RMarkdown Exercise

Shreya Mohapatra

2023-03-29

Contents

An Exercise in Importance of Data Visualisation

I dino what to tell ya, but here() it goes anyway

The Datasaurus Dozen, a misnomer, contains 13 datasets: Alberto Cairo's Datasaurus or as he prefers Anscombosaurus, and 12 others created by Justin Matejka & George Fitzmaurice. Each dataset has the same summary statistics (mean, sd, and Pearson'r r) but results in very different visualizations. Similar in principle to Anscombe's Quartet, *the purpose of this dataset is to highlight the importance of graphical representation.*

Here's a very fun tool created by Robert Grant to create your own dataset based on what you want the scatterplot to look like! A great way to understand both stats and visualization.

Alright, let's begin trying to recreate the Datasaurus Dozen plots.

Loading the data

There's a couple of options:

- Download the CSV file from the Datasaurus Dozen link in the previous section.
- Use it as a **package**! Yup. The beautiful R community has turned these datasets into an R package, led by Steph Locke & Lucy McGowan. You can view this project on their GitHub!

To load that data in as a package:

- you can either use the latest stable version available on CRAN

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

- or you can get the latest dev version from GitHub

```
devtools::install_github("jumpingrivers/datasauRus")
```

Next, let's get some libraries loaded in

```
library(ggplot2)
library(tidyverse)
library(here)
library(gganimate)
library(gifski)
library(datasauRus)
```

Let's take a look at the dataframe

The first column is "dataset". This groups the rows into the 13 datasets. This will be helpful when we write the code for plotting and animating because the code can then iterate through these groups. The second and third columns are "x" for x co-ordinates and "y" for y co-ordinates.

```
# shows the first six rows of the df
head(datasaurus_dozen, 6)
```

```
## # A tibble: 6 x 3
##   dataset     x     y
##   <chr>   <dbl> <dbl>
## 1 dino     55.4  97.2
## 2 dino     51.5  96.0
## 3 dino     46.2  94.5
## 4 dino     42.8  91.4
## 5 dino     40.8  88.3
## 6 dino     38.7  84.9
```

```
# shows the last six rows of the df
tail(datasaurus_dozen, 6)
```

```
## # A tibble: 6 x 3
##   dataset        x     y
##   <chr>      <dbl> <dbl>
## 1 wide_lines  34.7  19.6
## 2 wide_lines  33.7  26.1
## 3 wide_lines  75.6  37.1
## 4 wide_lines  40.6  89.1
## 5 wide_lines  39.1  96.5
## 6 wide_lines  34.6  89.6
```

Now, if we look at their descriptive statistics, these datasets will appear similar.

```
summary <- datasaurus_dozen %>%
  # grouping the data by the column dataset
  group_by(dataset) %>%
  # creating a summary of the datasets' means, standard deviation, and correlation to show that their d
  summarize(mean_x = mean(x),
            mean_y = mean(y),
            std_dev_x = sd(x),
            std_dev_y = sd(y),
            corr_x_y  = cor(x, y))
```

```
# to print/ view the results
print(summary)
```

```
## # A tibble: 13 x 6
##    dataset    mean_x mean_y std_dev_x std_dev_y corr_x_y
##    <chr>       <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1 away         54.3   47.8      16.8      26.9  -0.0641
##  2 bullseye     54.3   47.8      16.8      26.9  -0.0686
##  3 circle       54.3   47.8      16.8      26.9  -0.0683
##  4 dino         54.3   47.8      16.8      26.9  -0.0645
##  5 dots         54.3   47.8      16.8      26.9  -0.0603
##  6 h_lines      54.3   47.8      16.8      26.9  -0.0617
##  7 high_lines   54.3   47.8      16.8      26.9  -0.0685
##  8 slant_down   54.3   47.8      16.8      26.9  -0.0690
##  9 slant_up     54.3   47.8      16.8      26.9  -0.0686
## 10 star         54.3   47.8      16.8      26.9  -0.0630
## 11 v_lines      54.3   47.8      16.8      26.9  -0.0694
## 12 wide_lines   54.3   47.8      16.8      26.9  -0.0666
## 13 x_shape      54.3   47.8      16.8      26.9  -0.0656
```

It is easy to assume at this stage that these datasets would look exactly the same as each other when you plot them. But is that the case?

Time for the fun graphs!

