# Class3_20231018_DataVisualization_Apichat

Apichat Photi-A

18/03/2023

```r
#attach the libraries
library(socviz)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.0     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.1.8
## v purrr     1.0.1     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

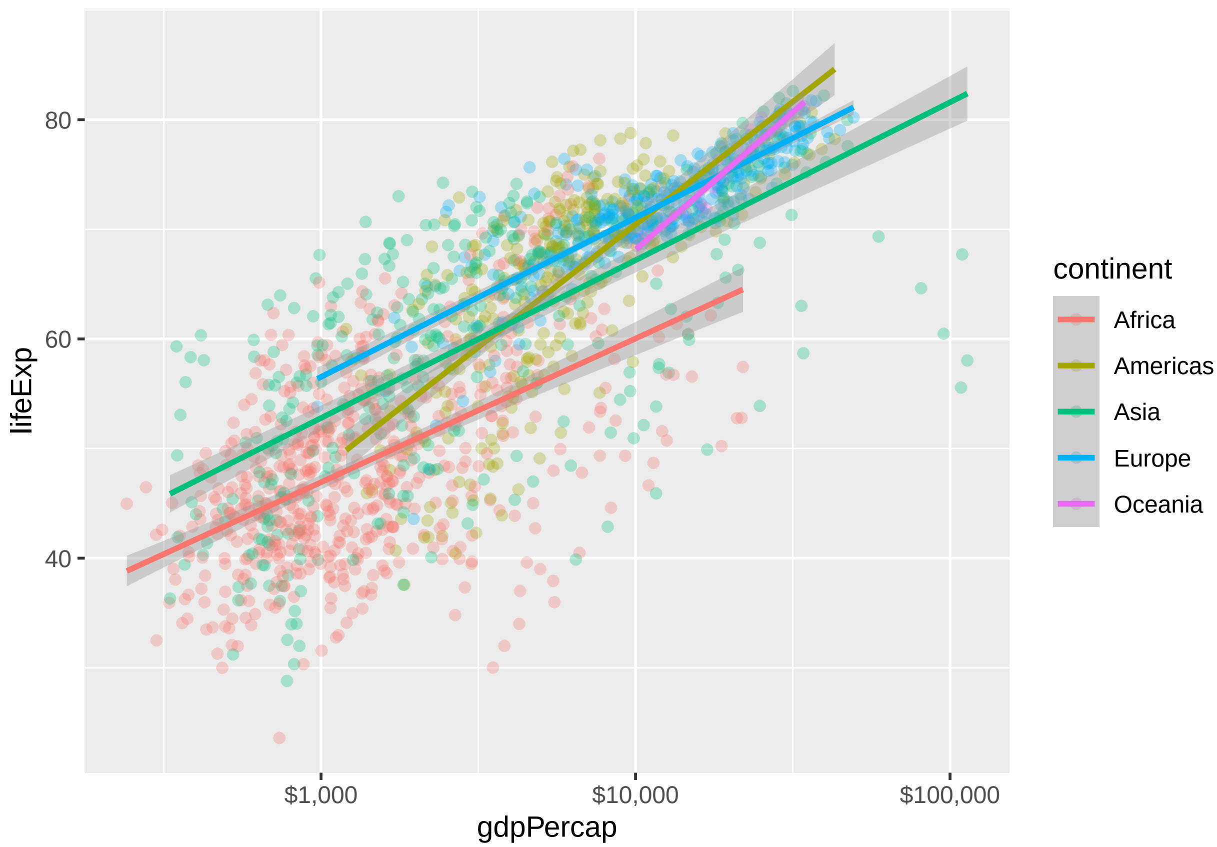
```r
#install.packages("gapminder")
library(gapminder)

#attach the data
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

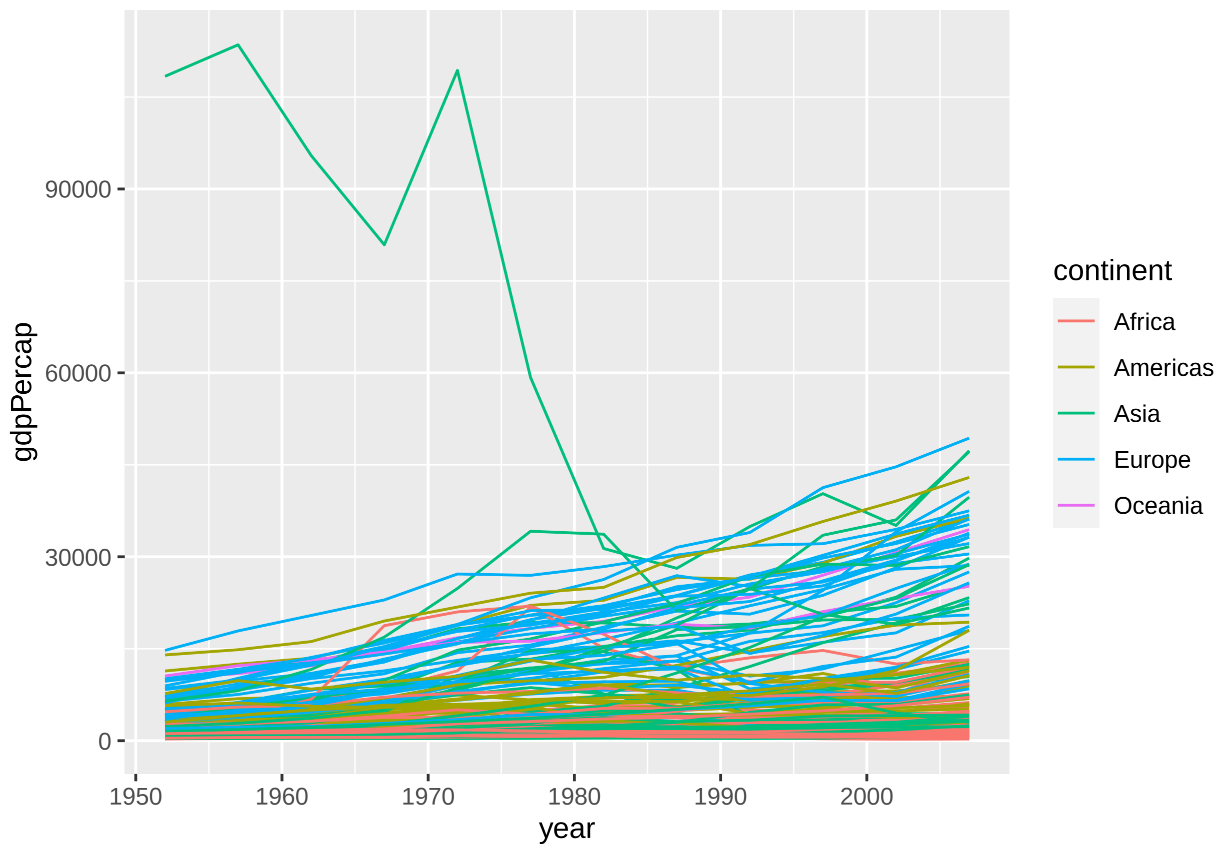
```r
#plot the graph of life expectancy over GDPperCap (from last class)
p <- ggplot(data = gapminder,
       mapping = aes(x=gdpPercap, y = lifeExp,
                     color = continent))
```

```r
p + geom_point(alpha = 0.3) + geom_smooth(method = 'lm') + scale_x_log10(labels = scales::dollar)
```

