Session 3: Data Visualization

Andrew McCormack

2018-11-14

The `ggplot2` package

`ggplot2` is a powerful package for data visualization that allows you to create many types of plots with a great deal of flexibility. `ggplot2` is based on the *Grammar of Graphics*—quantitative plots are composed of elements that convey precise and clear messages much like the grammatical elements of sentences. To create quantitative plots, we work with a number layered elements. The strength of `ggplot2` is that each of these elements can be added iteratively (i.e. we can add one element at a time to create highly customized plots). Let's start from scratch with the first and most important element: data.

Data

In order to visualize our data, we need to have a well-prepared data frame. This is the basis for every plot.

First, set the working directory as the folder where your data file is stored:

```
setwd("/Users/andrewmccormack/Documents/DSC/")
civilwar <- read.csv("fearon03.csv")

# Use head() to get acquainted with the first few rows of data
head(civilwar)
```

```
##   X.1 X country year war onset    pop polity2 gdppc
## 1   1 1     USA 1945   0     0 140969      10 7.626
## 2   2 2     USA 1946   0     0 141936      10 7.654
## 3   3 3     USA 1947   0     0 142713      10 8.025
## 4   4 4     USA 1948   0     0 145326      10 8.270
## 5   5 5     USA 1949   0     0 147987      10 8.040
## 6   6 6     USA 1950   0     0 152273      10 8.772
##                           region   mtn oil   ethfrac cow
## 1 western democracies and japan -99.0   0 0.3569501 USA
## 2 western democracies and japan  23.9   0 0.3569501 USA
## 3 western democracies and japan  23.9   0 0.3569501 USA
## 4 western democracies and japan  23.9   0 0.3569501 USA
## 5 western democracies and japan  23.9   0 0.3569501 USA
## 6 western democracies and japan  23.9   0 0.3569501 USA
```

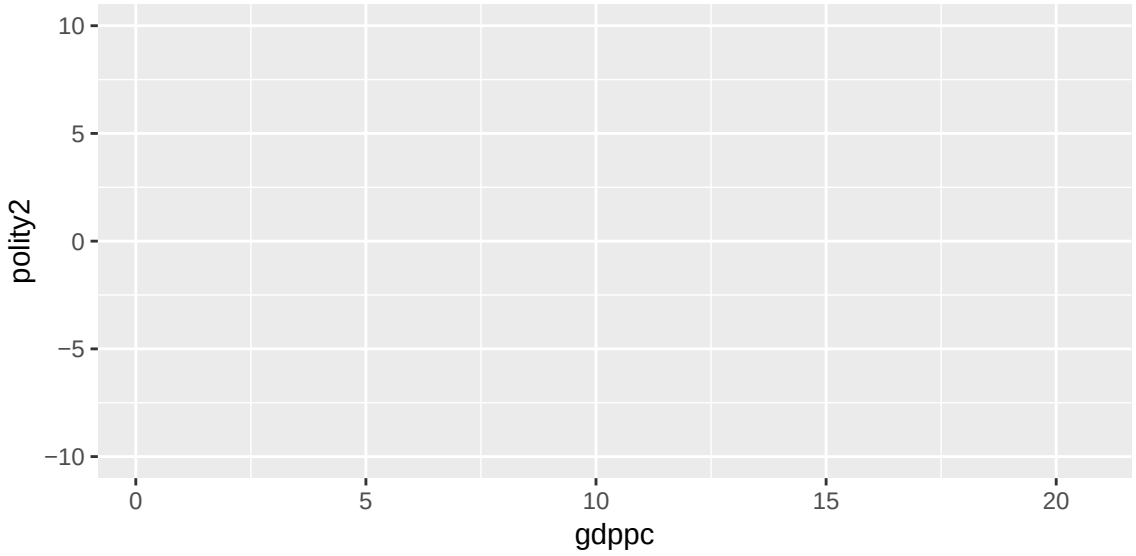
To keep things simple, let's focus on one year of the data: 1998. Let's also create a new variable `log_gdp` that is logged GDP per capita.

```
## Filter the data to include only observations from 1998
cw98 <- civilwar %>%
  filter(year == 1998) %>%
  mutate(log_gdp = log(gdppc))
```

Aesthetic mapping

Aesthetics refer to the variables we want to present. Aesthetic mappings in `ggplot2`, which go inside the `aes()` argument, define the variables that will be represented on our vertical (y) and horizontal (x) axes as well as how the data will be grouped. Let's initialize a `ggplot2` object with `cw98` as our data, democracy scores (`polity2`) as our `x` variable, and logged GDP per capita (`log_gdp`) as our `y` variable:

```
cwplot <- ggplot(data = cw98, mapping = aes(gdppc, polity2))
cwplot
```

