SPEC Lab R Resources: Data Visualization 1 Walkthrough pt 2 – Aesthetics

Alix Ziff, Gaea Morales, Zachary Johnson, and Ben Graham
Based on earlier materials by Therese Anders

Summer 2021, Version: April 14

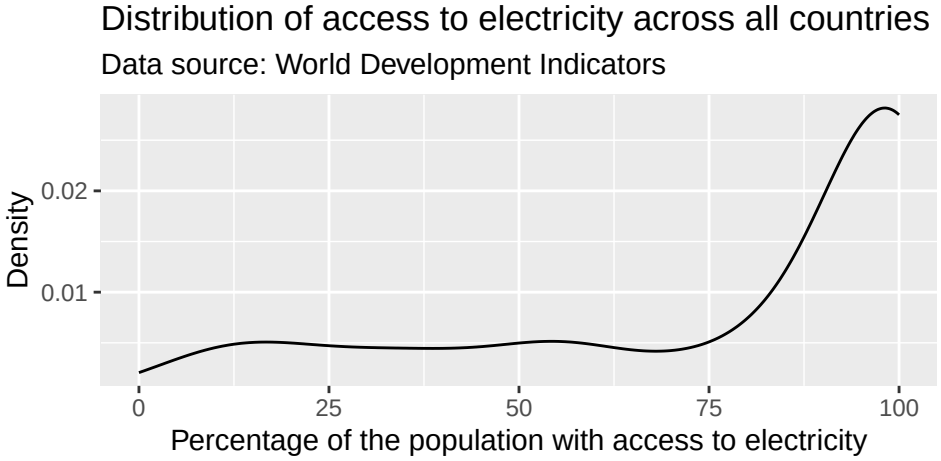
Making pretty pictures

The default graphs we have produced so far are not (yet) ready for publication. In particular, they lack informative labels. In addition, we might want to change the appearance of the graph in terms of size, color, linetype, etc.

Labeling our pretty pictures: title, subtitle, and axes titles

We can specify titles and axes labels within the `labs()` argument.

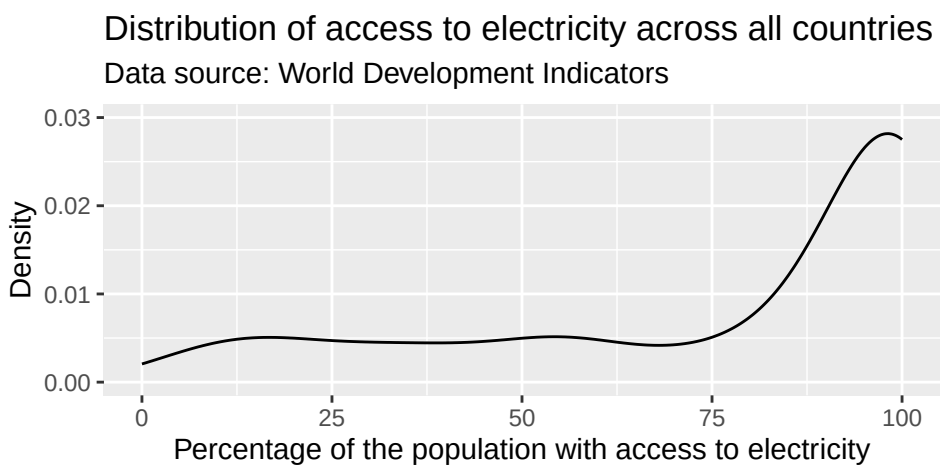
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")
```



How to change the range of the axes

By default, `ggplot()` adjusted the y-axis to start not at zero but at approximately 0.2 to reduce the amount of empty space in the plot. This might confuse the viewer, as it visually underestimates the height of the density curve–in particular for lower values of the variable. We can manually adjust the range of the axes using the `coord_cartesian()` parameter.