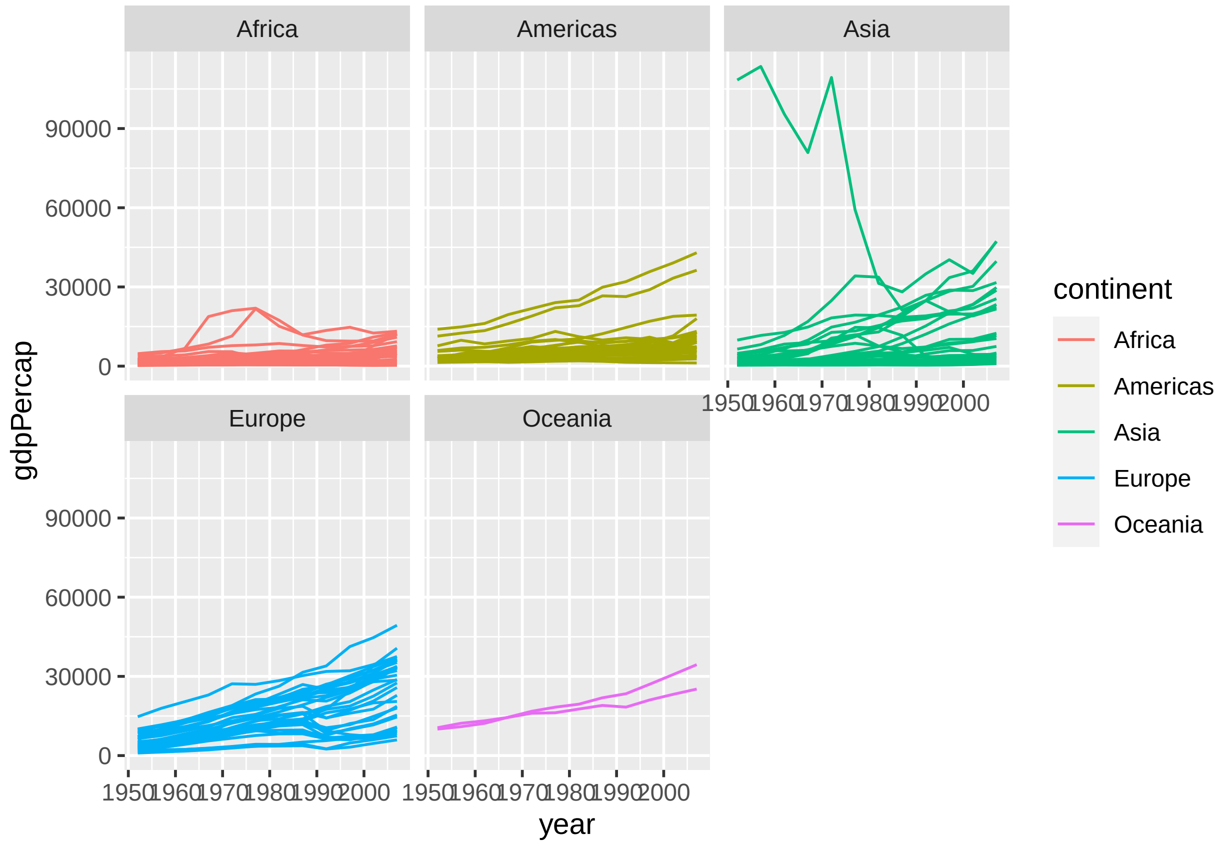
```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
#plot the graph of GDP over time per country
p <- ggplot(data = gapminder,
       mapping = aes(x= year, y = gdpPercap,
                     color = continent))
p + geom_line(aes(group = country))
```




```r
#add the facet_wrap to break up the data into pieces too make a small multiple plot
p <- ggplot(data = gapminder,
       mapping = aes(x= year, y = gdpPercap,
                     color = continent))
p + geom_line(aes(group = country)) + facet_wrap(~continent)
```

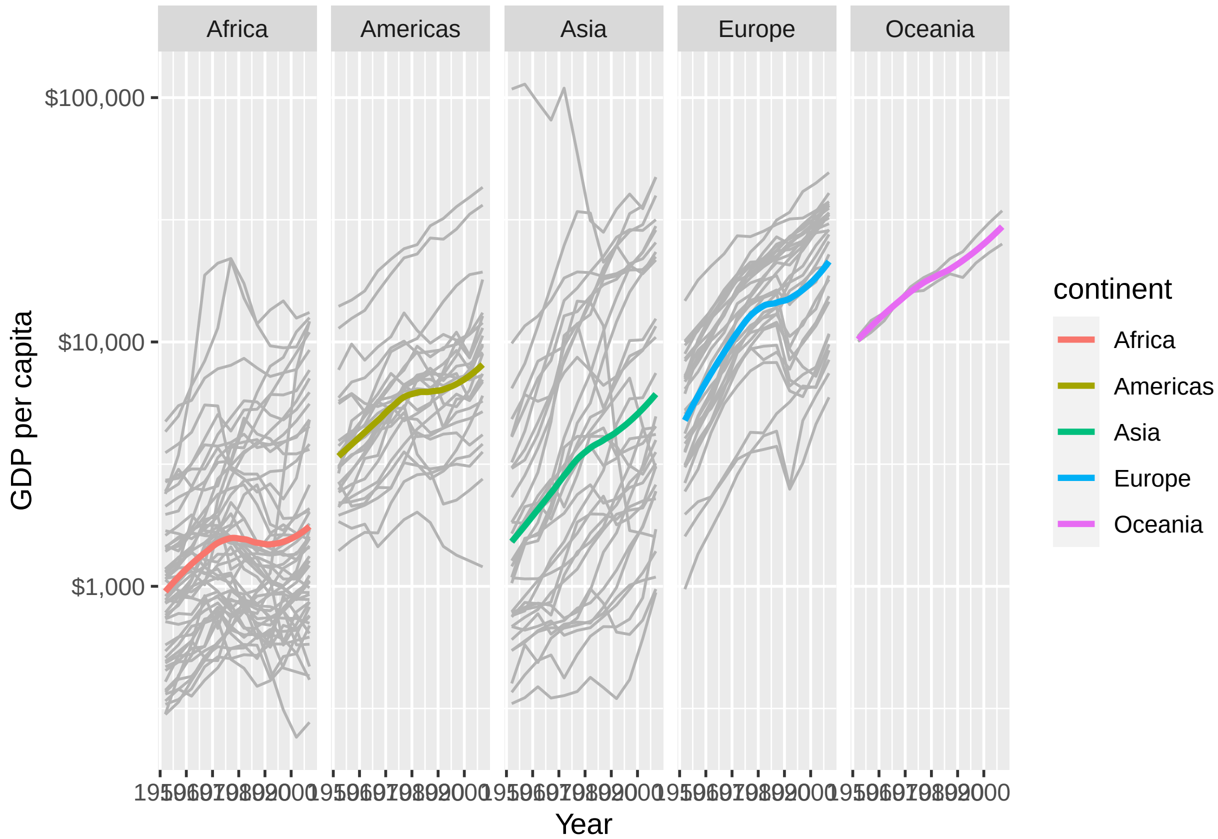



**Improving our plot** aesthetic, substantive, and perceptual characteristics of our plot: 1. Make country trends light grey colour 2. Add a trend line 3. Make y axis logarithmic and show that values are in dollars 4. Try to fit all five facets on a single row (5 columns) 5. Add axis labels and graph title

```
#add the facet_wrap to break up the data into pieces too make a small multiple plot
p <- ggplot(data = gapminder,
       mapping = aes(x= year, y = gdpPercap,
                     color = continent))
p + geom_line(color = "gray70", aes(group = country)) +
  geom_smooth(size=1.1, method="loess", se = FALSE) +
  scale_y_log10(labels = scales::dollar) +
  facet_wrap(~continent, ncol = 5) +
  labs(x = "Year",
       y = "GDP per capita",
       Title = "GDP per capita on Five Continents")
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

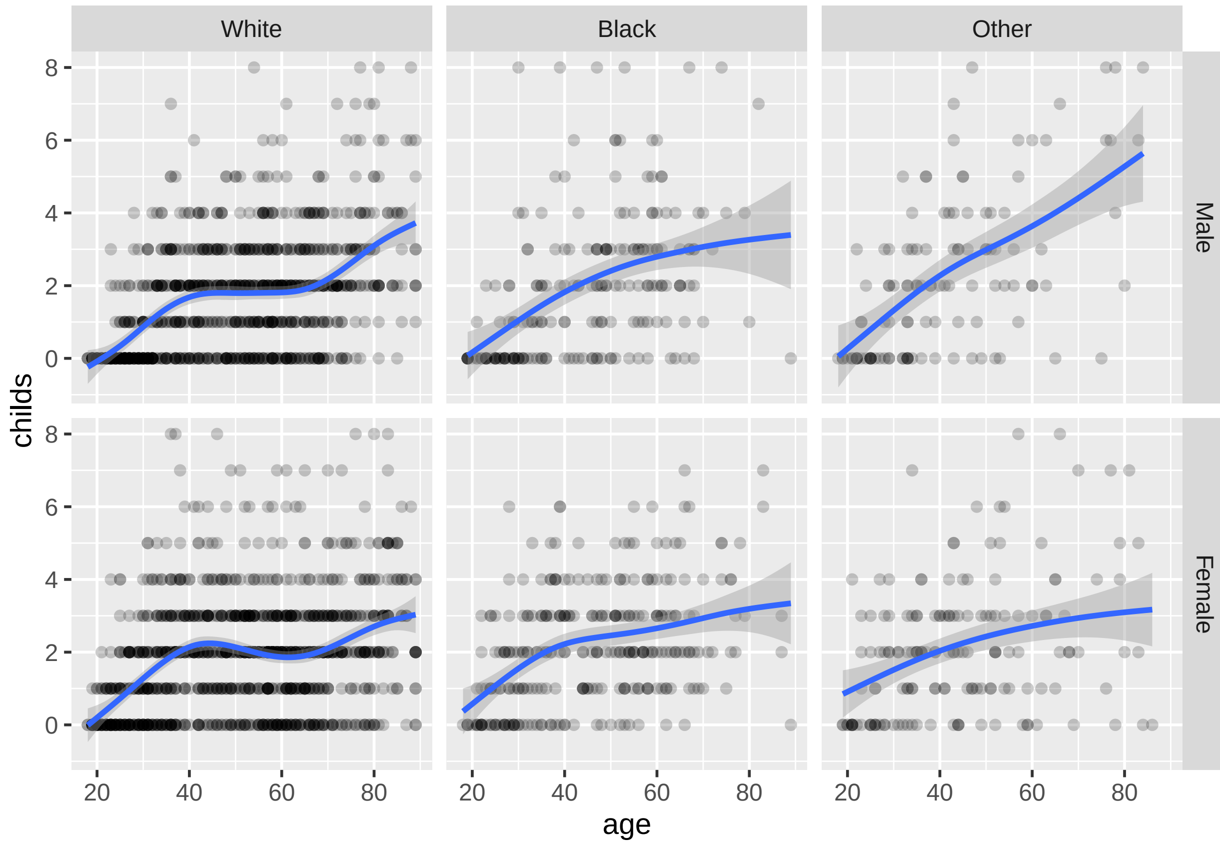



```r
#attach new dataset
attach(gss_sm)
#A dataset containing an extract from the 2016 General Social Survey.
#See http://gss.norc.org/Get-Documentation for full documentation of the variables.

