

# Lab 01 - Hello R!

Meredith Franklin

January 08, 2025

## Learning goals

- Get acquainted with R and RStudio, which we will be using throughout the course to analyze data as well as to learn the statistical concepts discussed in the course.
- Appreciate the value of visualization in exploring the relationship between variables.
- Start using R for building plots and calculating summary statistics.

## Deliverables

- Complete the data analysis exercises 1-5 by adding code chunks to the .Rmd
- Make sure you can knit without error
- Upload the .html and .Rmd to Quercus under Lab 1

## Terminology

We've already thrown around a few new terms, so let's define them before we proceed.

- **R**: Name of the programming language we will be using throughout the course.
- **RStudio**: An integrated development environment for R. In other words, a convenient interface for writing and running R code.

I like to think of R as the engine of the car, and RStudio is the dashboard.

## Starting slow

As the labs progress, you are encouraged to explore beyond what the labs dictate; a willingness to experiment will make you a much better programmer. Before we get to that stage, however, you need to build some basic fluency in R. Today we begin with the fundamental building blocks of R and RStudio: the interface, reading in data, and basic commands.

And to make versioning simpler, this is a solo lab. Additionally, we want to make sure everyone gets a significant amount of time at the steering wheel.

## Getting started

### 1. Download project

Go to the lab page for this week on the website. You can download the lab materials from the course github.

## 2 Download R

**If you don't have R installed.**

Go to CRAN and download R, make sure you get the version that matches your operating system.

**If you have R installed**

If you have R installed run the following code

```
R.version

##
## platform      _
## arch          aarch64-apple-darwin20
## arch          aarch64
## os            darwin20
## system        aarch64, darwin20
## status
## major         4
## minor         4.1
## year          2024
## month         06
## day           14
## svn rev       86737
## language      R
## version.string R version 4.4.1 (2024-06-14)
## nickname      Race for Your Life
```

This should tell you what version of R you are currently using. If your R version is lower than 4.3.0 I would strongly recommend updating. In general it is a good idea to update your R version, unless you have a project right now that depend on a specific version of R.

### 2.2 Download RStudio

We recommend using RStudio as your IDE if you don't already have it installed. You can go to the RStudio website to download and install the software.

### 2.3 Launch RStudio

You have 2 ways to open the lab project we will be working with. Inside the folder is a `lab01.Rproj` file, opening this file should launch your RStudio project.

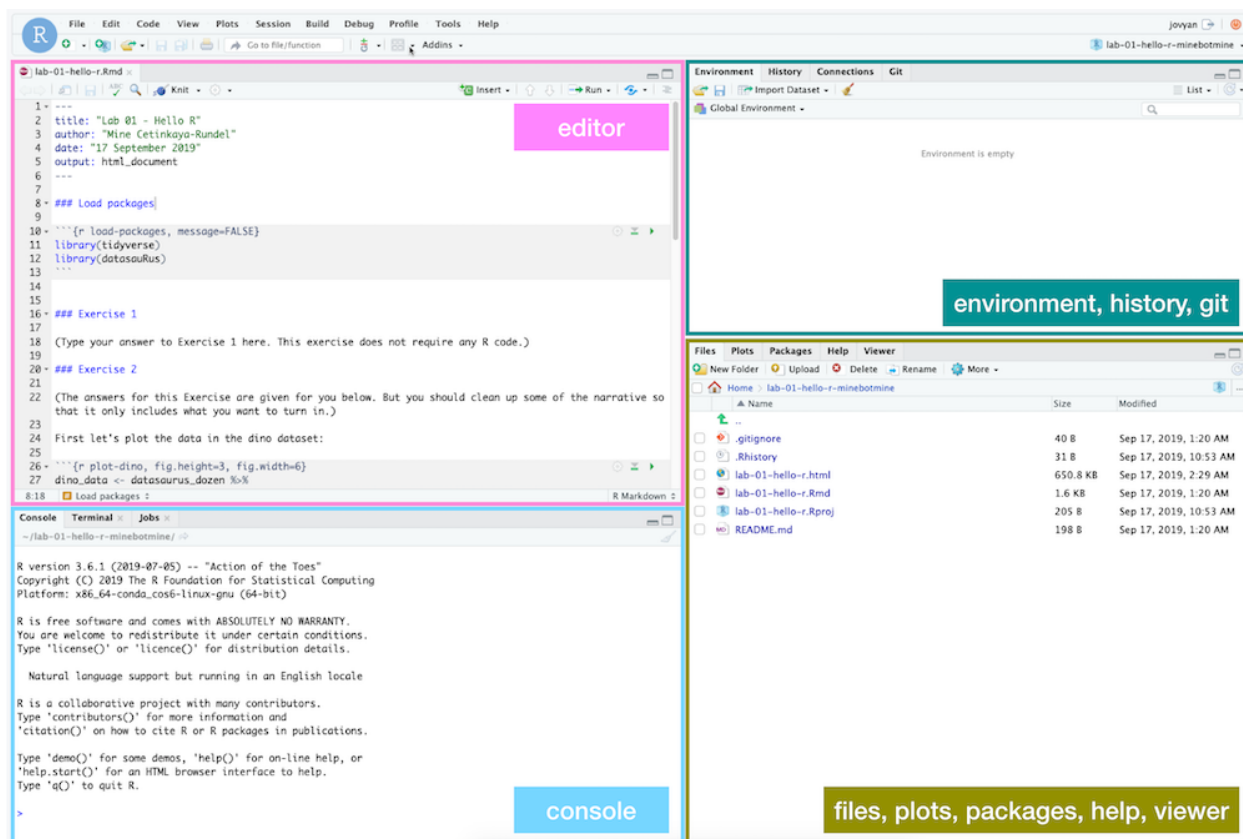
You can also open the RStudio application first and then open the project by going `file -> open project...`

