually
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
l1
```

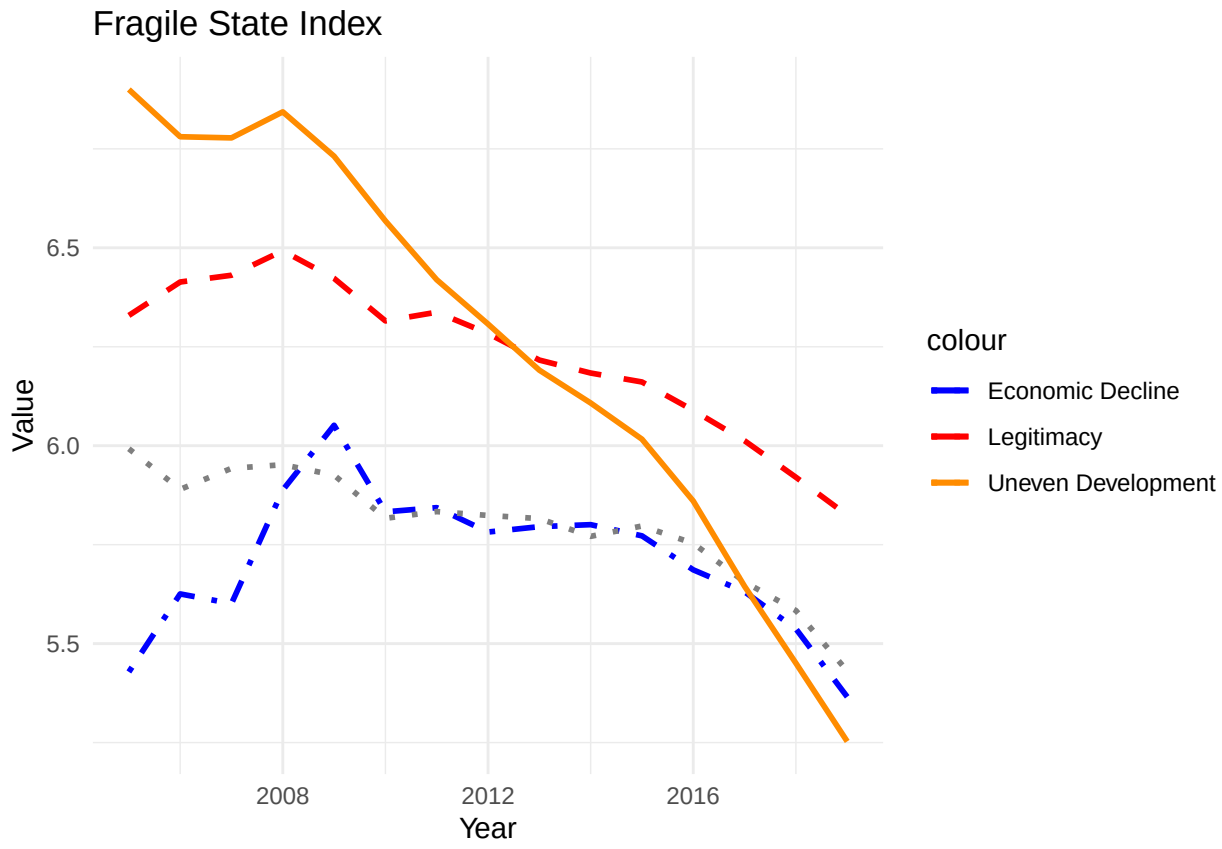



```
#Economic indicators of selected countries over time: gdp of several countries.
newdata <-filter(total,country %in% c('China','United Kingdom','United States','Japan'),
                  year > 1960)|>
  select("country", "polity", "year", "gle_gdp")
colors= c("#DA0000", "#304c00","#0872b9", "darkorange")

l2<-ggplot(newdata |>filter(!is.na(year),
                            !is.na(gle_gdp),
                            !is.na(country)),
           aes(year, gle_gdp, color =  country)) +
  geom_line(linewidth =1.2) +
  labs(title = "",
       subtitle = "",
       x = "Year",
       y = "",
       color = "") +
  scale_color_manual(values = colors)+
  ggthemes::theme_fivethirtyeight() +
theme(
plot.title = element_text(size = 30),
      axis.title.y = element_text(size = 20),
      axis.title.x = element_text(size = 20))
l2
```



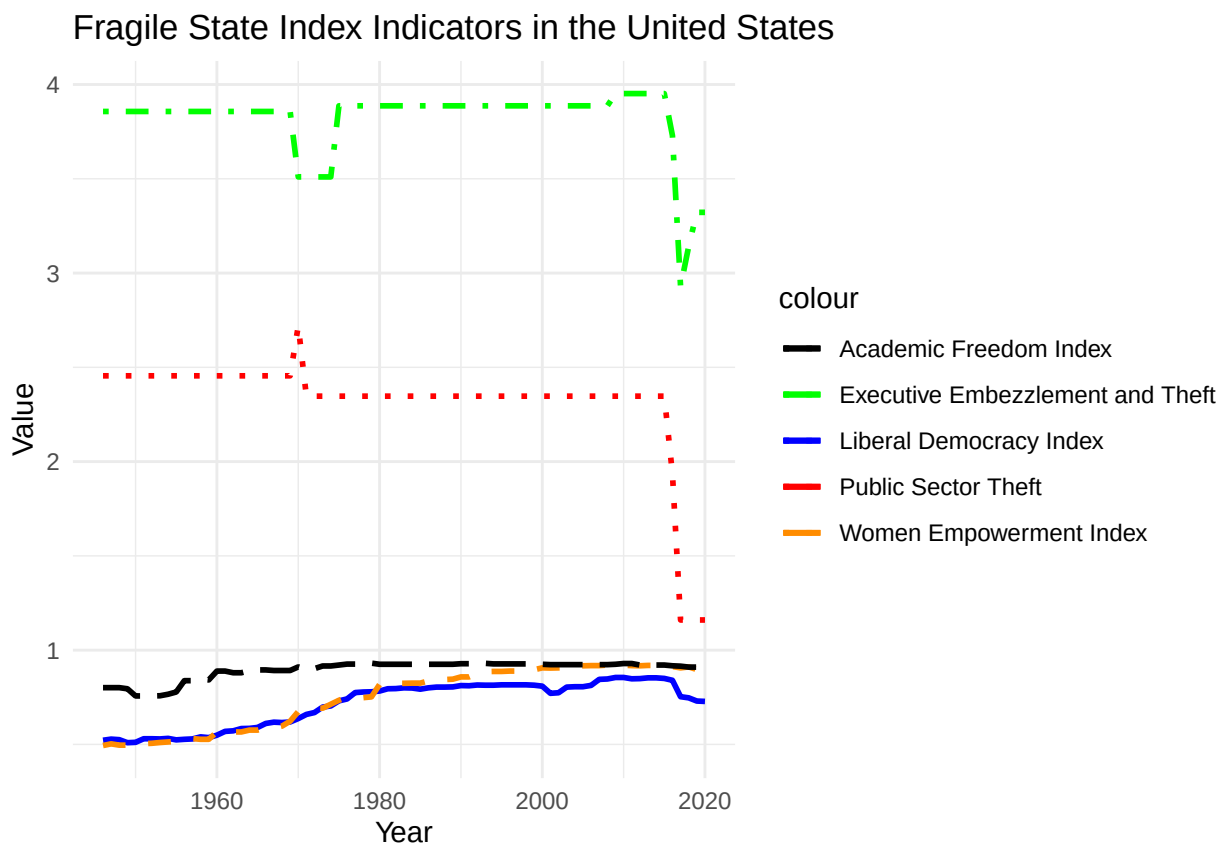

```
#explore changes in certain variables over year.
newdat<-total|>
  group_by(year)|>
  summarise(m_eco=mean(ffp_eco, na.rm = T),
            m_sl=mean(ffp_sl, na.rm = T),
            m_hr=mean(ffp_hr, na.rm = T),
            m_ued=mean(ffp_ued, na.rm = T) )|>
  na.omit()


l3<-ggplot(newdat, aes(x = year)) +
  geom_line(aes(y = m_eco,
                color = "Economic Decline"),
                linetype = "dotdash", size = 1) +
  geom_line(aes(y = m_sl,
                color = "Legitimacy"),
                linetype = "dashed", size = 1) +
  geom_line(aes(y = m_hr,
                color = "Human Rights "),
                linetype = "dotted", size = 1) +
  geom_line(aes(y = m_ued,
                color = "Uneven Development"),
                linetype = "solid",size=1) +
  labs(x = "Year",
       y = "Value",
       title = "Fragile State Index") +
```