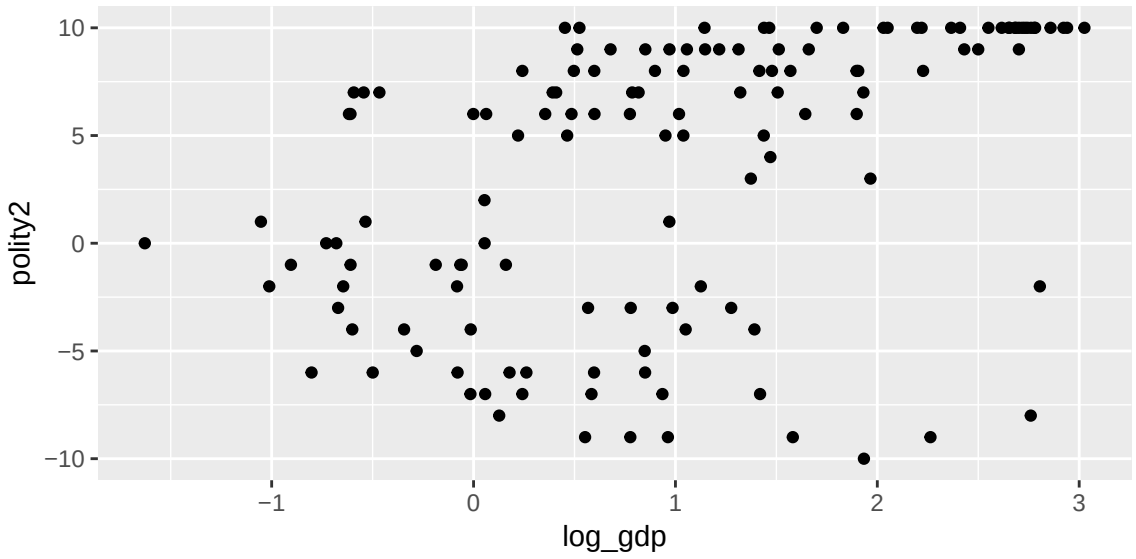


We've loaded our data and variables into a `ggplot2` object (which we have assigned to `cwplot`). You will notice that the plot is not very exciting at this point. We need to tell `ggplot2` the type of visual elements we want to plot.

Geometries

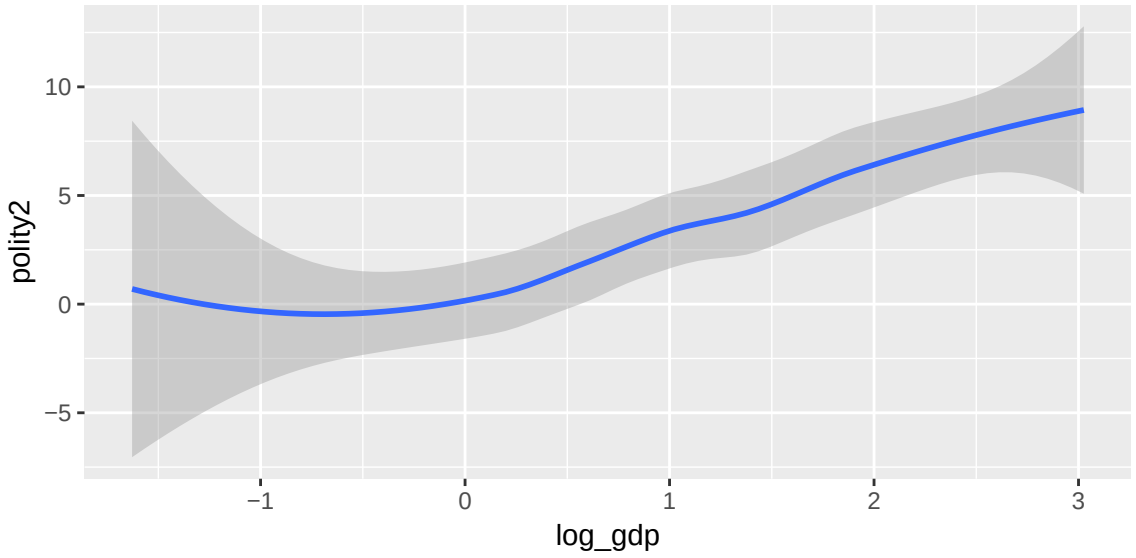
Visual elements in `ggplot2` are called `geoms` (as in geometric objects). The appearance and location of these geoms are controlled by the aesthetic properties. There are many different `geoms` to choose from in `ggplot2`. Let's start with `geom_point()` to create a scatterplot:

```
cwplot <- ggplot(data = cw98, mapping = aes(log_gdp, polity2))
cwplot +
  geom_point()
```