## 3. Get working!

Open the R Markdown (Rmd) file called `lab01-hello-R.Rmd`. It will likely ask you if you would like to install a package that is required, click Install.

## Hello RStudio!

RStudio is comprised of four panes.



- On the bottom left is the Console, this is where you can write code that will be evaluated. Try typing `2 + 2` here and hit enter, what do you get?
- On the bottom right is the Files pane, as well as other panes that will come handy as we start our analysis.
- If you click on a file, it will open in the editor, on the top left pane.
- Finally, the top right pane shows your Environment. If you define a variable it would show up there. Try typing `x <- 2` in the Console and hit enter, what do you get in the Environment pane?

## Packages

R is an open-source language, and developers contribute functionality to R via packages. In this lab we will work with three packages: **datasauRus** which contains the dataset, and **tidyverse** which is a collection of packages for doing data analysis in a “tidy” way.

If you haven’t installed these packages yet, you can install these packages by running the following command. (Note that R package names are case-sensitive)

```
install.packages(c("tidyverse", "datasauRus"))
```

Load them by running the following:

```
library(tidyverse)
library(datasauRus)
```

Note that the packages are also loaded with the same commands in your R Markdown document.

## Warm up

Before we introduce the data, let's warm up with some simple exercises.

The top portion of your R Markdown file (between the three dashed lines) is called YAML. It stands for “YAML Ain't Markup Language”. It is a human friendly data serialization standard for all programming languages. All you need to know is that this area is called the YAML (we will refer to it as such) and that it contains meta information about your document.

## YAML

Open the R Markdown (Rmd) file in your project, change the author name to your name, and knit the document.



## Data

The data frame we will be working with today is called `datasaurus_dozen` and it's in the `datasauRus` package. Actually, this single data frame contains 12 datasets, designed to show us why data visualisation is important and how summary statistics alone can be misleading. The different datasets are made by the `dataset` variable.

To find out more about the dataset, type the following in your Console: `?datasaurus_dozen`. A question mark before the name of an object will always bring up its help file. This command must be ran in the Console.

1. Based on the help file, how many rows and how many columns does the `datasaurus_dozen` file have? What are the variables included in the data frame? Add your responses to your lab report.

