1 Exercises

1.1 Formatting in ggplot2

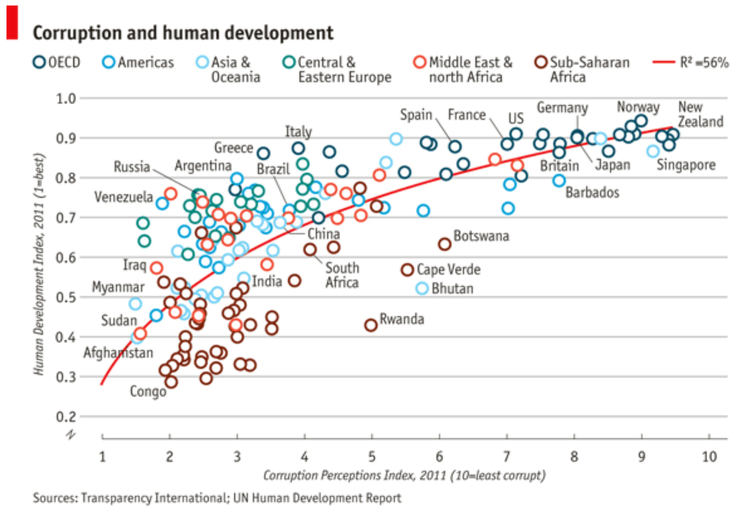


Figure 1.1: Graph from *The Economist* (see `https://www.economist.com/graphic-detail/2011/12/02/corrosive-corruption`) showing the relationship between corruption and the human development index.

A large portion of thoughts and concepts in this exercise are inspired by Harvard's tutorial *Introduction to R Graphics with* `ggplot2`. Available at `http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html`.

THE use of public office for private gain benefits a powerful few while imposing costs on large swathes of society. Transparency International's annual Corruption Perceptions Index, published on December 1st, measures the perceived levels of public-sector graft by aggregating independent surveys from across the globe. Just five non-OECD countries make the top 25: Singapore, Hong Kong, Barbados, Bahamas and Qatar. The bottom is formed mainly of failed states, poor African countries and nations that either were once communist (Turkmenistan) or are still run along similar lines (Venezuela, Cuba). Comparing the corruption index with the UN's Human Development Index (a measure combining health, wealth and education), demonstrates an interesting connection. When the corruption index is between approximately 2.0 and 4.0 there appears to be little relationship with the human development index, but as it rises beyond 4.0 a stronger connection can be seen. Outliers include small but well-run poorer countries such as Bhutan and Cape Verde, while Greece and Italy stand out among the richer countries. (**?**)

Figure 1.1 appeared in *The Economist* at the end of 2011. For this exercise, you will be led through code that reproduces this graphic, but using 2017 data (`corruptDF`) as available in the `causact` package. At the end, you will be asked to extend the reproduced graphic and submit a working R-script that reproduces the figure.

The `ggplot()` function is used to initialize the basic graph structure, then we add to it. The structure of a `ggplot()` function call looks like this:

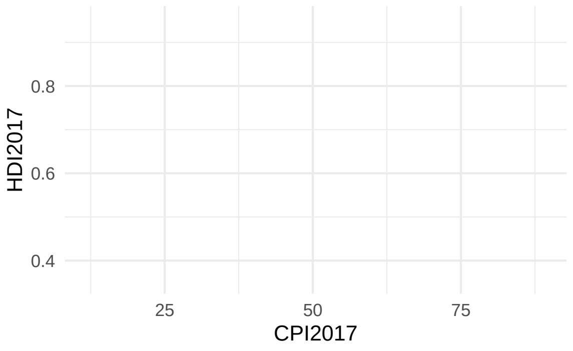
```
ggplot(data = <default data set>,
       aes(x = <default x axis variable>,
           y = <default y axis variable>,
           ... <other default aesthetic mappings>),
       ... <other plot defaults>) +

        geom_<geom type>(aes(size = <size variable for this geom>,
                      ... <other aesthetic mappings>),
                  data = <data for this point geom>,
                  stat = <statistic string or function>,
                  position = <position string or function>,
                  color = <"fixed color specification">,
                  <other arguments, possibly passed to the _stat_ function) +

        scale_<aesthetic>_<type>(name = <"scale label">,
                     breaks = <where to put tick marks>,
                     labels = <labels for tick marks>,
                     ... <other options for the scale>) +

        theme(plot.background = element_rect(fill = "gray"),
        ... <other theme elements>)
```

1.1.1 Getting the Basics Right - Data, Geoms, and Coordinates

To begin the 2017 reproduction and extension of Figure 1.1, we first need to get the data:

```
library("dplyr")
library("ggplot2")
library("causact")
# retrieve the data
data("corruptDF")
```

```
# define the coordinate system
ggplot(corruptDF, aes(x = CPI2017, y = HDI2017))
```



and choose the geom to display the points of interest:

```
plot1 = ggplot(corruptDF, aes(x = CPI2017, y = HDI2017)) + geom_point()
plot1
```