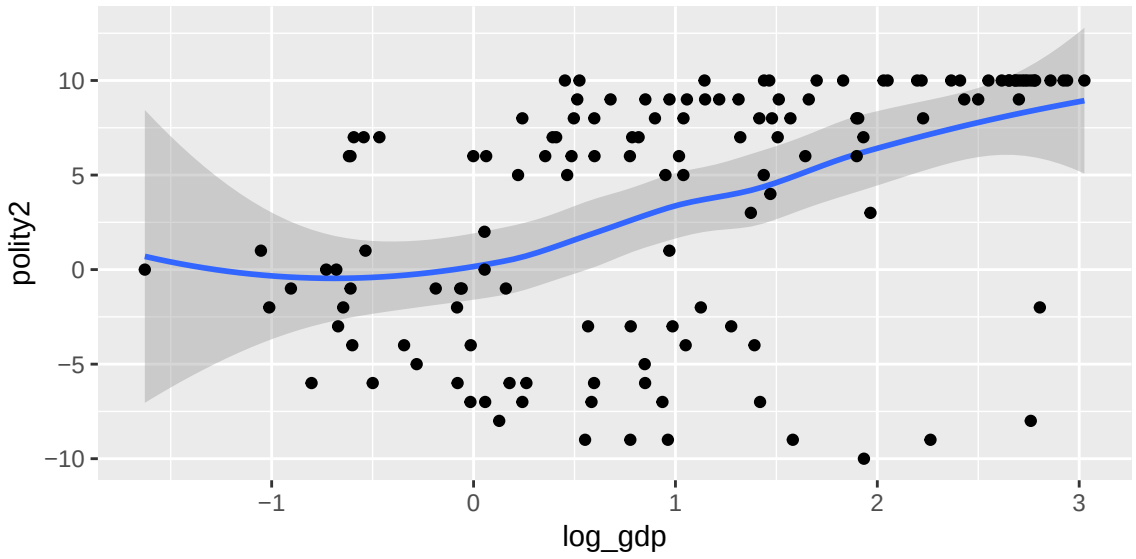
Note that `ggplot2` *inherits* the `x` and `y` variables from the the original `cwplot` object. Let's try a different `geom_` function with our plot:

```
cwplot <- ggplot(data = cw98, mapping = aes(log_gdp, polity2))
cwplot +
  geom_smooth()
```



`geom_smooth` calculates a smoothed line for us that gives us a better sense of the relationship between democracy and logged GDP per alone. Perhaps we want both visual elements (the smoothed line and the points) in the same plot:

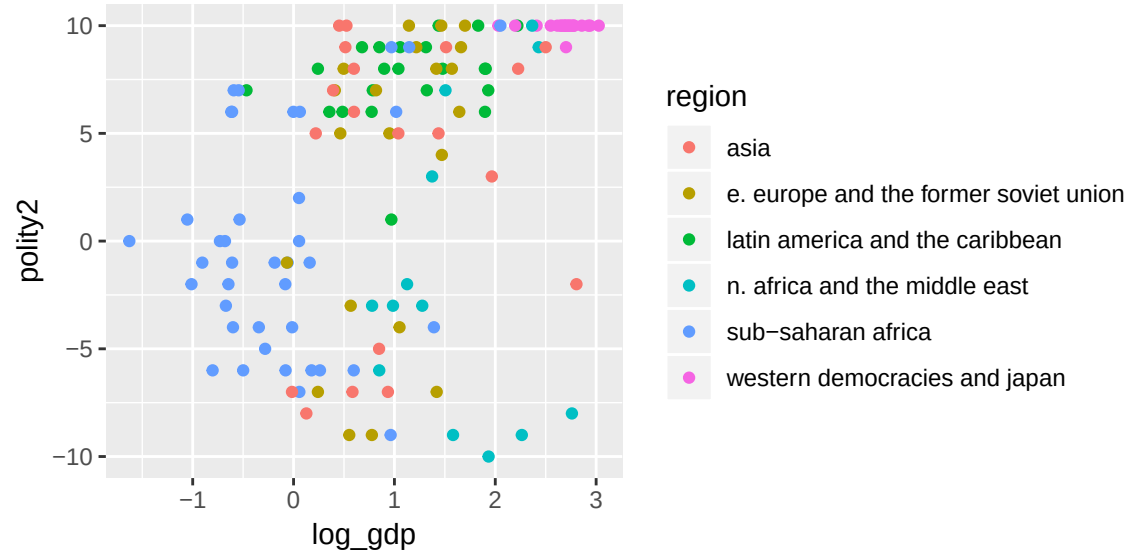
```
cwplot <- ggplot(data = cw98, mapping = aes(log_gdp, polity2))

cwplot +
  geom_smooth() +
  geom_point()
```



Other aesthetic settings

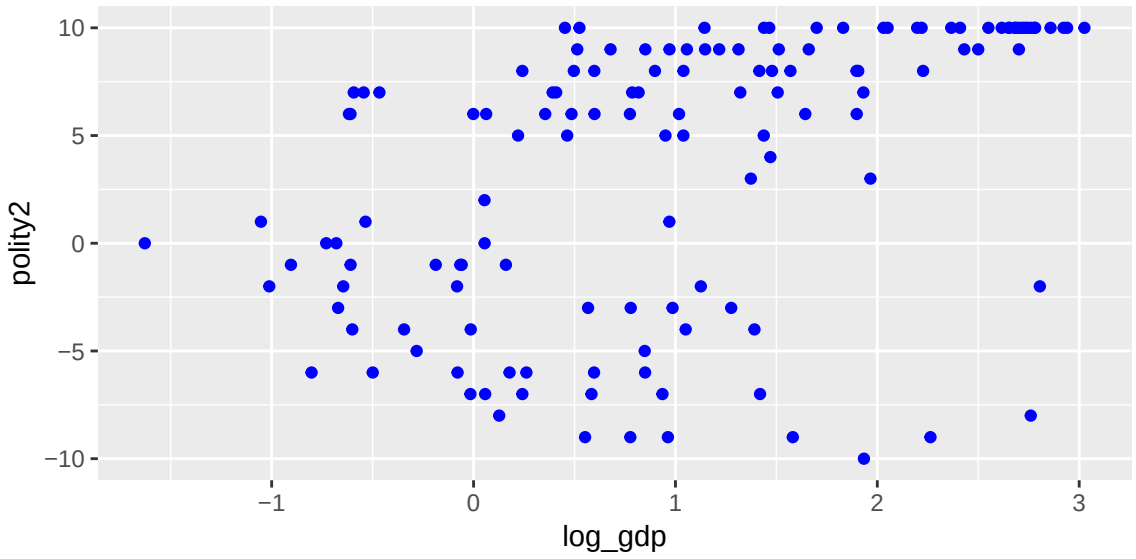
So far, we have mapped GDP per capita to `x` and Polity scores to `y`. We can do much more than this. Some other aesthetics to consider are `colour`, `shape`, and `size`. Let's start with colour:

```
cwplot <- ggplot(cw98, aes(x = log_gdp, y = polity2, colour = region))

cwplot +
  geom_point()
```



