Data Visualization

Graphing our Data: Graph Tables, Add Labels, Make Notes

Ciara Zogheib

Data Sciences Institute, University of Toronto

Prerequisites

- You have the code from class open in RStudio

Coming up, we will...

- Learn new geoms and alter ggplot defaults as intermediate users
- Work through practice code from Chapter 5 (*Graph Tables, Add Labels, Make Notes*) of Healy, K. (2018). Data Visualization: A Practical Introduction. Princeton University Press. This code will let us
    - Organize continuous variables by group or category (in boxplots or scatter plots)
    - Plot text
    - Label outliers
    - Write and draw on our plots

Continuous variables by groups or categories

Continuous variables by group or category

- We will use a new sample dataset (**organdata**, from the **socviz** library) containing numerical and categorical information on organ donations in seventeen OECD countries
- We will use a pipeline to select five rows at random and view the first six columns for those rows:

```
organdata |> select(1:6) |> sample_n(size = 10)
```

(Healy, 2018)

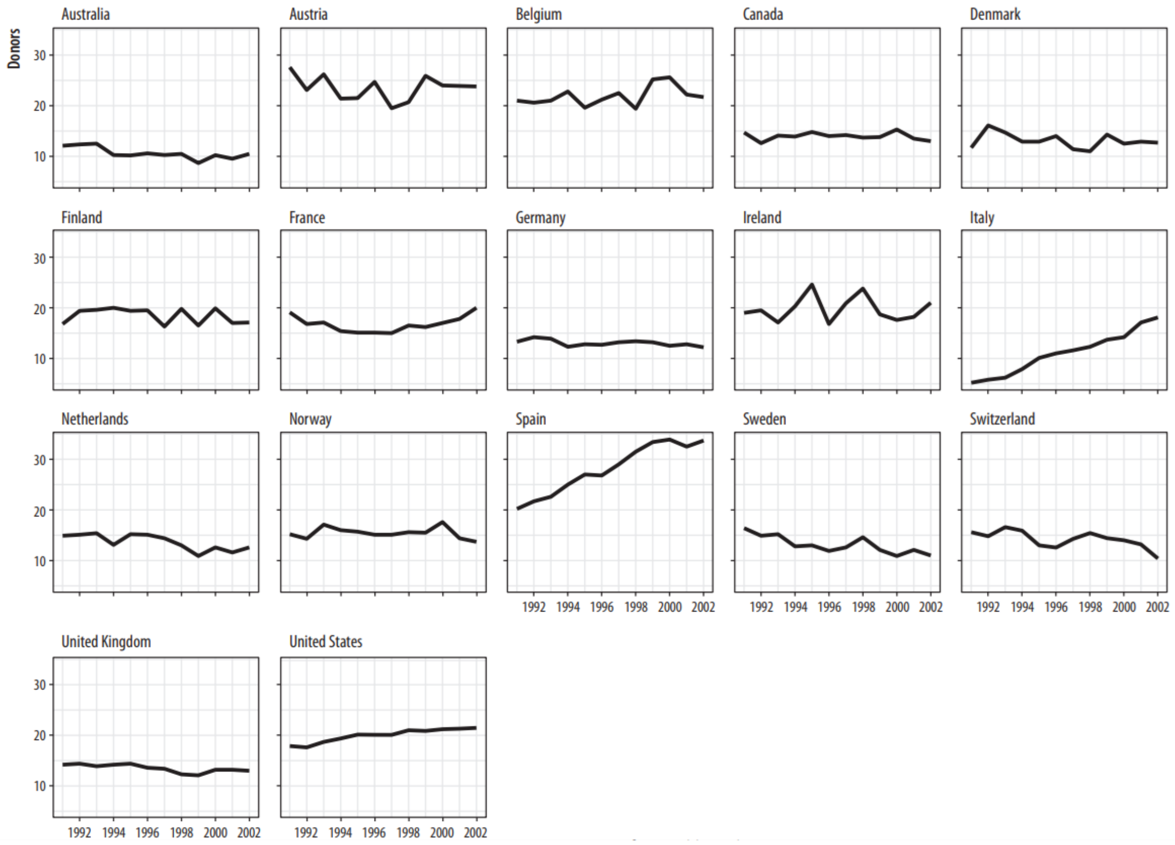
Recall: Groups and facets

- We can use `geom_line()` to explore each country's time series, incorporating grouping and faceting:

```
p <- ggplot(data = organdata, mapping = aes(x = year, y = donors))

p + geom_line(aes(group = country)) + facet_wrap(~country)
```

(Healy, 2018)

Recall: Groups and facets - Result



(Healy, 2018)