#Using facet_grid()
p <- ggplot(data=gss_sm, mapping = aes(x=age, y=childs))
p + geom_point(alpha=0.2) + geom_smooth() + facet_grid(sex~race)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

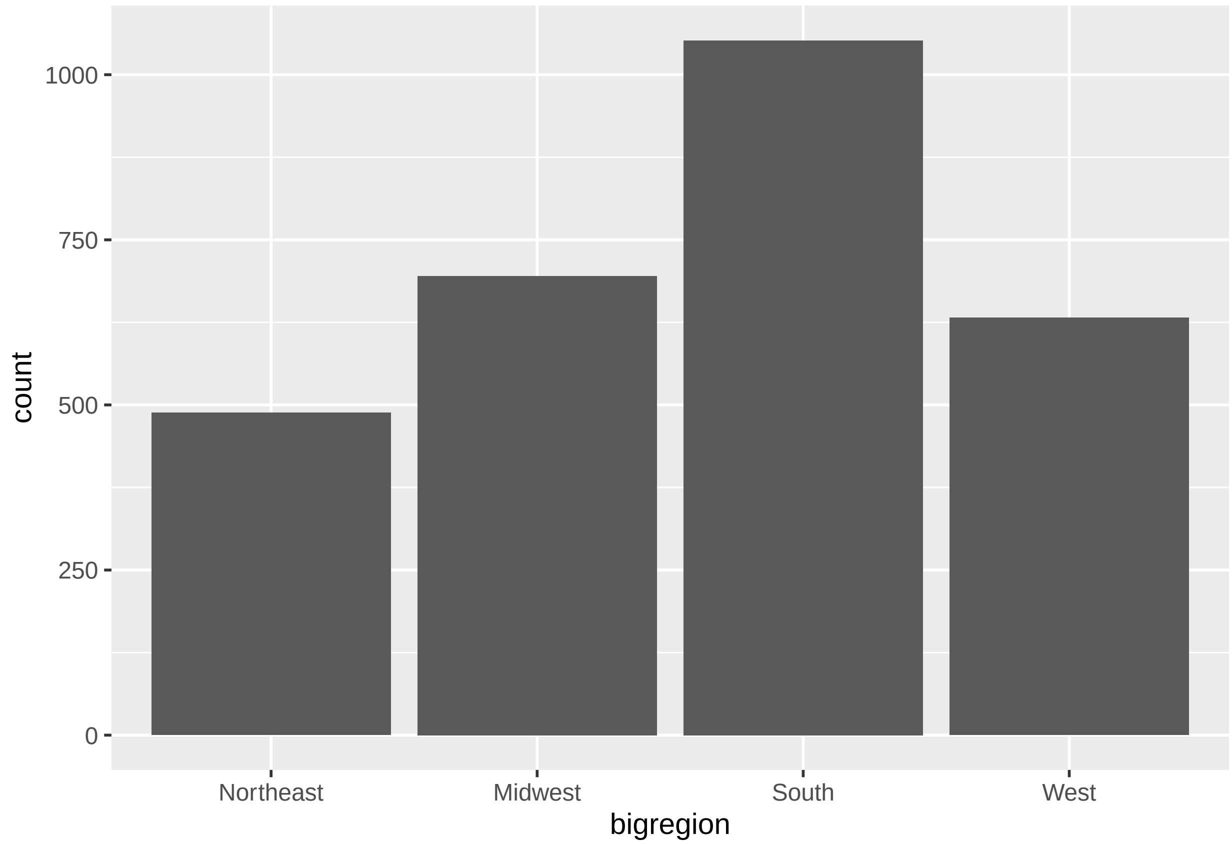
```
## Warning: Removed 18 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```



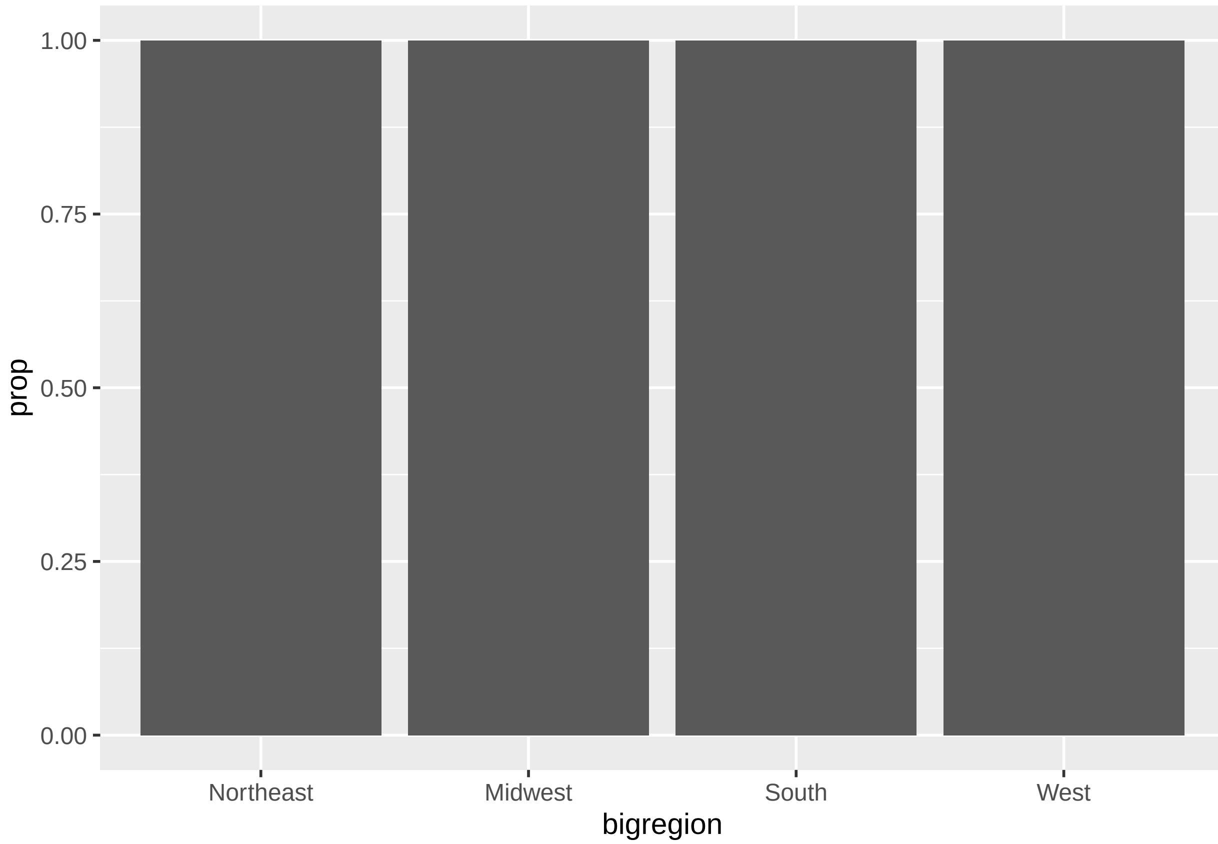

let's try stat_functions