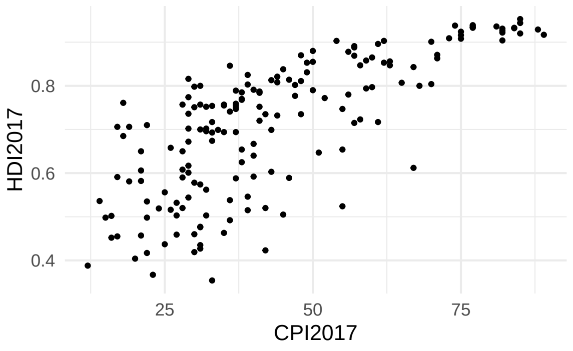


Figure 1.2: A basic plot of the corruption data.

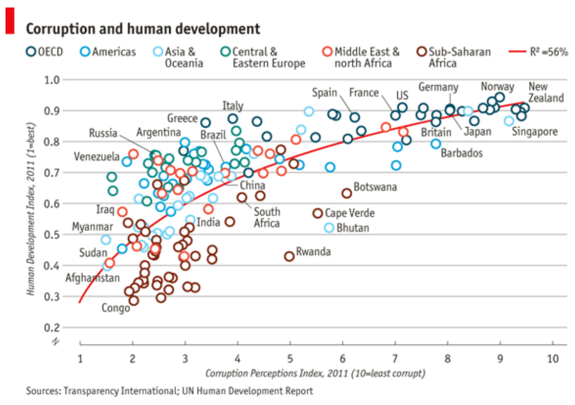


Figure 1.3: Are there any similarities between the ggplot figure and the economist figure?

As basic as Figure 1.2 may look, the core information showing the relationship between human development and corruption is observed.

However, a good graphic can convey much more information through the use of additional aesthetics.

In ggplot aesthetic means *something you can see*. Examples include:

1. position (i.e., on the x and y axes)
2. color ("outside" color)
3. fill ("inside" color)
4. shape (of points)
5. linetype (dashed, dotted, etc.)
6. size

Each type of geom accepts only a subset of all aesthetics - refer to the geom help pages to see what mappings each geom accepts. For our purposes, we can take advantage of color.

```r
ggplot(corruptDF, aes(x = CPI2017, y = HDI2017)) +
  geom_point(aes(color=region), size = 4)
```

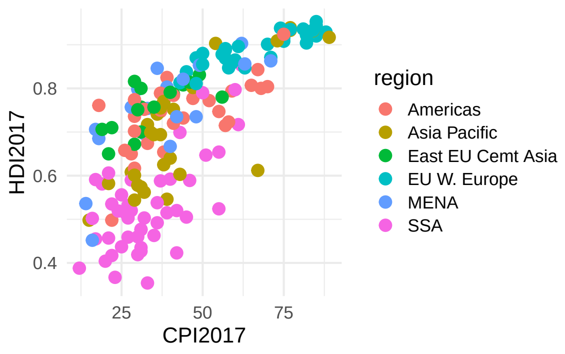


Figure 1.4: A basic plot of the corruption data.

Mapping region to the color aesthetic works well because there are only 7 regions. If we recall, color is useful for data that is not ordered and has only a few values.

Outside of aesthetic mapping, the above call to `geom_point()` specified a size argument was used to make the points bigger and more prominent. However, this is not an aesthetic mapping as it is a **non-data** related property. These properties which are not directly tied to data are often better handled through the use of themes.

The ggplot2 theme system handles non-data plot elements such as axis labels, plot background, legend appearance. There are some built-in themes we can use (`theme_gray()` (default), `theme_bw()`, `theme_classic()`):

```r
library(gridExtra)
plotA = plot1 + theme_classic()
```

```
plotB = plot1 + theme_minimal()
grid.arrange(plotA,plotB, nrow = 1)
```

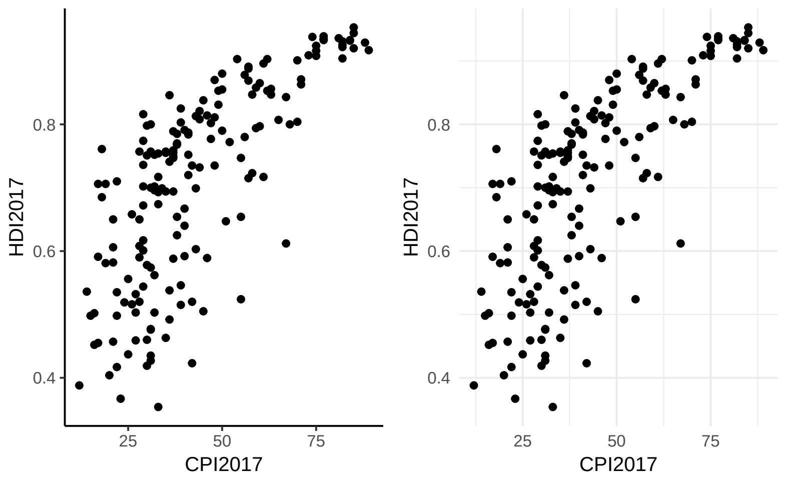


Figure 1.5: Examples of `theme_classic()` (left) and `theme_minimal()` (right) applied to the previous scatterplot.