```
  scale_linetype_manual(values = c("dotdash", "dashed", "dotted", "solid")) +
  scale_color_manual(values = c("Economic Decline" = "blue",
                                "Legitimacy" = "red",
                                "Human Rights" = "green",
                                "Uneven Development" = "darkorange")) +
  theme_minimal()
13
```




```
#changes in a single country: United States
usa<-total |>
  select(c("vdem_libdem",
           "vdem_gender",
           "vdem_exthftps",
           "vdem_exembez",
           "vdem_academ",
           "year",
           "country"))|>
  filter(country=="United States") |>
  na.omit()


14<-ggplot(usa, aes(x = year)) +
  geom_line(aes(y = vdem_libdem,
                color = "Liberal Democracy Index"),
                linetype = "solid", size = 1) +
```

```
  geom_line(aes(y = vdem_gender,
                color = "Women Empowerment Index"),
                linetype = "dashed", size = 1) +
  geom_line(aes(y = vdem_exthftps,
                color = "Public Sector Theft"),
                linetype = "dotted", size = 1) +
  geom_line(aes(y = vdem_exembez,
                color = "Executive Embezzlement and Theft"),
                linetype = "dotdash", size = 1) +
  geom_line(aes(y = vdem_academ,
                color = "Academic Freedom Index"),
                linetype = "longdash", size = 1) +
  labs(x = "Year",
       y = "Value",
       title = "Fragile State Index Indicators in the United States") +
  scale_linetype_manual(values = c("solid", "dashed", "dotted", "dotdash", "longdash")) +
  scale_color_manual(values = c("Liberal Democracy Index" = "blue",
                                "Women Empowerment Index" = "darkorange",
                                "Public Sector Theft" = "red",
                                "Executive Embezzlement and Theft" = "green",
                                "Academic Freedom Index"="black")) +
  theme_minimal()
14
```

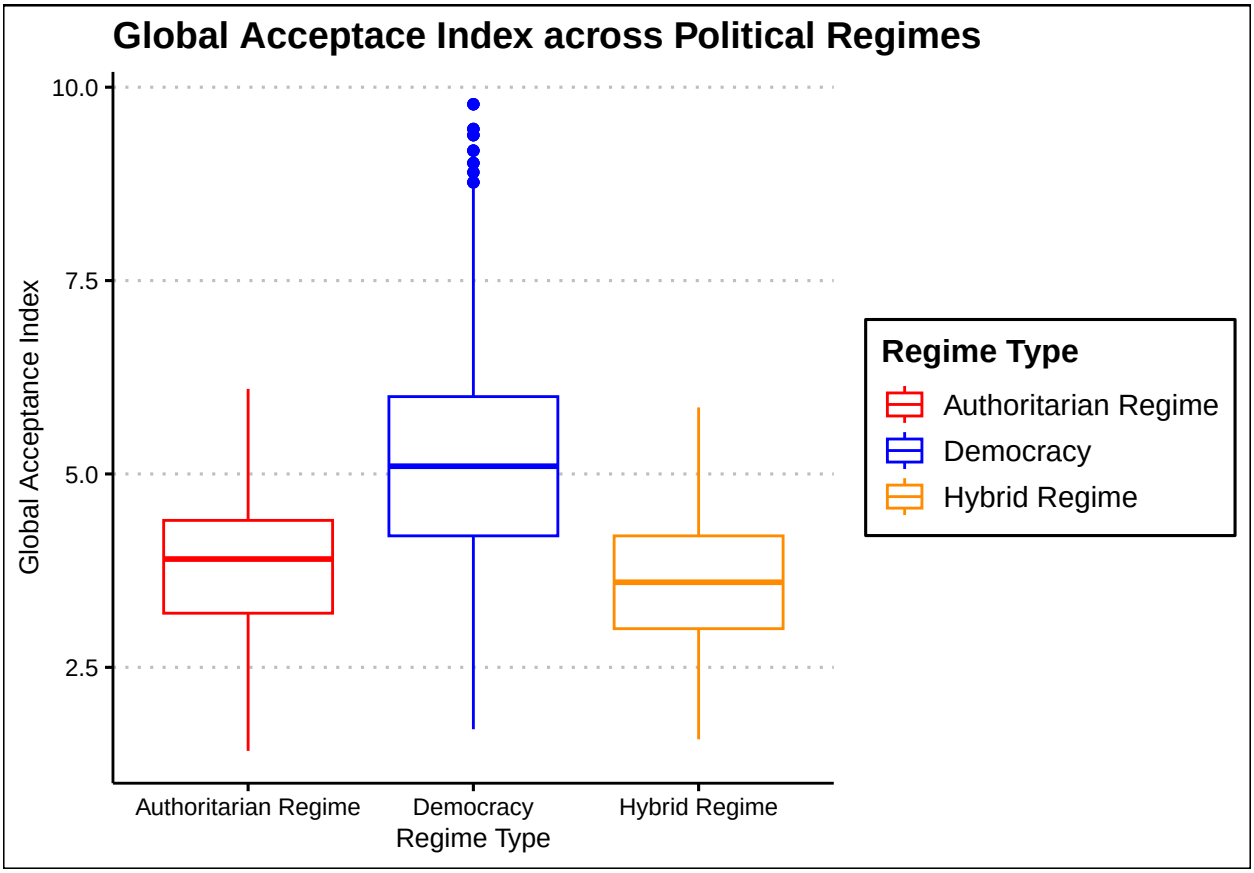


Box Plot

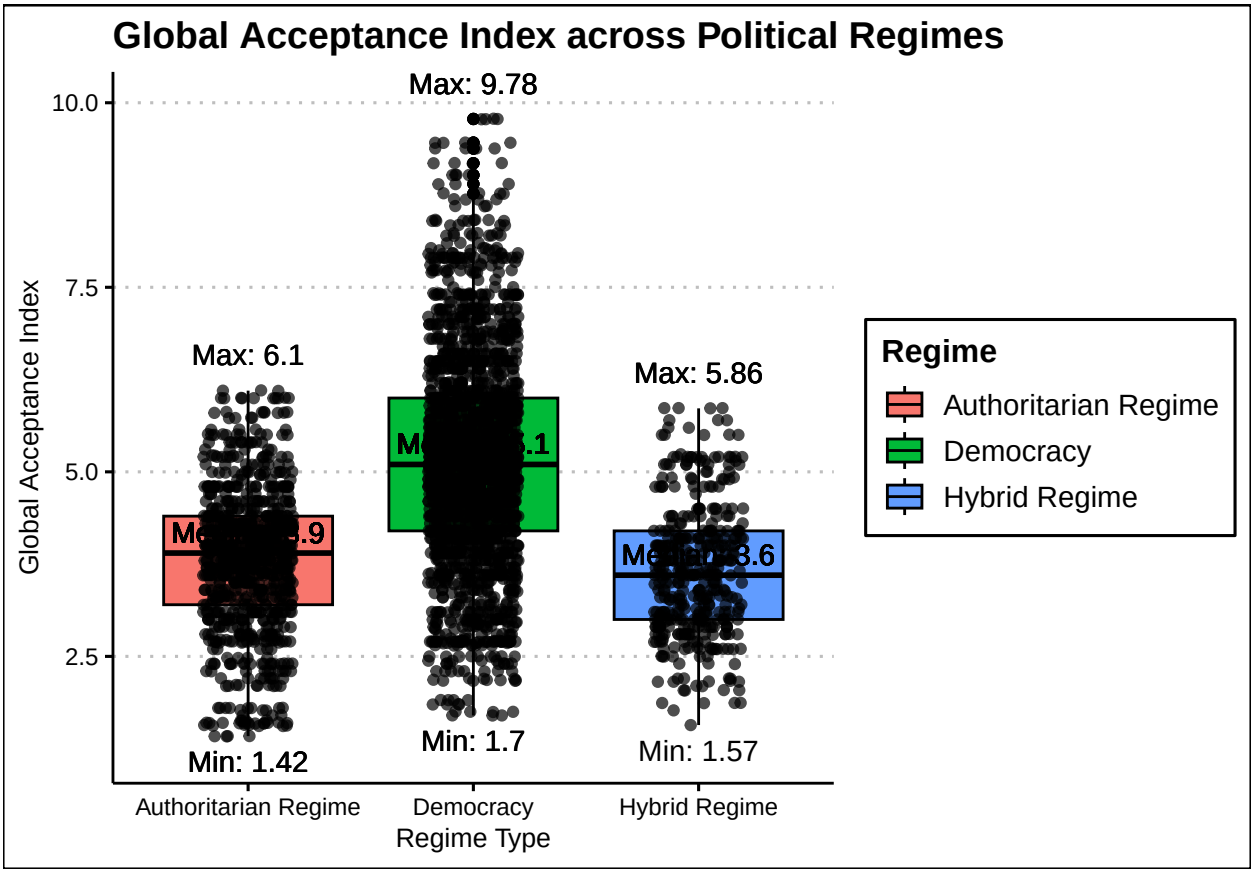
A box plot(aka a box-and-whisker plot) displays summary statistics about the central tendency, spread, and skewness of the data. The box plot is useful for identifying outliers and comparing distributions across different categories or groups. The key components of box plot:

- Box: The central box represents the interquartile range (IQR). It contains the middle 50% of the data. The top and bottom edges of the box represent the upper and lower quartiles (Q3 and Q1, respectively), while the line inside the box represents the median (Q2).
- Whiskers: Lines extending from the box (whiskers) typically reach out to a maximum of 1.5 times the IQR. Data points beyond this range are often considered outliers and are plotted individually as dots.
- Outliers: Data points that fall beyond the whiskers are considered outliers. They are plotted individually as dots.