```
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p+ geom_bar()
```

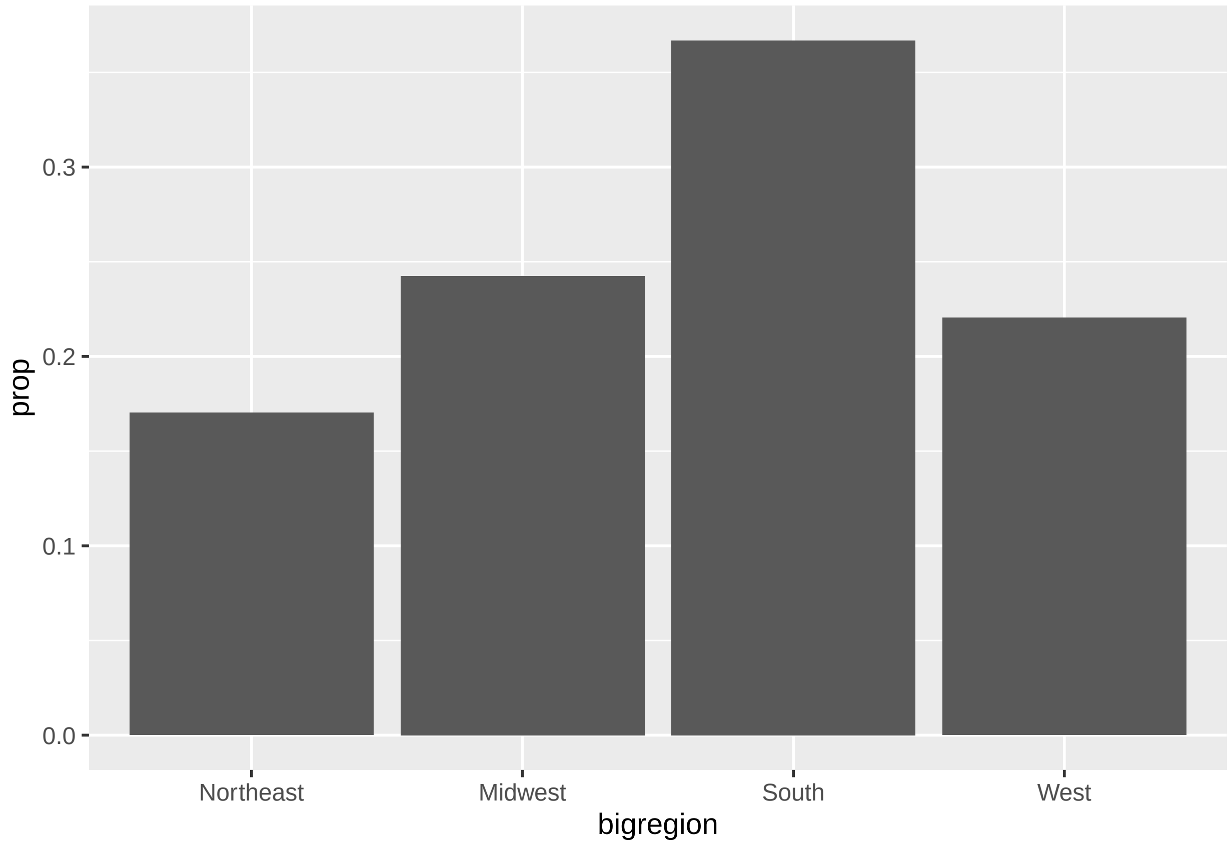



```
#show the relative frequencies rather than counts, we can use the prop (proportion)
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p+ geom_bar(mapping = aes(y=..prop..))
```

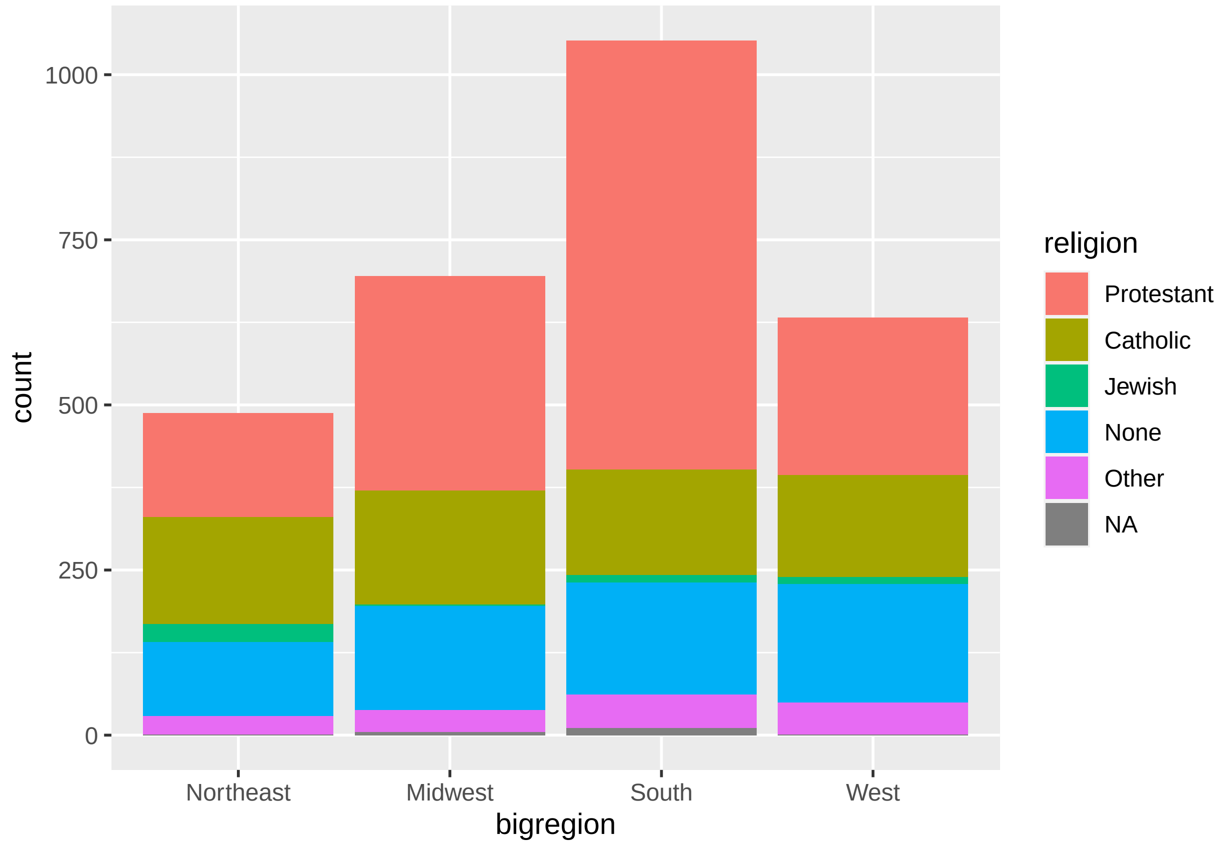
```
## Warning: The dot-dot notation (`..prop..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(prop)` instead.
```



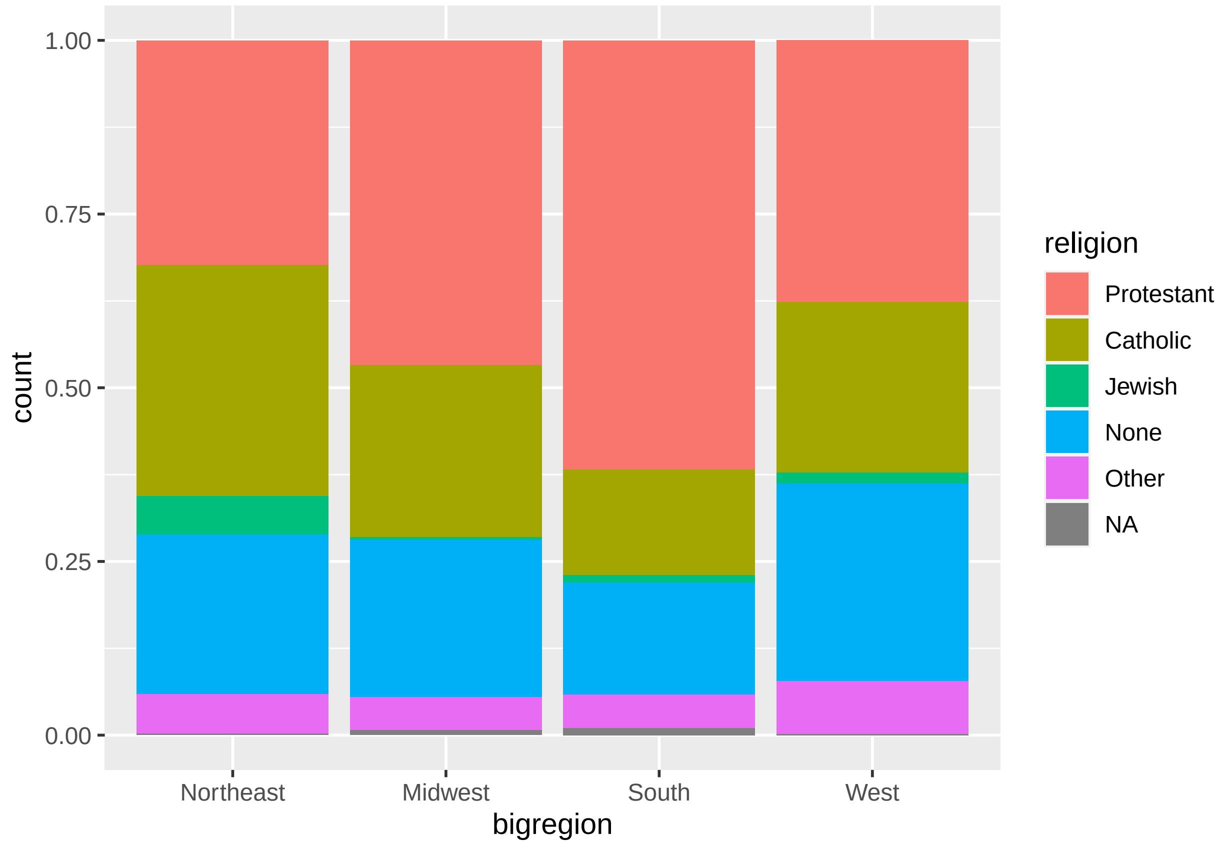