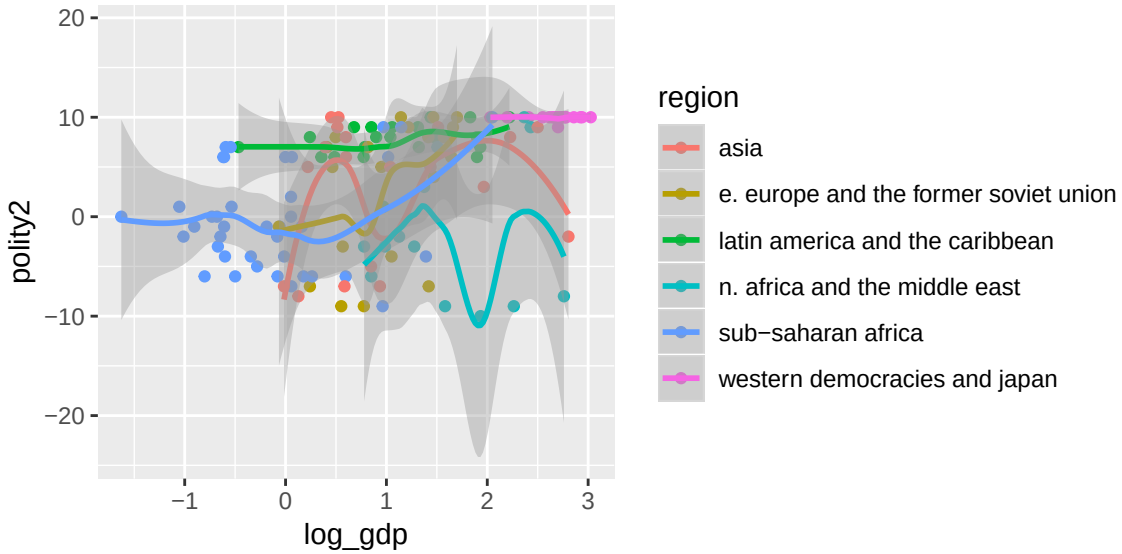
Note that the `color` argument does not give instructions like: colour the points blue. It says: the property color will will represent the variable `region`. If we wanted to turn all the points in in the figure blue, we would do this outside the aesthetic mapping:

```
cwplot <- ggplot(cw98, aes(x = log_gdp, y = polity2))

cwplot +
  geom_point(colour = "blue")
```



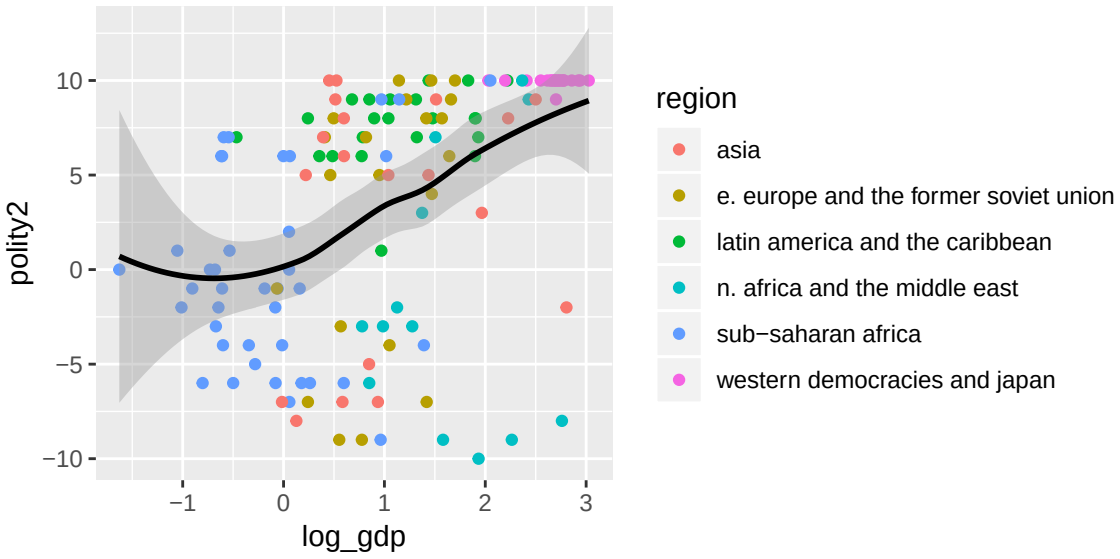
What happens when we include multiple `geoms` with a color mapping specified?

```
cwplot <- ggplot(cw98, aes(x = log_gdp, y = polity2, colour = region))

cwplot +
  geom_point() +
  geom_smooth()
```