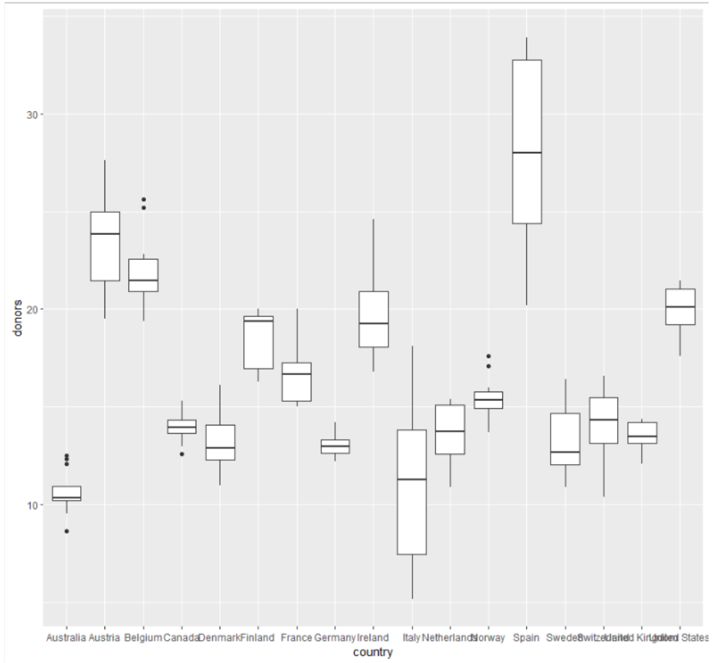
geom_boxplot()

- We can use `geom_boxplot()` to focus on variation across countries rather than a time trend
- The `stat_boxplot()` function works in `geom_boxplot()` to calculate statistics allowing a box and whiskers to be drawn
- We tell `geom_boxplot()` the variable we want to categorize by (`country`) and the continuous variable we want to be summarized (`donors`)

```
p <- ggplot(data = organdata, mapping = aes(x = country, y =
donors))

p + geom_boxplot()
```

(Healy, 2018)

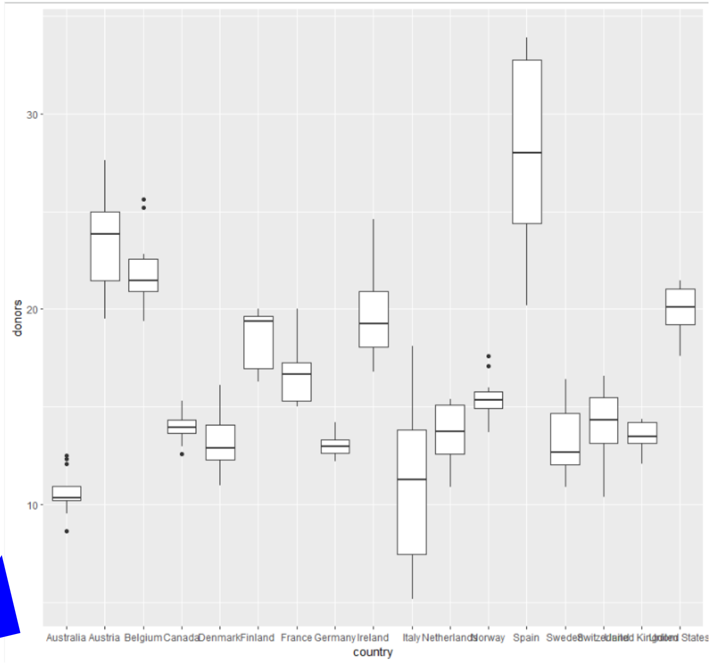
geom_boxplot()



```
p <- ggplot(data = organdata, mapping = aes(x = country, y = donors))

p + geom_boxplot()
```

(Healy, 2018)

geom_boxplot()



By default, country names on the x axis overlap.

```
p <- ggplot(data = organdata, mapping = aes(x = country, y = donors))

p + geom_boxplot()
```

(Healy, 2018)

Improving our boxplot

- We use `coord_flip()` to switch the axes and not the mappings (so our country text will not overlap)

```
p <- ggplot(data = organdata, mapping = aes(x = country, y =
donors))

p + geom_boxplot() + coord_flip()
```

Improving our boxplot

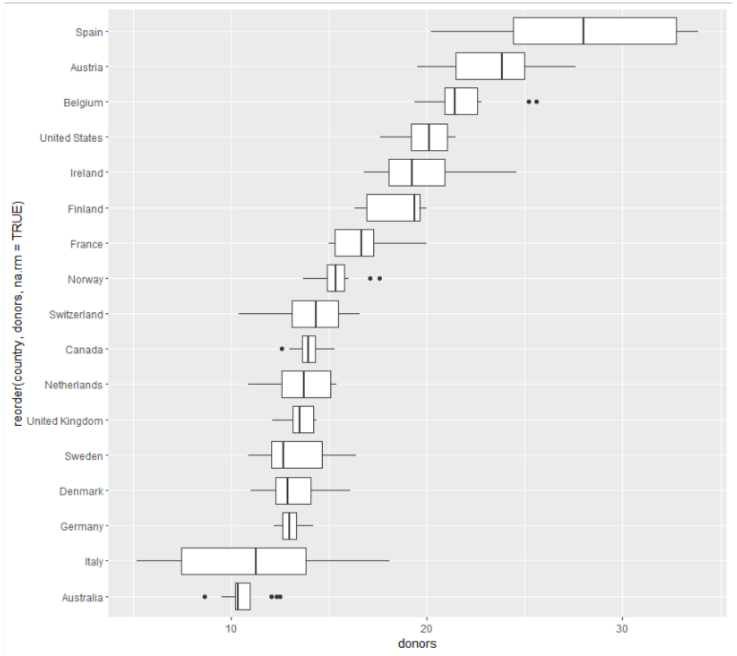
- To make the plot easier to read, we can list the countries from high to low mean donation using the `reorder()` function

```
p <- ggplot(data = organdata, mapping = aes(x = reorder(country,
donors, na.rm = TRUE), y = donors))

p + geom_boxplot() + coord_flip()
```