```
#We create a 'dummy group' with group = 1 to tell ggplot to use the whole dataset when establishing the
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p+ geom_bar(mapping = aes(y=..prop.., group  = 1))
```



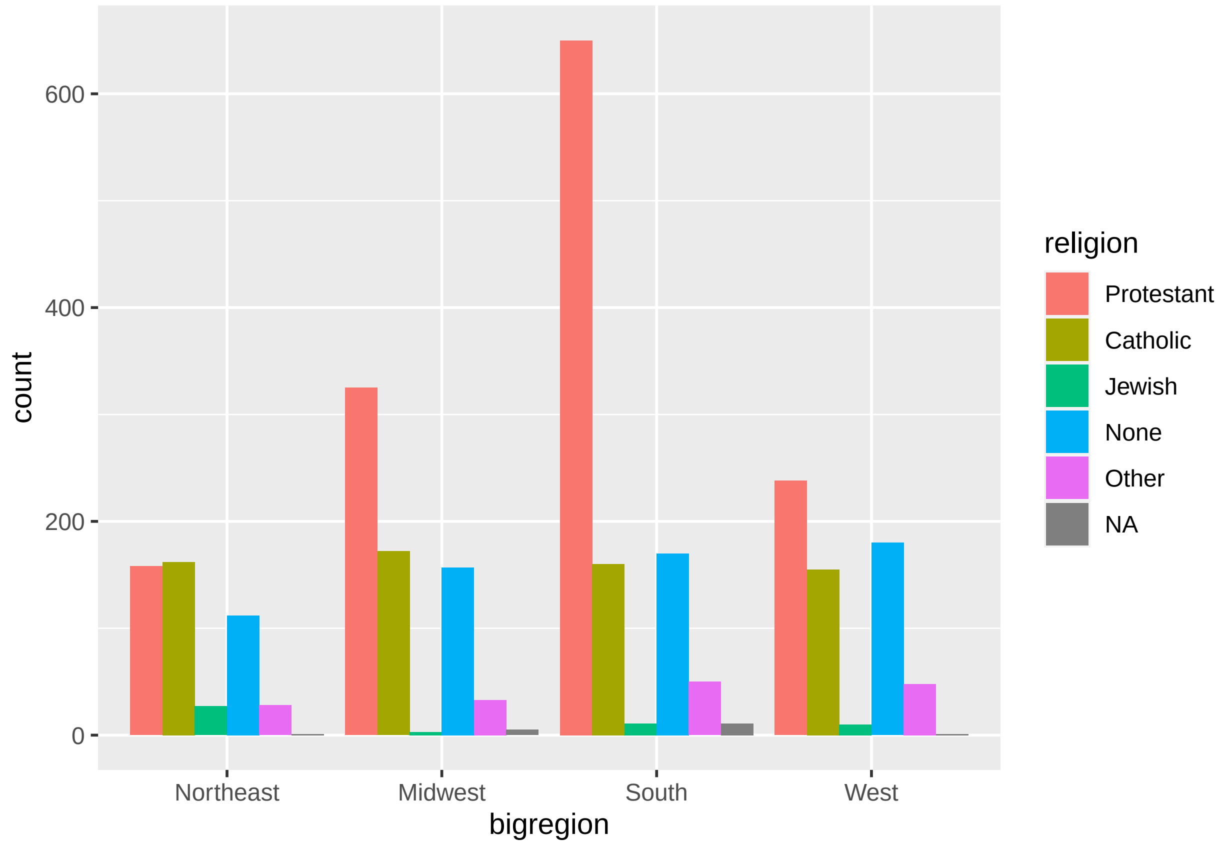

```
#The variable religion is a categorical variable about participants' religious affiliation
# We can plot religious affiliation by census region as a bar chart, and colour the bars differently fo
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion, fill = religion))
p+ geom_bar()
```




```
#Now we can compare proportions across groups
# BUT we cannot see the relative size of each cut with respect to the overall total
# Our frequency chart can benefit from faceting!
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion, fill = religion))
p+ geom_bar(position = "fill")
```

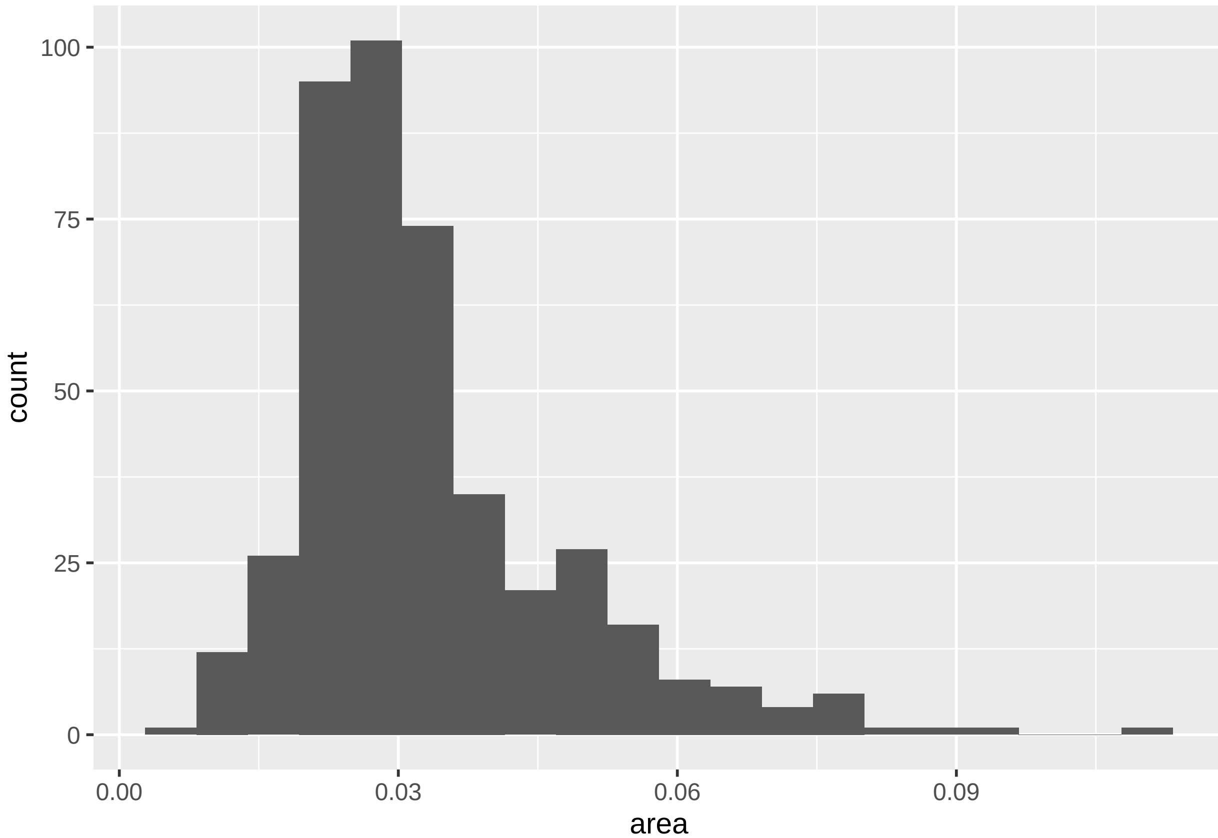



