STAT 231: Problem Set 0

YOUR NAME HERE

due by 5 PM on Thursday, August 27

This problem set is a modified version of a lab developed by Mine Cetinkaya-Rundel.

The main goal of this lab is to introduce you to working with R and RStudio in conjunction with git and GitHub, all of which we will be using throughout the course.

Packages

In this lab we will work with two packages: `datasauRus` which contains the dataset, and `tidyverse` which is a collection of packages for doing data analysis in a "tidy" way.

If you are using R on your local machine, you may need to install these packages by running the following in the console:

```r
install.packages("tidyverse")
install.packages("datasauRus")
```

Now that the necessary packages are installed, you should be able to Knit your document and see the results.

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(datasauRus)
```

Warm up

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

YAML:

The top portion of the 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.

In the YAML, change the author name to your name, and knit the document.

Commiting changes:

Then Go to the Git pane in your RStudio.

If you have made changes to your Rmd file, you should see it listed here. Click on it to select it in this list and then click on **Diff**. This shows you the *diff*erence between the last committed state of the document and its current state that includes your changes. If you're happy with these changes, write "Update author name" in the **Commit message** box and hit **Commit**.

You don't have to commit after every change, this would get quite cumbersome. You should consider committing states that are *meaningful to you* for inspection, comparison, or restoration. In the first few assignments, I will tell you exactly when to commit and in some cases, what commit message to use. As the semester progresses, I will let you make these decisions.

Pushing changes:

Now that you have made an update and committed this change, it's time to push these changes to the web! Or more specifically, to your repo on GitHub. Why? So that others can see your changes. And by others, we mean the course teaching team (your repos in this course are private to you and us, only).

In order to push your changes to GitHub, click on **Push**.

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 13 datasets (have you heard of a baker's dozen?), designed to show us why data visualisation is important and how summary statistics alone can be misleading. The different datasets are identified 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.

EXERCISE 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? After you've added your response below, commit your changes with the commit message "Added answer for Ex 1", and push.

> RESPONSE: 1846 rows, 3 variables: dataset, x, y

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) %>%
  print(13)
```

```
## # 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
```

Data visualization and summary

EXERCISE 2.

Calculate the correlation coefficient between `x` and `y` for the `dino` 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. What does this correlation coefficient imply about the relationship between `x` and `y` in the `dino` dataset?

RESPONSE:

```
# Start with the ‘datasaurus_dozen‘ and pipe it into the ‘filter‘ function to filter for observations w
dino_data <- datasaurus_dozen %>%
  filter(dataset == "dino")

# Calculate a summary statistic that we call ‘r‘ equal to the correlation coefficient between ‘x‘ and ‘
dino_data %>%
  summarize(r = cor(x, y))
```

```
## # A tibble: 1 x 1
##         r
##     <dbl>
## 1 -0.0645
```