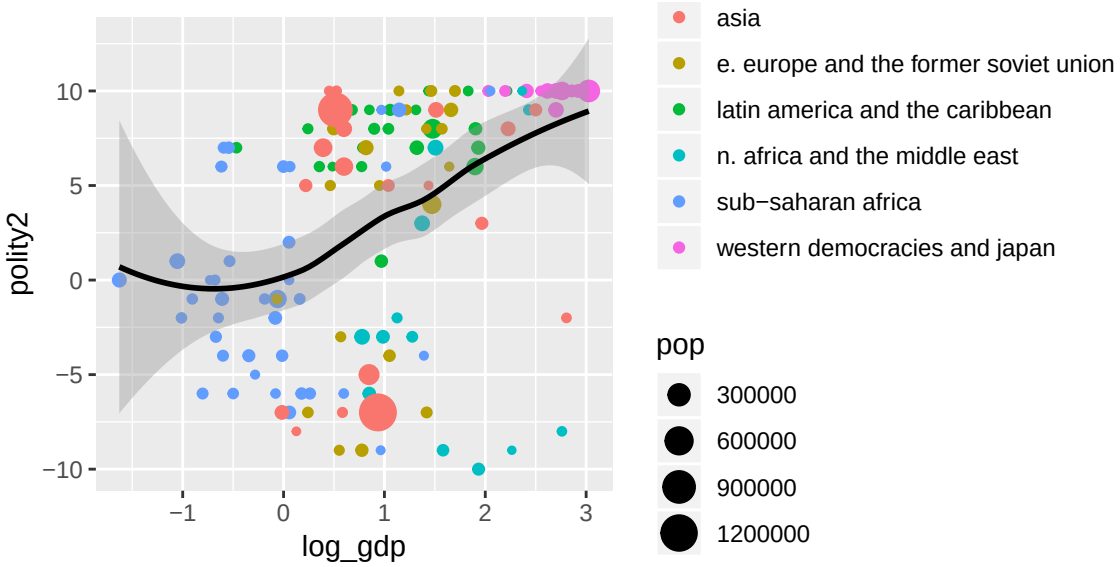
The graph quickly becomes too busy. Instead, we just want to have the points coloured by the `region` variable. To do this, we specify the colour mapping on a geom-by-geom basis.

```
cwplot <- ggplot(cw98, aes(x = log_gdp, y = polity2))

cwplot +
  # Set the colour aesthetic within geom_point() to only colour the points
  geom_point(aes(colour = region)) +
  # Specify colour outside of aes() to set actual colours
  geom_smooth(colour = "black")
```



Perhaps we also want the size of each point (country) on the plot to reflect relative population size:

```
cwplot <- ggplot(cw98, aes(x = log_gdp, y = polity2))

cwplot +
  # Set the colour aesthetic within geom_point() to only colour the points
  geom_point(aes(colour = region, size = pop)) +
  # Specify colour outside of aes() to set actual colours
  geom_smooth(colour = "black")
```



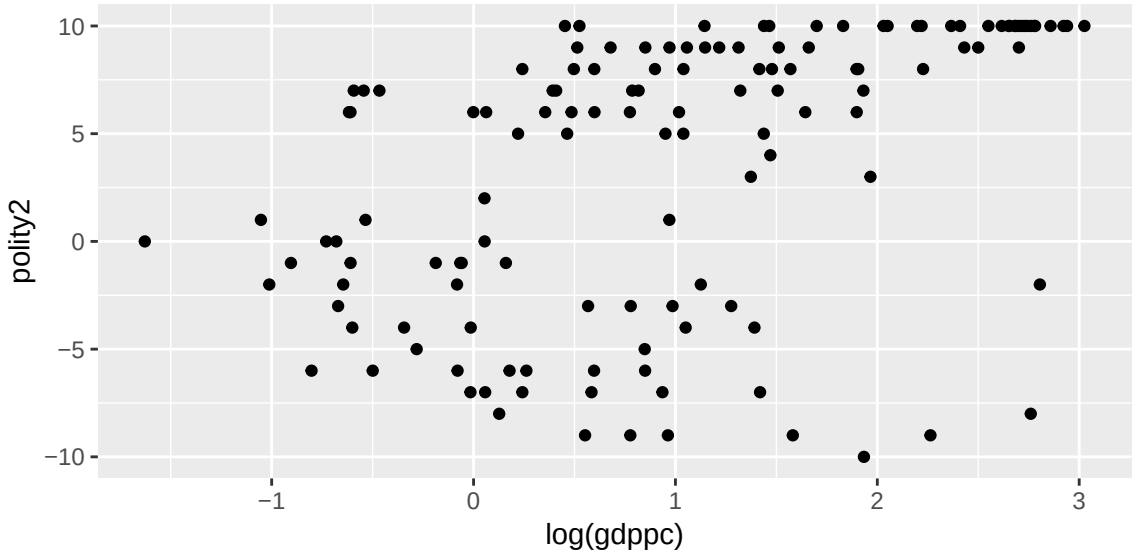
Facetting

```
cw98 <- civilwar %>%
  filter(year==1998)
```

Perhaps we want greater detail than regional trends and instead want to plot country-level trends of democracy

