RMarkdown Exercise: Datasaurus Dozen

Shreya Mohapatra

first published: 2023-03-29; last updated: 2023-04-20

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's 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

You may need to install one or more of these packages first by using `install.package()`

```
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 coordinates and "y" for y coordinates.

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

```
# shows the last six rows of the df
tail(datasaurus_dozen, 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 descriptive stats, in this case, are
  # almost identical!
  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.*

VERSION ONE First, we need to create a ggplot that we will then use for our animation. The way your animation runs will depend on this ggplot you create. This will become evident when you look at VERSION TWO below.

```r
# here we 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
```

```r
# to save the image generated
# you can change the image type to .jpeg/ .TIFF etc
# change "figs" to your folder of choice within the working directory
ggsave(here("figs", "simpleplot.png"))
```

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

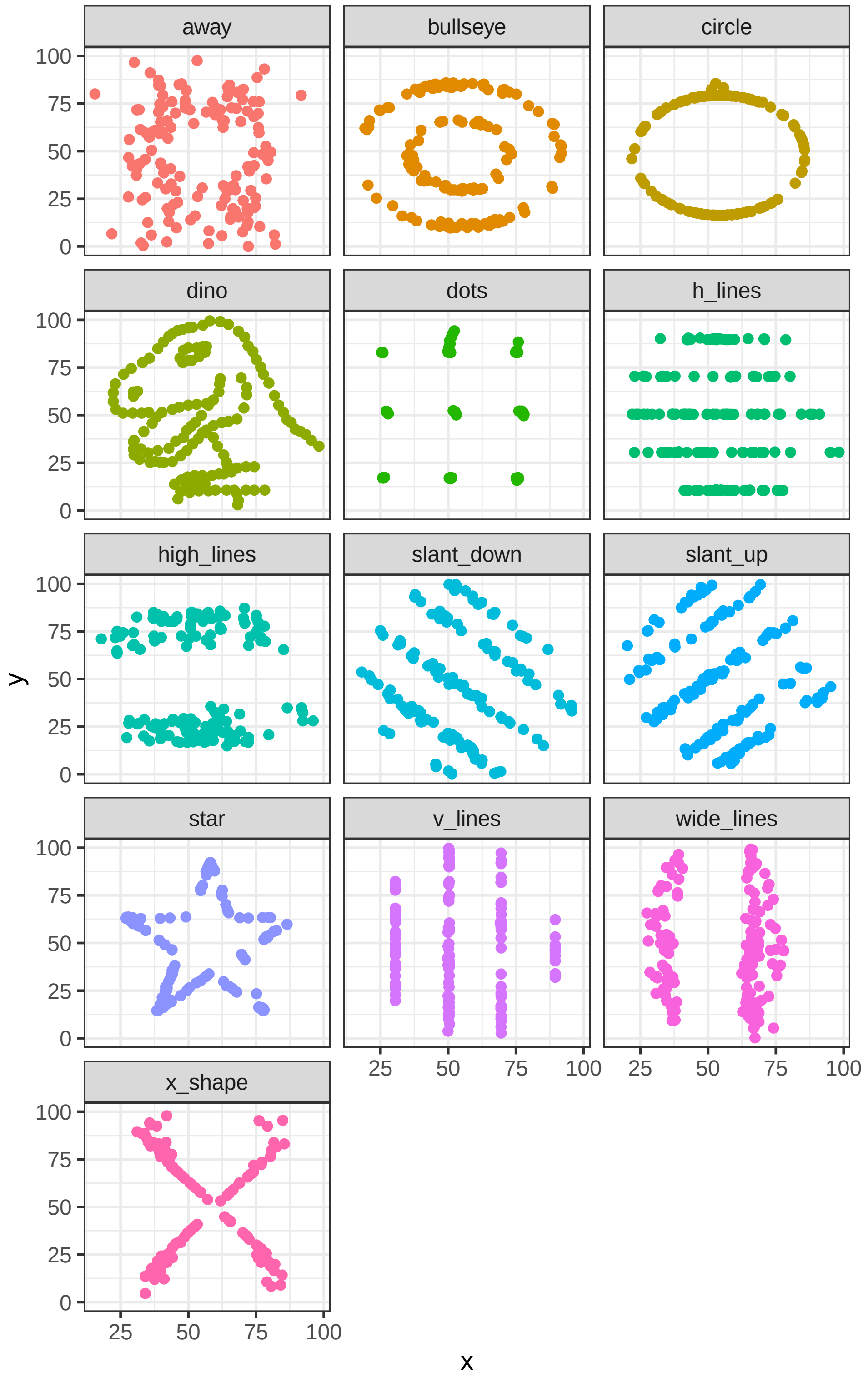

Figure 1: The DD ggplot #1

```r
# this will add the animation to cycle through the plots
scatter_viz <- simple +
  # to transition between the groups in column dataset
  transition_states(dataset,3,3) +
  # style of transition or animation
  ease_aes('cubic-in-out') +
  # label of each plot, will iterate through each "state"
  labs(title = "{closest_state}") +
  # to specify other elements of the plot
  theme(plot.title = element_text(size=22,hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())

animate(scatter_viz)

# to save the gif generated; works the same as ggsave() function
# change "figs" to your folder of choice within the working directory
anim_save(here("figs", "simpleanimated.gif"))

# to view the gif generated
scatter_viz
```

view here

Satisfying! But can we refine it more?