```
#A basic boxplot: Global Acceptance Index across Political Regimes
b1<-total |>
  filter(!is.na(regime_status_name),
         !is.na(gaiscore)) |>
  ggplot(aes(x=regime_status_name, y=gaiscore, color=regime_status_name))+
  geom_boxplot()+
  scale_color_manual(values = c("red", "blue", "darkorange"))+
  labs(title="Global Acceptace Index across Political Regimes",
       x="Regime Type",
       y="Global Acceptance Index",
       color="Regime Type")+
  theme_clean()
b1
```

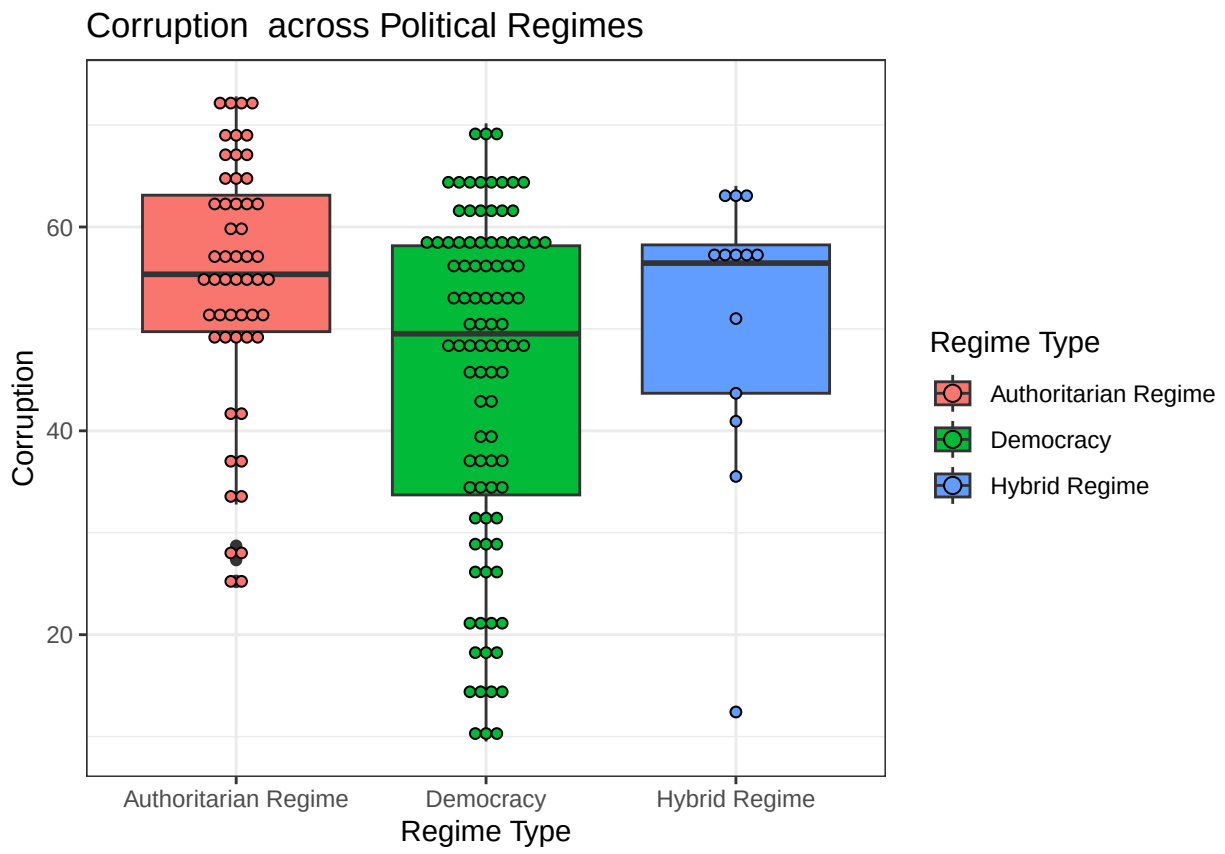



```r
#Make it a bit advanced
b2<- total |>
  filter(!is.na(regime_status_name),
         !is.na(gaiscore)) |>
  ggplot(aes(x = regime_status_name, y = gaiscore)) +
  geom_boxplot(aes(fill = regime_status_name), color = "black") +
  geom_jitter(width = 0.2, alpha = 0.7) +                    # Add jittered points
  labs(title = "Global Acceptance Index across Political Regimes",
       x = "Regime Type",
       y = "Global Acceptance Index",
       fill="Regime") +
  theme_clean() +
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(gaiscore == max(gaiscore)),
            aes(label = paste("Max:", round(gaiscore, 2)),
                y = gaiscore + 0.2), vjust = -0.5) +          # Label max values
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(gaiscore == min(gaiscore)),
            aes(label = paste("Min:", round(gaiscore, 2)),
                y = gaiscore - 0.2), vjust = 1) +             # Label min values
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(gaiscore == median(gaiscore)),
            aes(label = paste("Median:", round(gaiscore, 2)),
                y = gaiscore), vjust = -0.5)                  # Label median values
b2
```



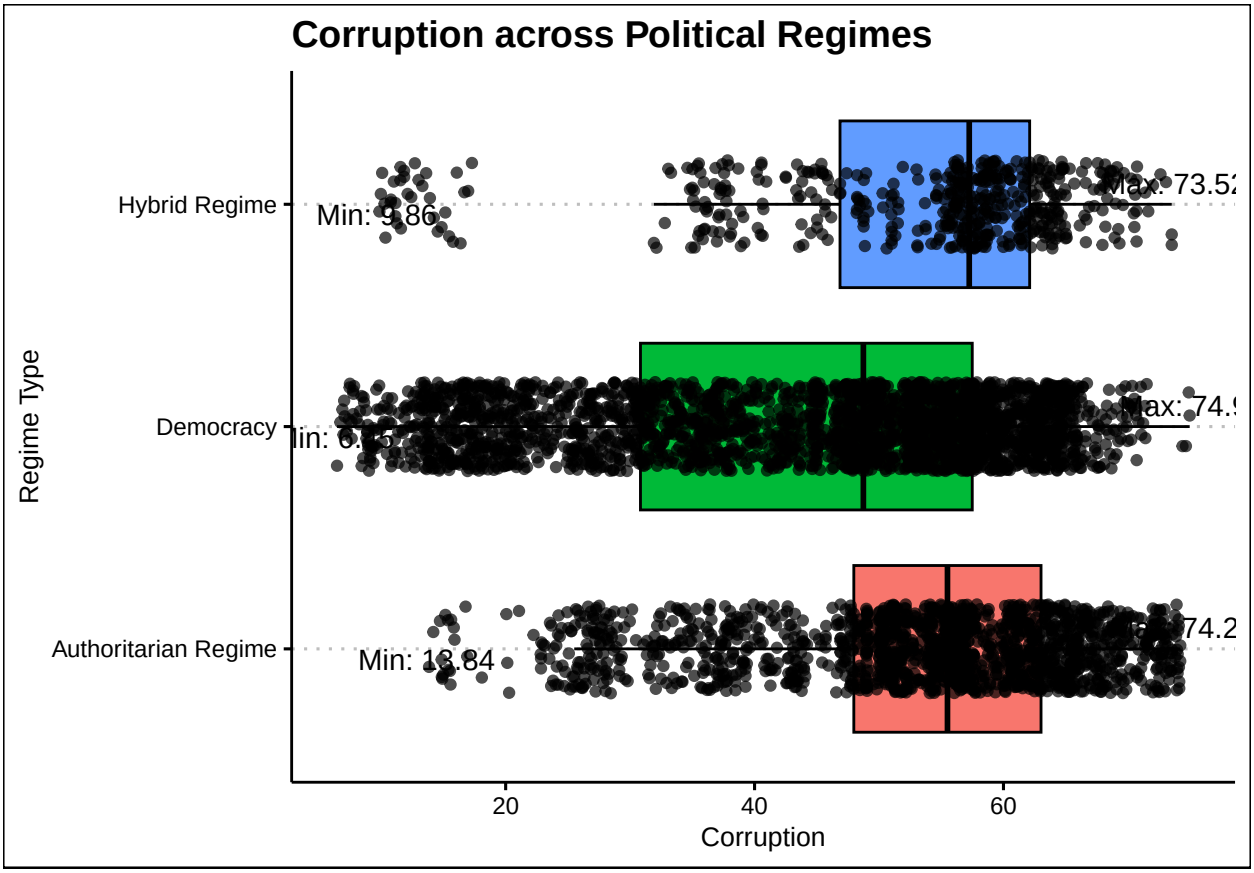