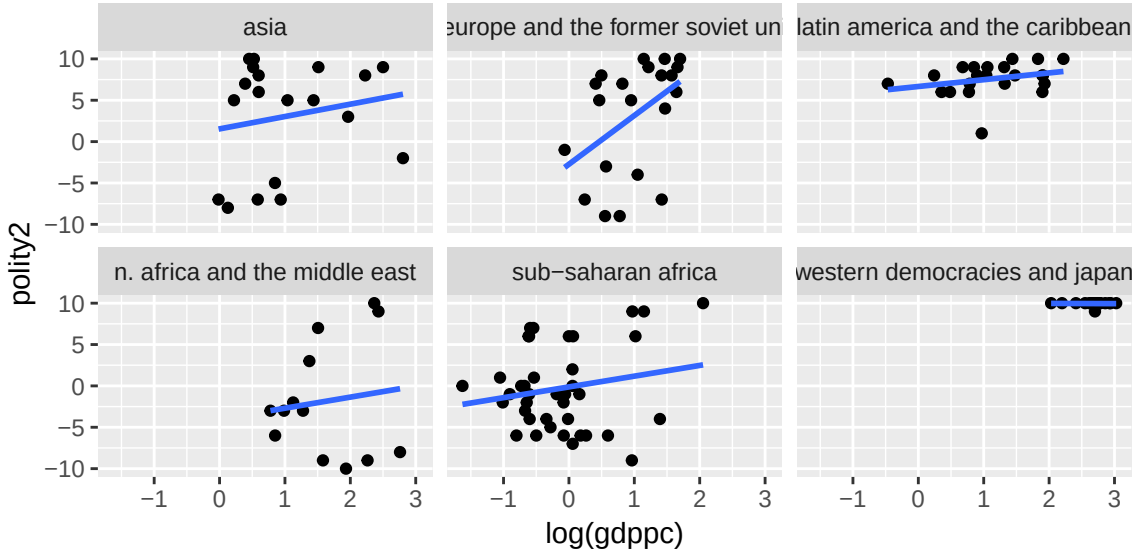
across time. We can work with the original `civilwar` dataframe to do this. Let's use `group` instead of `colour` as our aesthetic for the `country` variable:

```
ggplot(cw98, aes(x = log(gdppc), y = polity2)) +
  geom_point(aes(group = country))
```



Perhaps we want to see how democracy scores and GDP per capita are distributed within each region. We can use the `facet_wrap()` function to create multiple plots that can be compared side-by-side:

```
ggplot(cw98, aes(x = log(gdppc), y = polity2)) +
  geom_point(aes(group = country)) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~region)
```



Line plots

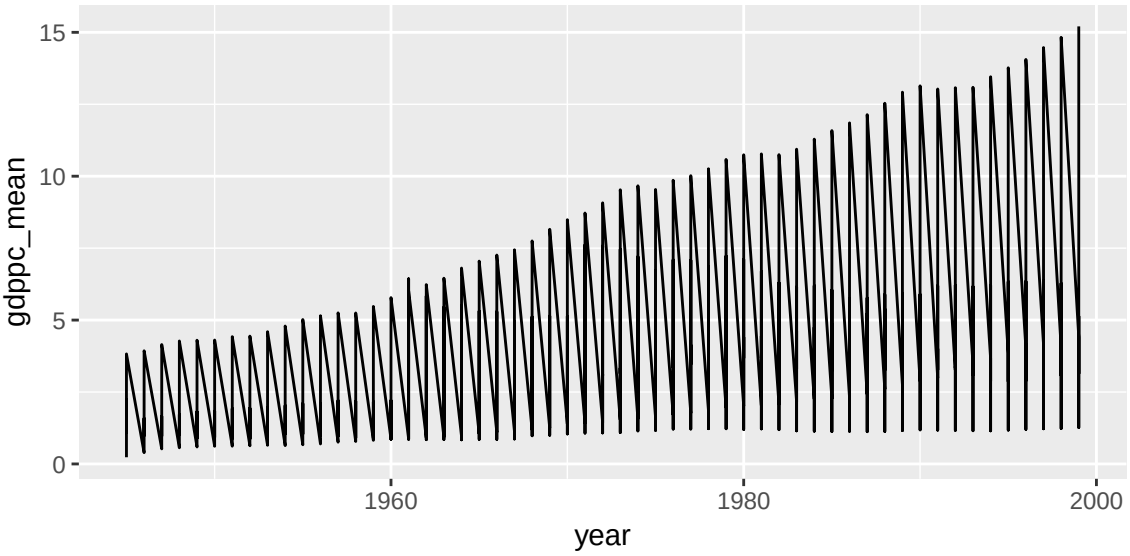
`ggplot2` can be used to create a wide variety of graphs, but sometimes this requires manipulating the data before plotting. For instance, perhaps we want to visualize how GDP per capita evolves over the years in

each region:

```
gdp_trend <- civilwar %>%
  group_by(region, year) %>%
  summarise(gdppc_mean = mean(gdppc, na.rm = T))
```

