Name: _____ LA Initials: [ ]

Lab 4: Graphing with ggplot

Learning Objectives

By the end of this lab, students will be able to:

- Describe the basic idea of building graphs in layers using ggplot2.
- Create a ggplot object by specifying a dataset and aesthetic mappings.
- Map numerical variables to the x- and y-axes to produce a scatterplot.
- Map a categorical variable to color to represent groups in the data.
- Use `facet_wrap()` to create small-multiple scatterplots.
- Recreate a scatterplot with aesthetics and facets using a new dataset.

Getting started

Before you begin the lab activities, complete the following steps to make sure you are fully set up.

1. **Get the Lab Handout.**

   Pick up a physical copy of the lab worksheet, or print one if you are working outside of class.
   Download Lab Worksheet (PDF, if needed)

2. **Open Posit Cloud and Start Lab 4.**

   Log in to **Posit Cloud** and navigate to the **course workspace**, and start the Lab 4 assignment.

3. **Install required packages**

   Install the required packages for this lab, including **tidyverse**, **palmerpenguins**, and **gapminder** using the Packages pane or by running the following code ***in the Console***:

```
install.packages(c("tidyverse", "palmerpenguins", "gapminder"))
```

4. **Create an R Script**

   Create a new R script and save it as **lab-3-script.R**.

5. **Load the packages**

   Copy and paste this code to your R script and run it to load the packages. Remember to run both library commands (Ctrl+Enter on one line, then the next).

```
# load packages -----------------------------------------------------------

library(tidyverse)
library(palmerpenguins)
library(gapminder)
```

Checkpoint

At this point, you should have:

- These instructions open in a web browser.
- Your Lab 4 project open in Posit Cloud in another browser window.
- The required packages installed and loaded without errors
- The Lab 4 worksheet in front of you.

Do not continue until all of the above steps are working correctly.

Overview: From Data to Graphs

In this lab, you will learn how to create your first graphs in R using the ggplot2 package. Rather than selecting a graph type from a menu, you will build a graph step by step by describing how variables in a dataset should be represented visually.

You will begin by working through examples using the Palmer Penguins dataset to learn the basic structure of a ggplot, how aesthetic mappings work, and how scatterplots are constructed. Once those ideas are in place, you will apply the same approach to a different dataset—the Gapminder dataset—to create a similar graph on your own.

The focus of this lab is on scatterplots and on understanding the core pieces that make up a graph: the data, the variables being plotted, and the visual mappings that connect data values to positions, colors, and panels. The goal is not to memorize commands, but to understand how the pieces fit together.

This lab is designed to be self-paced. As you work through the sections, you will follow along with provided examples and record notes and answers on the lab worksheet. The questions are intended to highlight key ideas and syntax that you will reuse later in the course, not to test you.

By the end of the lab, you should be comfortable reading ggplot code, understanding what each part of a plot is doing, and making small modifications to create similar graphs using a new dataset.

Tip

Read the following sections carefully **before** writing any code or answering handout questions. The lab will clearly indicate when you should run code or record an answer. Skipping ahead often leads to confusion and unnecessary frustration.

How ggplot2 Builds a Graph

ggplot2 is an R package for creating graphs of data. ggplot2 uses a conceptual framework that involves composing graphs in a stepwise manner using code rather than selecting a pre-defined graph from a list. You can read more about ggplot2 in this short vignette Introduction to ggplot2 and in the ggplot2 book.

A ggplot is composed of seven parts. The first three are always needed to produce a graph: data, a mapping, and a layer. The other four have sensible defaults and are used to fine-tune the graph: scales, facets, coordinates, and a theme.

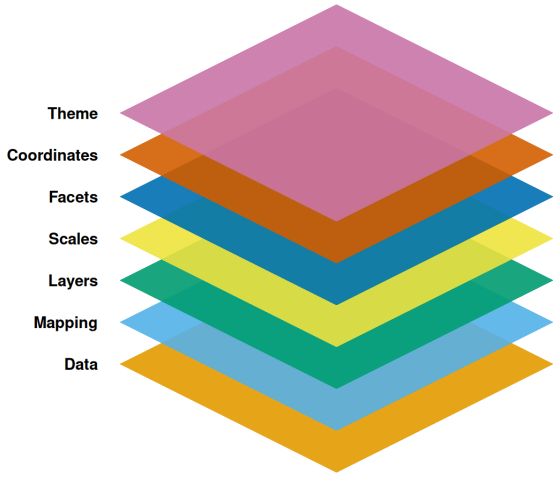

Figure 1: The 7 parts used to compose a ggplot. Credit: ggplot2 package webpage by Wickham et al.

Handout Q1

In your own words, what does it mean to "build" a graph in ggplot2 rather than choosing a graph type?

Data

Every ggplot begins with a dataset. The dataset tells ggplot where the data come from, but it does not yet specify what to draw or how to draw it. At this stage, ggplot knows what data are available, but it has no instructions for how to display them.

```
ggplot(data = penguins)
```

Because no variables or layers have been specified yet, nothing is drawn. This is expected. Think of this as creating an empty plotting canvas that knows which dataset it will use.