Let's take a look at what these datasets are. To do so we can make a *frequency table* of the dataset variable:

```
datasaurus_dozen %>%
  count(dataset)
```

```
## # A tibble: 13 x 2
##   dataset      n
##   <chr>    <int>
## 1 away      142
## 2 bullseye  142
## 3 circle    142
## 4 dino      142
## 5 dots      142
## 6 h_lines   142
## 7 high_lines 142
## 8 slant_down 142
## 9 slant_up   142
## 10 star      142
## 11 v_lines   142
## 12 wide_lines 142
## 13 x_shape   142
```

```
dim(datasaurus_dozen)
```

```
## [1] 1846    3
```

```
names(datasaurus_dozen)
```

```
## [1] "dataset" "x"      "y"
```

The original Datasaurus (**dino**) was created by Alberto Cairo in this great blog post. The other Dozen were generated using simulated annealing and the process is described in the paper *Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics* through Simulated Annealing by Justin Matejka and George Fitzmaurice. In the paper, the authors simulate a variety of datasets that the same summary statistics to the Datasaurus but have very different distributions.

## Data visualization and summary

2. Plot **y** vs. **x** for the **dino** dataset. Then, calculate the correlation coefficient between **x** and **y** for this dataset.

Below is the code you will need to complete this exercise. Basically, the answer is already given, but you need to include relevant bits in your Rmd document and successfully knit it and view the results.

Start with the **datasaurus\_dozen** and pipe it into the **filter** function to filter for observations where **dataset == "dino"**. Store the resulting filtered data frame as a new data frame called **dino\_data**.

```
dino_data <- datasaurus_dozen |>  
  filter(dataset == "dino")
```

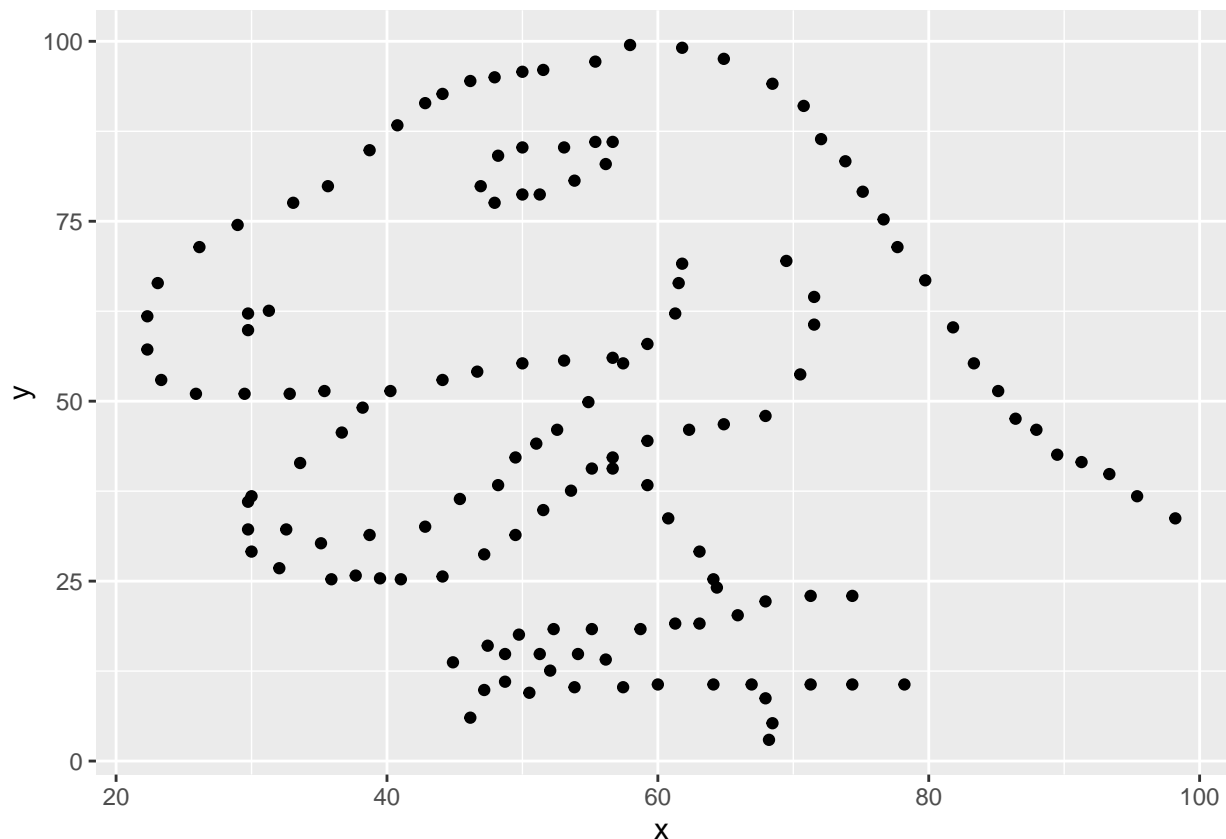
There is a lot going on here, so let's slow down and unpack it a bit.