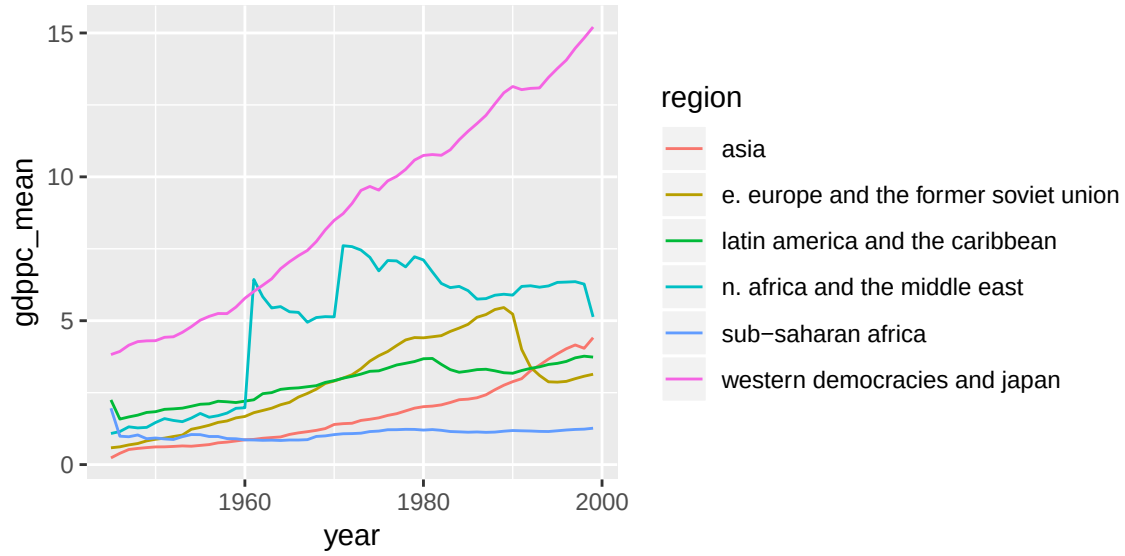
We will use this newly created `gdp_trend` dataframe as the basis for a new plot. Let's try adding `geom_line()` to plot the trend lines:

```
ggplot(gdp_trend, aes(x = year, y = gdppc_mean)) +
  geom_line()
```



This would have worked if there were only one region in the dataset, but we have multiple regions. We need to tell `ggplot2` to create a separate line for each region. We can do this by mapping `region` to the color aesthetic:

```
ggplot(gdp_trend, aes(x = year, y = gdppc_mean)) +
  geom_line(aes(colour = region))
```

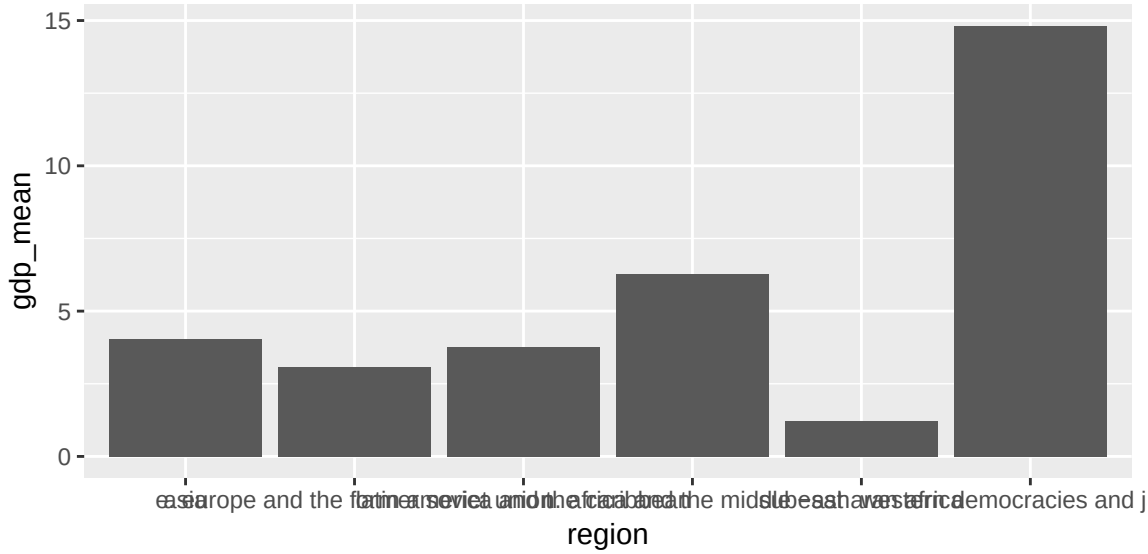


Bar plots

Let's create a bar plot that shows us the mean value of GDP per capita by region in 1998. First we will create a dataframe with the necessary means:

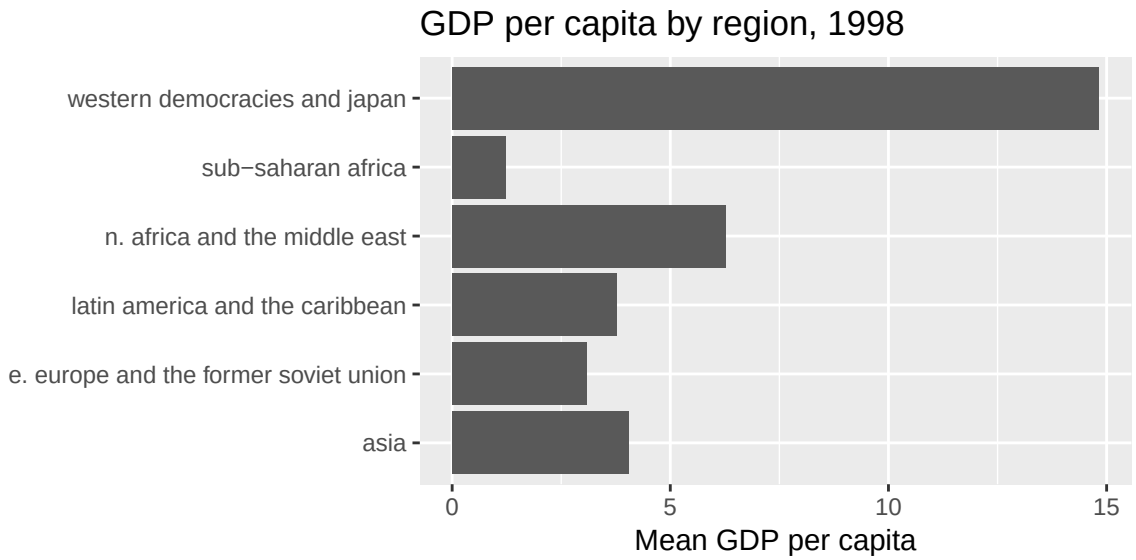
```
gdp98 <- civilwar %>%
  filter(year == 1998) %>%
  group_by(region) %>%
  summarise(gdp_mean = mean(gdppc, na.rm = T))
```

```
ggplot(data = gdp98, aes(x = region, y = gdp_mean)) +
  geom_col()
```



It appears that the labels for the plots overlap, making it hard to discern which bar represents which region. We can flip the coordinates using `coord_flip()` to make the bars horizontal rather than vertical, which creates more room for the labels. We can also clean up the axis labels with the `labs()` function:

```
ggplot(data = gdp98, aes(x = region, y = gdp_mean)) +
  geom_col() +
  coord_flip() +
  labs(title = "GDP per capita by region, 1998", x = NULL, y = "Mean GDP per capita")
```

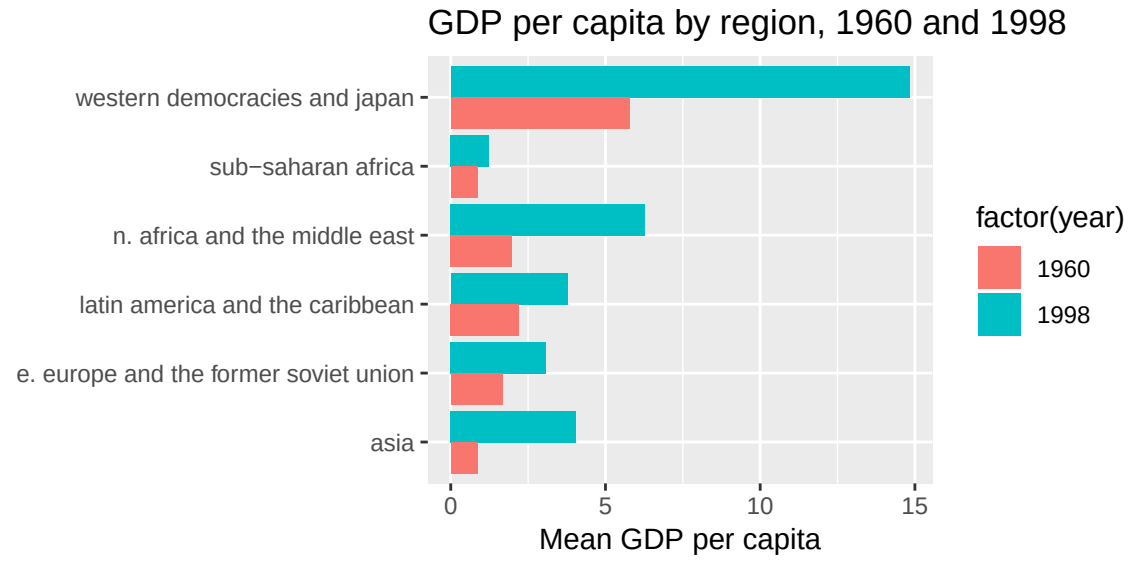


Perhaps we want to compare how GDP growth has changed between 1960 and 1998 for each region. To do this, we create another dataframe with the required information:

```
gdp60to98 <- civilwar %>%
  filter(year %in% c(1960, 1998)) %>%
  group_by(region, year) %>%
  summarise(gdp_mean = mean(gdppc, na.rm = T))
```

We use `geom_col()` again, but now use the argument `position = "dodge"` in conjunction with `fill = factor(year)` to create two separate bars per region. Note that we use `factor(year)` rather than just `year` so that `ggplot2` knows we are dealing with categories and not numeric values.

```
ggplot(data = gdp60to98, aes(x = region, y = gdp_mean, fill = factor(year))) +
  geom_col(position = "dodge") +
  coord_flip() +
  labs(title = "GDP per capita by region, 1960 and 1998",
       x = NULL,
       y = "Mean GDP per capita")
```



Additional resources

If you wish to continue your journey with `ggplot2`, we suggest the following resources:

- `ggplot2` cheat sheet
- Kieran Healy's "Data Visualization: A practical introduction" is an excellent resource on how to make the most out of `ggplot2` as well as the principles of good data visualization