We will create two versions to illustrate that even small changes can make big difference in how data can be presented.

```
# here were will generate a single image with a separate plot for each of the 13 datasets
simple <- datasaurus_dozen %>%
  # mapping the variables
  ggplot(aes(x = x, y = y, colour = dataset)) +
  # type of plot
  geom_point() +
  # background colour
  theme_bw() +
  # hiding legend because it would be redundant in this case
  theme(legend.position = "none") +
  # creates a ribbon of panels
  facet_wrap(~dataset, ncol = 3)
# to view the image generated
simple
```

```
# to save the image generated; you can change the image type to .jpeg or .TIFF etc; change "figs" to yo
ggsave(here("figs", "tester.png"))
```

v1

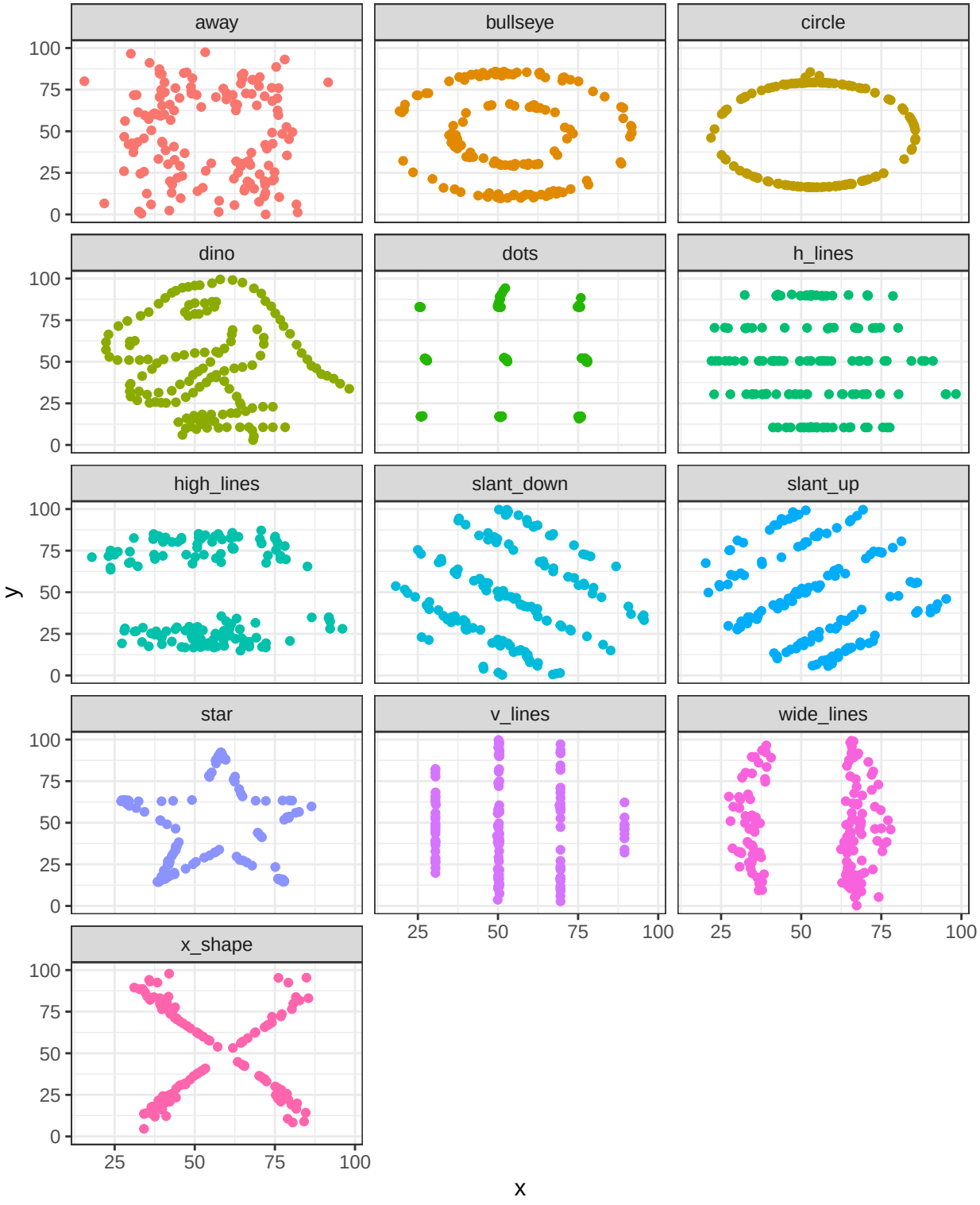

Figure 1: The DD #1

```
## Saving 6.5 x 8 in image
```

Now, this is a static image. It might be more fun to look at an animated version.