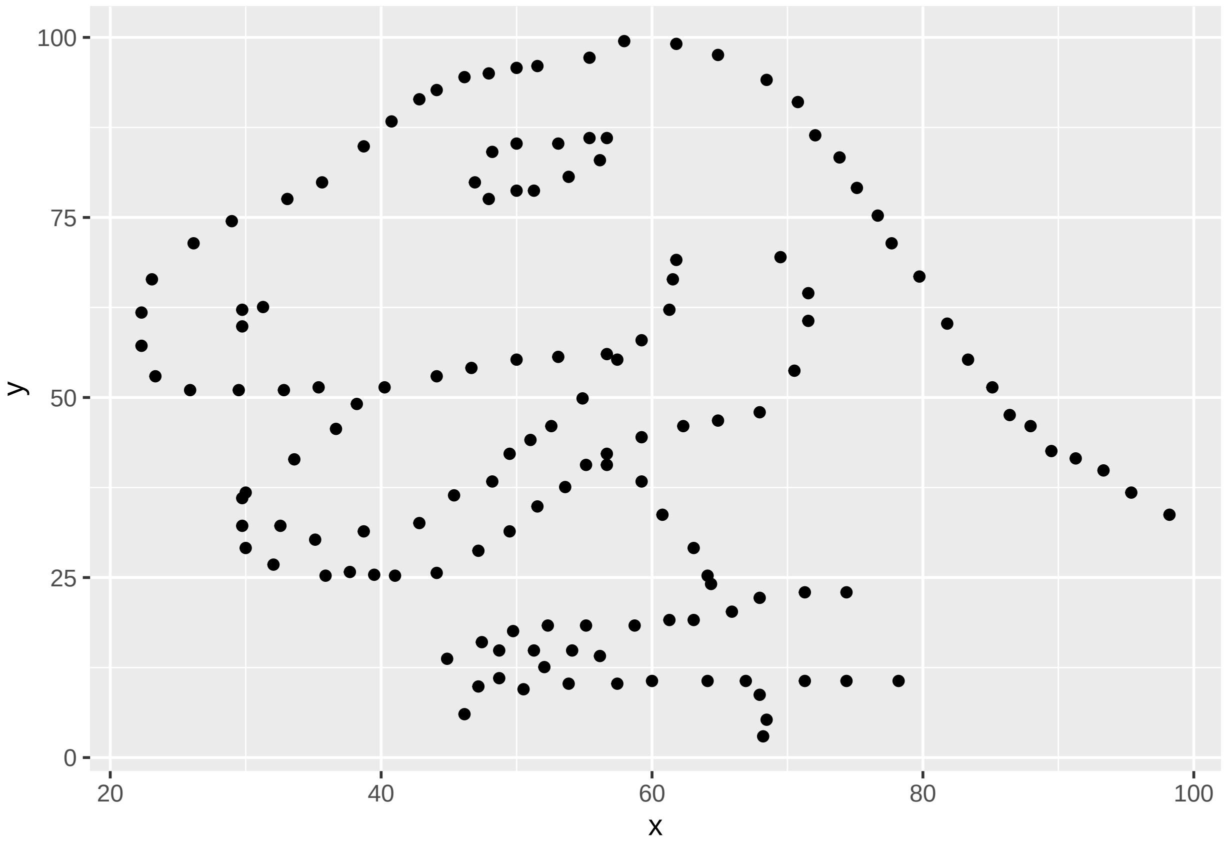
Now, plot `y` vs. `x` for the `dino` dataset. We will use the `ggplot` function for this. Its first argument is the data you're visualizing. Next we define the `aes`thetic mappings. In other words, the columns of the data that get mapped to certain aesthetic features of the plot, e.g. the `x` axis will 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`.

If this seems like a lot, don't worry. You will learn about the philosophy of building data visualizations in layer in detail next week. For now, follow along with the code that is provided.

What do you notice now about the relationship between x and y?

RESPONSE:They are correlated in dinosaur-shape. Remarkable.

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



This is a good place to pause, commit changes with the commit message "Added answer for Ex 2", and push.

EXERCISE 3.

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

RESPONSE: This correlation is only slightly less negative than the previous.

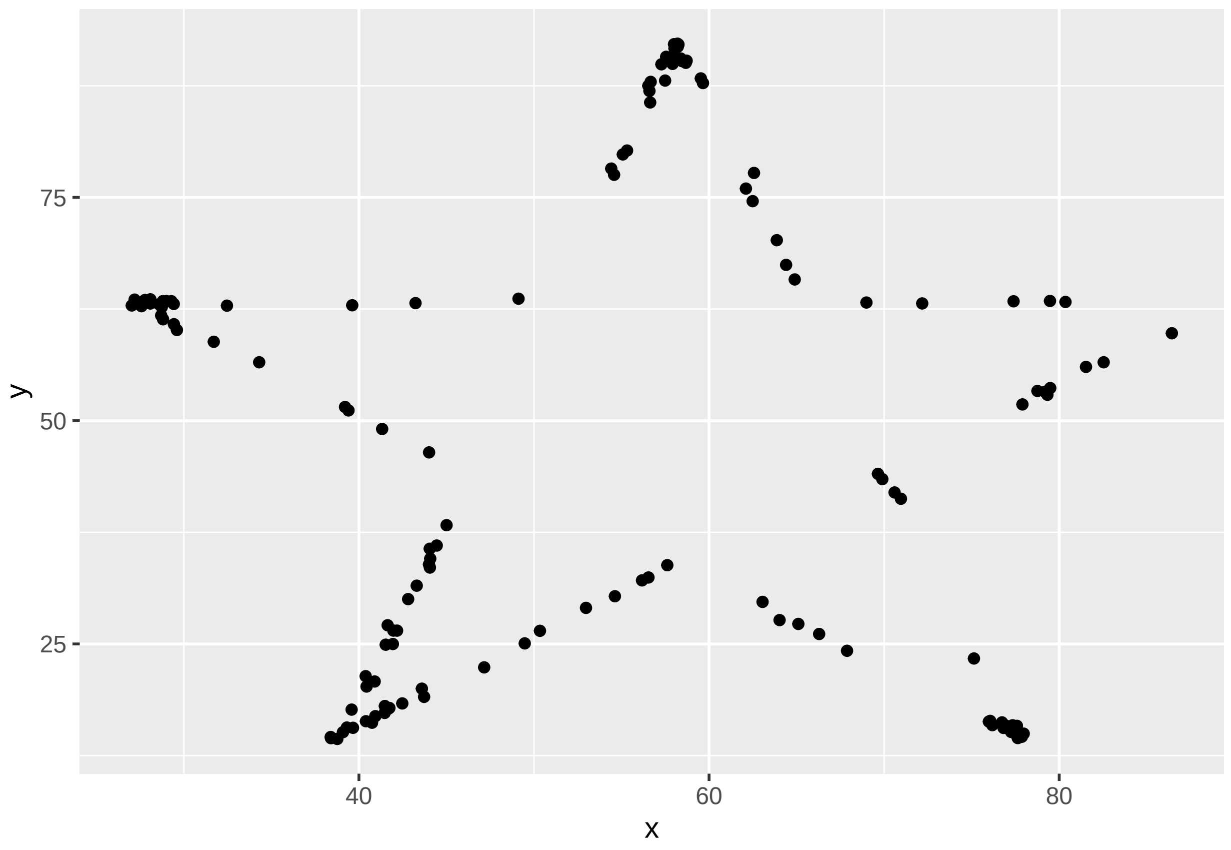
```r
dino_data <- datasaurus_dozen %>%
  filter(dataset == "star")
dino_data %>%
  summarize(r = cor(x, y))
```