```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```

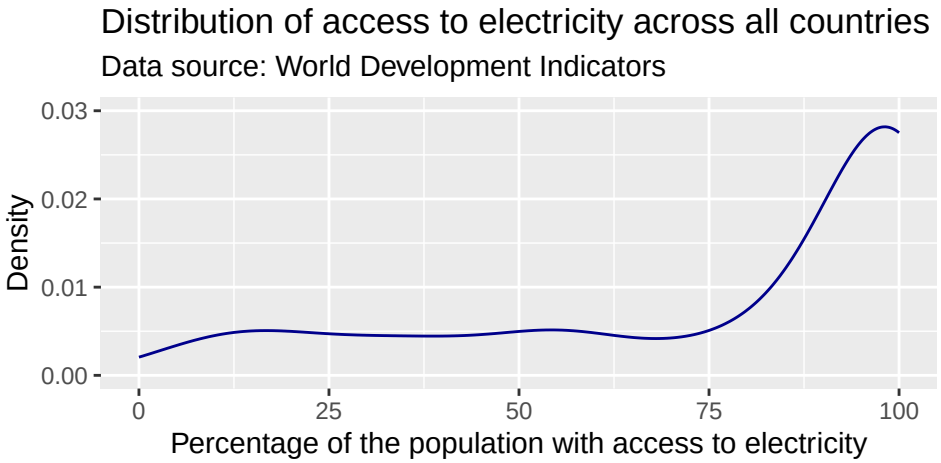


Caution!! You may see code in the Lab that uses `scale_y_continuous(limits = c(0, 0.03))` instead of `coord_cartesian(ylim = c(0, 0.03))`. Note that these are not the same. `coord_cartesian()` only adjusts the range of the axes (it "zooms" in and out), while `scale_y_continuous(limits = c())` subsets the data. For density plots, this does not make a difference. But there are other examples where it alters the actual shape of the graph, rather than just the part of the graph that is visible.

Color me!

Any changes to the appearance of the curve itself are made within the argument that specifies the geometric object to be plotted, here `geom_line()`. `R` knows many colors by name; for a great overview see http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf.

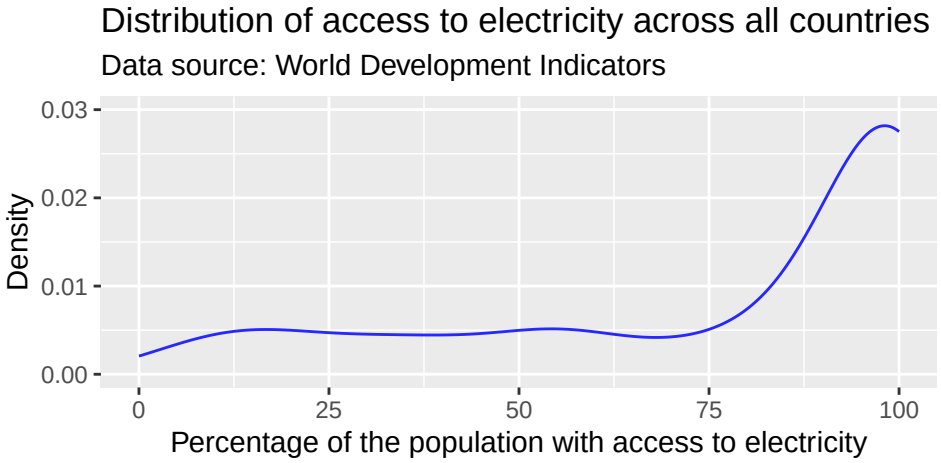
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density", color = "darkblue") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```



We can also use hexadecimal or RGB (red, green, blue) strings to specify colors. There are plenty of online tools to pick colors and extract hexadecimal or RBG strings. One of my favorites is http://www.colorhexa.com. This online tool allows you to specify a color name, hexadecimal, or RGB string, and returns information on color schemes, complementary colors, as well as alternative shades, tints, and tones. It also offers a color blindness simulator.

Suppose, I like the general tone of the darkblue color above, but am worried that it is a bit too dark for my plot. I enter the color "darkblue" into the search field at http://www.colorhexa.com and look for a brighter alternative. Suppose I really like the color displayed in the second tile from the left on the tints scale. I can extract this color's hexadecimal value of `#2727ff` by hovering over the tile of that color.

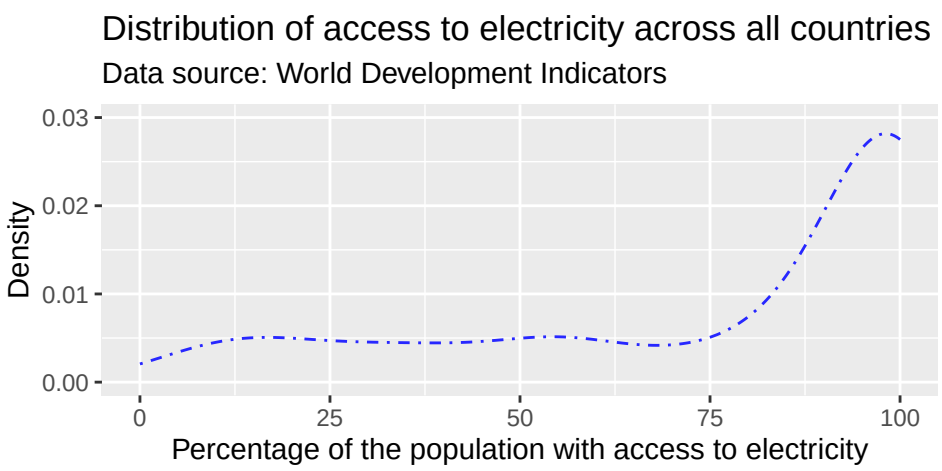
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density", color = "#2727ff") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```



Line Looks: type

We can adjust the type of the line via the `linetype` parameter within `geom_line()`. For an overview of line types see http://sape.inf.usi.ch/quick-reference/ggplot2/linetype.