First, the pipe operator: **%>%**, which has been updated to **|>** (but both work), takes what comes before it and sends it as the first argument to what comes after it. So here, we're saying **filter** the **datasaurus\_dozen** data frame for observations where **dataset == "dino"**.

Second, the assignment operator: **<-**, assigns the name **dino\_data** to the filtered data frame.

Next, we need to visualize these data. We will use the **ggplot** function for this. Its first argument is the data you're visualizing. Next we define the **aesthetic** mappings. In other words, the columns of the data that get mapped to certain aesthetic features of the plot, e.g. the **x** axis will represent the variable called **x** and the **y** axis will represent the variable called **y**. Then, we add another layer to this plot where we define which **geometric** shapes we want to use to represent each observation in the data. In this case we want these to be points, hence **geom\_point**.

```
ggplot(data = dino_data, mapping = aes(x = x, y = y)) +  
  geom_point()
```



If this seems like a lot, it is. And you will learn about the philosophy of building data visualizations in layer in detail in the upcoming class. For now, follow along with the code that is provided.

For the second part of this exercises, we need to calculate a summary statistic: the correlation coefficient. Correlation coefficient, often referred to as  $r$  in statistics, measures the linear association between two variables. You will see that some of the pairs of variables we plot do not have a linear relationship between them. This is exactly why we want to visualize first: visualize to assess the form of the relationship, and calculate  $r$  only if relevant. In this case, calculating a correlation coefficient really doesn't make sense since the relationship between  $x$  and  $y$  is definitely not linear – it's dinosaurial!

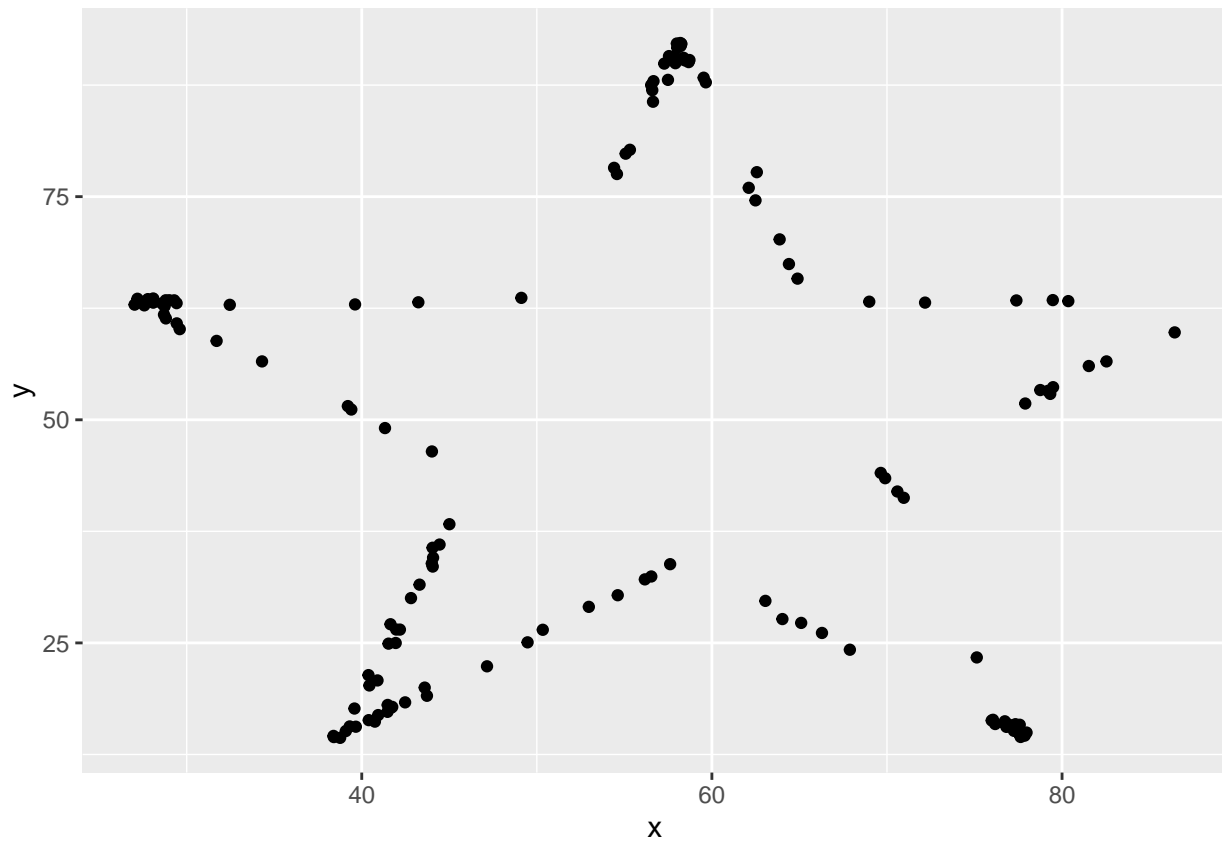
But, for illustrative purposes, let's calculate correlation coefficient between  $x$  and  $y$ .