GGANIMATE v1 CODE CHUNK HERE

Satisfying! But can we refine it more?

```
# shorter code.. hmm.
notsosimple <- datasaurus_dozen %>%
  # mapping the variables
  ggplot(aes(x= x, y= y)) +
  # type of plot
  geom_point() +
  # changing the background colour
  theme_set(theme_bw())

# to save this image; change "figs" to your folder of choice within the working directory
ggsave(here("figs", "tester2.png"))
```

v2

```
## Saving 6.5 x 4.5 in image
```

```
# to view the plot we have generated
# what do you see?
notsosimple
```

```
# a single plot with ALL the data points from ALL the datasets overlapping each other. Why? Let's see w
```

GGANIMATE v2 CODE CHUNK HERE

The kind of visualization you choose for your data will depend on what your goal is:

- what is the question you are trying to answer or message you are trying to convey
- what kind of data do you have
- how can the data be presented clearly and is more or less self-explanatory
- can it be engaging without losing accuracy or important information

Well, hope that was fun and educational. `That's all for now.`

References

- Acknowledging Tom
- Animation code
- gganimate cheat code
- RMarkdown styling
- knitr

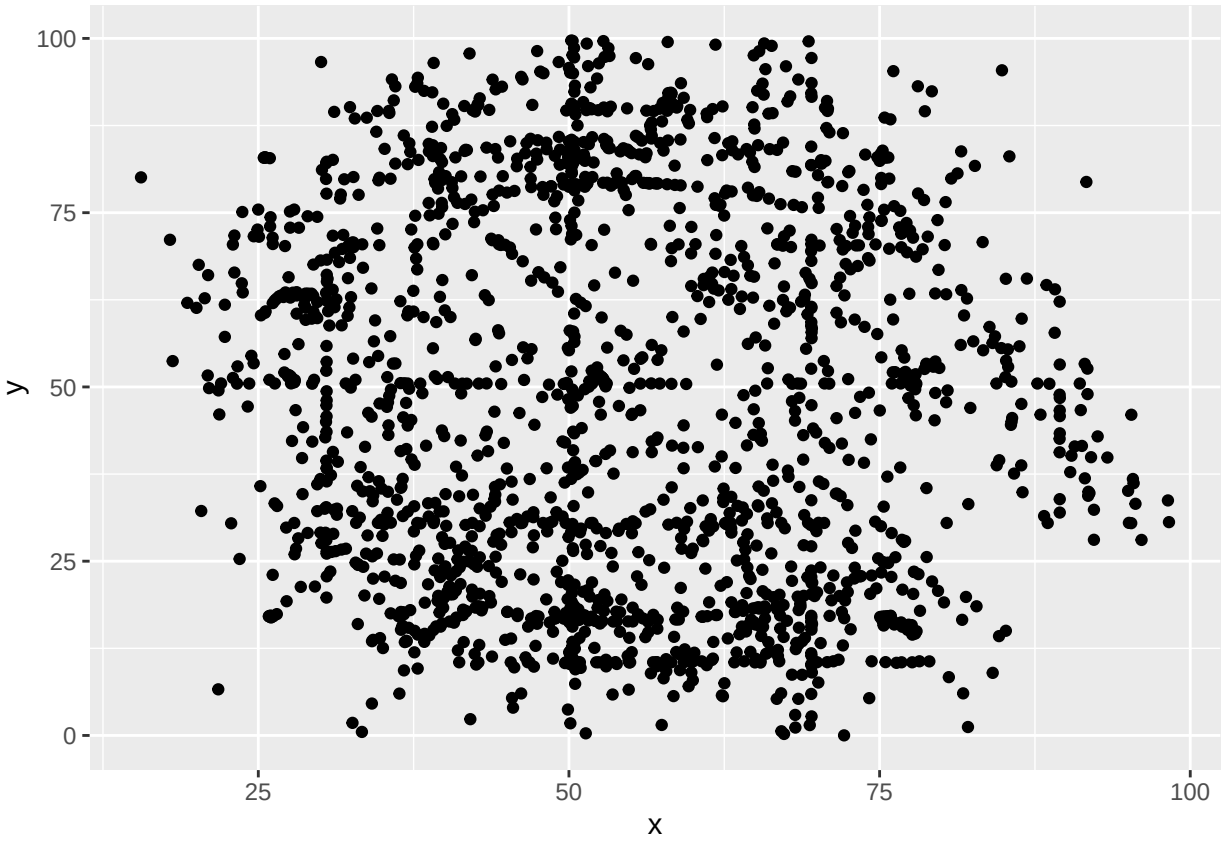

Figure 2: The DD #3