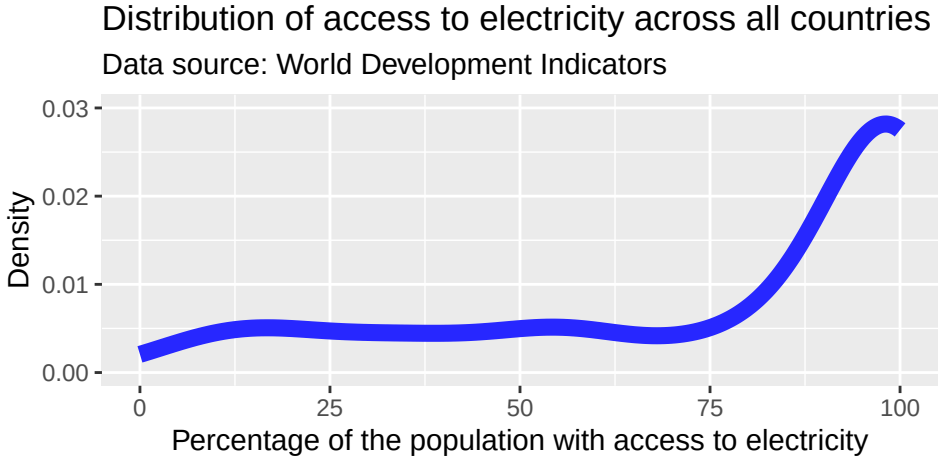
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density", color = "#2727ff", linetype = "dotdash") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```



Line Looks: width

We can adjust the width of the line via the `size` parameter within `geom_line()`. Note that the `size` parameter is universal in the way that it controls line width in line plots and point size in scatter plots.

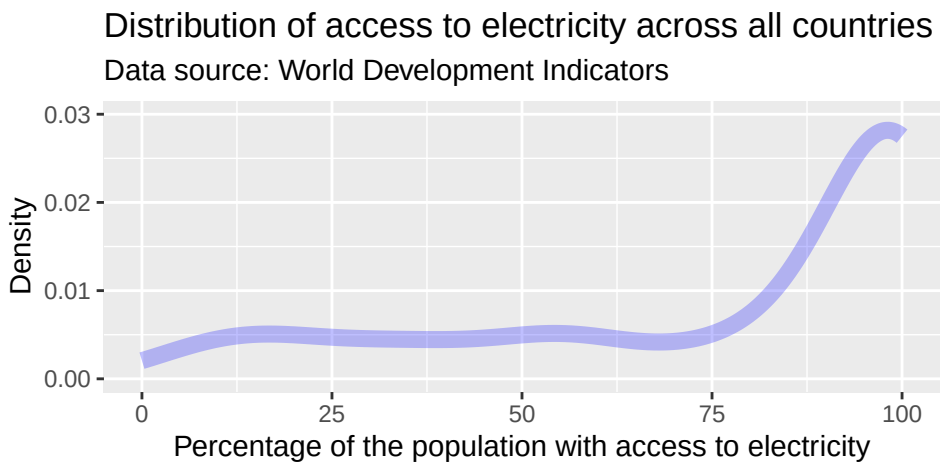
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density", color = "#2727ff", size = 3) +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```



Line Looks: opacity

We can adjust the opacity of the line via the `alpha` parameter within any geometric object. The `alpha` parameter ranges between zero and one. Adjusting the opacity of the geometric objects is especially important when plotting multiple lines (or objects) in the same graph to reduce overplotting.

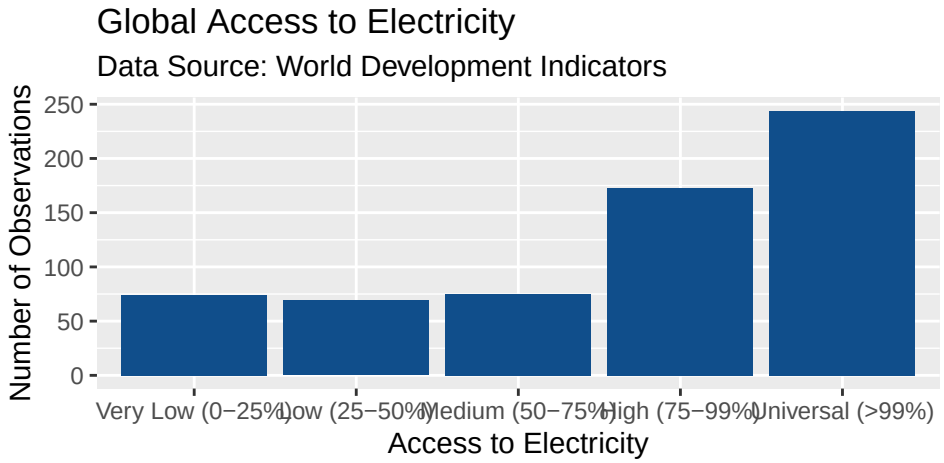
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density", color = "#2727ff", size = 3, alpha = 0.3) +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density") +
  coord_cartesian(ylim = c(0, 0.03))
```



Now a Prettier Bar Plot

Let's take the bar plot we made in the first DV1 walkthrough and pretty it up a bit.

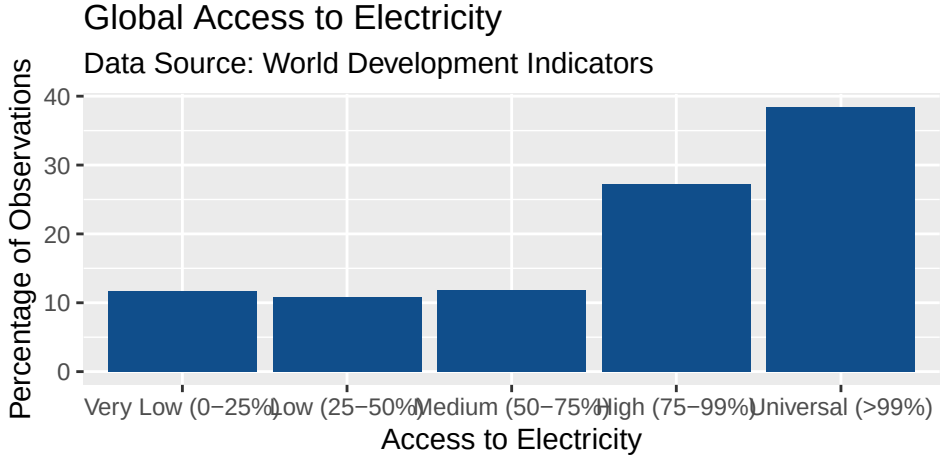
```
dat2 <- dat %>%
  mutate(electricity_pop_ord = cut(electricity_pop, breaks = c(0,25, 50, 75, 99, 100)))%>%
  filter(electricity_pop_ord != "NA")

ggplot(dat2, aes(x = electricity_pop_ord)) +
  geom_bar(fill = "dodgerblue4") +
  labs(title = "Global Access to Electricity",
       subtitle = "Data Source: World Development Indicators",
       y = "Number of Observations",
       x = ("Access to Electricity")) +
      scale_x_discrete(labels=c("Very Low (0-25%)","Low (25-50%)","Medium (50-75%)","High (75-99%)", "U
```



To make this a bit nicer still, we might think about rescaling the height of each bar data by share of observations (i.e. having percentage of cases on the Y axis instead of number of observations).

To get percentages on the Y axis, we have to use `aes()` to tell R that we don't want to use a count as the measure of Y, which is the default for 'geom_bar()', but rather to take the count and divide it by the total number of observations. As with anything, there are a lot of different ways we can right this code, but `..count..` gives us one method.

```
ggplot(dat2, aes(x = electricity_pop_ord)) +
  geom_bar(aes(y =(100*(..count..)/sum(..count..))), fill = "dodgerblue4") +
  labs(title = "Global Access to Electricity",
       subtitle = "Data Source: World Development Indicators",
       y = "Percentage of Observations",
       x = ("Access to Electricity")) +
      scale_x_discrete(labels=c("Very Low (0-25%)","Low (25-50%)","Medium (50-75%)","High (75-99%)", "U
```



Graphing distributions across groups

Color Coordination

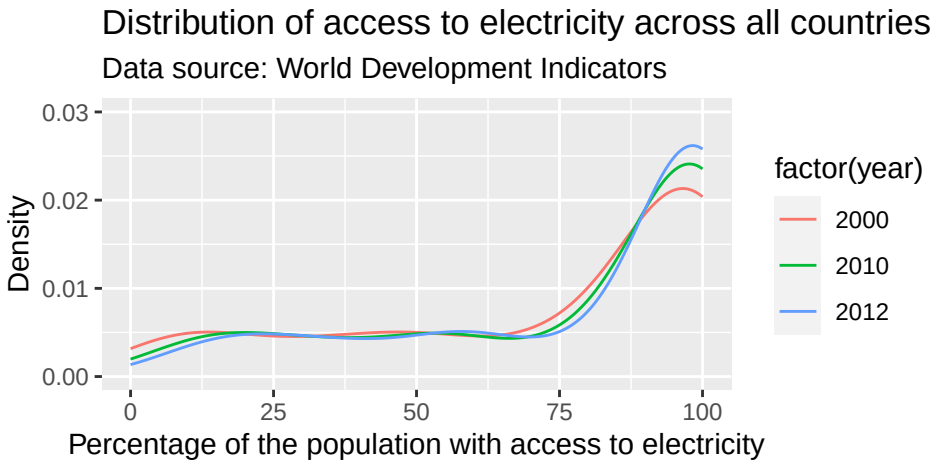
Sometimes, we want to compare distributions across different groups in our data set. Suppose, we wanted to assess how the distribution of the access to electricity changes over time. We have three years of observations for our variable: 2000, 2010, and 2012.