```r
#Now we can compare proportions across groups
# BUT we cannot see the relative size of each cut with respect to the overall total
# Our frequency chart can benefit from faceting!
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion, fill = religion))
p+ geom_bar(position = "dodge", mapping = aes(y = ..prop.., group = bigregion) +
              facet_wrap(~bigregion, ncol = 1))
```

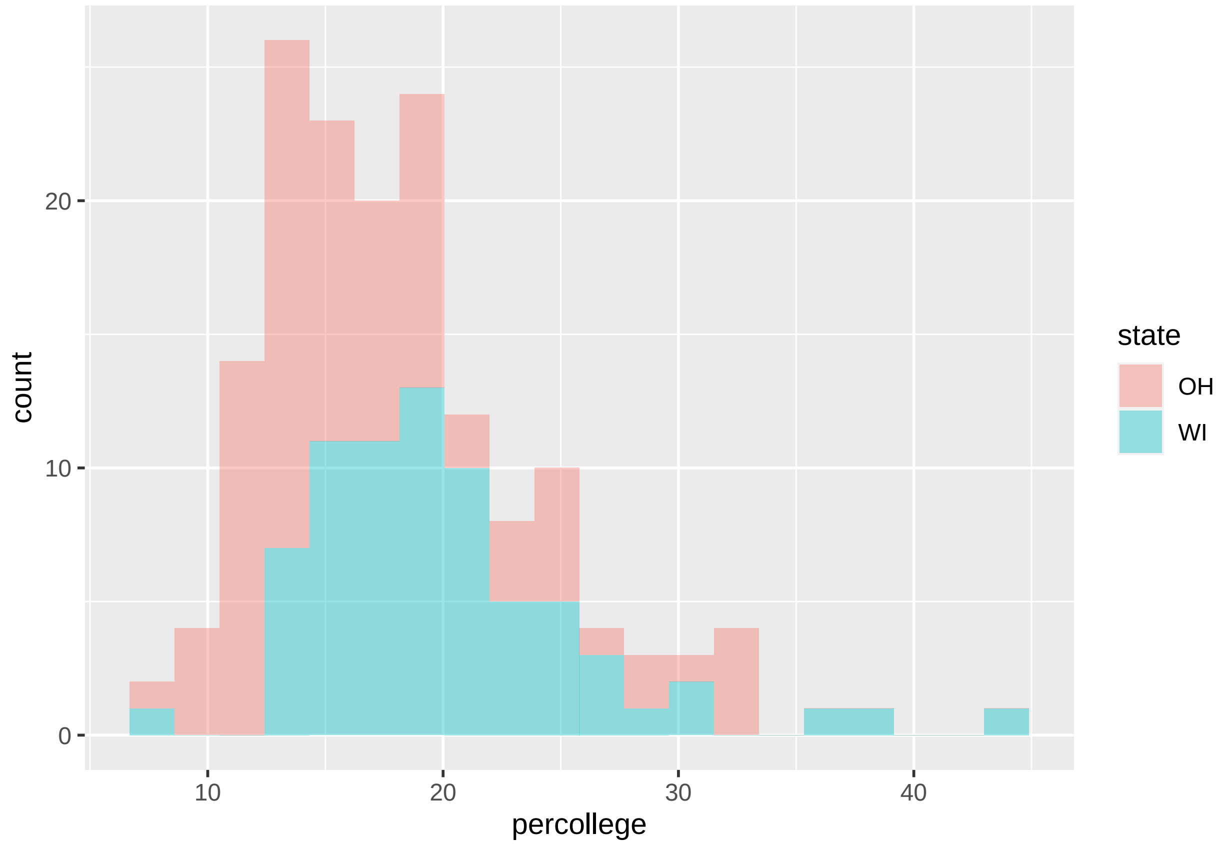



Histograms - Choosing bins

```
attach(midwest)
p <- ggplot(data=midwest, mapping = aes(x = area))
p + geom_histogram(bins = 20)
```



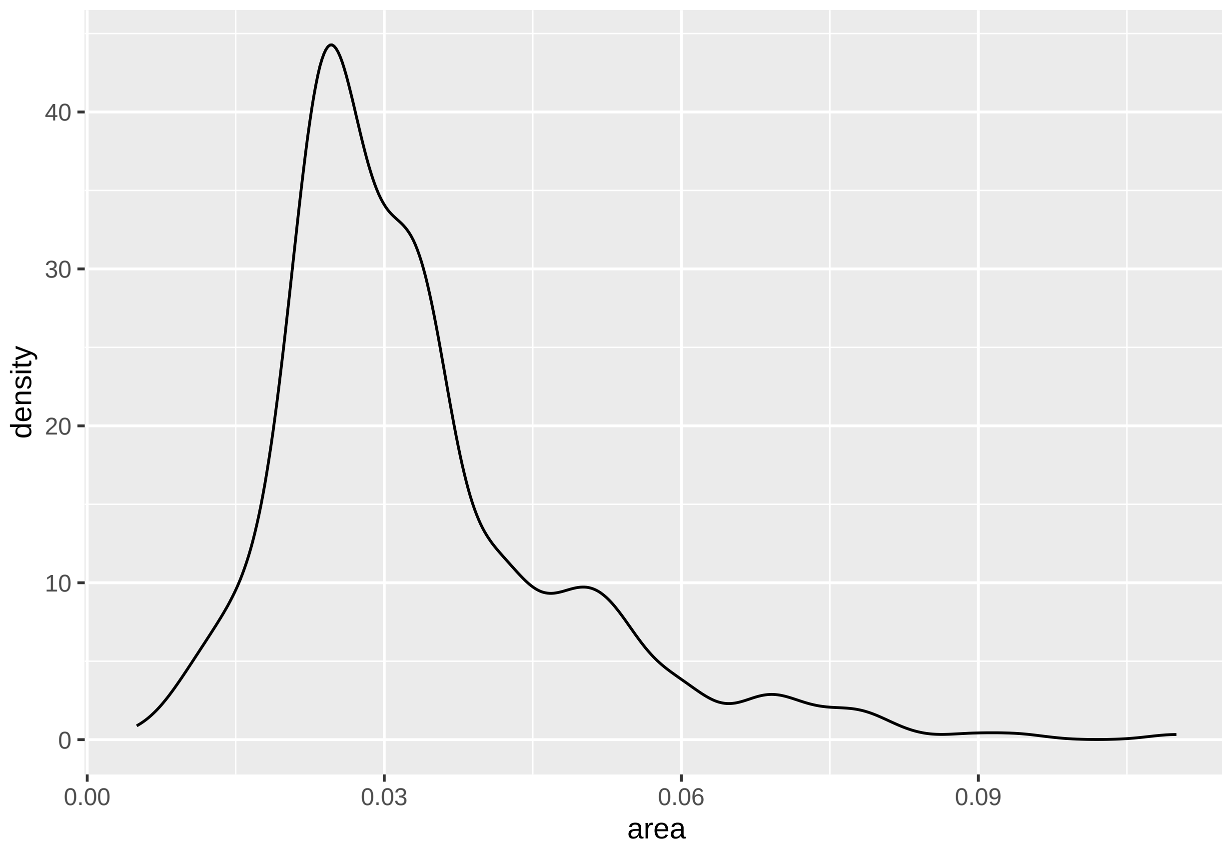

```
# comparing histograms
oh_wi <- c("OH", "WI")

p <- ggplot(data = subset(midwest, subset = state %in% oh_wi),
            mapping = aes(x = percollege, fill = state))
p + geom_histogram(alpha = 0.4, bins = 20)
```



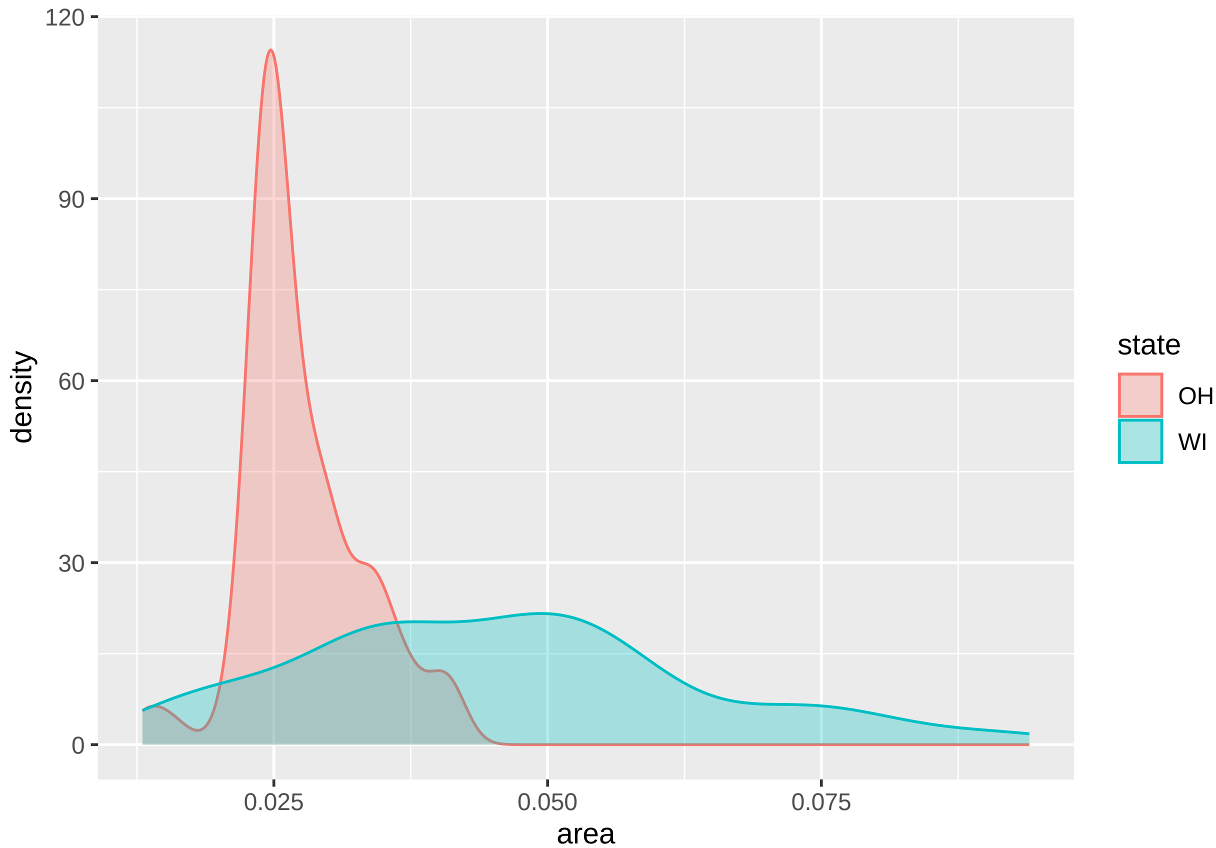