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

Then, plot `y` versus `x` for the `star` dataset. Does the plot look the same as the plot of the `dino` data?

> RESPONSE:

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



This is another good place to pause, commit changes with the commit message "Added answer for Ex 3", and push.

EXERCISE 4.

Calculate the correlation coefficient between `x` vs. `y` for the `circle` dataset. You can (and should) reuse code we introduced above, just replace the dataset name with the desired dataset. How does this value compare to the correlation coefficients from the other two datasets?

RESPONSE: This correlation is slightly more negative than both of the previous.

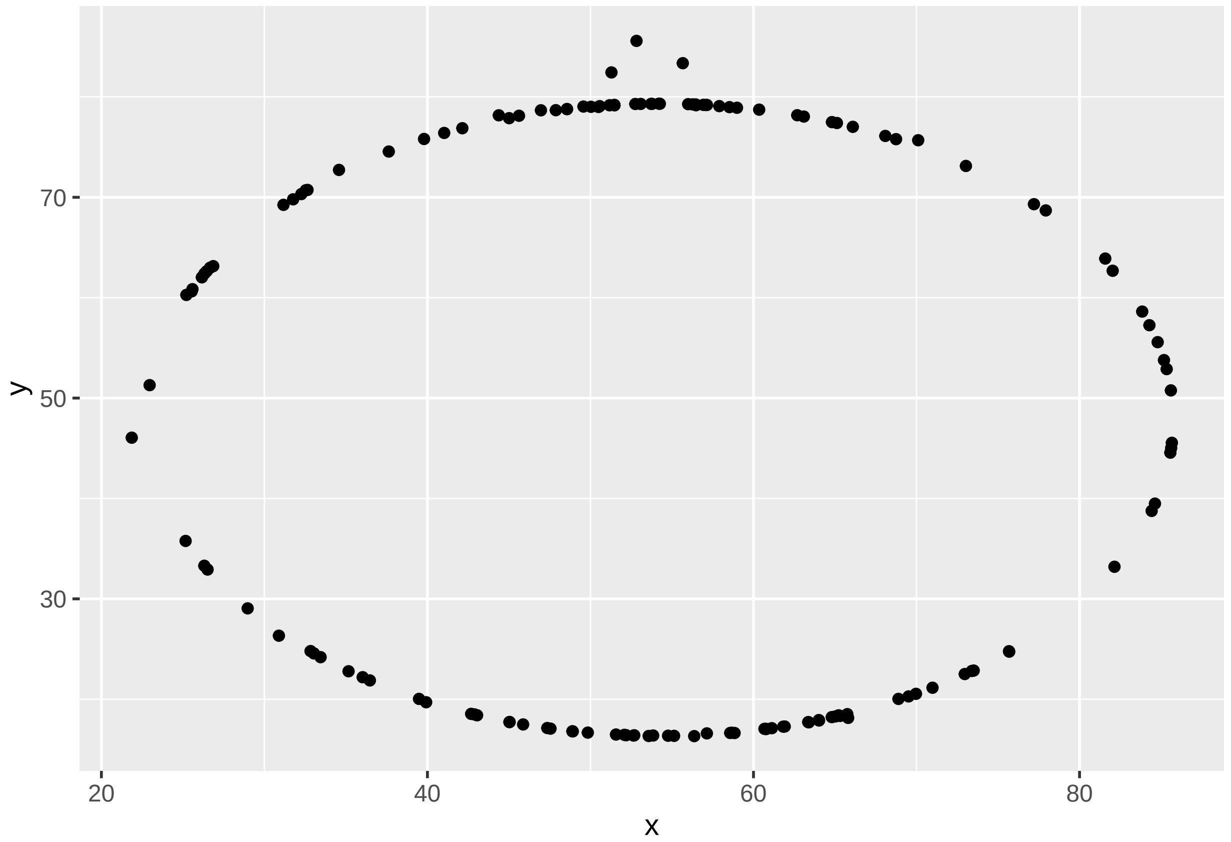
```
dino_data <- datasaurus_dozen %>%
  filter(dataset == "circle")
dino_data %>%
  summarize(r = cor(x, y))
```