```
#Show all dots: relationship between corruption level and regime type.
b3<-total |>
  filter(!is.na(regime_status_name),
         !is.na(bci_bci),
         year==2000) |>
  ggplot(aes(x=regime_status_name, y=bci_bci, fill=regime_status_name))+
  geom_boxplot()+
  labs(title="Corruption  across Political Regimes",
       x="Regime Type", y="Corruption",
       fill= "Regime Type")+
  geom_dotplot(binaxis = "y", stackdir = "center", dotsize = 0.5)+  #all dots
  theme_bw()
b3
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.
```




```r
# flip it and remove the legend
b4 <- total |>
  filter(!is.na(regime_status_name),
         !is.na(bci_bci)) |>
  ggplot(aes(x = regime_status_name, y = bci_bci)) +
  geom_boxplot(aes(fill = regime_status_name), color = "black", outlier.shape = NA) +
  guides(fill = "none")+                                          #removes legend
  geom_jitter(width = 0.2, alpha = 0.7) +
  labs(title="Corruption across Political Regimes",
       x="Regime Type",
       y="Corruption",
       fill="regime") +
  theme_clean() +
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(bci_bci == max(bci_bci)),
            aes(label = paste("Max:", round(bci_bci, 2)),
                y = bci_bci + 0.2), vjust = -0.5) +               #Max value
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(bci_bci == min(bci_bci)),
            aes(label = paste("Min:", round(bci_bci, 2)),
                y = bci_bci - 0.2), vjust = 1) +                  #Min value
  geom_text(data = . %>% group_by(regime_status_name) %>%
              filter(bci_bci == median(bci_bci)),
            aes(label = paste("Median:", round(bci_bci, 2)),
                y = bci_bci), vjust = -0.5)+                      #Median
  coord_flip()
```

```
b4
```



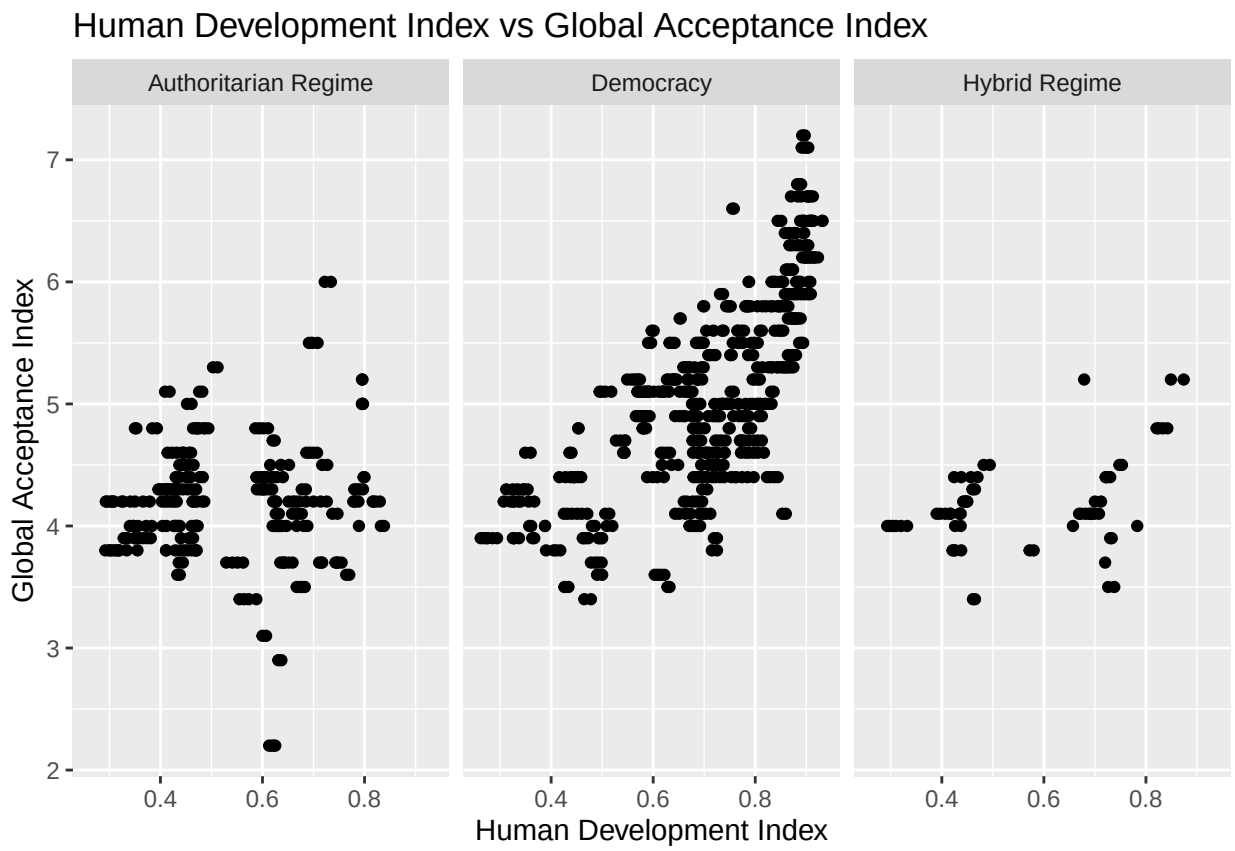
Faceting

Faceting in ggplot2 is a technique for creating multiple plots within a single larger plot. Each small plot displays a subset of the data, making it easier to compare different parts of the data or explore variations across categories or groups. There are two main faceting functions in ggplot2:

- **`facet_wrap()`**: This function is used when you want to facet by a single variable. It arranges the small plots in rows or columns, with each row or column containing multiple plots.
- **`facet_grid()`**: This function creates a grid of plots by specifying which variables define the rows and columns of the grid. You can also customize the arrangement of the panels as needed.

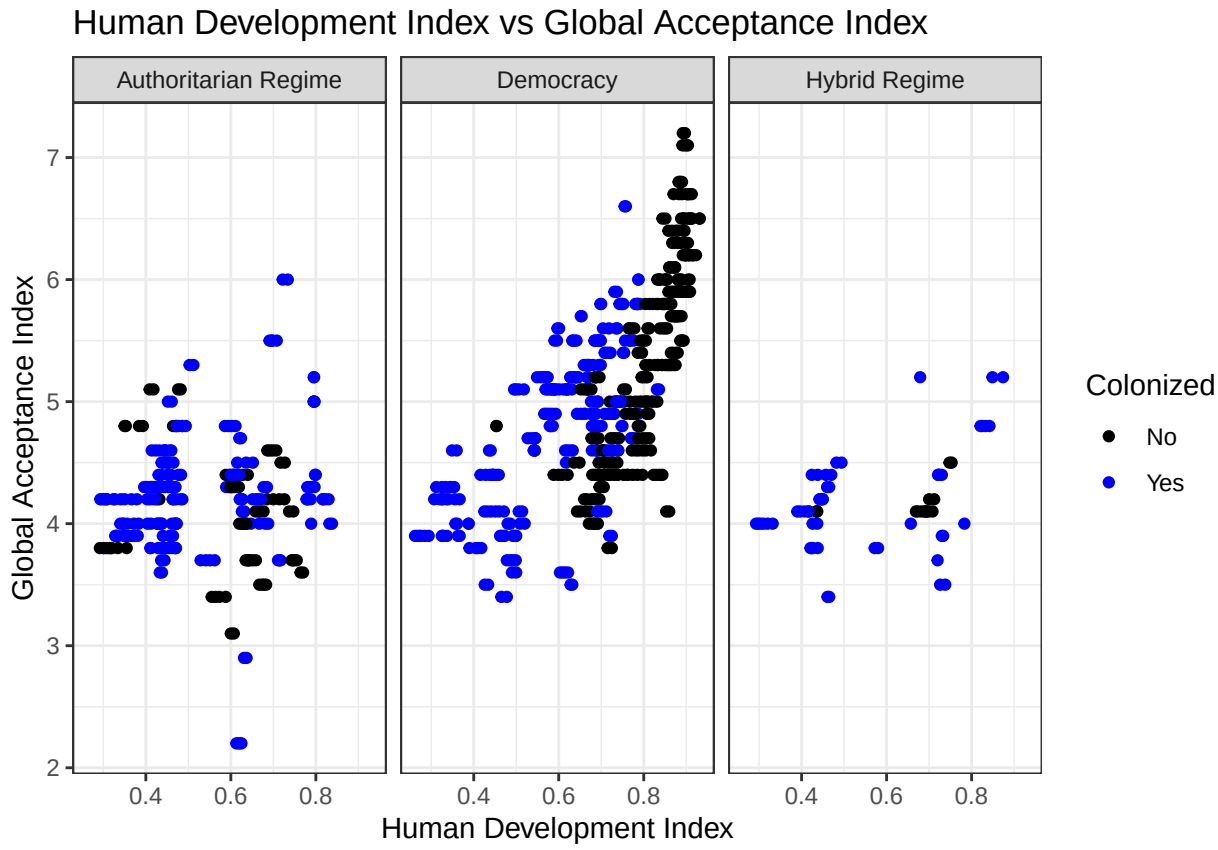
```
#A basic faceting graph
f1 <- total |>
  filter(
    !is.na(gaiscore),
    !is.na(undp_hdi),
    !is.na(regime_status_name),
    year %in% 2000:2005
  ) |>
  ggplot(aes(x = undp_hdi, y = gaiscore)) +
  geom_point() +
  facet_wrap(~ regime_status_name) +
```