```
midwest
```

```
## # A tibble: 437 x 28
##      PID county    state  area poptotal popden~1 popwh~2 popbl~3 popam~4 popas~5
##    <int> <chr>     <chr> <dbl>    <int>    <dbl>   <int>   <int>   <int>   <int>
##  1   561 ADAMS     IL    0.052    66090    1271.   63917    1702      98     249
##  2   562 ALEXANDER IL    0.014    10626     759     7054    3496      19      48
##  3   563 BOND      IL    0.022    14991     681.   14477     429      35      16
##  4   564 BOONE     IL    0.017    30806    1812.   29344     127      46     150
##  5   565 BROWN     IL    0.018     5836     324.    5264     547      14       5
##  6   566 BUREAU    IL    0.05     35688     714.   35157      50      65     195
##  7   567 CALHOUN   IL    0.017     5322     313.    5298       1       8      15
##  8   568 CARROLL   IL    0.027    16805     622.   16519     111      30      61
##  9   569 CASS      IL    0.024    13437     560.   13384      16       8      23
## 10   570 CHAMPAIGN IL    0.058   173025    2983.  146506   16559     331    8033
## # ... with 427 more rows, 18 more variables: popother <int>, percwhite <dbl>,
## #   percblack <dbl>, percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>, and abbreviated
## #   variable names 1: popdensity, 2: popwhite, 3: popblack, ...
```

```
#Density plot
p <- ggplot(data =  midwest, mapping = aes(x = area))
p+ geom_density()
```




```r
#create a subset
oh_wi <- c("OH", "WI")
p <- ggplot(data =  subset(midwest, subset = state %in% oh_wi),
mapping = aes(x = area, fill = state, color = state))
p+ geom_density(alpha = 0.3)
```




***Part2***

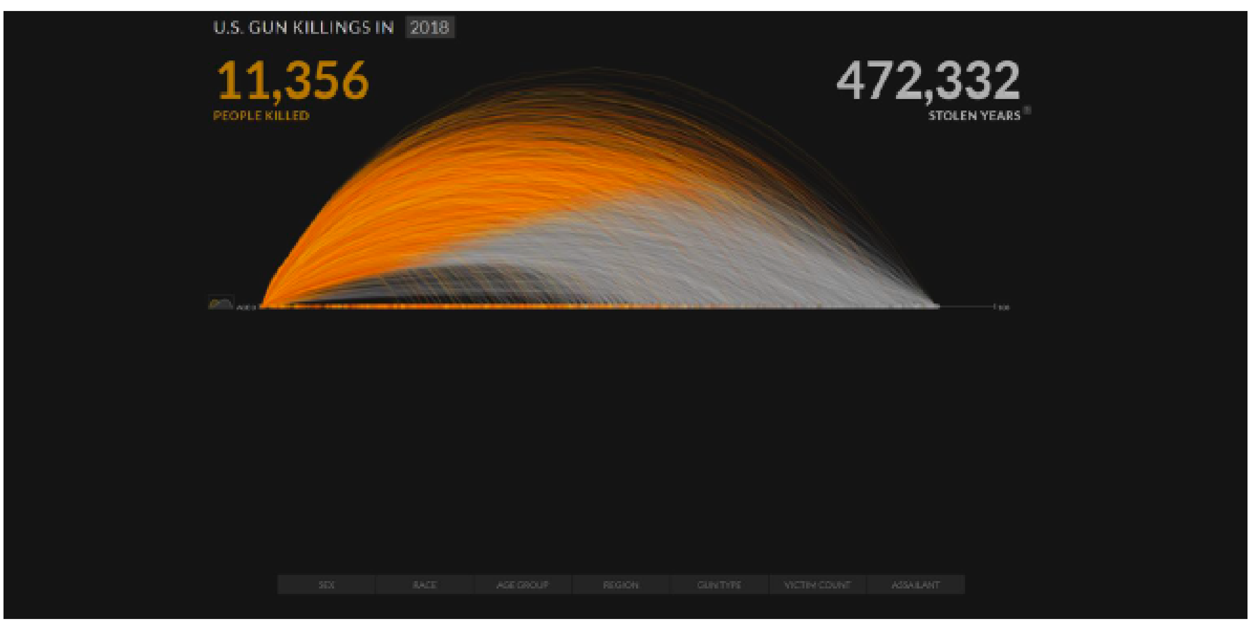



Figure 1: Resource picture

- What information can we learn from this visualization?

**Answer** Relationship between the stolen guns and the people killed # people killed, # of projected years cut short; not objective as projected years are speculative (assuming based on statistical likelihood)

- Is this an example of objective, neutral data visualization? Why or why not?

**Answer** It shows only the numbers.

What we want our data viz to do?

**Intended purpose** - Persuading - Comparison - Evaluation - Exploring

**Intended audience** - Age - Education - Expertise - Accessibility

**Intended medium** - Print - Web - Poster - Presentation

How is our data visualization perceived?

---

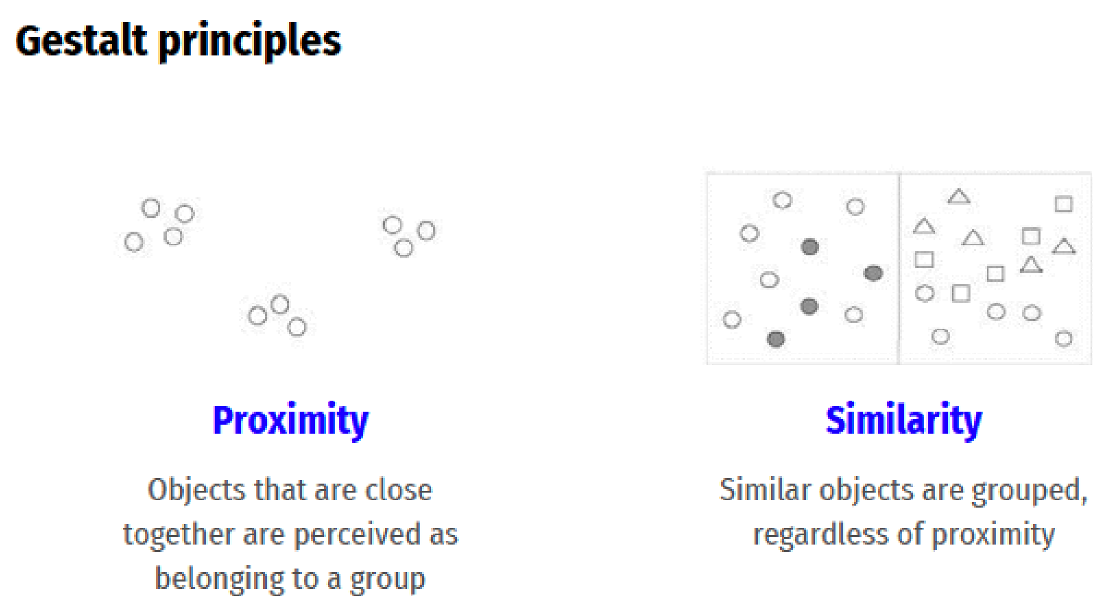



Figure 2: Gestalt#1

***Cognitive load*** can be divided into: Intrinsic (the intrinsic complexity of the new Germane (the audience's familiarity with the Extraneous (complexity from how the information is presented)

To find the useful dataviz catalogue, visit the DataViz catalogue

Interactive version

Stationary version

***Part3***