and we can create a new theme, as in the following example:

```
plot1 = ggplot(corruptDF, aes(x = CPI2017, y = HDI2017)) +
  geom_point(aes(color=region))

theme_new <- theme_classic() +
  theme(plot.background = element_rect(size = 1, color = "blue", fill = "black"),
        text=element_text(size = 16, color = "darkblue"),
        axis.text.y = element_text(colour = "purple"),
        axis.text.x = element_text(colour = "red"),
        panel.background = element_rect(fill = "pink"),
        strip.background = element_rect(fill = "orange"))

plot1 + theme_new
```

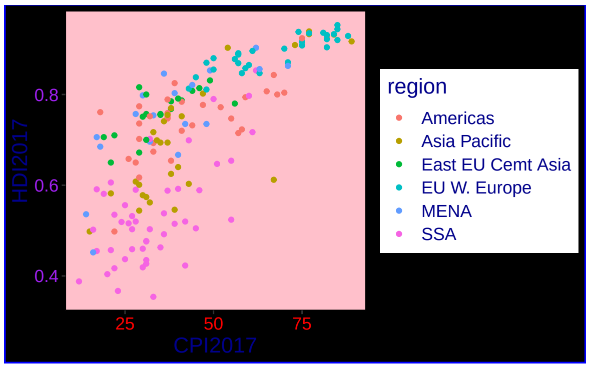


Figure 1.6: A custom theme example.

This graph is a little too audacious. Let's return to a basic plot from which we can build on.

```r
plot2 = ggplot(corruptDF, aes(x = CPI2017, y = HDI2017)) +
  theme_minimal()
plot2 + geom_point(aes(color=region))
```

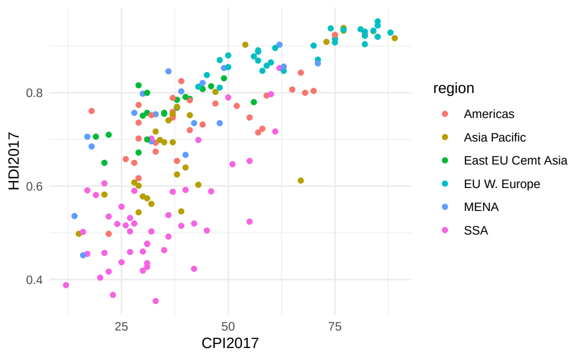
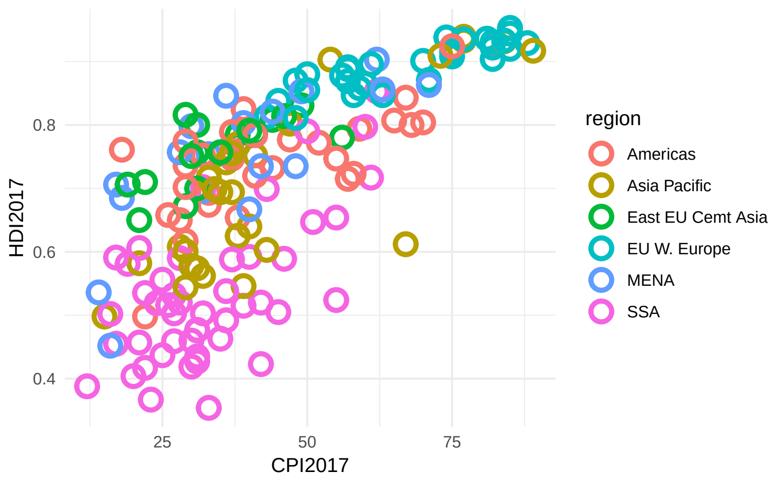


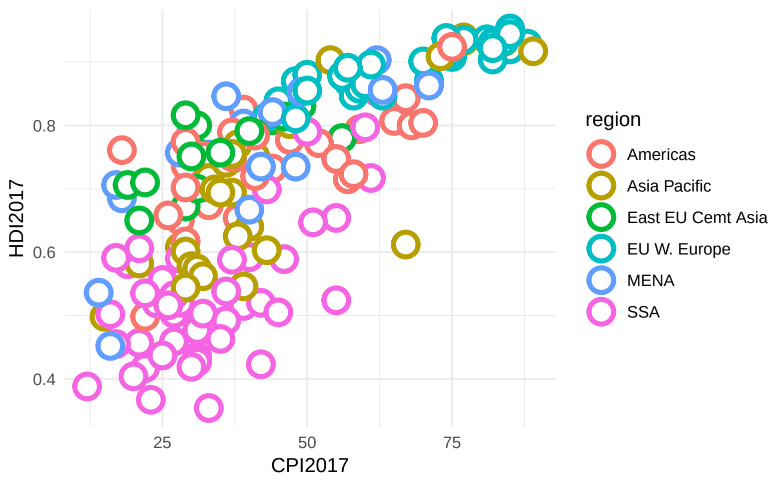
Figure 1.7: The basic plot from which we build.