```
  labs(
    title = "Human Development Index vs Global Acceptance Index",
    x = "Human Development Index",
    y = "Global Acceptance Index"
  )
f1
```



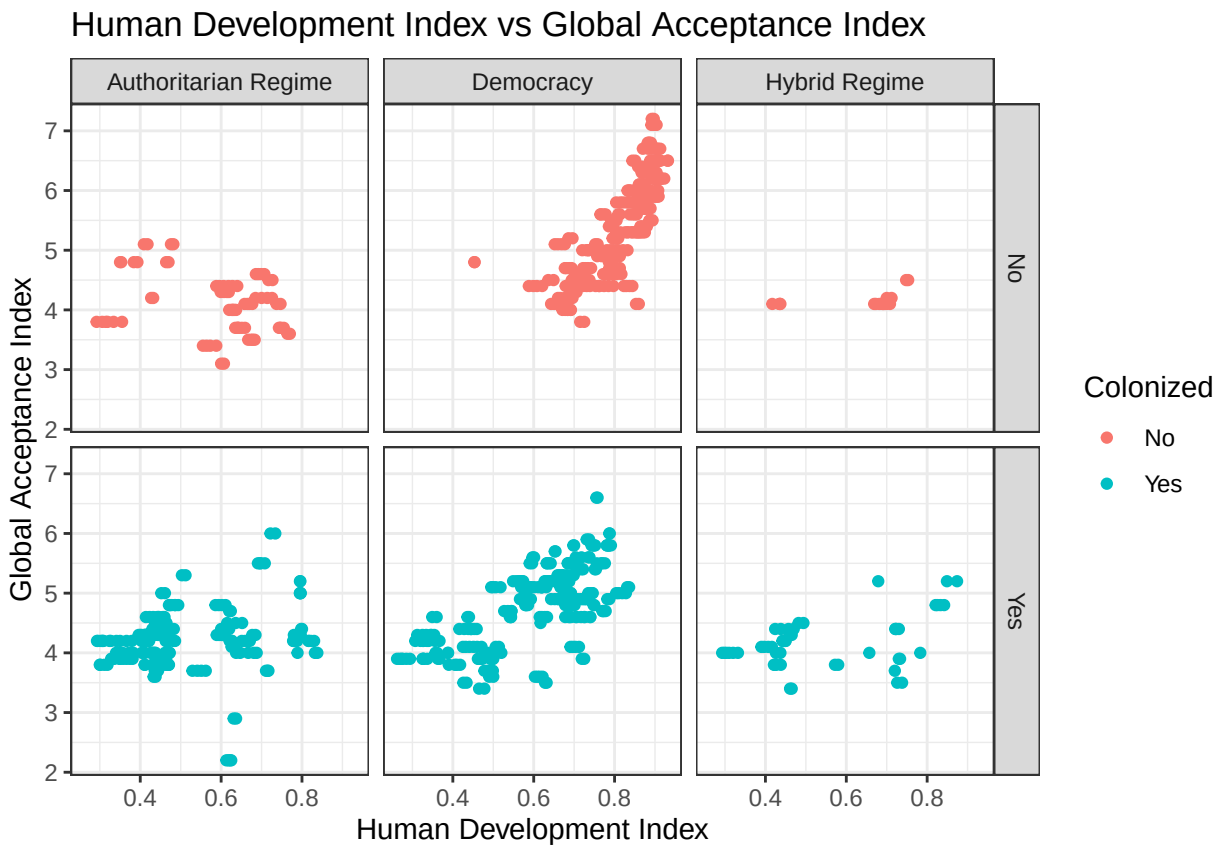

```
#Add a third variable and other aesthetics
f2 <- total |>
  filter(
    !is.na(gaiscore),
    !is.na(undp_hdi),
    !is.na(regime_status_name),
    !is.na(colonial),
    year %in% 2000:2005
  ) |>
  mutate(colonial= ifelse(colonial == 0, "No", "Yes")) |>
  ggplot(aes(x = undp_hdi, y = gaiscore, color = colonial)) +
  geom_point() +
  scale_color_manual(values = c("No" = "black", "Yes" = "blue")) +
  facet_wrap(~ regime_status_name) +
  labs(
    title = "Human Development Index vs Global Acceptance Index",
    x = "Human Development Index",
    y = "Global Acceptance Index",
```

```
    color = "Colonized"
  ) +
  theme_bw()
f2
```




```
#How about facet_grid? How is it different from wrap?
f3 <- total |>
  filter(
    !is.na(gaiscore),
    !is.na(undp_hdi),
    !is.na(regime_status_name),
    !is.na(colonial),
    year %in% 2000:2005
  ) |>
  mutate(colonial= ifelse(colonial == 0, "No", "Yes")) |>
  ggplot(aes(x = undp_hdi, y = gaiscore, color = colonial)) +
  geom_point() +
  facet_grid(colonial~regime_status_name ) +        # Use two variables
  labs(
    title = "Human Development Index vs Global Acceptance Index",
    x = "Human Development Index",
    y = "Global Acceptance Index",
    color = "Colonized"
  ) +
  theme_bw()
```