- The three arguments in the reorder() function are
  - The categorical variable we want to reorder (`country`)
  - The variable we want to reorder by (`donors`)
  - Removing missing values when calculating the mean (`na.rm=TRUE`)

(Healy, 2018)

Improving our boxplot - Result



```
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, donors, na.rm = TRUE), y = donors))

p + geom_boxplot() + coord_flip()
```

(Healy, 2018)

Activity - More grouped continuous data

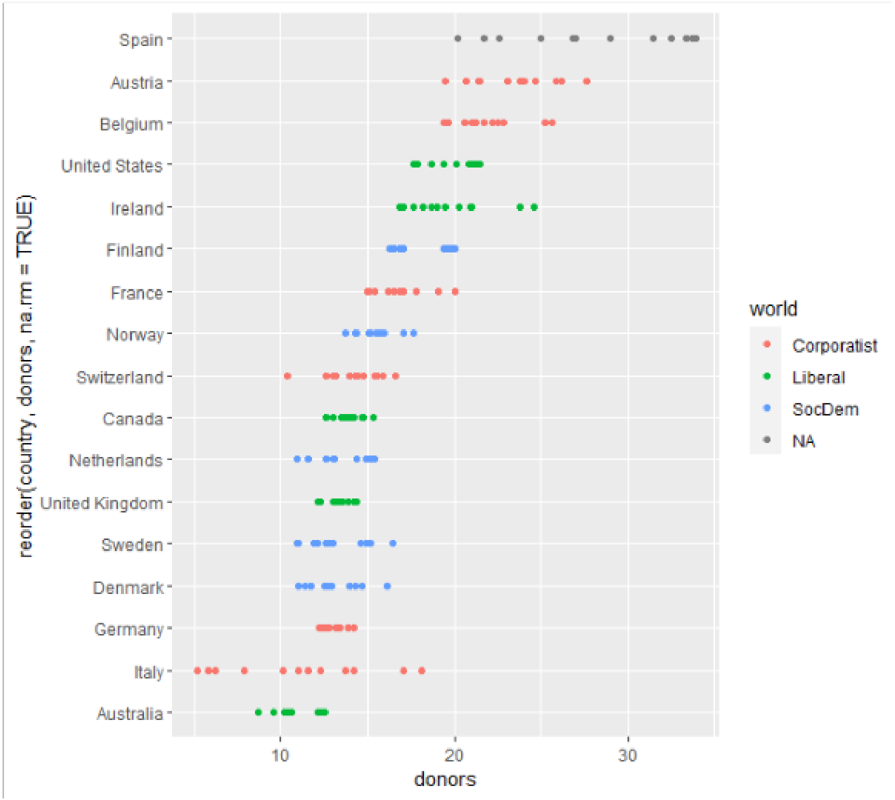
- If we have a small number of observations, it can be helpful to show individual observations (points) instead of boxplots
- We can also colour our points based on another variable (`world`)

Recalling previous classes, can you modify our code (below) to plot individual points and to map the ‘world’ variable to colour?

```
p <- ggplot(data = organdata, mapping = aes(x = x = reorder(country,
donors, na.rm = TRUE, y = donors))

p + geom_boxplot() + coord_flip()
```

(Healy, 2018)

Activity - More grouped continuous data



```
p <- ggplot(data = organdata, mapping =
aes(x = reorder(country, donors,
na.rm=TRUE),y = donors, color = world))

p + geom_point() +  coord_flip()
```

We can see that some of our observations overlap. This makes it challenging to tell how many observations are at a given value.