Geometric objects are the actual marks we put on a plot. To enlarge the points of Figure 1.7 and to make them hollow circles, we will want to alter those marks. From our data visualization cheatsheet (`https://www.rstudio.com/resources/cheatsheets/`), we can see that `geom_point()` has shape and size arguments. In addition, although not very clear, shapes 21-25 shown on page 2 of the cheatsheet have both *stroke color* and *fill* aesthetics. The size of the filled part is controlled by `size`, the size of the stroke is controlled by `stroke`. Each is measured in mm, and the total size of the point is the sum of the two. For our purposes, `size = 4` and `stroke = 2` will get us the perfect points we need.

```r
plot2 + geom_point(aes(color=region), shape = 21, size = 4, stroke = 2)
```



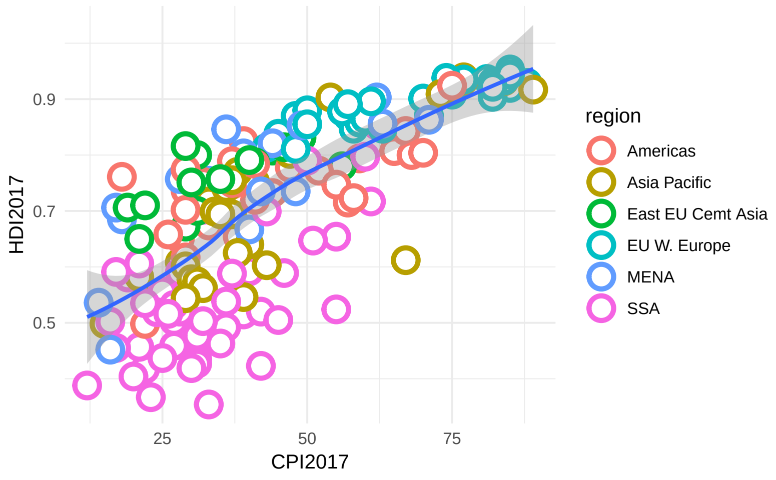
```
###and after a few adjustments
plot3 = plot2 +
  geom_point(aes(color=region), shape = 21, size = 5, stroke = 2, fill = "white")
plot3
```



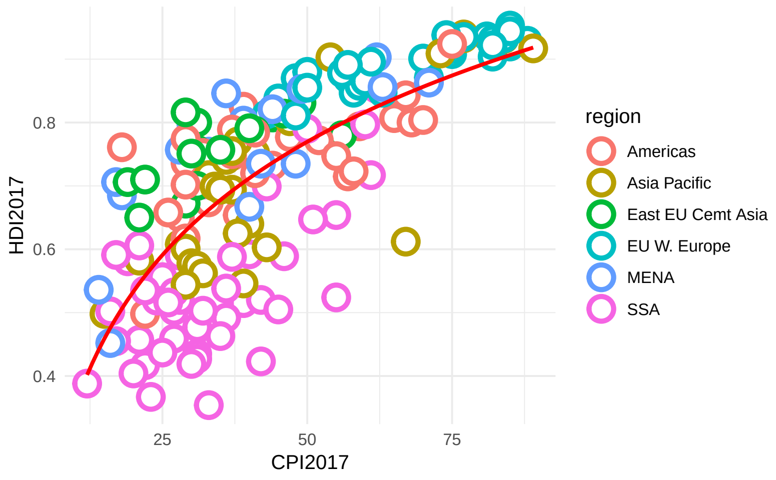
See `https://ggplot2.tidyverse.org/reference/geom_smooth.html` for more information on the `geom_smooth` arguments.

Statistical transformations within ggplot2 are `geoms` that manipulate the data prior to plotting it. In ggplot2, many layers (such as `geom_point` i.e. scatterplots) do not require transformations - each point is plotted at x and y coordinates equal to the original value. Other plots, such as boxplots, histograms, prediction lines etc. require statistical transformations. Or in our case, we seek to add a smooth curve through the data points. While we could explicitly model the curve and then plot it, we can also take advantage of built-in ggplot functionality, namely `geom_smooth()` to transform our y-values into predicted values prior to plotting.