```
f3
```



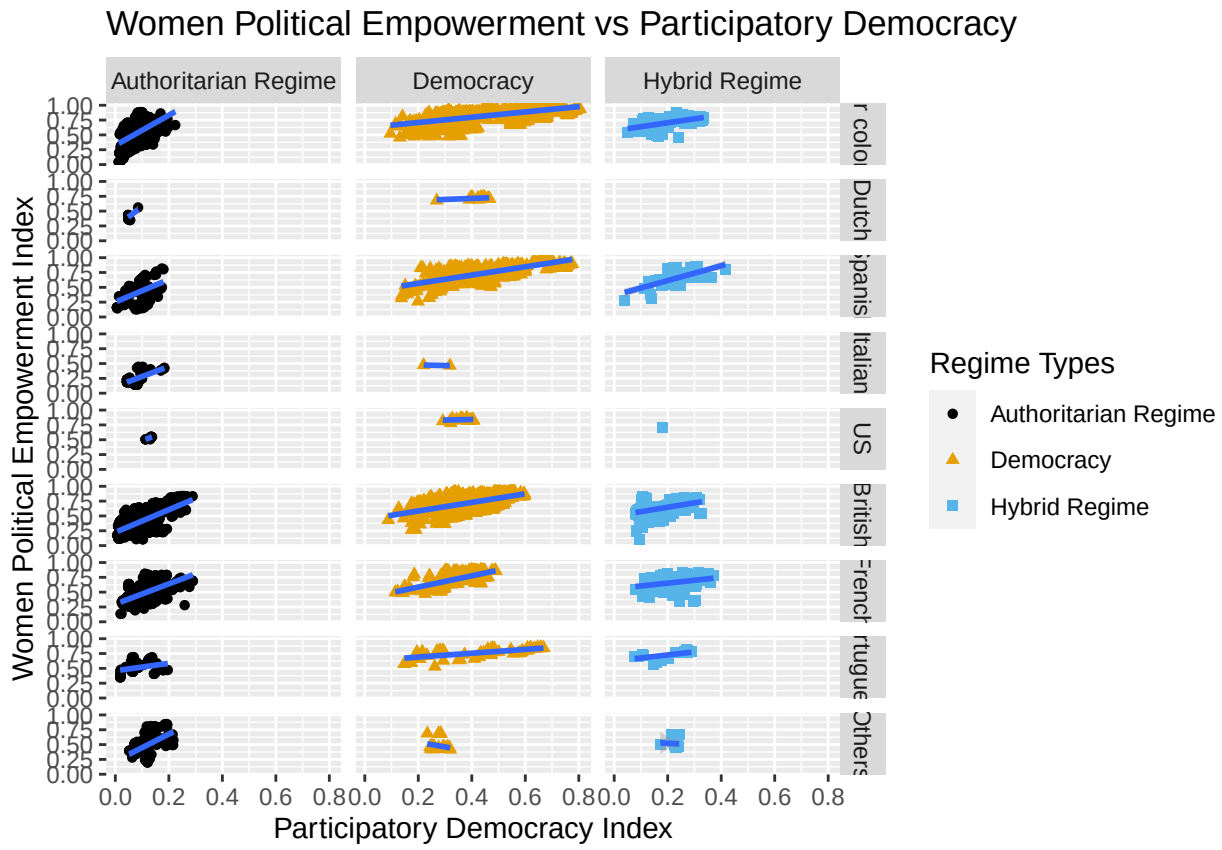

```
#Let's explore one more facet_grid with more panels
gen_par <- total |>                                  # data transformation
  select(vdem_partipdem, vdem_gender, regime_status_name, ht_colonial) |>
  mutate(
    colonial = recode_factor(
      ht_colonial,
      '0' = "Never colonized",
      '1' = "Dutch",
      '2' = "Spanish",
      '3' = "Italian",
      '4' = "US",
      '5' = "British",
      '6' = "French",
      '7' = "Portuguese",
      '8' = "Others",
      '9' = "British-French",
      '10' = "Others"
    )
  ) |>
  na.omit()

f4<-ggplot(
  data = gen_par,
```

```
  mapping = aes(x = vdem_partipdem, y = vdem_gender)
) +
  geom_point(aes(color = regime_status_name, shape = regime_status_name), na.rm = T)+
  geom_smooth(method = "lm") +
  labs(
    title = "Women Political Empowerment vs Participatory Democracy",
    x = "Participatory Democracy Index",
    y = "Women Political Empowerment Index",
    color = "Regime Types",
    shape = "Regime Types"
  ) +
  scale_color_colorblind() +
  facet_grid(colonial ~ regime_status_name)
f4
```

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

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



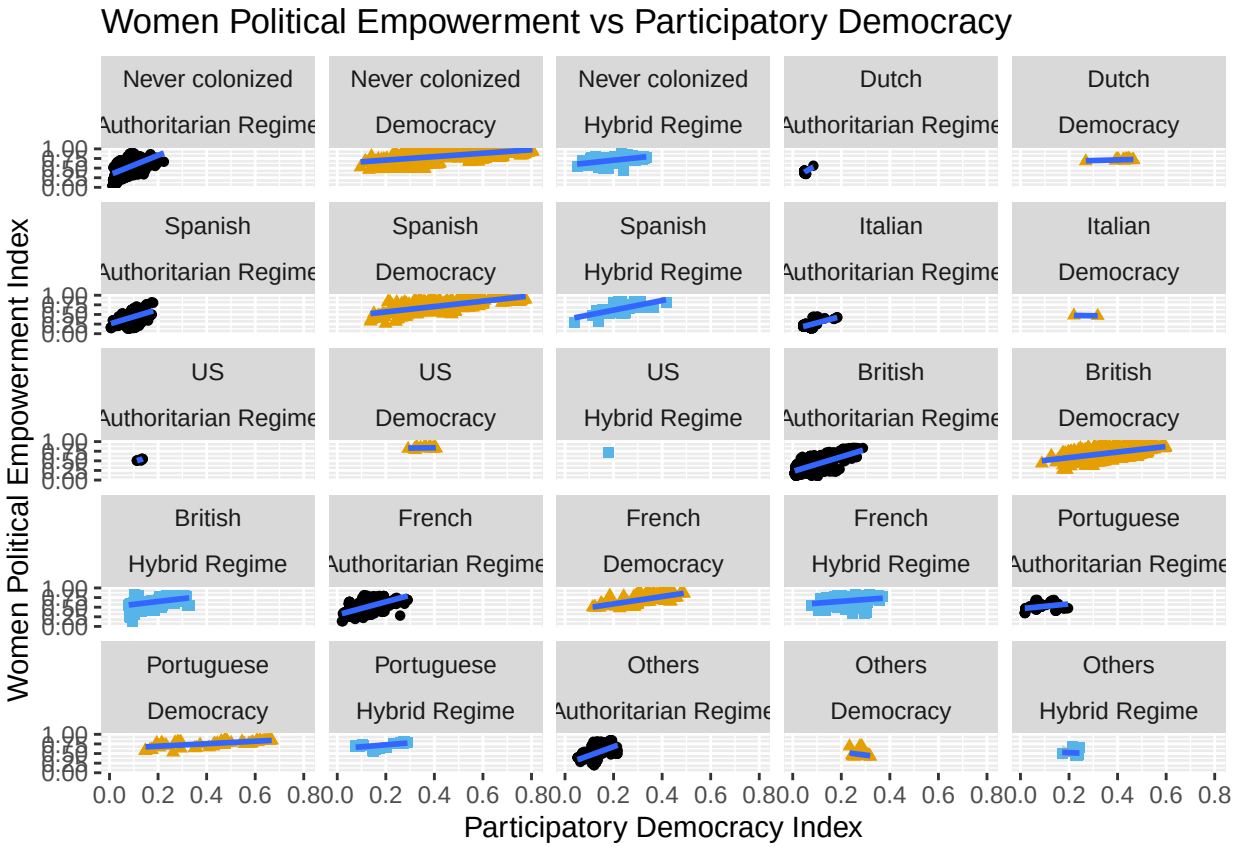