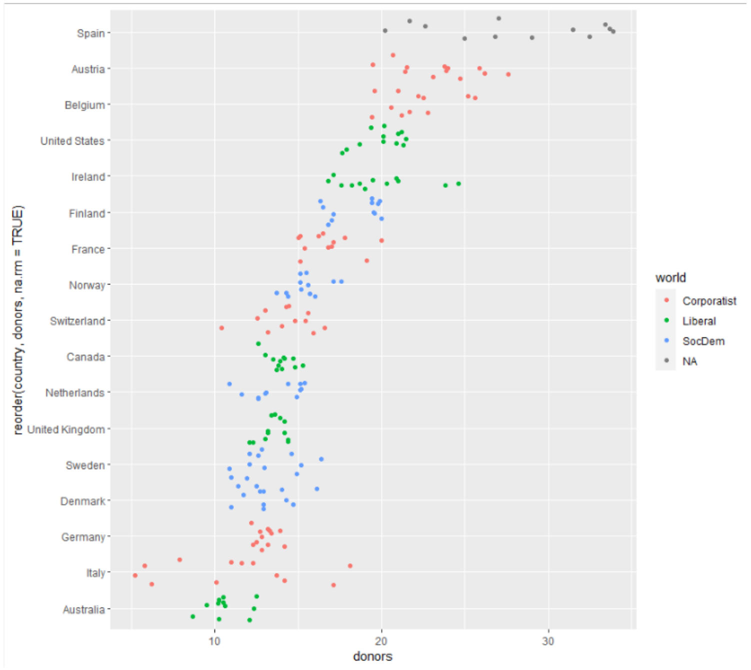
(Healy, 2018)

Addressing overlapping observations

- By slightly perturbing our data, we can better see how many observations there are at a given value
- We do this by using `geom_jitter()` to randomly and slightly nudge each observation

```
p <- ggplot(data = organdata, mapping = aes(x = x = reorder(country,
donors, na.rm = TRUE), y = donors, color=world))

p + geom_jitter() + coord_flip()
```

Addressing overlapping observations - Result



```
p <- ggplot(data = organdata, mapping = aes(x = x = reorder(country, donors, na.rm = TRUE,
y = donors, color=world))

p + geom_jitter() + coord_flip()
```

(Healy, 2018)

Categorical variables with a single point

- We can take a similar approach to summarizing categorical variables with a single point per category
- The output will be a **Cleveland dotplot**
- We will use a dplyr pipeline to aggregate our organ donor data into a smaller table of summary statistics for each country

(Healy, 2018)

Summarizing our data

```
by_country <- organdata |> group_by(consent_law, country) |>
summarize_if(is.numeric, funs(mean, sd), na.rm = TRUE) |>
ungroup()
```

- The above code
  - Creates a new dataset called `by_country`
  - Groups our data by `consent_law` and `country`
  - Uses the `summarize_if()` function to create new variables that are the mean or standard deviation of numeric variables from our original dataset
  - Ungroups the data so our result is a tibble

(Healy, 2018)

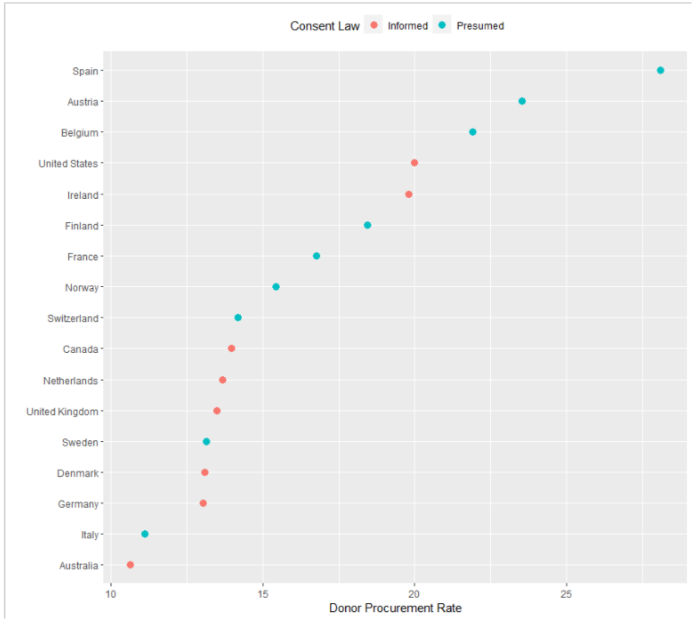
Categorical variables with a single point

- Now that we have summarized our data, we can make our Cleveland dotplot using geom_point()
- We will also
  - Colour our results by the consent law for each country
  - Move our legend to the top of our plot
  - Add axis labels

```
p <- ggplot(data = by_country, mapping = aes(x = donors_mean,
y = reorder(country, donors_mean), color = consent_law))

p + geom_point(size=3) + labs(x = "Donor Procurement Rate", y
= "", color = "Consent Law") + theme(legend.position="top")
```

(Healy, 2018)

Categorical variables with a single point - Result



```
p <- ggplot(data = by_country, mapping = aes(x = donors_mean, y = reorder(country, donors_mean),
color = consent_law))

p + geom_point(size=3) + labs(x = "Donor Procurement Rate", y = "", color = "Consent Law") +
theme(legend.position="top")
```

(Healy, 2018)