Handout Q2

What dataset is being used in this example?

Mapping

A **mapping** describes how variables in the dataset are connected to visual properties in the plot. These connections are called aesthetic mappings. The most common mappings are the x- and y-axes.

```
ggplot(penguins, mapping = aes(x = flipper_length_mm, y = body_mass_g))
```

Here, two numerical variables are mapped to position: one to the x-axis and one to the y-axis. At this point, ggplot knows where values should go, but it still does not know what type of marks to draw.

Handout Q3

Which variables are mapped to the x- and y-axes in this plot?

Layers

A **layer** tells ggplot how to represent the mapped data visually. Layers include information about geometry (what shape to draw), statistical transformations, and position adjustments. In this course, you will usually define only the geometry and rely on defaults for the rest.

To add a layer of points to the graph, use the `geom_point()` function as shown in the example below. You can also add other layers to the same graph. In the example below, we have also added a linear regression line with `geom_smooth()` to help visualize the relationship between the two variables.

```
ggplot(penguins, mapping = aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm")
```

Adding a geometry layer causes the data to appear on the graph. In this lab, you will focus on point geometries, which are used to create scatterplots.

You do not need to worry about the statistical transformation or position adjustment yet—ggplot handles these automatically unless you tell it otherwise.

Handout Q4

In this lab, what type of geometry is used to create scatterplots? Give the exact function name.

Scales

Scales control how data values are translated into visual values. When you map a variable to color, ggplot automatically creates a color scale. The same is true for the x- and y-axes.

In our example, we are plotting data from three penguin species. If we wanted to visualize which points represent each species, we can use an aesthetic mapping to link the `species` variable to the `color` aesthetic by adding `color = species` to the `aes()` function.

```
ggplot(penguins, mapping = aes(flipper_length_mm, body_mass_g, color = species)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm")
```

This provides a default scale for color (i.e. a default set of colors). We can then modify those defaults by specifying a color scale ourselves using one of the many `scale_*` functions. In this case we will use `scale_color_viridis_d()` to tell ggplot to use the **viridis** color scale. The viridis color scale is a common one that is color-blind friendly.

```
ggplot(penguins, mapping = aes(flipper_length_mm, body_mass_g, color = species)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm") +
  scale_color_viridis_d()
```

This example shows that scales can be customized without changing the underlying data or mappings. For now, the important idea is that scales exist and can be adjusted when needed.

Handout Q5

Do you need to manually define scales in order to make a useful plot? Why or why not?

Facets

Facets create small multiples: the same graph repeated for different subsets of the data. Each panel shows the same variables using the same scales, making comparisons across groups easier. Create facets using the `facet_wrap()` function like this:

```
ggplot(penguins, mapping = aes(flipper_length_mm, body_mass_g, color = species)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm") +
  scale_colour_viridis_d() +
  facet_wrap(~ species)
```

Faceting is often preferable to adding more colors or symbols, especially when comparing patterns across categories. You will use faceting frequently in future labs.

Handout Q6:
What changes and what stays the same when a plot is faceted.

> Tip
>
> When you use `facet_wrap()` the default behavior is to use the same limits for the x- and y-axes. If you want to let those limits vary to fit the data on each facet, add a comma followed by `scales = "free"` to `facet_wrap()`.

Guided Practice: Modifying an Existing Plot

This exercise is designed to help you connect specific lines of ggplot code to visible changes in the graph. Focus on what changes and why, not on getting a "correct" final plot.

Copy the the working ggplot code from above, and make the following modifications:

- put `body_mass_g` on the x-axis and `flipper_length_mm` on the y-axis.
- map the color aesthetic to `sex` instead of `species`.
- facet the graph using `year` instead of `species`.

If something does not work as expected, read error messages carefully and ask an LA for help. Small syntax issues are common at this stage.

Independent Practice: A New Scatterplot

In this final activity, you will apply what you learned using a new dataset. The goal is to recreate the structure of the previous scatterplots—not to invent a new type of graph.

You will work with the **Gapminder** dataset, which contains country-level measurements collected across multiple years. Each row represents one country in one year.

Print the gapminder dataset in your console to see what it looks like:

```
print(gapminder)
```

Your scatterplot should:

- use the `gapminder` data.

- use two numerical variables of your choice on the x- and y-axes.
- map a categorical variable to color. in practice, the one that works best is `continent`.
- use faceting to show the same plot across groups.

Use the earlier examples as a template. You should not need any new ggplot functions to complete this section.

Handout Q7

List the variables you used for the x and y axes, for color, and for facetting in the gapminder scatterplot.

Key Takeaways

- ggplot graphs are built by combining data, mappings, and layers.
- Aesthetic mappings connect variables to visual properties like position and colors
- Scatterplots require two numerical variables.
- Faceting allows you to compare patterns across groups using the same axes.

These ideas will carry forward as you learn additional graph types later in the course.

Wrap-up and submission

This lab is graded for completion.

1. Make sure your script is saved in your project on Posit Cloud.
2. Print the final bird dataset in the console and show it to an LA for grading.
3. Keep your handout and **show it to a Learning Assistant for a completion grade before you leave lab**. You may do this as soon as you finish.