```
table(dat$year[!is.na(dat$electricity_pop)])
```

```
##
## 2000 2010 2012
##  212  212  212
```

We pass a separate color to the distribution of the `electricity_pop` for each year by specifying the `color` parameter within the aesthetics. Since the year parameter is currently specified as a numerical variable, we also need to specify that `ggplot` should treat it as a `factor` (i.e. a categorical variable).

```
ggplot(dat, aes(x = electricity_pop, color = factor(year))) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")  +
  coord_cartesian(ylim = c(0, 0.03))
```



Question 4 Review: What is the difference between specifying the `color` parameter outside the `aes()` argument versus within the `aes()` argument?

See answer at the bottom of this walkthrough

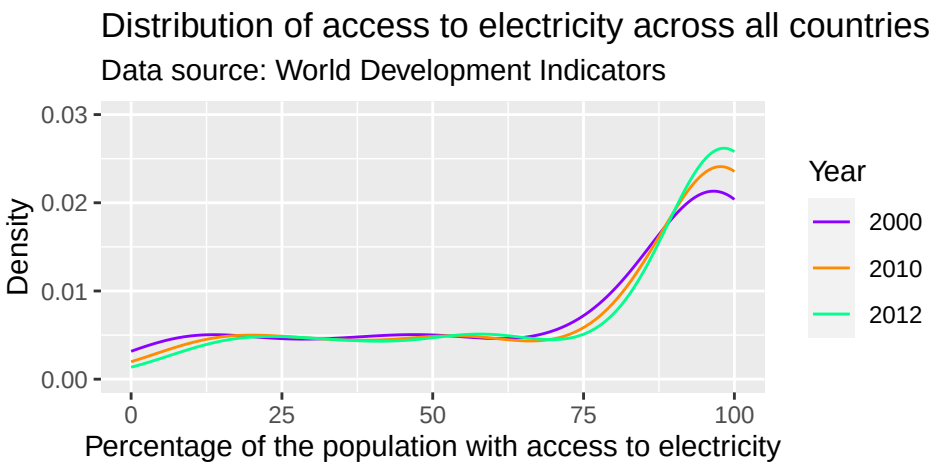
Question 5 How would you interpret this plot? How did the access to electricity change over time?

See answer at the bottom of this walkthrough

We can use custom colors to distinguish between the three years using the `scale_color_manual()` function. This will change the colors in both the plot and the legend. Let us choose triadic colors to "darkorange" from http://www.colorhexa.com. Within the `scale_color_manual()` argument, we can also specify a name and labels for the legend.

```
ggplot(dat, aes(x = electricity_pop, color = factor(year))) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")  +
  coord_cartesian(ylim = c(0, 0.03)) +
  scale_color_manual(values = c("#8c00ff",
                                "#ff8c00",
                                "#00ff8c"),
```

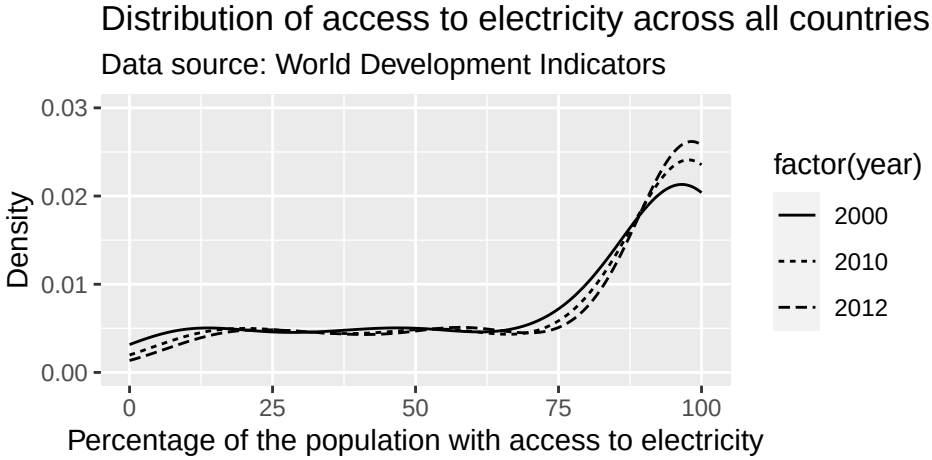
```
                    name = "Year")
```



Line Looks

Many academic journals will only accept graphs on a gray scale. This means that color will not be enough to differentiate the three lines. We can use different line types instead by specifying the `linetype` parameter within the `aes()` argument.. This also makes the graph more color blind friendly.

