Graphics With ggplot2

Sean Hellingman ©

Data Visualization and Manipulation through Scripting (ADSC1010)

shellingman@tru.ca

Fall 2024

THOMPSON RIVERS UNIVERSITY

Topics

2. Introduction
3. Basic Plots
4. Geometries
5. Aesthetic Mappings
6. Customization
7. Coordinate Systems
8. Facets
9. Labels and Annotations
10. Maps
11. Exercises and References

Introduction

- Demonstrated that base R can be very useful for generating graphics.
- The *ggplot2* package is built on the *Grammar of Graphics*.
- Widely considered the primary package for generating *static* plots in R.
- Will allow you to customise your plots exactly to your liking.

Grammar of Graphics I

- Helps you construct graphical features out of different visual elements using the following components:
  - The **data** being plotted.
  - The **geometric objects** (circles/lines) that appear in the plot.
  - The **aesthetics** (appearance) of the geometric objects, and the *mappings* from variables in the data due to those aesthetics.
  - A **position adjustment** for placing elements on the plot so they do not overlap.
  - A **scale** (range of values) for each aesthetic mapping used.
  - A **coordinate system** used to organise the geometric objects.
  - The **facets** or groups of data shown in different plots.

Grammar of Graphics II

- These components are all organised into *layers* by *ggplot2*.
- Each layer displays a single type of *geometric object*.
- Each plot is a set of layers of images, where the appearance of each image is based on some aspect of the dataset.
- We can use this standard *grammar* to easily create specific images about our data.

`ggplot()` function

- The package uses a set of functions that correspond to the *Grammar of Graphics*.
- To create a plot we use the `ggplot(data = data.frame)` function.
    - This will return a blank canvas that we can add *layers* to.
- Each layer contains a specific *geometry* (lines, points, ect.) that can be drawn onto the blank canvas.

Example 1

- Initialise the *ggplot2* package in your Environment.
- Import the *midwest* dataset from the *ggplot2* package.
- Use the `ggplot()` function to generate a scatterplot of the education (*percollege*) and poverty (*peradultpoverty*) rates.
  - *Hint: Use the example code to help you.*

Basic Steps

1. Create a blank canvas using the `ggplot()` function.

2. Specify the type of geometric object (geom) using one of the *geom_functions*.
   - Example 1 used `geom_point()` to generate a layer of dots.

3. Within the *geom_function* you must specify the **aesthetic mapping** (`aes()`).
   - Takes a set of named arguments where the *argument name* is the visual property to map to and the argument value (variable) is the data to *map from*.
   - Example 1 set `x` as *percollege* and `y` as *peradultpoverty*.

- Add layers to the plot using the + operator

Specifying Geometries

- The geometric objects you select will provide the largest differences between plots.
- Some of the geometric functions:
  - `geom_point()` for drawing individual points (scatterplot).
  - `geom_line()` for drawing lines (line chart).
  - `geom_smooth()` for drawing smoothed lines (trend lines).
  - `geom_col()` for drawing columns (bar chart).
  - `geom_polygon()` for drawing arbitrary shapes (an area in a coordinate plane).

geom_functions Arguments

- Each *geom_function* requires a set of aesthetic mappings as an argument (`aes()`).
- Almost all *geom_functions* require an `x` and a `y` mapping.
- You can also map a data feature to the shapes of your points or the type of your lines.

Example 2

- Using the *midwest* dataset and various *geom_functions* generate the following plots using *ggplot2*:

  1. Generate a barplot of the population totals (*poptotal*) by state (*state*) (*Hint:* `geom_col()`)

  2. Generate a trend line (smoothed line) for the education (*percollege*) to the adult poverty (*peradultpoverty*) rates. (*Hint:* `geom_smooth()`)

Multiple *geom_functions*

- You can add multiple *geometries* to a plot.
  - This allows you to create complex graphics that include multiple aspects of your data.

- If you wish to keep the same *aesthetics* (`aes()`) for each geometry you can specify the default mapping in the `ggplot()` function.
  - `ggplot(data = data.frame, mapping = aes(x=var.1, y=var.2, ...))`

- You can also make adjustments to the *aesthetics* (`aes()`) as needed in the different layers.

Example 3

- Using the *midwest* dataset and various *geom_functions* generate the following plot using *ggplot2*:
    1. Generate a scatter plot of the education (*percollege*) and poverty (*peradultpoverty*) rates and add **a smoothed line** to the plot.

Statistical Transformations of Geometries

- Some *geom_functions* also perform statistical transformations on your data before mapping that data.
- Many of the operations can be performed using the *dplyr* package (`group_by()`, `summarize()`)
- *You can find more about doing this on the tidyverse website.*

Graphical Encodings

- We can use aesthetic mappings to take properties of the data and use them to influence *visual channels*.
- These aesthetic mappings are used to express different data values.
- These aesthetic mappings are driven by the (`aes()`) function.
- We can add colours by using the `color =` argument **within** the (`aes()`) function.
  - Add `color =` is added outside to set everything to be the same colour.

Example 4

- Using the *midwest* dataset and various *geom_functions* generate the following plot using *ggplot2*:
    1. Generate a scatter plot of the education (*percollege*) and poverty (*peradultpoverty*) rates and add a colour layer for each state (*state)* that the observation is in.

Layouts

- Building on the basics we have covered we can generate many different plots.
- We can use the geometry and aesthetics to construct plots.
- Plots can be further customised using additional functions.

Position Adjustments

- Each geometry has a default positional adjustment.
- You can use the (`position()`) argument to specify a different position.
- These arguments are included in the geometry but outside of the aesthetics.

Example 5

- Using the *midwest* dataset, the code for the *state_education* dataset and various *geom_functions* generate the following plots using *ggplot2*:

1. Generate a barplot of the population totals (*poptotal*) by state (*state*) coloured by education level (*Education*). (*Hint:* `geom_col()` & `fill =`)

2. Generate a barplot of the population totals (*poptotal*) by state (*state*) coloured by education level (*Education*) and fill each bar to 100%. (*Hint:* `position = "fill"`)

3. Generate a barplot of the population totals (*poptotal*) by state (*state*) coloured by education level and compare the education levels (*Education*) within each state using a grouped column format. (*Hint:* `position = "dodge"`

Styling with Scales

- *ggplot2* uses a particular **scale** to determine the *range of values*.
- Each scale can be represented by `scale_` followed by the name of the aesthetic (e.g. `x` or `color`), followed by an _ and the type of the scale (e.g. `continuous` or `discrete`).
  - `continuous` is usually used for numeric data.
  - `discrete` is usually used for a short set of possibilities (colours).
- The default scales are often sufficient.
- Example scales:
  - `scale_x_reverse()`: change the direction of the x-axis
  - `scale_x_log10()`: plot data on a logarithmic scale

Colour Scales

- One of the most common scales to change is the colour.
- Using the ColorBrewer palettes: `scale_color_brewer_(palette = "palette.name")`
  - Pass the `palette` as the argument name.
- You can also define your own colour schemes using `scale_color_manual()` & `scale_color_gradient()`

Example 6

- Using the *midwest* dataset, the code for the *state_education* dataset and various *geom_functions* generate the following plot using *ggplot2*:

1. Generate a barplot of the population totals (*poptotal*) by state (*state*) coloured by education level (*Education*). (*Hint:* `geom_col()` & `fill =`)
    - Reverse the direction of the y-axis (*Hint:* `scale_y_reverse()`)
    - Change the colour palette of the fill to *Dark2*.

Coordinate Systems

- We can also specify the **coordinate system** that organises the geometric objects.

- As with scales the functions are formatted `coord_` followed by the function name.

- Some coordinate systems:
  - `coord_cartesian()`: The *default* Cartesian (x, y) coordinate system.
  - `coord_flip()`: A Cartesian system with x and y flipped.
  - `coord_fixed()`: A Cartesian system with a *fixed* aspect ratio (e.g. 1.78 for *widescreen*).
  - `coord_polar()`: A plot using polar coordinates (pie chart).
  - `coord_quickmap()`: A coordinate system that approximates a good aspect ratio for maps.

Example 7

- Using the *midwest* dataset, the code for the *state_education* dataset and various *geom_functions* generate the following plots using *ggplot2*:

  1. Generate a barplot of the population totals (*poptotal*) by state (*state*) coloured by education level (*Education*). (*Hint:* `geom_col()` & `fill =`)
     - Use `coord_flip()` to make a horizontal bar chart.
       We DO NOT need to change the x and y arguments, R does it for us.
  2. Use `coord_polar()` to change the coordinate system.

Facets

- Something we have covered using the *lattice* package in base R.
- **Facets** allow us to group visualizations into different subplots.
- Generally use `facet_wrap(~variable.name)` to produce subplots for each category of `variable.name`.
  - Need to put ~ in front of the variable we are grouping by in the `facet_` functions.

Example 8

- Using the *midwest* dataset, generate the following plot using *ggplot2*:

  1. Generate a scatter plot of the education (*percollege*) and poverty (*peradultpoverty*) rates. Facet your plot by *state* and colour your points by *inmetro* (as a factor Metro area or not).

Labels I

- It is important to clearly express the meanings of different elements of visualisations.
- Use the `labs()` function in R to add labels.
- The `labs()` function takes arguments for each aspect to label.
- Some examples:
  - `title = "Plot Title"`
  - `x = "x-axis label"`
  - `y = "y-axis label"`
  - `color = "legend label for the color property"`
  - `fill = "legend label for the fill property"`

Labels II

- We can also add labels inside the plot (to each point or line).
- *Essentially adding an extra set of data values that happen to be the value names.*
- Some functions we can use to do this:
  - `geom_text()`: Adds plain text to the plot.
  - `geom_label()`: Adds boxed text to the plot.
  - `geom_label_repel()`: Provides labels that do not overlap (*ggrepel* package)

Example 9

- Run the code under Example 9 to create the *most_poverty* dataset (will be used for labels).

- Using the *midwest* dataset and your new *most_poverty* dataset, generate the following plot using *ggplot2*:

  1. Generate a scatter plot of the education (*percollege*) and poverty (*peradultpoverty*) rates. Facet your plot by *state* and colour your points by *location* (created in Example 8).
     - Add an appropriate title, subtitle and axes labels.
     - Use the `geom_label_repel()` and your *most_poverty* dataset to add labels to the counties with the highest poverty.

Building Maps

Maps with *ggplot2*

- We will generally use a Cartesian layout to create geographic visualisations.

- There are two types of maps we will create:
  1. **Choropleth:**
     - Different geographic areas are shaded based on data about each region.
     - Useful to visualise regionally aggregated data (heatmaps).
  2. **Dot distribution:**
     - Markers are placed at specific coordinates.
     - Useful to visualise observations that occur at specific points.

- *Our examples will use the USA because ggplot2 already has the shapefile.*

Choropleth Maps I

- We need to use the `geom_polygon()` function to draw the outlines
- We will need to load the **shapefile** that contains the outlines.
- *You can find and download shapefiles online.*
- We can get the shapefiles from *ggplot* using `map_data()`.
- To keep an appropriate aspect ratio we can use the `coord_map()` function.

Example 10

- Follow the example code to draw an outline map of the United States.

Choropleth Maps II

- If we want each geographical area to express different data through a visual channel such as colour:
  1. *Load* your data
  2. *Join* the data to the shapefile.
  3. Map the `fill` of each polygon.

- Usually, the hardest part if formatting your data correctly to plot.

Example 11

- Use your USA map from Example 10, the *2022_US_Population.csv* data, and the example code to create a USA map filled by state populations.

Dot Distribution Maps

- We can plot data at discrete locations on a map because we are already using a coordinate system.
- Generally we can use an additional data set that contains the locations and information about the locations we would like to plot.
- Use the `geom_point()` function to add the additional data.
  - You can add aesthetics to the points as you would in a scatterplot.

Example 12

- Use your USA map from Example 11, the *US_Cities.csv* data, and the example code to create a USA map filled by state populations with marked US cities.

- Change the aesthetics to express the population sizes of the cities.

Thoughts on Maps

- We have introduced some examples of creating maps in R.
- You can combine the *ggmap* package with *ggplot2* to create more granular maps.
- We will talk about creating maps using other packages.

Exercise 1

- The best way to improve your understanding of the plotting functions in *ggplot2* is to take some time to generate some plots, make customisations, and think about what your plots actually mean.

- We have only covered a few of the geometries (`geom_` functions). Go on the website and see if you can use *ggplot2* to replicate the plots we made using base R:
    - Histograms
    - Boxplots/violin plots
    - Use the `facet_wrap(~variable.name)` function to facet your plots by groups.
    - Be sure to include the appropriate labels.

- I encourage you to take some time with your project data (or other real data) and try to generate some insightful plots using *ggplot2*.

Exercise 2

- Select a region of 4 or 5 bordering US states and create a map of that region using *ggplot2*.
- Add information about the state populations (or other information you are interested in) as a fill.
- Add the state capitals to your map.
- Be sure to appropriately label everything.
- **If you are feeling ambitious, select a shapefile for a different country or region and see if you can generate an insightful map using *ggplot2*.**

References & Resources

1. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to R*. Retrieved from https://intro2r.com/

2. Michael Freeman, Joel Ross, *Programming Skills for Data Science: Start Writing Code to Wrangle, Analyze, and Visualize Data with R*, 2019, ISBN-13: 978-0-13-513310-1

- https://ggplot2.tidyverse.org/
- https://cran.r-project.org/web/packages/ggmap/readme/README.html
- https://cran.r-project.org/web/packages/maps/index.html
- https://cran.r-project.org/web/packages/hexbin/index.html