Start with `dino_data` and calculate a summary statistic that we will call `r` as the correlation between  $x$  and  $y$ .

```
dino_data |>
  summarize(r = cor(x, y))
```

3. Plot  $y$  vs.  $x$  for the `star` dataset. You can (and should) reuse code we introduced above, just replace the dataset name with the desired dataset. Then, calculate the correlation coefficient between  $x$  and  $y$  for this dataset. How does this value compare to the  $r$  of `dino`?

```
ggplot(data = star_data, mapping = aes(x = x, y = y)) +
  geom_point()
```

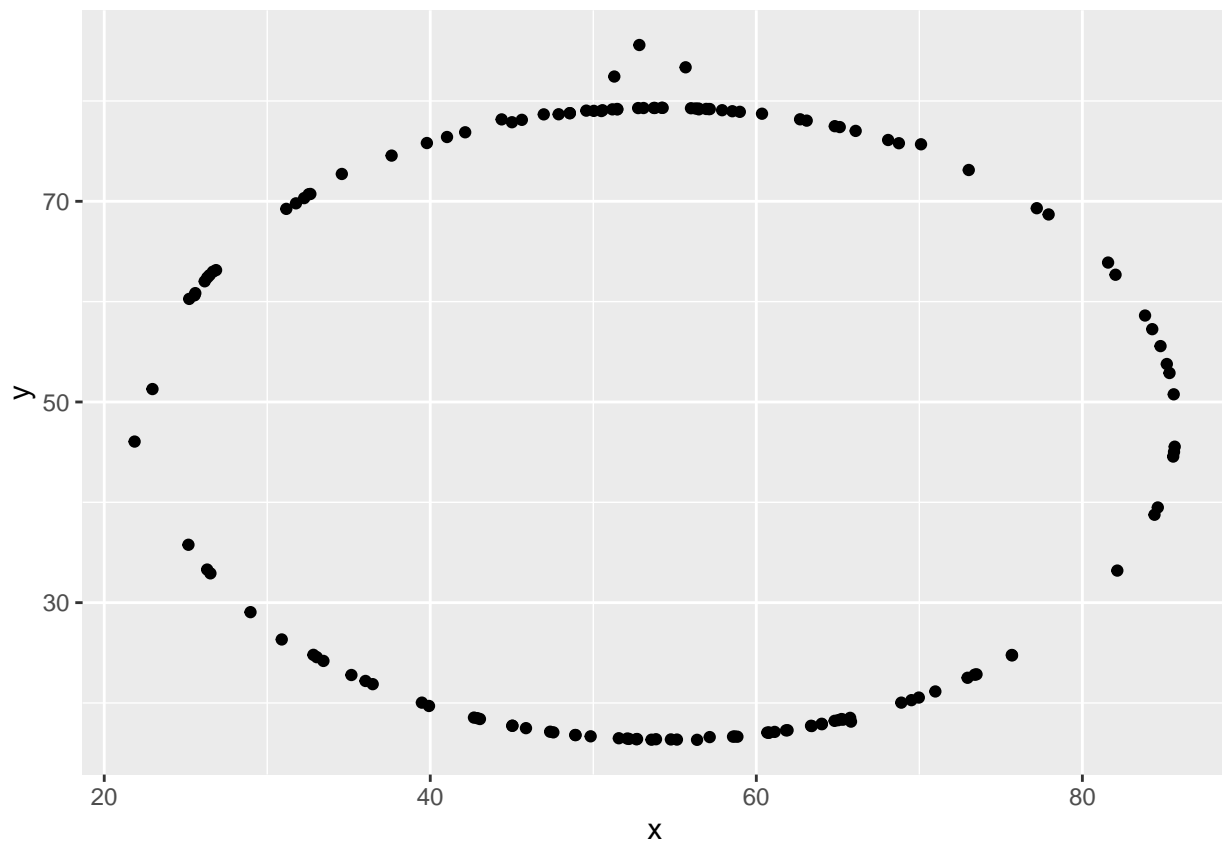


```
star_data |>
  summarize(r = cor(x, y))
```

```
## # A tibble: 1 x 1
##       r
##   <dbl>
## 1 -0.0630
```