Plotting text

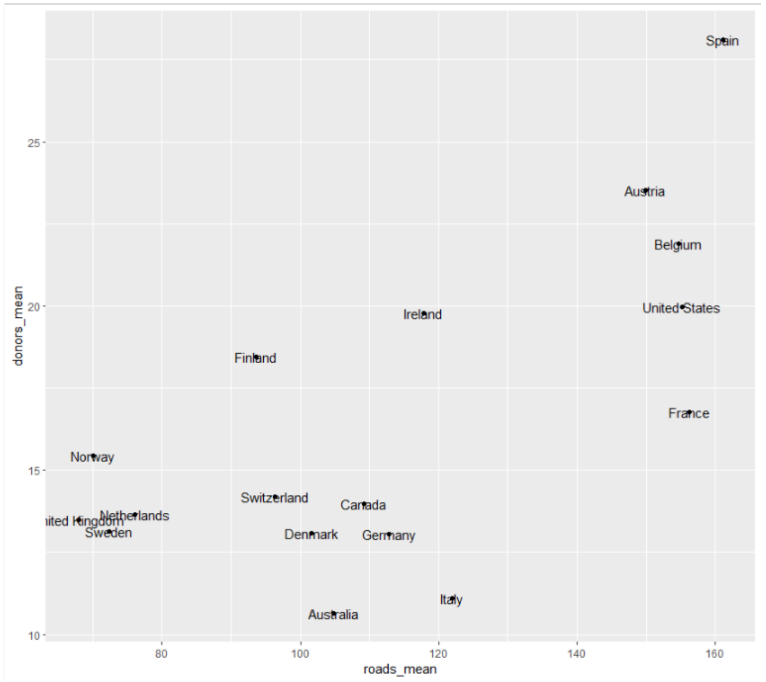
Data labels

- Adding labels to the points in our scatterplots can help to make our plots more informative
- We can add text to our plots using `geom_text()`

```
p <- ggplot(data = by_country, mapping = aes(x = roads_mean,
y = donors_mean))
p + geom_point() + geom_text(mapping = aes(label = country))
```

(Healy, 2018)

Data labels - Result

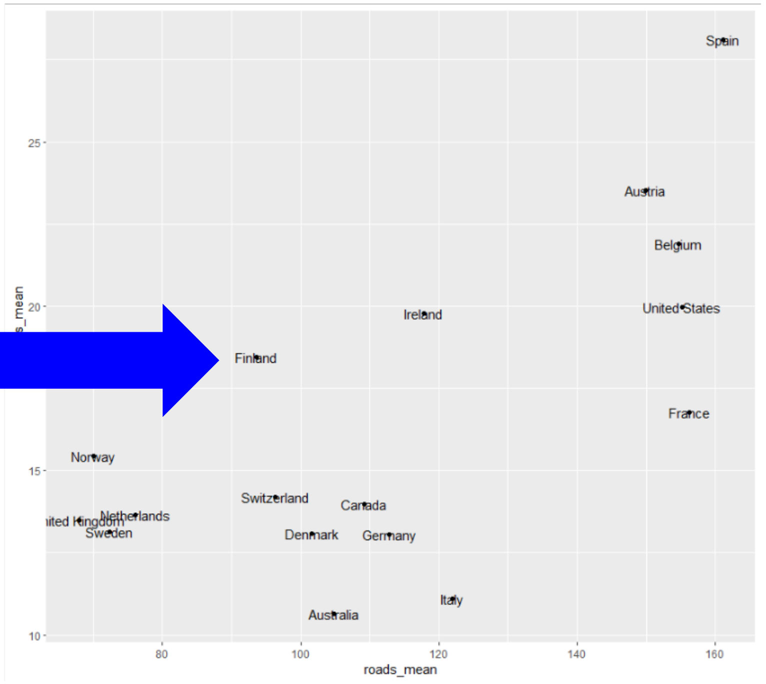


```
p <- ggplot(data = by_country, mapping = aes(x = roads_mean,y = donors_mean))

p + geom_point() + geom_text(mapping = aes(label = country))
```

(Healy, 2018)

Data labels - Result

The text in our image is plotted directly on top of the points because both use the same x and y mapping.



```
p <- ggplot(data = by_country, mapping = aes(x = roads_mean,y = donors_mean))

p + geom_point() + geom_text(mapping = aes(label = country))
```

(Healy, 2018)