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

Then, plot `y` versus `x` for the `circle` dataset. Does the plot look the same as either plot from the other two datasets?

RESPONSE:

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

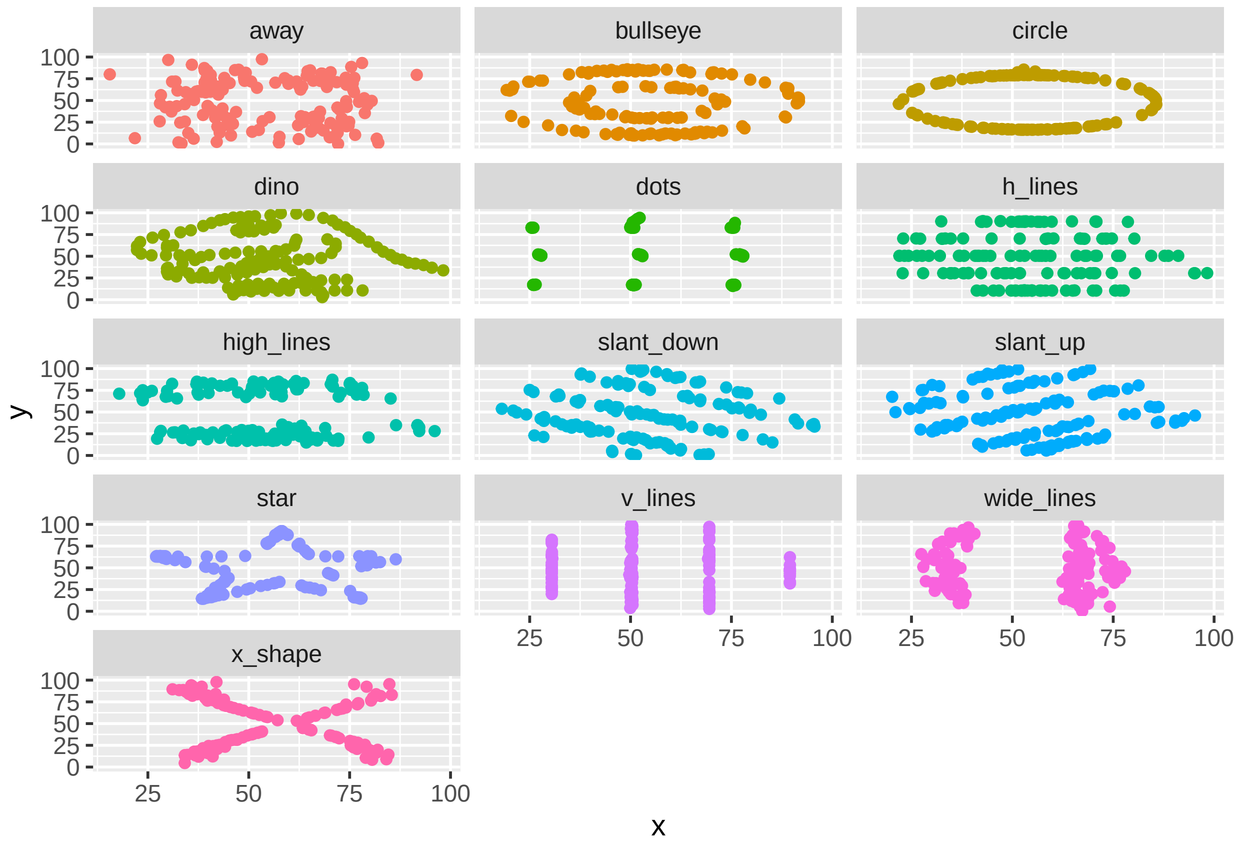


You should pause again, commit changes with the commit message "Added answer for Ex 4", and push.

EXERCISE 5.

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

```r
# Scatterplots by dataset
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 the summary correlation coefficients.

```r
# Correlations by dataset
datasaurus_dozen %>%
  group_by(dataset) %>%
  summarize(r = cor(x, y)) %>%
  print(13)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 13 x 2
##    dataset          r
##    <chr>        <dbl>
##  1 away       -0.0641
##  2 bullseye   -0.0686
##  3 circle     -0.0683
##  4 dino       -0.0645
##  5 dots       -0.0603
##  6 h_lines    -0.0617
##  7 high_lines -0.0685
##  8 slant_down -0.0690
##  9 slant_up   -0.0686
## 10 star       -0.0630
## 11 v_lines    -0.0694
## 12 wide_lines -0.0666
## 13 x_shape    -0.0656
```

What do you notice? Is the correlation coefficiant an appropriate summary for any of these datasets? Why or why not?

> RESPONSE: All the coefficients are very similar. They are a poor summary, since a linear regression would fit these datasets poorly. The fact that a similar coefficient is used to summarize them all demonstrates this.

You should pause again, commit changes with the commit message "Added answer for Ex 5", and push.

Submitting to Gradescope

Lastly, knit this file to a PDF, push the PDF to GitHub, then upload the PDF to Gradescope as your submission for Problem Set 0. Note that whether you're working in RStudio via the Amherst server or your own machine, you can upload the PDF to gradescope by finding the file in your GitHub folder (i.e. you do not need to download the file from the server to your machine if working on the server).