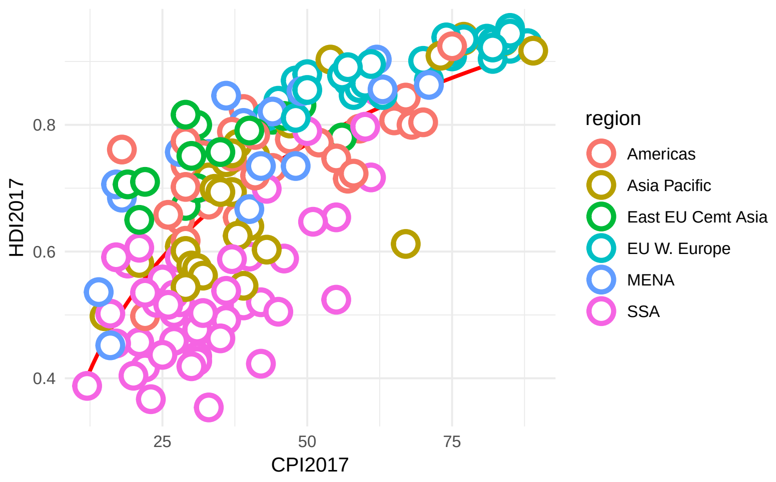
```r
plot3 + geom_smooth()
```



```r
###with a few adjustments
plot3 + geom_smooth(method = "lm", formula = y ~ log(x), se = FALSE, color = "red")
```



```r
##placing the line layer behind the points layer
plot4 = plot2 +
  geom_smooth(method = "lm", formula = y ~ log(x), se = FALSE, color = "red") +
  geom_point(aes(color=region), shape = 21, size = 5, stroke = 2, fill = "white")
plot4
```

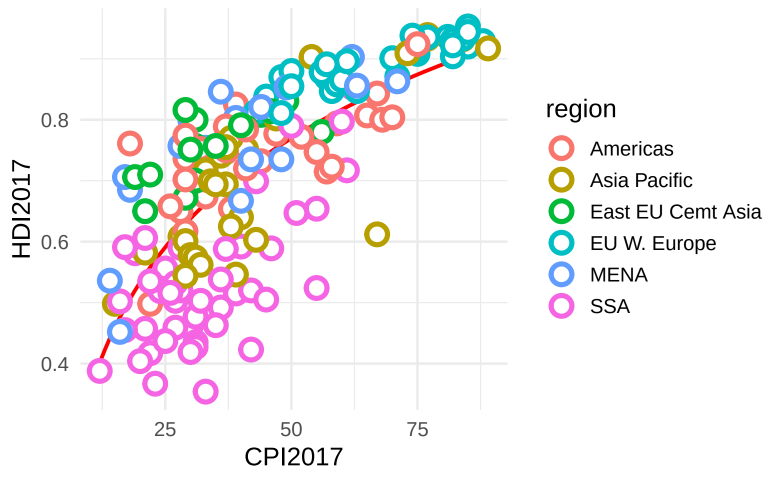


LABELLING POINTS is tricky because there is no attribute in the data that separates points that should be labelled from points that should not be labelled. Whenever this is the case, we need to add data to the dataframe that contains the labels for the points we want labelled and contains NA for the points we do not want labelled.

```
countriesToLabel = c("Russia", "Venezuela", "China", "Kenya", "Syria",
                     "Afghanistan", "Greece", "Argentina", "Brazil",
                     "India", "Italy", "Spain","Botswana", "France",
                     "United States", "Germany", "Britain", "Barbados",
                     "Norway", "Japan","New Zealand", "Singapore")

corruptDF2 = corruptDF %>%
  mutate(cLabel =ifelse(country %in% countriesToLabel, country, NA))
corruptDF2
```

```
## # A tibble: 174 x 8
##    country   region   countryCode regionCode
##    <chr>     <chr>    <chr>       <chr>
##  1 Afghanis~ Asia Pa~ AFG         AP
##  2 Albania   East EU~ ALB         ECA
##  3 Algeria   MENA     DZA         MENA
##  4 Angola    SSA      AGO         SSA
##  5 Argentina Americas ARG         AME
##  6 Armenia   East EU~ ARM         ECA
##  7 Australia Asia Pa~ AUS         AP
##  8 Austria   EU W. E~ AUT         WE/EU
##  9 Azerbaij~ East EU~ AZE         ECA
## 10 Bahamas   Americas BHS         AME
## # ... with 164 more rows, and 4 more
## #   variables: population <int>,
## #   CPI2017 <int>, HDI2017 <dbl>,
## #   cLabel <chr>
```

```
plot5 = ggplot(corruptDF2, aes(x = CPI2017, y = HDI2017)) +
  theme_minimal(14) +
  geom_smooth(method = "lm", formula = y ~ log(x), se = FALSE, color = "red") +
  geom_point(aes(color=region), shape = 21, size = 4, stroke = 2, fill = "white")
plot5
```