```
ggplot(dat, aes(x = electricity_pop, linetype = factor(year))) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")  +
  coord_cartesian(ylim = c(0, 0.03))
```

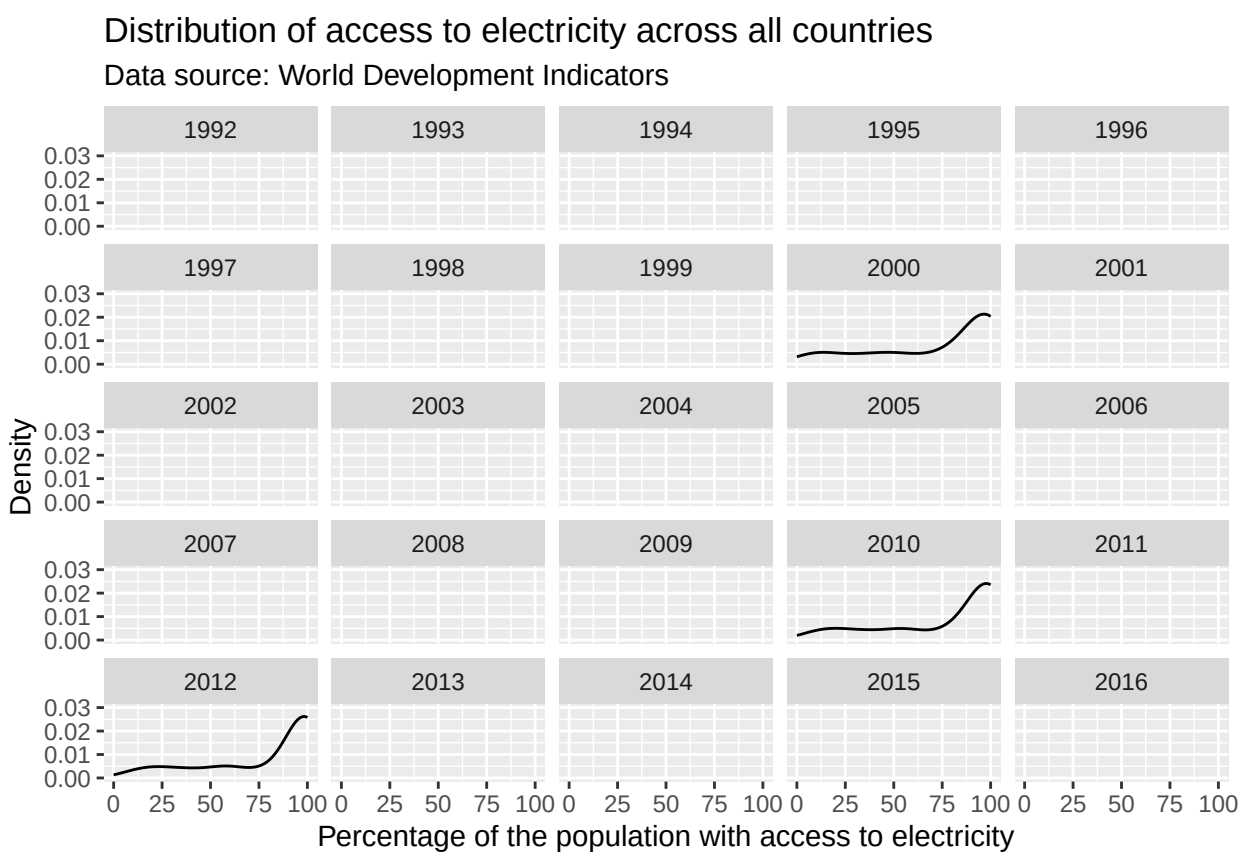


Faceting

Another option to graph different groups is to use faceting. This means to plot each value of the variable upon which we facet in a different panel within the same plot. Here, we will use the `facet_wrap()` function. We could also use the `facet_grid()` which allows faceting across more than one variable.

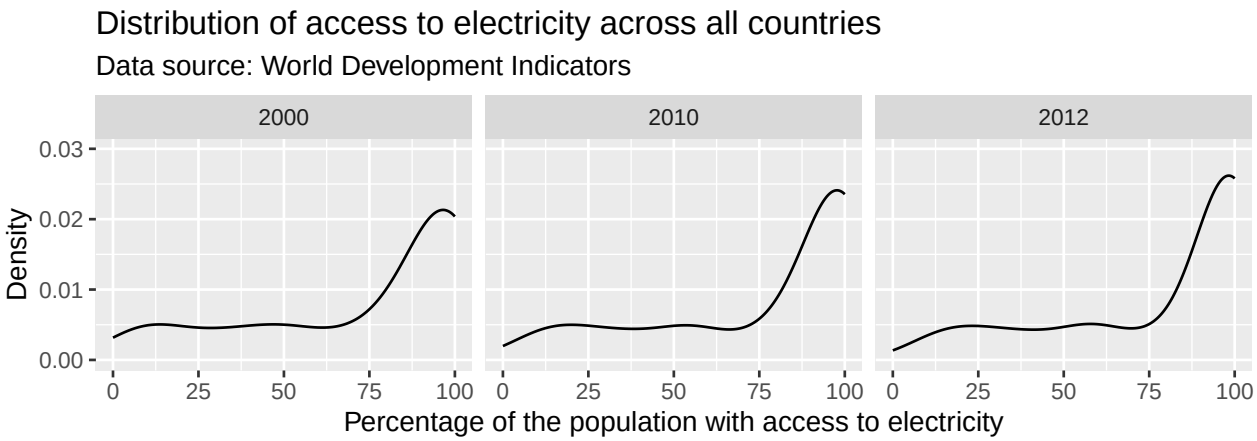
```
ggplot(dat, aes(x = electricity_pop)) +
  geom_line(stat = "density") +
```

```
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")  +
  coord_cartesian(ylim = c(0, 0.03)) +
  facet_wrap(~ year)
```



The `facet_wrap()` function draws a separate plot for each year in the data set. Since we only have data for 2000, 2010, and 2012, we subset the plot to omit empty panels. We can either create a new subsample data frame, or use the `subset()` command directly within `ggplot()`. Here, we explicitly choose the years 2000, 2010, and 2012. Alternatively, we could also subset to all non-missing observations on the variable `electricity_pop`.

```
ggplot(subset(dat, year %in% c(2000, 2010, 2012)), aes(x = electricity_pop)) +
  geom_line(stat = "density") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Percentage of the population with access to electricity",
       y = "Density")  +
  coord_cartesian(ylim = c(0, 0.03)) +
  facet_wrap(~ year)
```



Question 6 For this particular application, which alternative to show the distribution of the variable across groups do you think is the most appropriate? Why?

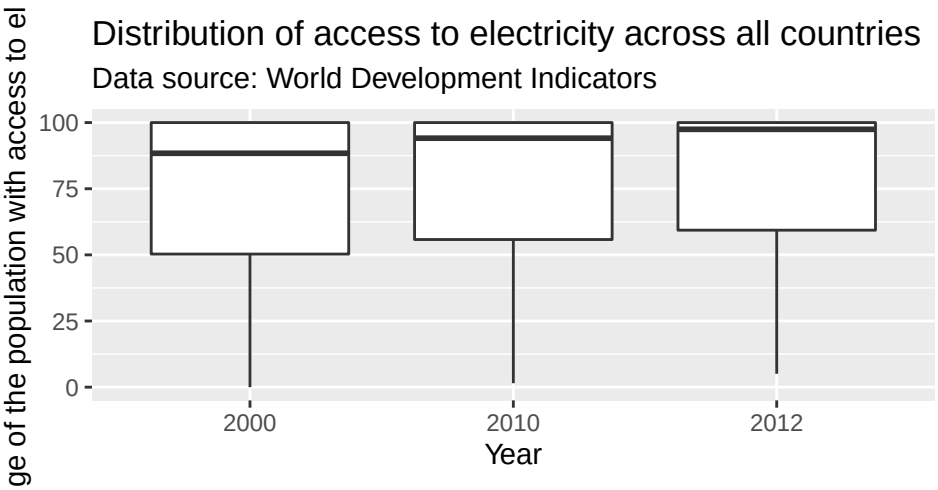
Boxplots

Another way to show the distribution of variables across groups are boxplots. Boxplots graph different properties of a distribution:

- The borders of the box denote the 25th and 75th percentile.
- The line within the box denotes the median.
- The position of the whiskers (vertical lines) denote the first quartile value minus 1.5 times the interquartile range and the third quartile value plus 1.5 times the interquartile range. We will not go into details here.
- Dots denote outliers (values that lie outside the whiskers), if applicable.

In `ggplot2` we can graph boxplots across multiple variables using the `geom_boxplot()` geometric object.

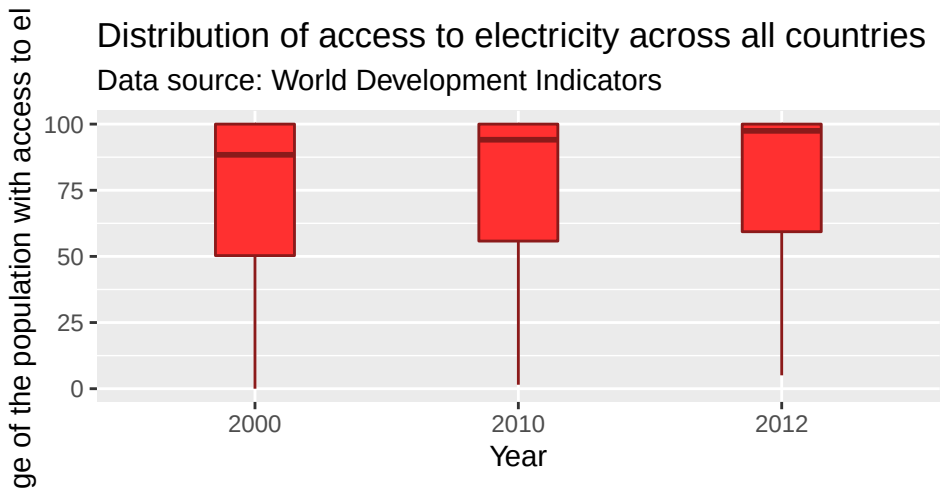
```
ggplot(subset(dat, year %in% c(2000, 2010, 2012)),
       aes(x = factor(year), y = electricity_pop)) +
  geom_boxplot() +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Year",
       y = "Percentage of the population with access to electricity")
```



We can change the appearance of the boxplot using standard graphing parameters such as `color`, `fill`, and

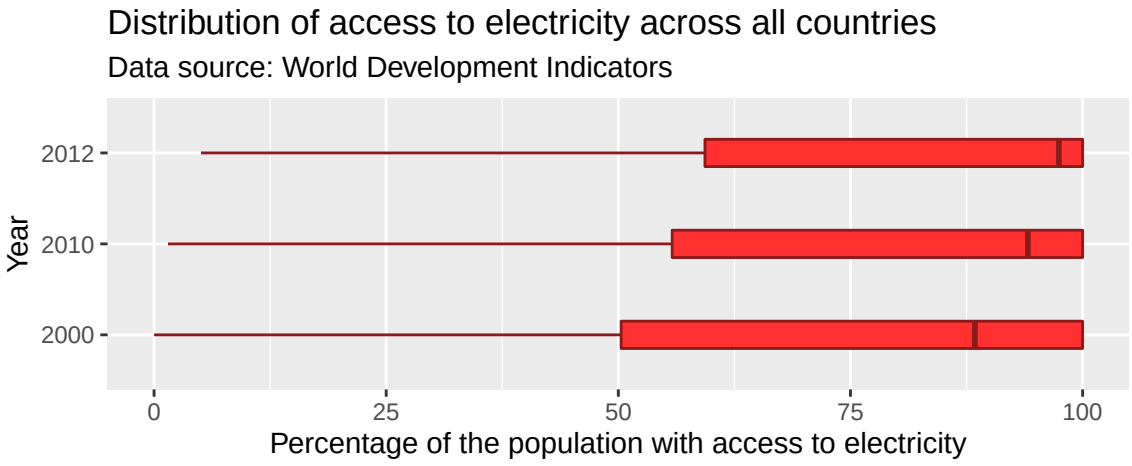
width.