When you submit to Gradescope, you will be asked to select which of your pages are associated with a particular problem (e.g. if your answer to Exercise 1 runs across pages 2 and 3, then you need to assign pages 2 and 3 to Exercise 1). For problem sets going forward, be sure you assign your solutions correctly and fully – *you will NOT get credit for work on unassigned pages (e.g., if you only selected page 2 but not page 3 for Exercise 1, you would lose points for any page 3 work that the grader can't see).* It's a lot of work for the grader to try to track down the rest of your answer if the pages aren't selected when you upload your work, so please be mindful to do this correctly.

Please see me if you have any questions or difficulty in submitting!

[Note that this assignment is designed to ensure you've successfully submitted an assignment to Gradescope before the first real assignment is due. This submission will not be graded for correctness; you will receive full credit for having the assignment submitted on time.]

References

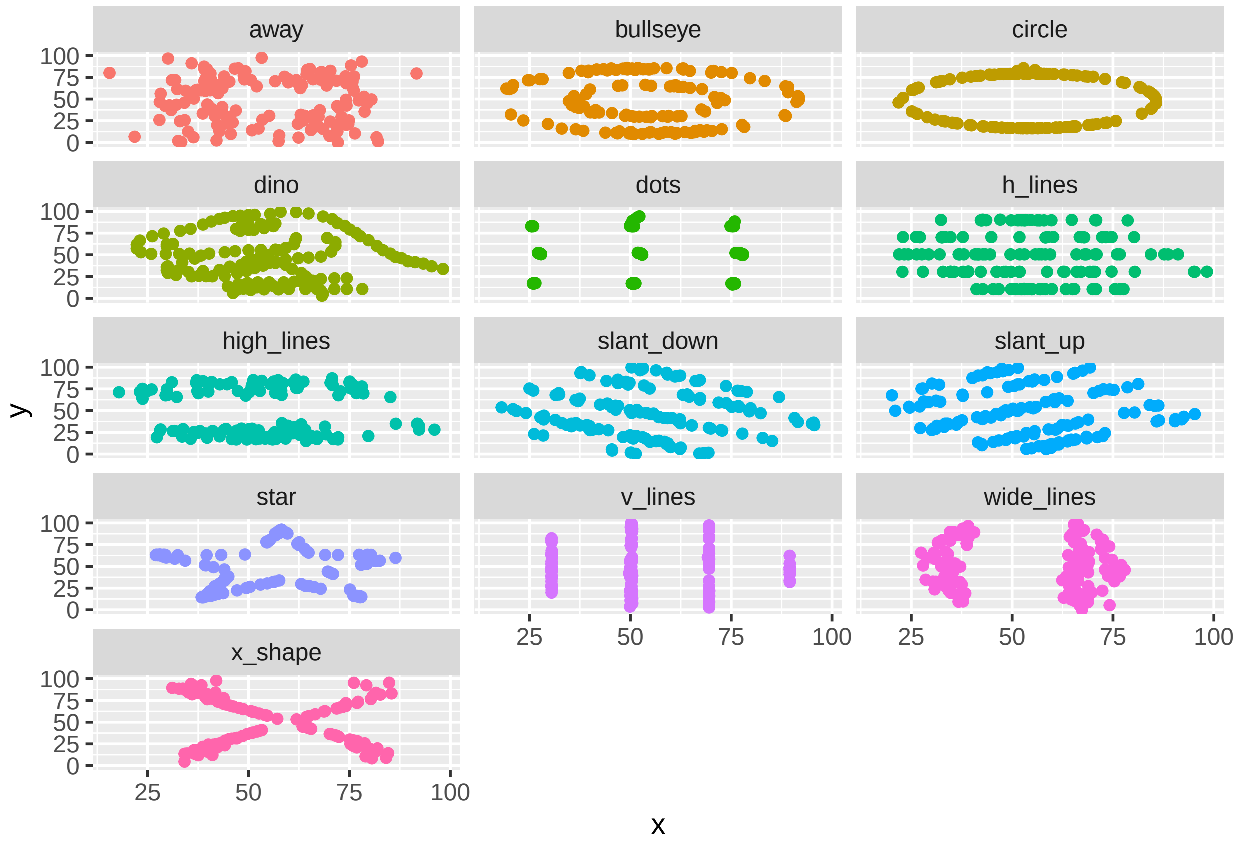
The original Datasaurus (`dino`) was created by Alberto Cairo in this great blog post. The other Dozen were generated using simulated annealing and the process is described in the paper *Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated 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.

Extra (R Chunk Options)

Done early? You might have noticed that this .Rmd lacks the usual global chunk options that most/all of my .Rmd files have. Let's explore how these options work when compared to the default settings. Try adding the following options (ONE AT A TIME) to the code chunk below (this is the same code as in Exercise 5) and re-knitting the PDF. Try to identify what each option is doing to the PDF output.

- echo=FALSE
- eval=FALSE
- include=FALSE
- comment=NULL
- comment=";)"
- prompt=TRUE
- collapse=TRUE

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