4. Plot  $y$  vs.  $x$  for the `circle` dataset. You can (and should) reuse code we introduced above, just replace the dataset name with the desired dataset. Then, calculate the correlation coefficient between  $x$  and  $y$  for this dataset. How does this value compare to the  $r$  of `dino`?

```
ggplot(data = circle_data, mapping = aes(x = x, y = y)) +
  geom_point()
```



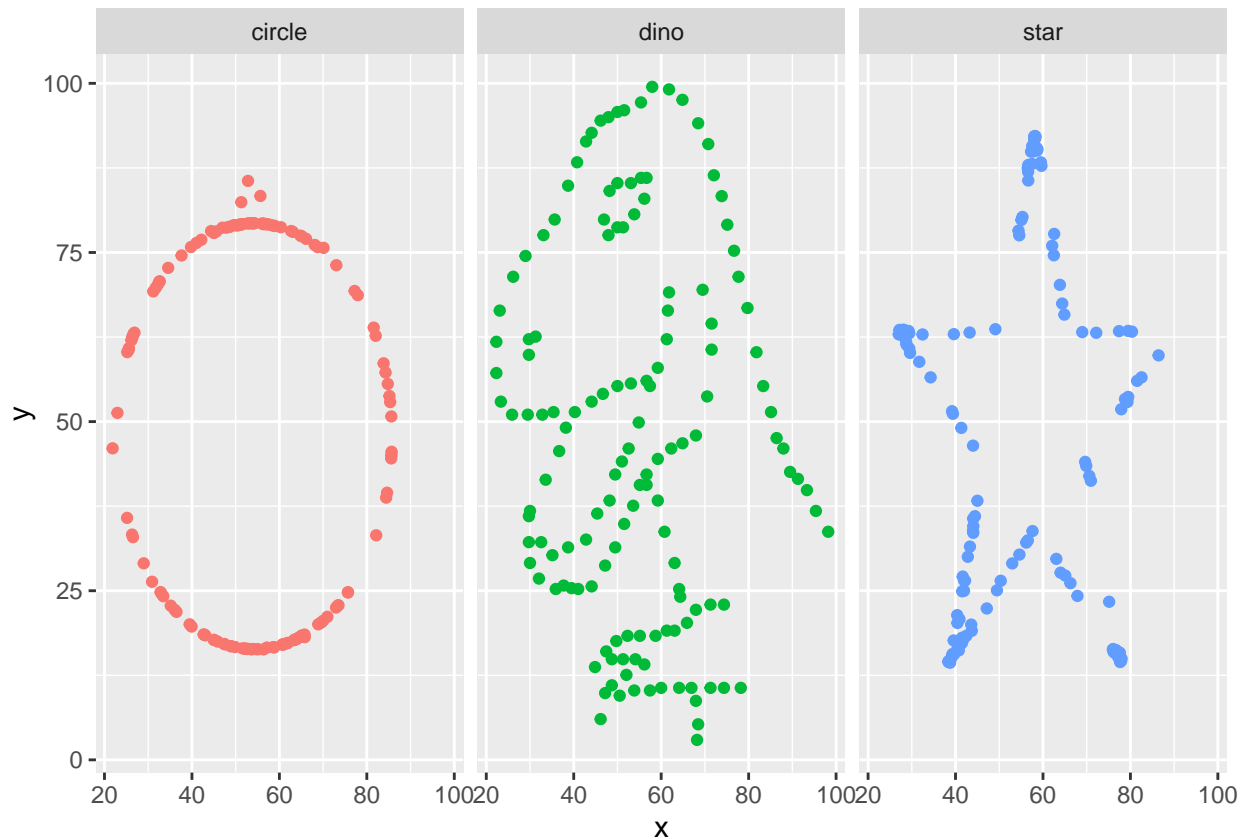
```
circle_data |>
  summarize(r = cor(x, y))
```

```
## # A tibble: 1 x 1
##       r
##   <dbl>
## 1 -0.0683
```

```
selected_datasets <- datasaurus_dozen |>
  filter(dataset %in% c("dino", "star", "circle"))
```

```
ggplot(selected_datasets, aes(x = x, y = y, color = dataset)) +
  geom_point() +
  facet_wrap(~dataset, ncol = 3) +
  theme(legend.position = "none")
```