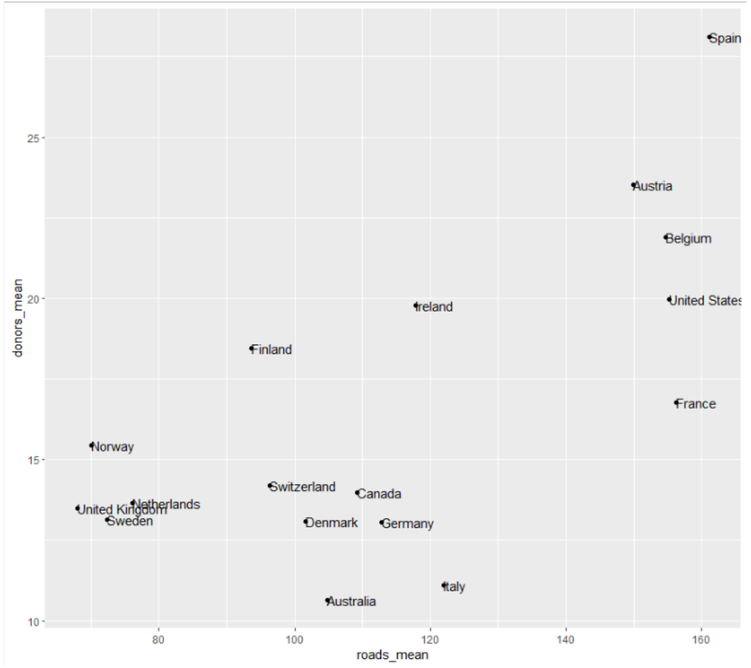
Positioning our text

- We can address this problem by removing geom_point() (so we only see the text) OR by adjusting the position of our text
- We move our data labels by adding the `hjust` argument to `geom_text()`
  - `hjust = 0` left-justifies our labels
  - `hjust = 1` right-justifies our labels

```
p <- ggplot(data = by_country, mapping = aes(x = roads_mean,

y = donors_mean))

p + geom_point() + geom_text(mapping = aes(label = country),
hjust=0)
```

(Healy, 2018)

Positioning our text - Result



```
p <- ggplot(data = by_country, mapping = aes(x = roads_mean, y =
donors_mean))

p + geom_point() + geom_text(mapping = aes(label = country), hjust=0)
```

(Healy, 2018)

ggrepel

- For more flexible text formatting, install and load the `ggrepel` library

```
install.packages(“ggrepel”)

library(ggrepel)
```

- To explore what we can do with ggrepel, we will use a new dataset about historical United States presidents (`elections_historic`) from the `socviz` library we previously loaded

(Healy, 2018)

ggrepel

```
p <- ggplot(elections_historic, aes(x = popular_pct, y = ec_pct,

label = winner_label))

p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray80") +

geom_vline(xintercept = 0.5, size = 1.4, color = "gray80") +

geom_point() +

geom_text_repel() +

scale_x_continuous(labels = scales::percent) +

scale_y_continuous(labels = scales::percent) +

labs(x = “Winner’s share of Popular Vote”, y = “Winner’s share of Electoral
College Votes”, title = “Presidential Elections: Popular & Electoral
College Margins”, subtitle = “1824-2016”)
```

(Healy, 2018)

ggrepel

```
p <- ggplot(elections_historic, aes(x = popular_pct, y = ec_pct,

label = winner_label))

p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray80") +

geom_vline(xintercept = 0.5, size = 1.4, color = "gray80") +

geom_point() +

geom_text_repel() +

scale_x_continuous(labels = scales::percent)

scale_y_continuous(labels = scales::percent)

labs(x = "Winner's share of Popular Vote", y
College Votes", title = "Presidential Elections: Popular & Electoral
College Margins", subtitle = "1824-2016")
```

Vote shares are stored as proportions rather than percents, so we adjust the labels of the scales.

ggrepel

```
p <- ggplot(elections_historic, aes(x = popular_pct, y = ec_pct,

label = winner_label))

p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray80") +

geom_vline(xintercept = 0.5, size = 1.4, color = "gray80") +

geom_point() +

geom_text_repel() +

scale_x_continuous(labels = scales::percent)

scale_y_continuous(labels = scales::percent)

labs(x = “Winner’s share of Popular Vote”, y
College Votes”, title = “Presidential Elections: Popular & Electoral
College Margins”, subtitle = “1824-2016”)
```

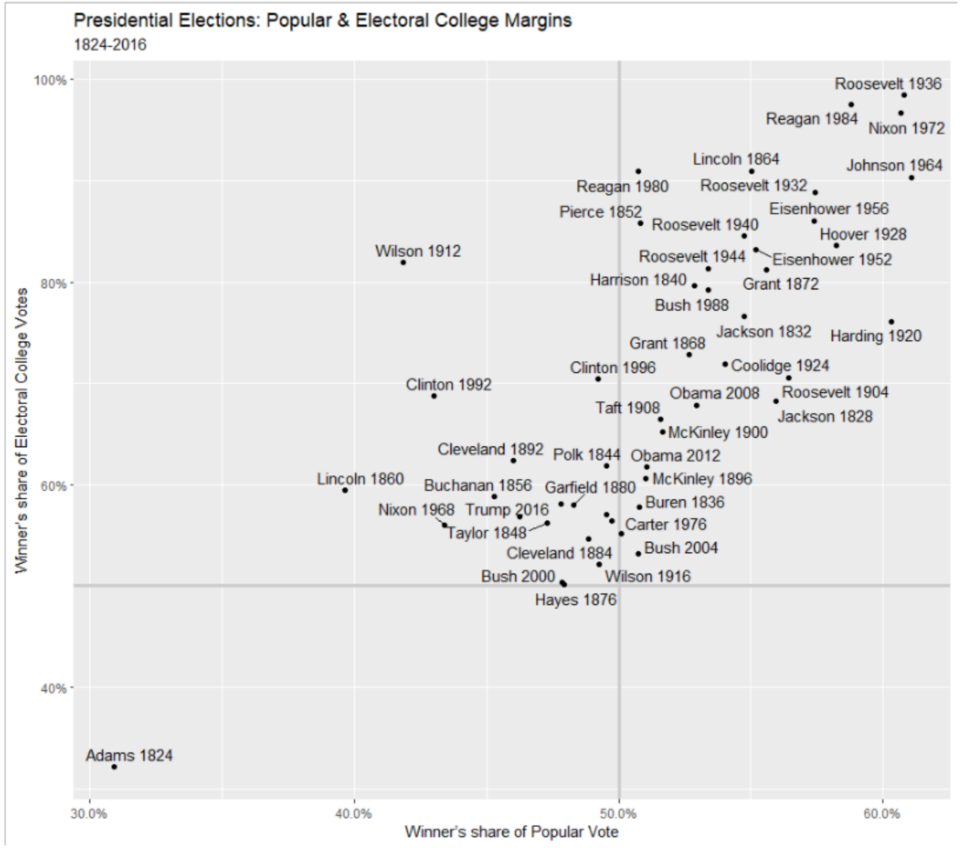
We add reference lines so we can see the 50% threshold of votes on each axis.

ggrepel

```
p <- ggplot(elections_historic, aes(x = popular_pct, y = ec_pct,

label = winner_label))

p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray80") +

geom_vline(xintercept = 0.5, size = 1.4, color = "gray80") +

geom_point() +

geom_text_repel() +

scale_x_continuous(labels = scales::percent)

scale_y_continuous(labels = scales::percent)

labs(x = "Winner's share of Popular Vote", y = "Winner's share of Electoral
College Votes", title = "Presidential Elections: Popular & Electoral
College Margins", subtitle = "1824-2016")
```

geom_text_repel() will ensure that our data labels do not overlap.

ggrepel - Result



(Healy, 2018)

Labeling outliers

Labeling specific points

- Sometimes, we only want to label specific points of interest (such as outliers) rather than every single item
- We do this by using the `subset()` function to tell `geom_text_repel()` to use a different dataset from the one being used by `geom_point()`
- We will return to our `by_country` dataset to explore labeling outliers

(Healy, 2018)

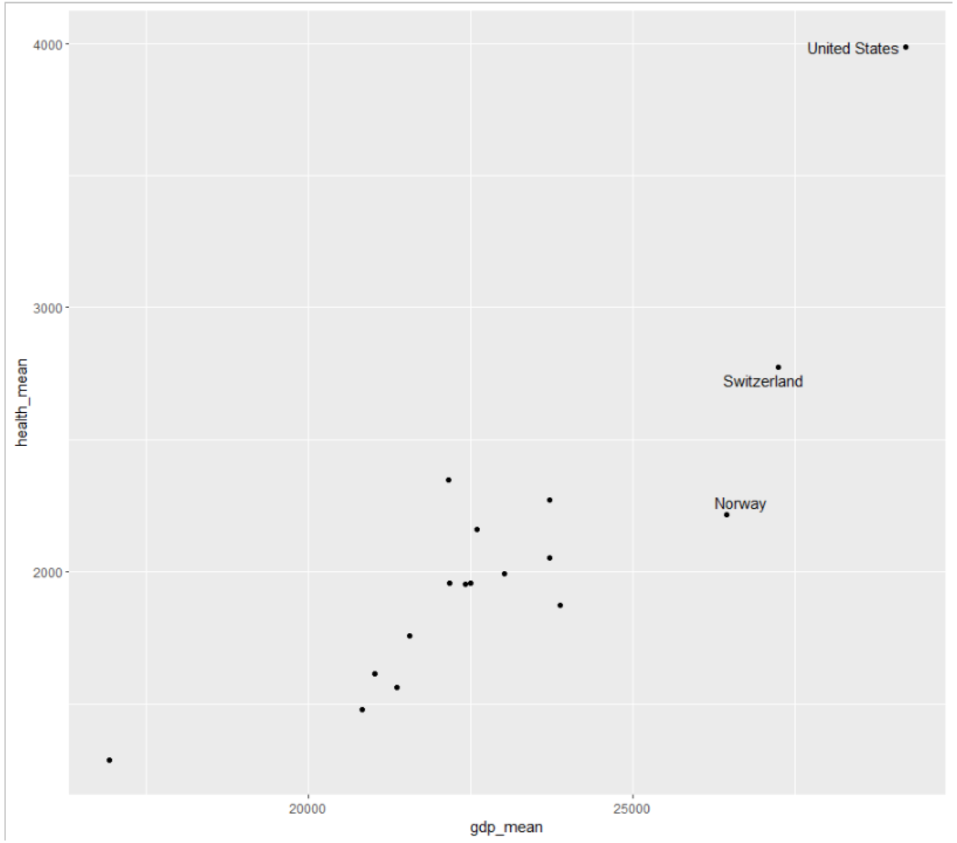
Labeling specific points

- In the below code, we use `subset()` to select only the cases from our dataset (`by_country`) where `gdp_mean` is over 25000
- Only points within this subset are labeled

```
p <- ggplot(data = by_country, mapping = aes(x = gdp_mean, y =
health_mean))

p + geom_point() +

geom_text_repel(data = subset(by_country, gdp_mean > 25000),

mapping = aes(label = country))
```