```r
# Correlations by dataset
datasaurus_dozen %>%
  group_by(dataset) %>%
  summarize(r = cor(x, y)) %>%
  print(13)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 13 x 2
##    dataset          r
##    <chr>        <dbl>
##  1 away       -0.0641
##  2 bullseye   -0.0686
##  3 circle     -0.0683
##  4 dino       -0.0645
##  5 dots       -0.0603
##  6 h_lines    -0.0617
##  7 high_lines -0.0685
##  8 slant_down -0.0690
##  9 slant_up   -0.0686
## 10 star       -0.0630
## 11 v_lines    -0.0694
## 12 wide_lines -0.0666
## 13 x_shape    -0.0656
```

RESPONSE:

There are a lot more R code chunk options, most of which we'll not use in this course. But if you're interested in the full list, check out the RMarkdown Reference Guide

Extra Extra (Inline R Code)

Done exploring R code chunk options? Wow, you're fast! Here's another fun functionality of RMarkdown: You can use inline R code chunks to place R numerical outout in your text. For instance:

The dino dataset contains 142 observations and has 3 variables. The mean x value is 54.2673197 units and the mean y value is 47.8377173 units.

You can use the `round` command to get a sensible number of digits: The mean x value is 54.3 units and the mean y value is 47.8 units.

Your turn: Use inline R coding to get the standard deviation of x in the dino dataset reported to 1 decimal spot.

> RESPONSE: