09 Data Visualization in practice

Soyoung Park

Pusan National University
Department of Statistics

Data Visualization in practice

In this chapter, we will demonstrate how relatively simple **ggplot2** code can create insightful and aesthetically pleasing plots. As motivation we will create plots that help us better understand trends in world health and economics.

Case study: new insights on poverty

Hans Rosling was the co-founder of the Gapminder Foundation, an organization dedicated to educating the public by using data to dispel common myths about the so-called developing world. The organization uses data to show how actual trends in health and economics contradict the narratives that emanate from sensationalist media coverage of catastrophes, tragedies, and other unfortunate events.

Case study: new insights on poverty

Hans Rosling conveyed actual data-based trends in a dramatic way of his own, using effective data visualization. Specifically, in this section, we use `gapminder` dataset provided in **dslabs** to attempt to answer the following two questions:

1. Is it a fair characterization of today's world to say it is divided into western rich nations and the developing world in Africa, Asia, and Latin America?

2. Has income inequality across countries worsened during the last 40 years?

Case study: new insights on poverty

```
library(tidyverse)
library(dslabs)
data(gapminder)
# gapminder %>% as_tibble()
# ?gapminder
```

Hans Rosling's quiz

For each of the six pairs of countries below, which country do you think had the highest child mortality rates in 2015? Which pairs do you think are most similar?

1. Sri Lanka or Turkey
2. Poland or South Korea
3. Malaysia or Russia
4. Pakistan or Vietnam
5. Thailand or South Africa

Hans Rosling's quiz

To answer these questions with data, we can use **dplyr**. For example, for the first comparison we see that:

```
gapminder %>%
  filter(year == 2015 & country %in% c("Sri Lanka","Turkey")) %>%
  select(country, infant_mortality)
```

Turkey has the higher infant mortality rate.

Hans Rosling's quiz

We can use above code on all comparisons and find the following:

country	infant mortality	country	infant mortality
Sri Lanka	8.4	Turkey	11.6
Poland	4.5	South Korea	2.9
Malaysia	6.0	Russia	8.2
Pakistan	65.8	Vietnam	17.3
Thailand	10.5	South Africa	33.6

Hans Rosling's quiz

We see that the European countries on this list have higher child mortality rates.

It turns out that when Hans Rosling gave this quiz to educated groups of people, the average score was less than 2.5 out of 5, worse than what they would have obtained had they guessed randomly.

This implies that more than ignorant, we are misinformed. In this chapter we see how data visualization helps inform us.

Scatterplots

In order to analyze this world view, our first plot is a scatterplot of life expectancy versus fertility rates (average number of children per woman). We start by looking at data from about 50 years ago, when perhaps this view was first cemented in our minds.

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy)) +
  geom_point()
```

Scatterplots

Most points fall into two distinct categories:

1. Life expectancy around 70 years and 3 or fewer children per family.
2. Life expectancy lower than 65 years and more than 5 children per family.

Scatterplots

To confirm that indeed these countries are from the regions we expect, we can use color to represent continent.

```
filter(gapminder, year == 1962) %>%
  ggplot( aes(fertility, life_expectancy, color = continent))
  geom_point()
```

In 1962, "the West versus developing world" view was grounded in some reality. Is this still the case 50 years later?

Faceting

We could easily plot the 2012 data in the same way we did for 1962. To make comparisons, however, side by side plots are preferable. In **ggplot2**, we can achieve this by *faceting* variables

To achieve faceting, we add a layer with the function `facet_grid`, which automatically separates the plots. Here is an example of a scatterplot with facet_grid added as the last layer:

```
filter(gapminder, year%in%c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_grid(continent~year)
```

Faceting

We see a plot for each continent/year pair. However, this is just an example and more than what we want, which is simply to compare 1962 and 2012.

In this case, there is just one variable and we use `.` to let facet know that we are not using one of the variables:

```
filter(gapminder, year%in%c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_grid(. ~ year)
```

Faceting

This plot clearly shows that the majority of countries have moved from the *developing world* cluster to the *western world* one.

In 2012, the western versus developing world view no longer makes sense. This is particularly clear when comparing Europe to Asia, the latter of which includes several countries that have made great improvements.

facet_wrap

To explore how this transformation happened through the years, we can make the plot for several years. For example, we can add 1970, 1980, 1990, and 2000. If we do this, we will not want all the plots on the same row. The function `facet_wrap` permits us to do this instead of `facet_grid`.

```
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_wrap(~year)
```

This plot clearly shows how most Asian countries have improved at a much faster rate than European ones.

Fixed scales for better comparisons

The default choice of the range of the axes is important. When not using `facet`, this range is determined by the data shown in the plot.

When using `facet`, this range is determined by the data shown in all plots and therefore kept fixed across plots. This makes comparisons across plots much easier.

Fixed scales for better comparisons

For example, in the above plot, we can see that life expectancy has increased and the fertility has decreased across most countries.

We see this because the cloud of points moves. This is not the case if we adjust the scales:

```
filter(gapminder, year%in%c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_wrap(. ~ year, scales = "free")
```

In the plot above, we have to pay special attention to the range to notice that the plot on the right has a larger life expectancy.

Time series plots

Time series plots have time in the x-axis and an outcome or measurement of interest on the y-axis. For example, here is a trend plot of United States fertility rates:

```
gapminder %>%
  filter(country == "United States") %>%
  ggplot(aes(year, fertility)) +
  geom_point()
```

We see that the trend is not linear at all. Instead there is sharp drop during the 1960s and 1970s to below 2. Then the trend comes back to 2 and stabilizes during the 1990s.

Time series plots

When the points are regularly and densely spaced, as they are here, we create curves by joining the points with lines, to convey that these data are from a single series, here a country. To do this, we use the `geom_line` function instead of `geom_point`.

```
gapminder %>%
  filter(country == "United States") %>%
  ggplot(aes(year, fertility)) +
  geom_line()
```

This is particularly helpful when we look at two countries.

Time series plots

If we subset the data to include two countries, one from Europe and one from Asia, then adapt the code above:

```
countries <- c("South Korea","Germany")

gapminder %>% filter(country %in% countries) %>%
  ggplot(aes(year,fertility)) +
  geom_line()
```

Unfortunately, this is not the plot that we want. Rather than a line for each country, the points for both countries are joined. This is actually expected since we have not told ggplot anything about wanting two separate lines.

Time series plots

To let ggplot know that there are two curves that need to be made separately, we assign each point to a group, one for each country:

```
countries <- c("South Korea","Germany")

gapminder %>% filter(country %in% countries & !is.na(fertility
  ggplot(aes(year, fertility, group = country)) +
  geom_line()
```

But which line goes with which country? We can assign colors to make this distinction.

Time series plots

A useful side-effect of using the `color` argument to assign different colors to the different countries is that the data is automatically grouped:

```
countries <- c("South Korea","Germany")

gapminder %>% filter(country %in% countries & !is.na(fertility
  ggplot(aes(year,fertility, col = country)) +
  geom_line()
```

The plot clearly shows how South Korea's fertility rate dropped drastically during the 1960s and 1970s, and by 1990 had a similar rate to that of Germany.

Labels instead of legends

For trend plots we recommend labeling the lines rather than using legends since the viewer can quickly see which line is which country. We define a data table with the label locations and then use a second mapping just for these labels:

```
labels <- data.frame(country = countries, x = c(1975,1965), y

gapminder %>%
  filter(country %in% countries) %>%
  ggplot(aes(year, life_expectancy, col = country)) +
  geom_line() +
  geom_text(data = labels, aes(x, y, label = country), size =
  theme(legend.position = "none")
```

Labels instead of legends

The plot clearly shows how an improvement in life expectancy followed the drops in fertility rates.

In 1960, Germans lived 15 years longer than South Koreans, although by 2010 the gap is completely closed. It exemplifies the improvement that many non-western countries have achieved in the last 40 years.

Data transformations

We now shift our attention to the second question related to the commonly held notion that wealth distribution across the world has become worse during the last decades.

When general audiences are asked if poor countries have become poorer and rich countries become richer, the majority answers yes.

By using stratification, histograms, smooth densities, and boxplots, we will be able to understand if this is in fact the case.

First we learn how transformations can sometimes help provide more informative summaries and plots.

Data transformations

The `gapminder` data table includes a column with the countries' gross domestic product (GDP). GDP measures the market value of goods and services produced by a country in a year.

The GDP per person is often used as a rough summary of a country's wealth. Here we divide this quantity by 365 to obtain the more interpretable measure *dollars per day*.

Using current US dollars as a unit, a person surviving on an income of less than $2 a day is defined to be living in absolute poverty. We add this variable to the data table:

```
gapminder <- gapminder %>%
  mutate(dollars_per_day = gdp/population/365)
```

Data transformations

The GDP values are adjusted for inflation and represent current US dollars, so these values are meant to be comparable across the years. Of course, these are country averages and within each country there is much variability. All the graphs and insights described below relate to country averages and not to individuals.

Log transformation

Here is a histogram of per day incomes from 1970:

```
past_year <- 1970
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(dollars_per_day)) +
  geom_histogram(binwidth = 1, color = "black")
```

We use the `color = "black"` argument to draw a boundary and clearly distinguish the bins.

Log transformation

In this plot, we see that for the majority of countries, averages are below $10 a day. However, the majority of the x-axis is dedicated to the 35 countries with averages above $10. So the plot is not very informative about countries with values below $10 a day.

It might be more informative to quickly be able to see how many countries have average daily incomes of about $1 (extremely poor), $2 (very poor), $4 (poor), $8 (middle), $16 (well off), $32 (rich), $64 (very rich) per day.

These changes are multiplicative and log transformations convert multiplicative changes into additive ones: when using base 2, a doubling of a value turns into an increase by 1.

Log transformation

Here is the distribution if we apply a log base 2 transform:

```
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(log2(dollars_per_day))) +
  geom_histogram(binwidth = 1, color = "black")
```

In a way this provides a *close-up* of the mid to lower income countries.

Which base?

In the case above, we used base 2 in the log transformations. Other common choices are base e(the natural log) and base 10.

In general, we do not recommend using the natural log for data exploration and visualization. This is because while power of numbers are easy to compute in our heads rather than power of the natural log, so the scale is not intuitive or easy to interpret.

In the dollars per day example, we used base 2 instead of base 10 because the resulting range is easier to interpret. The range of the values being plotted is 0.327, 48.885.

Which base?

In base 10, this turns into a range that includes very few integers: just 0 and 1. With base two, our range includes -2, -1, 0, 1, 2, 3, 4, and 5. It is easier to compute 2^x and 10^x when x is an integer and between -10 and 10, so we prefer to have smaller integers in the scale.

Another consequence of a limited range is that choosing the binwidth is more challenging. With log base 2, we know that a binwidth of 1 will translate to a bin with range x to $2x$.

Which base?

For an example in which base 10 makes more sense, consider population sizes. A log base 10 is preferable since the range for these is:

```
past_year <- 1970
filter(gapminder, year == past_year) %>%
  summarize(min = min(population), max = max(population))
```

```
##      min       max
## 1 46075 808510713
```

Which base?

Here is the histogram of the transformed values:

```
gapminder %>%
  filter(year == past_year) %>%
  ggplot(aes(log10(population))) +
  geom_histogram(binwidth = 0.5, color = "black")
```

In the above, we quickly see that country populations range between ten thousand and ten billion.

Transform the values or the scale?

There are two ways we can use log transformations in plots. We can log the values before plotting them or use log scales in the axes.

Both approaches are useful and have different strengths. If we log the data, we can more easily interpret intermediate values in the scale.

The advantage of using logged scales is that we see the original values on the axes. However, the advantage of showing logged scales is that the original values are displayed in the plot, which are easier to interpret.

Transform the values or the scale?

As we learned earlier, if we want to scale the axis with logs, we can use the `scale_x_continuous` function. Instead of logging the values first, we apply this layer:

```
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(dollars_per_day)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans = "log2")
```

Note that the log base 10 transformation has its own function: `scale_x_log10()`.

Transform the values or the scale?

There are other transformations available through the `trans` argument.

- The square root transformation (`sqrt`) : useful when considering counts
- The logistic transformation (`logit`) : useful when plotting proportions between 0 and 1
- The `reverse` transformation : useful when we want smaller values to be on the right or on top

Comparing multiple distributions with boxplots and ridge plots

A histogram showed us that the 1970 income distribution values show a dichotomy. However, the histogram does not show us if the two groups of countries are *west* versus the *developing* world.

Let's start by quickly examining the data by region. We reorder the regions by the median value and use a log scale.

```
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  mutate(region = reorder(region, dollars_per_day,
                          FUN = median)) %>%
  ggplot(aes(dollars_per_day, region)) +
  geom_point() +
  scale_x_continuous(trans = "log2")
```

Comparing multiple distributions with boxplots and ridge plots

We can already see that there is indeed a "west versus the rest" dichotomy. We define groups based on this observation:

```
gapminder <- gapminder %>%
  mutate(group = case_when(
    region %in% c("Western Europe", "Northern Europe",
                  "Southern Europe", "Northern America",
                  "Australia and New Zealand") ~ "West",
    region %in% c("Eastern Asia",
                  "South-Eastern Asia") ~ "East Asia",
    region %in% c("Caribbean", "Central America",
                  "South America") ~ "Latin America",
    continent == "Africa" &
      region != "Northern Africa" ~ "Sub-Saharan",
    TRUE ~ "Others"))
```

Comparing multiple distributions with boxplots and ridge plots

We turn this group variable into a factor to control the order of the levels:

```
gapminder <- gapminder %>%
  mutate(group = factor(group, levels = c("Others",
                                          "Latin America",
                                          "East Asia",
                                          "Sub-Saharan",
                                          "West")))
```

In the next section we demonstrate how to visualize and compare distributions across groups.

Boxplots

The exploratory data analysis above has revealed two characteristics about average income distribution in 1970. We now want to compare the distribution across these five groups. The number of points in each category is large enough that a summary plot may be useful.

We could generate five histograms or five density plots, but it may be more practical to have all the visual summaries in one plot. We therefore start by stacking boxplots next to each other.

Note that we add the layer `theme(axis.text.x = element_text(angle = 90, hjust = 1))` to turn the group labels vertical, since they do not fit if we show them horizontally, and remove the axis label to make space.

Boxplots

```
p <- gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(group, dollars_per_day)) +
  geom_boxplot() +
  scale_y_continuous(trans = "log2") +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
p
```

Boxplots have the limitation that by summarizing the data into five numbers, we might miss important characteristics of the data.

Boxplots

One way to avoid this is by showing the data.

```
p + geom_point(alpha = 0.5)
```

Ridge plots

Boxplots help with this by providing a five-number summary, but this has limitations too. In cases in which we are concerned that the boxplot summary is too simplistic, we can show stacked smooth densities or histograms. We refer to these as *ridge plots*.

The package **ggridges** provides a convenient function for doing this. Here is the income data shown above with boxplots but with a *ridge plot*.

```
library(ggridges)
p <- gapminder %>%
  filter(year == past_year & !is.na(dollars_per_day)) %>%
  ggplot(aes(dollars_per_day, group)) +
  scale_x_continuous(trans = "log2")
p  + geom_density_ridges()
```

Ridge plots

Note that we have to invert the `x` and `y` used for the boxplot. A useful `geom_density_ridges` parameter is `scale`, which lets you determine the amount of overlap, with `scale = 1` meaning no overlap and larger values resulting in more overlap.

If the number of data points is small enough, we can add them to the ridge plot using the following code:

```
p + geom_density_ridges(jittered_points = TRUE)
```

By default, the height of the points is jittered and should not be interpreted in any way.

Ridge plots

To show data points, but without using jitter we can use the following code to add what is referred to as a *rug representation* of the data.

```
p + geom_density_ridges(jittered_points = TRUE,
                        position =
                          position_points_jitter(height = 0),
                        point_shape = '|', point_size = 3,
                        point_alpha = 1, alpha = 0.7)
```

Example: 1970 versus 2010 income distributions

Data exploration clearly shows that in 1970 there was a "west versus the rest" dichotomy. But does this dichotomy persist? Let's use `facet_grid` see how the distributions have changed.

Example: 1970 versus 2010 income distributions

To start, we will focus on two groups: the west and the rest. We make four histograms.

```
past_year <- 1970
present_year <- 2010
years <- c(past_year, present_year)
gapminder %>%
  filter(year %in% years & !is.na(gdp)) %>%
  mutate(west = ifelse(group == "West", "West", "Developing")) %>%
  ggplot(aes(dollars_per_day)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans = "log2") +
  facet_grid(year ~ west)
```

Example: 1970 versus 2010 income distributions

Before we interpret the findings of this plot, we notice that there are more countries represented in the 2010 histograms than in 1970. One reason for this is that several countries were founded after 1970. Another reason is that data was available for more countries in 2010.

Example: 1970 versus 2010 income distributions

We remake the plots using only countries with data available for both years. we can use simple code using the intersect function:

```
country_list_1 <- gapminder %>%
  filter(year == past_year & !is.na(dollars_per_day)) %>%
  pull(country)

country_list_2 <- gapminder %>%
  filter(year == present_year & !is.na(dollars_per_day)) %>%
  pull(country)

country_list <- intersect(country_list_1, country_list_2)
```

These 108 account for 86% of the world population, so this subset should be representative.

Example: 1970 versus 2010 income distributions

Let's remake the plot, but only for this subset by simply adding `country %in% country_list` to the filter function:

```
past_year <- 1970
present_year <- 2010
years <- c(past_year, present_year)
gapminder %>%
  filter(year %in% years & !is.na(gdp) &
           country %in% country_list) %>%
  mutate(west = ifelse(group == "West", "West", "Developing"))
  ggplot(aes(dollars_per_day)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans = "log2") +
  facet_grid(year ~ west)
```

Example: 1970 versus 2010 income distributions

We now see that the rich countries have become a bit richer, but percentage-wise, the poor countries appear to have improved more. In particular, we see that the proportion of developing countries earning more than $16 a day increased substantially.

Example: 1970 versus 2010 income distributions

To see which specific regions improved the most, we can remake the boxplots we made above, but now adding the year 2010 and then using facet to compare the two years.

```
gapminder %>%
  filter(year %in% years & country %in% country_list) %>%
  ggplot(aes(group, dollars_per_day)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_y_continuous(trans = "log2") +
  xlab("") +
  facet_grid(. ~ year)
```

Example: 1970 versus 2010 income distributions

Here, we pause to introduce another powerful **ggplot2** feature.Because we want to compare each region before and after, it would be convenient to have the 1970 boxplot next to the 2010 boxplot for each region. In general, comparisons are easier when data are plotted next to each other.

So instead of faceting, we keep the data from each year together and ask to color (or fill) them depending on the year.

Example: 1970 versus 2010 income distributions

Note that we have to convert the year columns from numeric to factor.

```
gapminder %>%
  filter(year %in% years & country %in% country_list) %>%
  mutate(year = factor(year)) %>%
  ggplot(aes(group, dollars_per_day, fill = year)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_y_continuous(trans = "log2") +
  xlab("")
```

Example: 1970 versus 2010 income distributions

The previous data exploration suggested that the income gap between rich and poor countries has narrowed considerably during the last 40 years. We used a series of histograms and boxplots to see this. We suggest a succinct way to convey this message with just one plot.

Example: 1970 versus 2010 income distributions

Let's start by noting that density plots for income distribution in 1970 and 2010 deliver the message that the gap is closing:

```
gapminder %>%
  filter(year %in% years & country %in% country_list) %>%
  ggplot(aes(dollars_per_day)) +
  geom_density(fill = "grey") +
  scale_x_continuous(trans = "log2") +
  facet_grid(. ~ year)
```

Example: 1970 versus 2010 income distributions

In the 1970 plot, we see two clear modes: poor and rich countries. In 2010, it appears that some of the poor countries have shifted towards the right, closing the gap.

Example: 1970 versus 2010 income distributions

The next message we need to convey is that the reason for this change in distribution is that several poor countries became richer, rather than some rich countries becoming poorer. To do this, we can assign a color to the groups we identified during data exploration.

```
gapminder %>%
  filter(year %in% years & country %in% country_list) %>%
  mutate(group = ifelse(group == "West", "West", "Developing")
  ggplot(aes(dollars_per_day, fill = group)) +
  scale_x_continuous(trans = "log2") +
  geom_density(alpha = 0.2) +
  facet_grid(year ~ .)
```

Example: 1970 versus 2010 income distributions

This makes it appear as if there are the same number of countries in each group. To change this, we will need to learn to access computed variables with `geom_density` function.

Accessing computed variables

To have the areas of these densities be proportional to the size of the groups, we can simply multiply the y-axis values by the size of the group.

From the `geom_density`, we see that the functions compute a variable called `count` that does exactly this. We want this variable to be on the y-axis rather than the density.

Accessing computed variables

In **ggplot2**, we access these variables by surrounding the name with two dots. We will therefore use the following mapping:

```
aes(x = dollars_per_day, y = ..count..)
```

Accessing computed variables

We can now create the desired plot by simply changing the mapping in the previous code chunk.

```
p <- gapminder %>%
  filter(year %in% years & country %in% country_list) %>%
  mutate(group = ifelse(group == "West", "West", "Developing")
  ggplot(aes(dollars_per_day, y = ..count.., fill = group)) +
  scale_x_continuous(trans = "log2", limit = c(0.125, 300))

p + geom_density(alpha = 0.2) +
  facet_grid(year ~ .)
```

Accessing computed variables

If we want the densities to be smoother, we use the `bw` argument so that the same bandwidth is used in each density. We selected 0.75 after trying out several values.

```
p + geom_density(alpha = 0.2, bw = 0.75) + facet_grid(year ~
```

This plot now shows what is happening very clearly.

Accessing computed variables

To visualize if any of the groups defined above are driving this we can quickly make a ridge plot:

```
gapminder %>%
  filter(year %in% years & !is.na(dollars_per_day)) %>%
  ggplot(aes(dollars_per_day, group)) +
  scale_x_continuous(trans = "log2") +
  geom_density_ridges(adjust = 1.5) +
  facet_grid(. ~ year)
```

Accessing computed variables

Another way to achieve this is by stacking the densities on top of each other:

```
gapminder %>%
    filter(year %in% years & country %in% country_list) %>%
  group_by(year) %>%
  mutate(weight = population/sum(population)*2) %>%
  ungroup() %>%
  ggplot(aes(dollars_per_day, fill = group)) +
  scale_x_continuous(trans = "log2", limit = c(0.125, 300)) +
  geom_density(alpha = 0.2, bw = 0.75, position = "stack") +
  facet_grid(year ~ .)
```