```
ggplot(subset(dat, year %in% c(2000, 2010, 2012)),
       aes(x = factor(year), y = electricity_pop)) +
  geom_boxplot(width = 0.3, color = "firebrick4", fill = "firebrick1") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Year",
       y = "Percentage of the population with access to electricity")
```



We can change the orientation of the plot with the `coord_flip()` command, which will flip the axes.

```
ggplot(subset(dat, year %in% c(2000, 2010, 2012)),
       aes(x = factor(year), y = electricity_pop)) +
  geom_boxplot(width = 0.3, color = "firebrick4", fill = "firebrick1") +
  labs(title = "Distribution of access to electricity across all countries",
       subtitle = "Data source: World Development Indicators",
       x = "Year",
       y = "Percentage of the population with access to electricity") +
  coord_flip()
```



Saving plots

We can output your plots to many different format using the `ggsave()` function, including but not limited to `.pdf`, `.jpeg`, `.bmp`, `.tiff`, or `.eps`. Here, we output the graph as a Portable Network Graphics (.png) file. We can specify the size of the output graph as well as the resolution in dots per inch (dpi). If no graph is specified, `ggsave()` will save the last graph that was executed. For us, this is the boxplot in horizontal orientation. If we no not specify the complete file path, the plot will be saved to your working directory.

```
ggsave("boxplot_horizontal.png", width = 6, height = 3, dpi = 400)
```

ANSWERS

Question 4: Answer

If the color parameter is specified outside the `aes()` argument, one color is passed all geometric objects of the same type. If the color parameter is specified within the `aes()` argument, different colors are passed to each value of the variable that is passed to the `color` parameter. A separate geometric object will be plotted for value–each in a different color.

Question 5: Answer

Over time, the amount of country-years with high levels of access to electricity (90% to 100% of the population) has increased. In later years, there are fewer observations at very low levels of access to electricity and there are fewer country-years in which approximately 75% to 90% of the population had access to electricity.