VERSION TWO We will create a new ggplot.

```r
# 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", "notsosimpleplot.png"))

# to view the plot we have generated
notsosimple
```

```r
# what do you see?
```

We see a single plot with ALL the data points from ALL the datasets overlapping each other. Why? Let's see what happens when we animate this next!

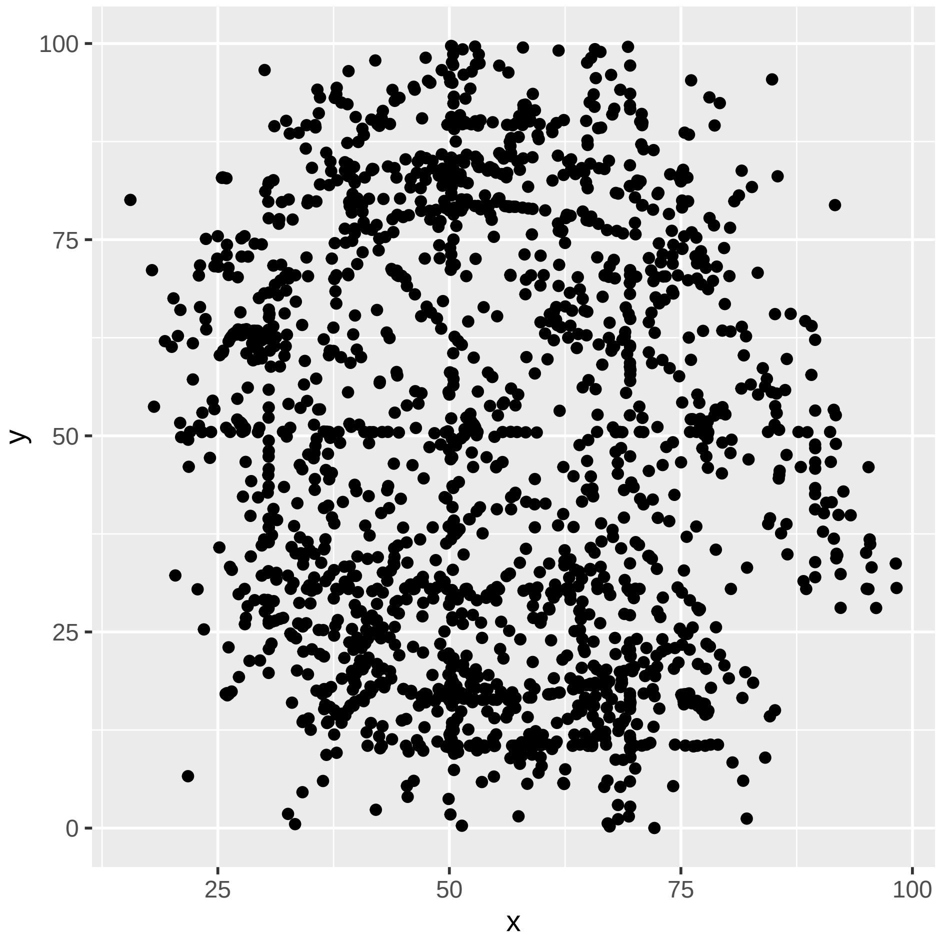

Figure 2: The DD ggplot #2

```
# creating animation
scatter_viz2 <- notsosimple +
  # to transition between the groups in column dataset
  transition_states(dataset,3,3) +
  # style of transition or animation
  ease_aes('cubic-in-out') +
  # label of each plot, will iterate through each "state"
  labs(title = "{closest_state}") +
  # to specify other elements of the plot
  theme(plot.title = element_text(size=22,hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
# don't forget to save this creation!
# change "figs" to your folder of choice within the working directory
anim_save(here("figs", "notsosimpleanimated.gif"))

# view animation
scatter_viz2
```

view here

Et, voila! The classic animation that you might have seen across the internet where it cycles through the different plots on the same graph.

You can now also appreciate the difference between creating facets in a graph (as in VERSION ONE) for the different datasets vs overlaying the datasets (as in VERSION TWO) in the graph.

CONCLUSION

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 Dr. Stafford
- Animation code
- gganimate cheat code
- RMarkdown styling
- RMarkdown basic syntax
- Advanced syntax
- Code chunk
- knitr