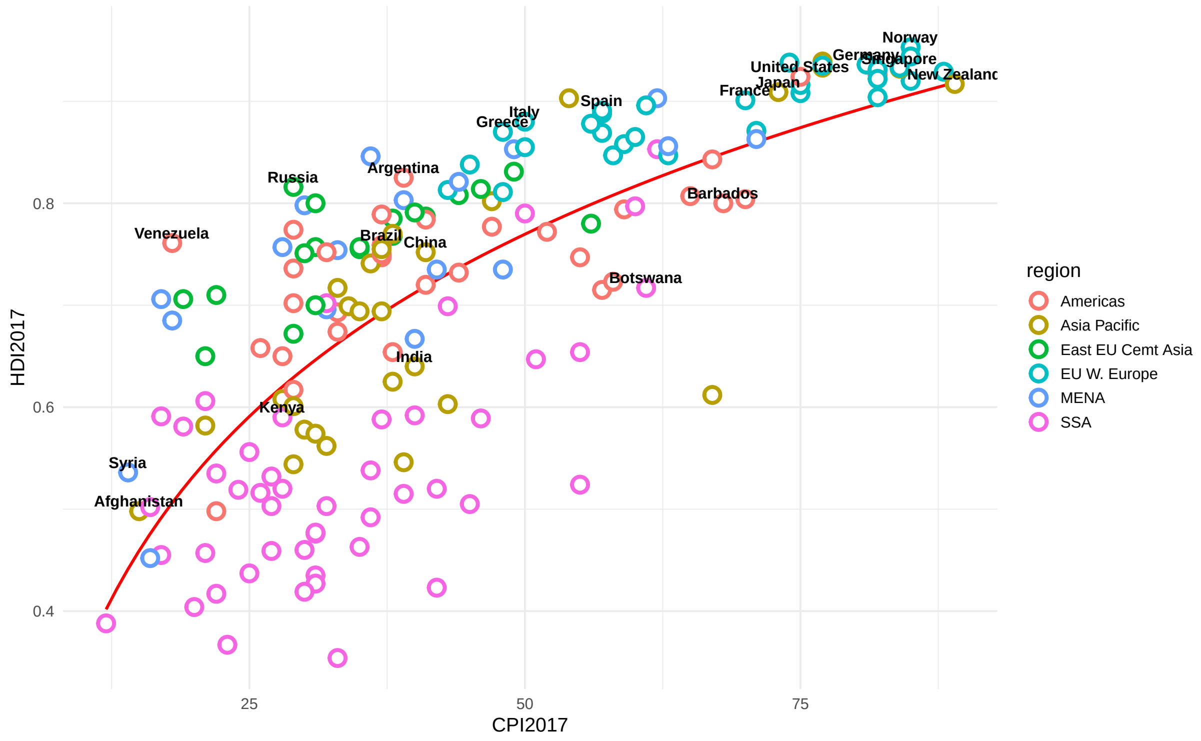


Now we can label these points using `geom_text`, like this:

```
plot6 = plot5 +
  geom_text(aes(label = cLabel),
            nudge_x = -0.05,
            nudge_y = 0.01,
            fontface = "bold")
plot6
```

Alternatively, we can take advantage of the `ggrepel` package to make it a little cleaner:

```
### use install.packages("ggrepel") as needed
library(ggrepel)
plot7 = plot5 +
  geom_text_repel(aes(label = cLabel),
                  force = 2,
                  point.padding = 0.75,
                  fontface = "bold")
plot7
```



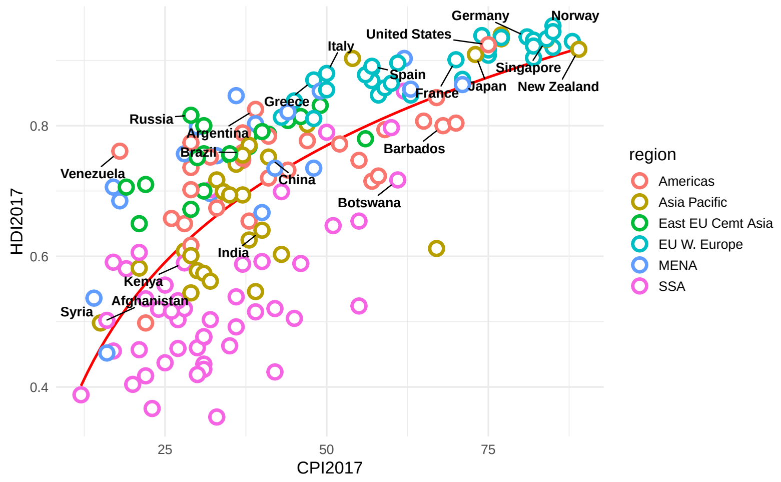


Figure 1.8: Plot with labels for selected countries.

While beyond the scope of this exercise, you can make the above plots interactive using the `plotly` package. This can allow for things like mouseover tooltips that display information for each point as you hover over them with your mouse. For more info, see: `https://plot.ly/ggplot2/user-guide/`.