```
#now change grid to wrap and everything else
gen_par <- total |>
```

```
  select(vdem_partipdem, vdem_gender, regime_status_name, ht_colonial) |>
  mutate(colonial = recode_factor(ht_colonial,
      '0' = "Never colonized",
      '1' = "Dutch",
      '2' = "Spanish",
      '3' = "Italian",
      '4' = "US",
      '5' = "British",
      '6' = "French",
      '7' = "Portuguese",
      '8' = "Others",
      '9' = "British-French",
      '10' = "Others"
    )
  ) |>
  na.omit()

f5<-ggplot(
  data = gen_par,
  mapping = aes(x = vdem_partipdem, y = vdem_gender)
) +
  geom_point(aes(color = regime_status_name, shape = regime_status_name), na.rm = TRUE) +
  geom_smooth(method = "lm") +
  labs(
    title = "Women Political Empowerment vs Participatory Democracy",
    x = "Participatory Democracy Index",
    y = "Women Political Empowerment Index",
    color = "Regime Types",
    shape = "Regime Types"
  ) +
  scale_color_colorblind() +
  guides(color="none", shape="none")+
  facet_wrap(colonial ~ regime_status_name)
f5
```

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

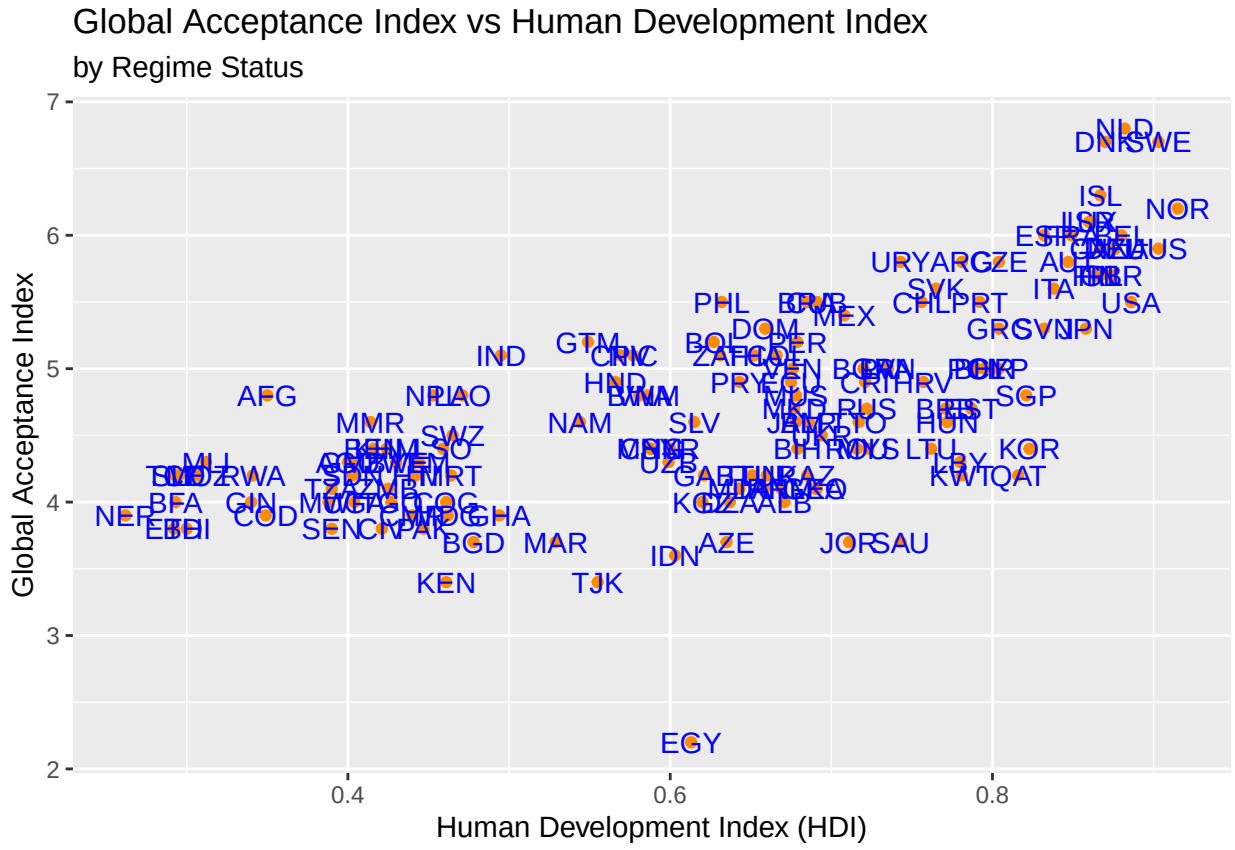
```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning in qt((1 - level)/2, df): no non-missing arguments to max; returning
## -Inf
```



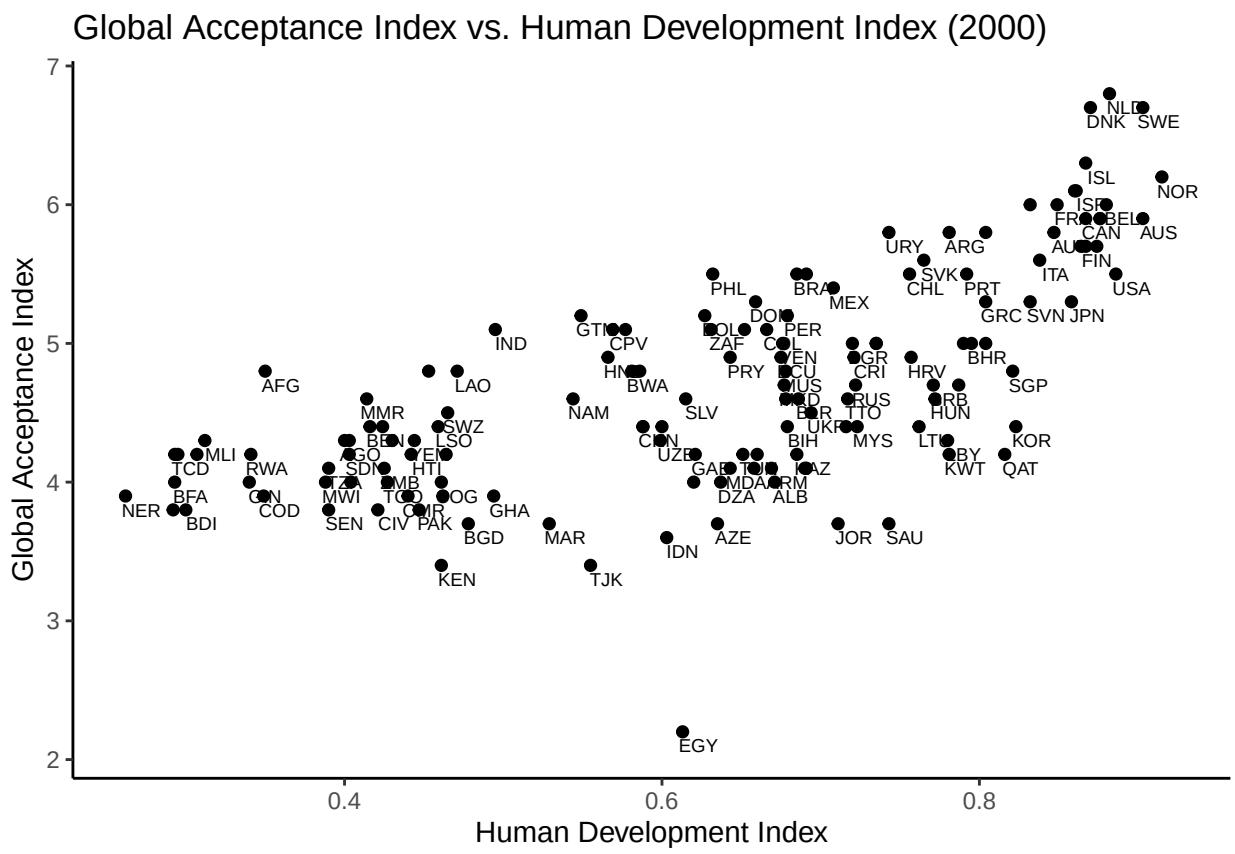
Geom_Text