(Healy, 2018)

Labeling specific points - Result



(Healy, 2018)

Writing and drawing in our plots

Annotating plots

- It can sometimes be useful to annotate a figure or place arbitrary text on a plot
    - That is, text that is not mapped to a variable
- We do this using the `annotate()` function, which is not a geom, but uses the features of geoms (such as the ability to modify size, colour, and x or y position)

(Healy, 2018)

Annotating plots with text

```
p <- ggplot(data = organdata, mapping = aes(x = roads, y = donors))

p + geom_point() + annotate(geom = "text", x = 91, y = 33, label =
"A surprisingly high \n recovery rate.", hjust = 0)
```

- The above code
  - Uses annotate() to add text
  - Positions the text at (91, 33) on our plot
  - Using the **newline code** (‘\n’) to force a line break
  - Left-justifies our text

(Healy, 2018)

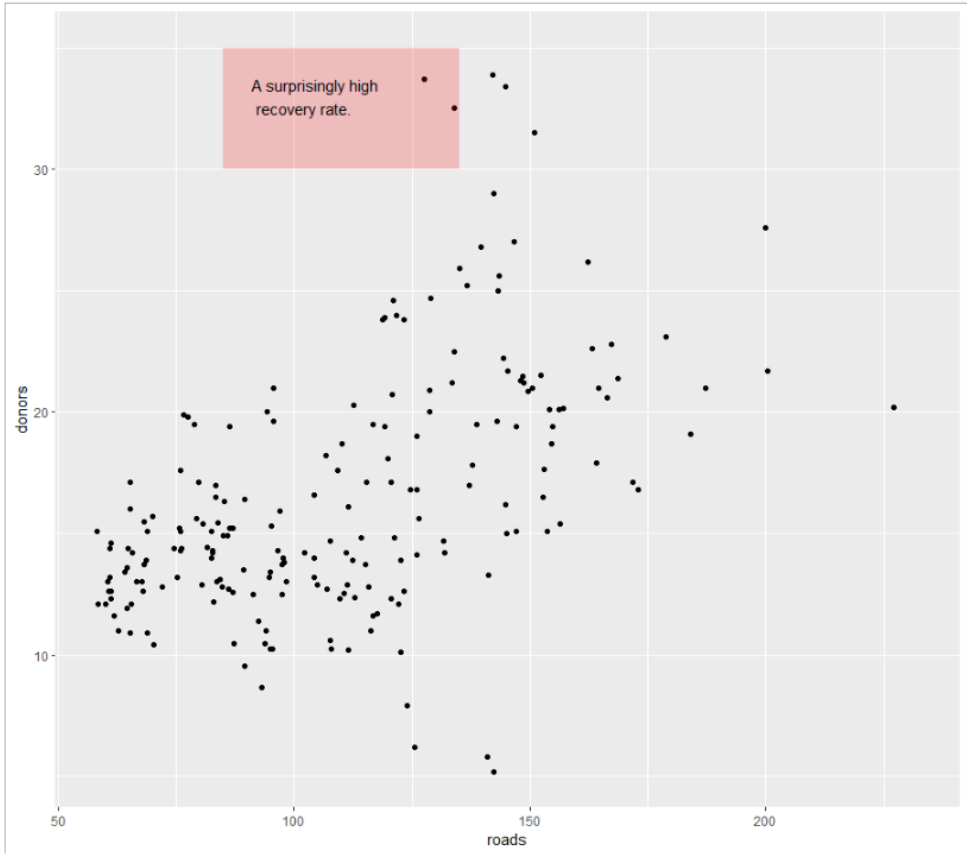
Annotating plots with shapes

- We can also use annotate to add shapes to our plot
- For example, we can add a red rectangle to our previous code by calling annotate() again with “rect” instead of “text”

```
p <- ggplot(data = organdata, mapping = aes(x = roads, y = donors))

p + geom_point() +

annotate(geom = "rect", xmin = 85, xmax = 135, ymin = 30, ymax = 35,
fill = "red", alpha = 0.2) +

annotate(geom = "text", x = 91, y = 33, label = "A surprisingly high
\n recovery rate.", hjust = 0)
```

(Healy, 2018)

Annotating plots - Result



(Healy, 2018)

Next...

- Visualization with Purpose
- Refining our ggplots in R, with consideration for aesthetic qualities of data visualization and colour theory
- Strategies for accessible data visualization (Ethics and inequity)
- Exploring data visualization as a tool for advocacy (Inequity)