SCALES ARE USED to control the details of how data gets mapped to aesthetics. For example, let's say we want to control how regions are mapped to the color aesthetic. Note, this level of control gets somewhat tedious; hence, only do this once you are confident that your plot is to be shared and that choosing specific colors is meaningful (otherwise, just use a color theme - see here: `http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/`). When you want to control color, a named vector is the recommended method. The vector values will be the actual colors for the plot and the vector names will be the data values that map to those colors. Let's get the unique values that will be the names:

```
valuesNeedingColors = unique(corruptDF2$region)
valuesNeedingColors
```

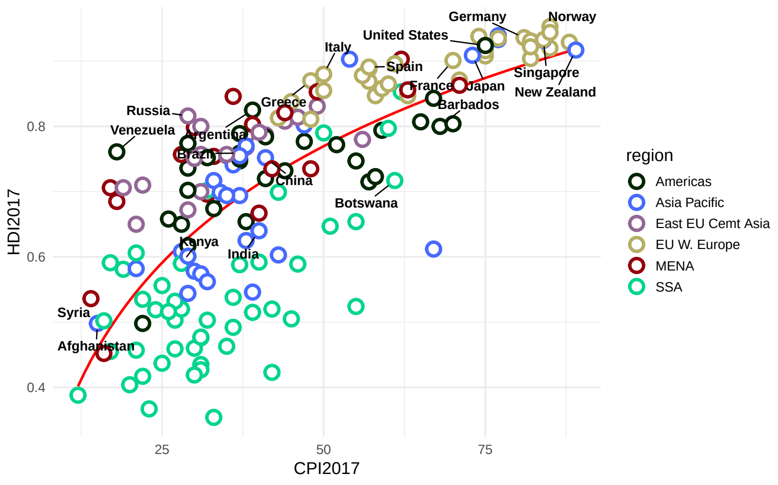
```
## [1] "Asia Pacific"      "East EU Cemt Asia"
## [3] "MENA"              "SSA"
## [5] "Americas"          "EU W. Europe"
```

Notice, there are 6 needed in our color pallette. My favorite site to generate distinct colors is `http://tools.medialab.sciences-po.fr/iwanthue/`. I leave it to the reader to explore that site, but after requesting a pallette of 6 colorblind friendly values, I can cut and paste the following HEX list into the `R` code:

```
## create list of colors using hexadecimal code - your list can be different
colorList = c("#466aff",
"#936796",
"#95000c",
"#02d48e",
"#002500",
"#b5ae64")
names(colorList) = valuesNeedingColors
```

With the named vector, if you have other plots using the same data, you can be consistent across plot's with the colors you choose. Here is the above color scheme applied to the plot:

```
plot7 + scale_color_manual(values = colorList)
```



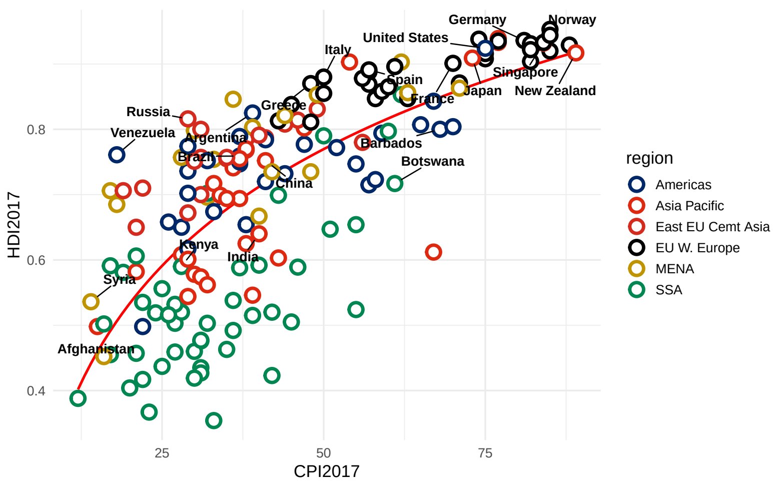
Now, let's say we want to associate meaning to the regions by picking our own colors. We can use the hex codes, as above, or used named colors in R (see more guidance here: `https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf`). To aid the plot viewer's ability to guess at the region purely based on color, I am going to look up the HEX codes for the colors on each region's largest country's flag. Here are the largest countries:

```
corruptDF %>%
  group_by(region) %>%
  top_n(n = 1, wt = population)
```

```
## # A tibble: 6 x 7
## # Groups:   region [6]
##   country   region    countryCode regionCode
##   <chr>     <chr>     <chr>       <chr>
## 1 China     Asia Pac~ CHN         AP
## 2 Egypt     MENA      EGY         MENA
## 3 Germany   EU W. Eu~ DEU         WE/EU
## 4 Nigeria   SSA       NGA         SSA
## 5 Russia    East EU ~ RUS         ECA
## 6 United S~ Americas  USA         AME
## # ... with 3 more variables:
## #   population <int>, CPI2017 <int>,
## #   HDI2017 <dbl>
```

So for the Americas, the United States flag has blue that is #002868. Similarly,

```
colorList2 = c("Asia Pacific" = "#DE2910", # based on China Red
             "East EU Cemt Asia" = "#D52B1E",  # based on Russia Red
             "MENA" = "#C09300", # based on Egypt Gold
             "SSA" = "#008751", # based on Nigeria Green
             "Americas" = "#002868", # based on United States Blue
             "EU W. Europe" = "#000000" # based on germany Black
)

plot7 + scale_color_manual(values = colorList2)
```