Facet by the dataset variable, placing the plots in a 3 column grid, and don't add a legend.

5. Finally, let's plot all datasets at once. In order to do this we will make use of facetting.


```
ggplot(datasaurus_dozen, aes(x = x, y = y, color = dataset)) +  
  geom_point() +  
  facet_wrap(~dataset, ncol = 3) +  
  theme(legend.position = "none")
```

And we can use the `group_by` function to generate all correlation coefficients.

```
datasaurus_dozen |>  
  group_by(dataset) |>  
  summarize(r = cor(x, y))
```

You're done with the data analysis exercises, but we'd like you to do two more things:

## Edit R Markdown Document Options

**Output Format:** HTML 

Recommended format for authoring (you can switch to PDF or Word output anytime).

General

Figures

Advanced

Default figure width in inches:

Default figure height in inches:

☐ Render figures with captions

OK

Cancel

- **Resize your figures:**

Click on the gear icon in on top of the R Markdown document, and select “Output Options...” in the dropdown menu. In the pop up dialogue box go to the Figures tab and change the height and width of the figures, and hit OK when done. Then, knit your document and see how you like the new sizes. Change and knit again and again until you’re happy with the figure sizes. Note that these values get saved in the YAML.

```

fig.height=3, fig.width=6}
asaurus_dozen %>%
  == "dino")

no_data, mapping = aes(x = x, y = y)) +

te the correlation between `x` and `y` in this

%>%
== "dino") %>%
cor(x, y))

```

Name:

Output:

☐ Show warnings

☐ Show messages

☐ Use paged tables

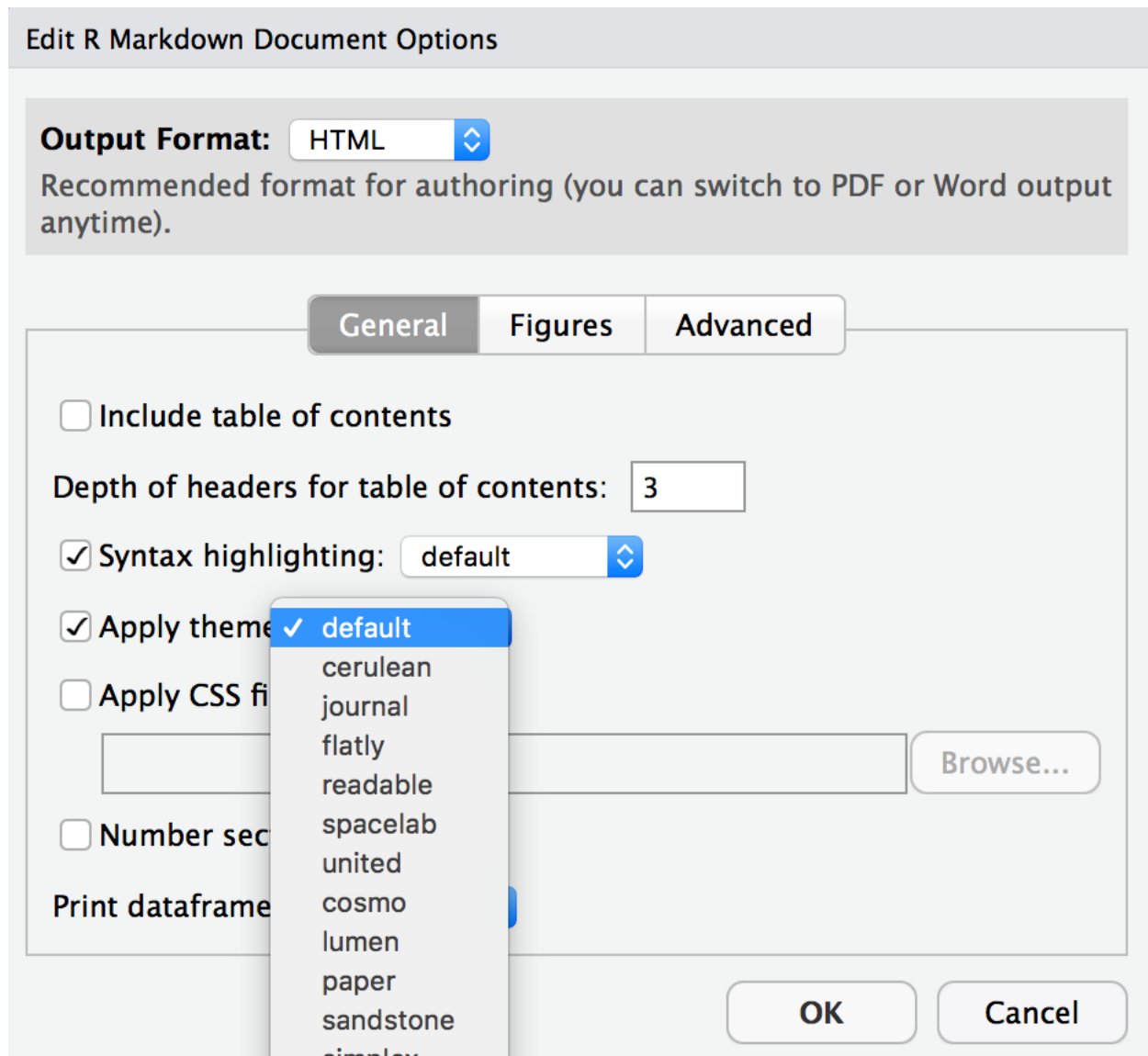
☒ Use custom figure size

Width (inches):

Height (inches):

[? Chunk options](#)

You can also use different figure sizes for different figures. To do so click on the gear icon within the chunk where you want to make a change. Changing the figure sizes added new options to these chunks: `fig.width` and `fig.height`. You can change them by defining different values directly in your R Markdown document as well.



- **Change the look of your report:**

Once again click on the gear icon in on top of the R Markdown document, and select “Output Options...” in the drop-down menu. In the General tab of the pop up dialogue box try out different syntax highlighting and theme options. Hit OK and knit your document to see how it looks. Play around with these until you’re happy with the look.

## Optional

If you have time you can explore the different ways you can add styling to your rmarkdown document.

Here is a Rmarkdown cheatsheet

and a general markdown cheatsheet and some themes and more themes

---

This set of lab exercises have been adopted from Mine Çetinkaya-Rundel’s class Introduction to Data Science.