```
#A basic geom_text
t1<-total|>
  filter(!is.na(iso3c),
         !is.na(gaiscore),
         !is.na(undp_hdi),
         year==2000,
         !is.na(regime_status_name)) |>
  ggplot(aes(x=undp_hdi, y=gaiscore))+
  geom_point(color="darkorange")+
  geom_text(aes(label=iso3c), color="blue")+ # labels from 'iso3c' column
  labs(
    title = "Global Acceptance Index vs Human Development Index",
    subtitle = "by Regime Status",
    x = "Human Development Index (HDI)",
    y = "Global Acceptance Index")
t1
```



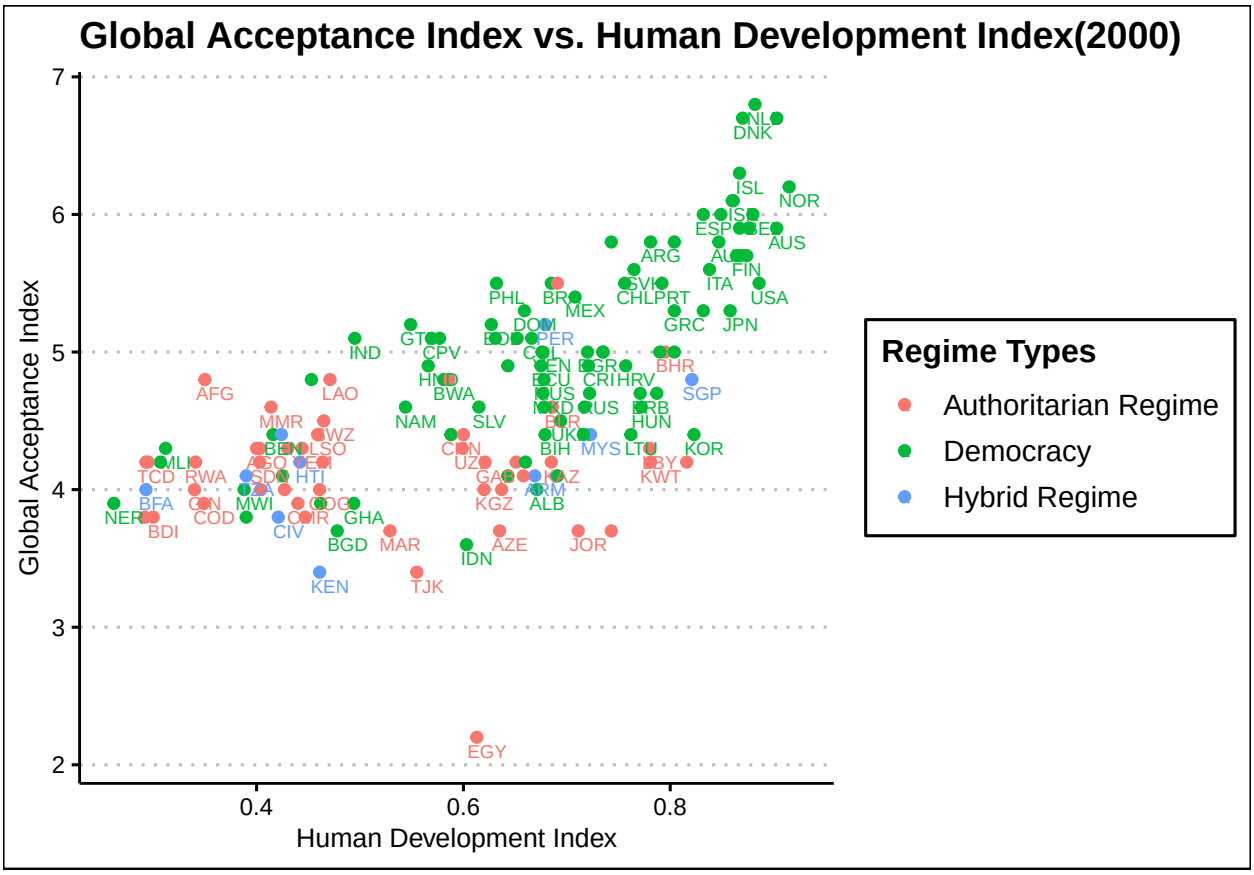

```
#let's make a customized plot
t2 <- total |>
  filter(
    !is.na(iso3c),
    !is.na(gaiscore),
    !is.na(undp_hdi),
    year==2000,
    !is.na(regime_status_name)
  ) |>
  ggplot(aes(x = undp_hdi, y = gaiscore)) +


  geom_point(                    # Create a scatterplot with  point aesthetics
    size = 1,                    # Adjust the size of the points
    alpha = 1,                   # Adjust the transparency of the points
    stroke = 1,                  # Add a border around points
    shape = 19                   # Use a solid circle as the point shape
  ) +
  geom_text(                     # Add labels for data points (ISO3 codes)
    aes(label = iso3c),
    nudge_x = 0.01,              # Adjust the horizontal position of labels
    nudge_y = -0.1,              # Adjust the vertical position of labels
    check_overlap = T,           # adjust label positions to avoid overlap
    size = 2.5                   # Adjust the size of the labels
  ) +
  labs(
```

```
    title = "Global Acceptance Index vs. Human Development Index (2000)",
    x = "Human Development Index",
    y = "Global Acceptance Index") +
  theme_classic()
t2
```




```
#let's add a color aesthetics(regime type)
t3 <- total |>
  filter(
    !is.na(iso3c),
    !is.na(gaiscore),
    !is.na(undp_hdi),
    year==2000,
    !is.na(regime_status_name)
  ) |>
  ggplot(aes(x = undp_hdi, y = gaiscore, color = regime_status_name)) +
  geom_point(                  #Create a scatter-plot with point aesthetics
    size = 1,                  # Adjust the size of the points
    alpha = 1,                 # Adjust the transparency of the points
    stroke = 1,                # Add a border around points
    shape = 19                 # Use a solid circle as the point shape
  ) +
  geom_text(                   #Add labels for data points (ISO3 codes)
    aes(label = iso3c),
    nudge_x = 0.01,            # Adjust the horizontal position of labels
```

```
    nudge_y = -0.1,            # Adjust the vertical position of labels
    check_overlap = T,         # Adjust label positions to avoid overlap
    size = 2.5                 # Adjust the size of the labels
  ) +
  labs(
    title = "Global Acceptance Index vs. Human Development Index(2000)",
    x = "Human Development Index",
    y = "Global Acceptance Index",
    color="Regime Types"
  ) +
  theme_clean()
t3
```



5. How Do I Know Which Code/Graph I Should Use?

Deciding which code or graph to use in **ggplot2** depends on the questions you want to answer, the message you want to convey with your visualization, and off course, the nature of your data. Here's a step-by-step guide to help you choose the right code and graph in **ggplot2**:

1. **Define Your Objectives:** What insights do you want to convey to your audience?
    - compare values
    - show distributions
    - highlight trends

- explore relationships
- or temporal and spatial relationships?

2. **Understand Your Data:** What type of data are you working with. For instance, numeric, categorical, time series? Are there any patterns, relationships, or trends you want to explore or communicate?
3. **Choose Right Aesthetics (aes):** x-axis, y-axis, color, size, shape, label that represent the variables you want to visualize.
4. **Select Right Graph Type (Geom)**: Choose a graph type that best represents your data. For instance:
   - Use bar charts for comparing categorical data.
   - Use scatter plots to show relationships between two continuous variables.
   - Use line charts for time series data.
5. **Set Scales and Themes:** You can customize axis labels, titles, fonts, colors, and other visual elements using `scale_` and `theme_` functions.
6. **Experiment and Iterate:**
   - Always experiment with different geoms, aesthetics, and settings.
   - Create multiple versions of your plot to see which one best conveys your message.
7. **Practice and Learn:**
   - Becoming proficient with `ggplot2` often requires practice and learning from experience.
   - Explore the `ggplot2` documentation, online tutorials, and galleries of data visualizations for inspiration.

Remember that data visualization is both art and science. The choice of code and graph should align with your goals and effectively convey the insights you want to share with your audience.

6. Further Resources

Learning and mastering `ggplot2` in R can greatly enhance your data visualization skills. Here are some valuable resources and steps to help you get started and become proficient with `ggplot2`:

1. **Official Documentation:**
   - The official `ggplot2` documentation is a comprehensive resource. Start with the **ggplot2 reference website**, which includes detailed documentation, examples, and references to related functions.
   - Tidyverse offers a **ggplot2 cheat sheet** that serves as a quick reference for creating different types of plots.
2. **Books:**
   - "ggplot2: Elegant Graphics for Data Analysis" by Hadley Wickham is the go-to resource. It provides a deep understanding of the package.
   - "R Graphics Cookbook" by Winston Chang offers practical solutions to common visualization challenges using `ggplot2`.
   - "R for Data Science" by Hadley Wickham, Mine Çetinkaya-Rundel, Garrett Grolemund. One of the best books for R beginners.
   - For more books, visit BOOKDOWN Webpage
3. **Online Tutorials and Courses:**
   - Websites like **R Graphics Cookbook** and **DataCamp** offer interactive courses on `ggplot2`.

- YouTube has numerous video tutorials explaining various aspects of `ggplot2`. Search for "ggplot2 tutorial" to find relevant videos.
- Many data science and R bloggers write informative articles and tutorials on `ggplot2`. Some popular blogs to explore include R-Bloggers and Towards Data Science on Medium.

4. **Online Communities:**

   - Join online communities like the **RStudio Community** and **Stack Overflow** where you can ask questions, share your challenges, and learn from others.

5. **GitHub Repositories:**

   - Explore GitHub repositories that contain `ggplot2` examples and projects. You can find code, datasets, and visualizations that may inspire you.