As you can see, this is both good and bad. The good is one can sort of know the region based on color without consulting the legend (i.e. based on the knonw context of flag color); the bad is the colors are now hard to distinguish from one another. Color is hugely important, but can also be a time trap. To that end, let's go with the more easily distnguishable colors and be done with it:

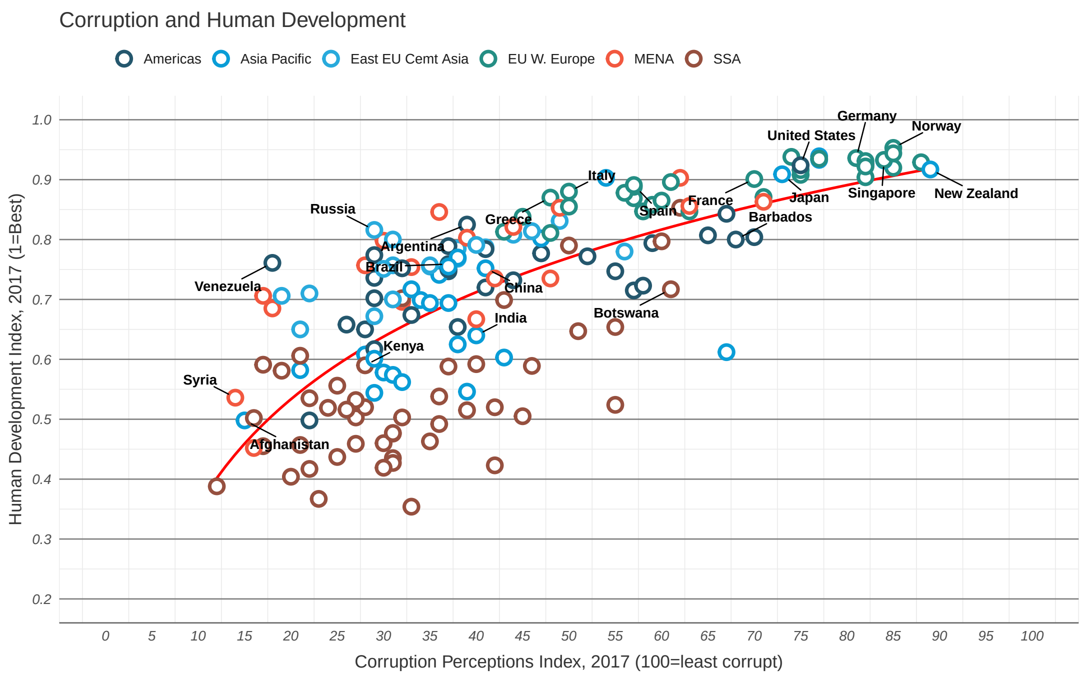
```
plot8 = plot7 + scale_color_manual(values = colorList)
```

** A list of scale functions can be found at `https://ggplot2.tidyverse.org/reference/#section-scales`.

Changing the legend position and other non-data components of a plot in `ggplot2` are modifications to theme elements.** Examples of applying multiple theme elements to reproduce the original *Economist* figure are shown below:

```
plot8 = plot7 +
  guides(col = guide_legend(nrow = 1, title = NULL)) +
  ggtitle("Corruption and Human Development") +
  theme(text = element_text(color = "gray20"),
        legend.position = c("top"), # position the legend in the upper left
        legend.direction = "horizontal",
        legend.justification = 0.1, # anchor point for legend.position.
        legend.text = element_text(size = 11, color = "gray10"),
        axis.text = element_text(face = "italic"),
        axis.title.x = element_text(vjust = -1), # move title away from axis
        axis.title.y = element_text(vjust = 2), # move away for axis
        axis.ticks.y = element_blank(), # element_blank() is how we remove elements
        axis.line = element_line(color = "gray40", size = 0.5),
        axis.line.y = element_blank(),
        panel.grid.major = element_line(color = "gray50", size = 0.5),
        panel.grid.major.x = element_blank()
        ) +
  scale_x_continuous(name = "Corruption Perceptions Index, 2017 (100=least corrupt)",
                     limits = c(0, 100),
                     breaks = seq(0, 100, by = 5)) +
  scale_y_continuous(name = "Human Development Index, 2017 (1=Best)",
                     limits = c(0.2, 1.0),
                     breaks = seq(0.2, 1.0, by = 0.1)) +
  scale_color_manual(name = "",
                     values = c("#24576D",
                                "#099DD7",
                                "#28AADC",
                                "#248E84",
                                "#F2583F",
                                "#96503F"))
plot8
```

70, 0.804



Exercise 1.1. Recreate the last plot - i.e. plot8 - (2 points) with the addition of a text label for *Uruguay*, (1 point) color the **text label** for Uruguay in purple, (0.5 points) and change the color of the SSA region points to your favorite shade of orange (0.5 points)

** A list of theme elements can be found at `https://ggplot2.tidyverse.org/reference/theme.html`.