```
organdata
```

```
## # A tibble: 238 x 21
##    country  year       donors    pop pop_d~1    gdp gdp_lag health healt~2 pubhe~3
```

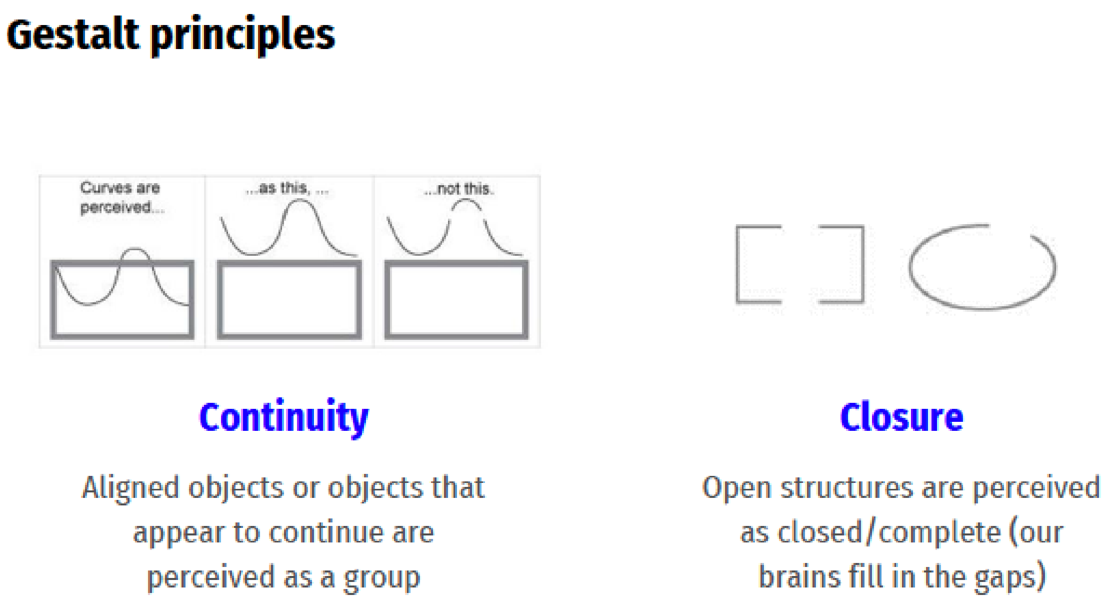



Figure 3: Gestalt#2

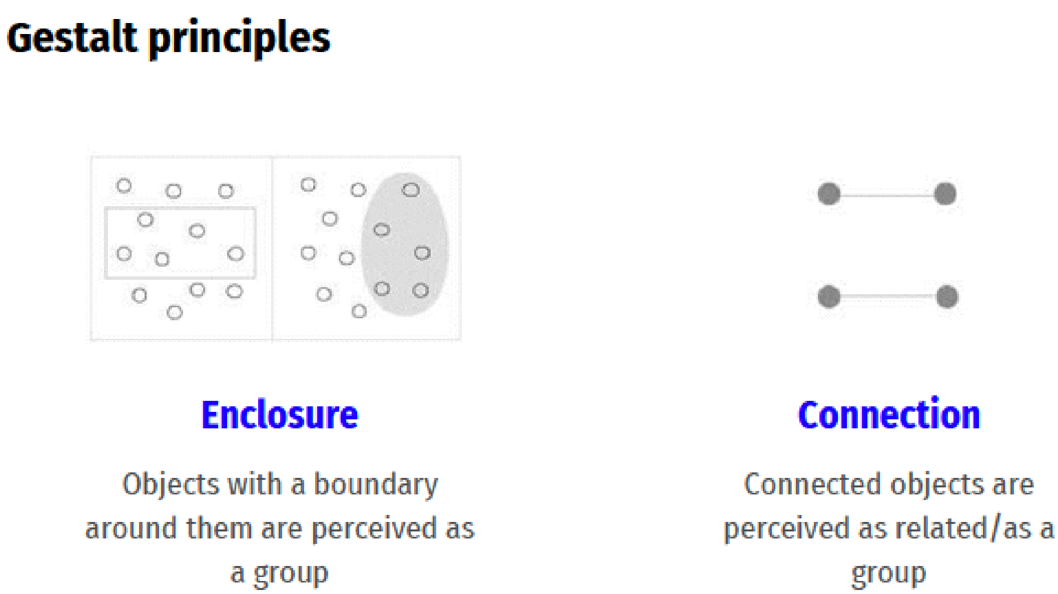



Figure 4: Gestalt#3

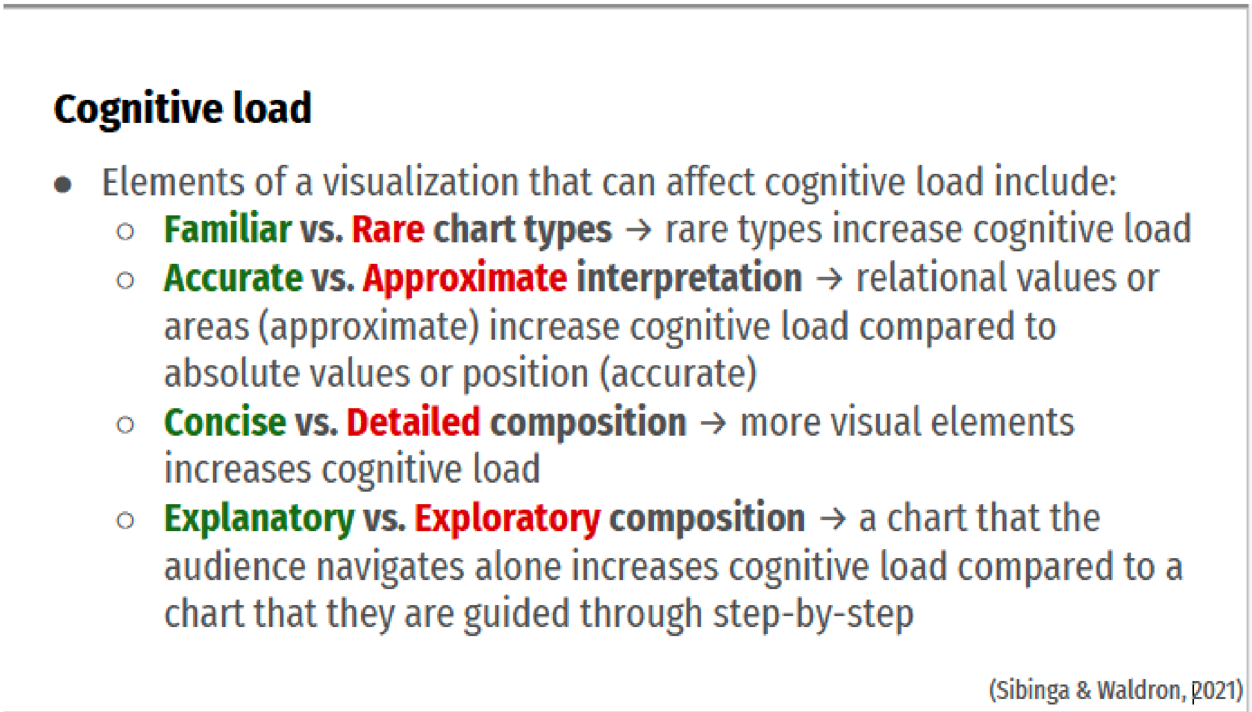



Figure 5: Cognitive load

```
##    <chr>    <date>       <dbl> <int>   <dbl> <int>   <int>  <dbl>    <dbl>    <dbl>
##  1 Austral~ NA           NA    17065   0.220 16774   16591   1300     1224      4.8
##  2 Austral~ 1991-01-01  12.1   17284   0.223 17171   16774   1379     1300      5.4
##  3 Austral~ 1992-01-01  12.4   17495   0.226 17914   17171   1455     1379      5.4
##  4 Austral~ 1993-01-01  12.5   17667   0.228 18883   17914   1540     1455      5.4
##  5 Austral~ 1994-01-01  10.2   17855   0.231 19849   18883   1626     1540      5.4
##  6 Austral~ 1995-01-01  10.2   18072   0.233 21079   19849   1737     1626      5.5
##  7 Austral~ 1996-01-01  10.6   18311   0.237 21923   21079   1846     1737      5.6
##  8 Austral~ 1997-01-01  10.3   18518   0.239 22961   21923   1948     1846      5.7
##  9 Austral~ 1998-01-01  10.5   18711   0.242 24148   22961   2077     1948      5.9
## 10 Austral~ 1999-01-01   8.67 18926   0.244 25445   24148   2231     2077      6.1
## # ... with 228 more rows, 11 more variables: roads <dbl>, cerebvas <int>,
## #   assault <int>, external <int>, txp_pop <dbl>, world <chr>, opt <chr>,
## #   consent_law <chr>, consent_practice <chr>, consistent <chr>, ccode <chr>,
## #   and abbreviated variable names 1: pop_dens, 2: health_lag, 3: pubhealth
```

```
#A dataset containing data on rates of organ donation for seventeen OECD countries between 1991 and 200
```

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

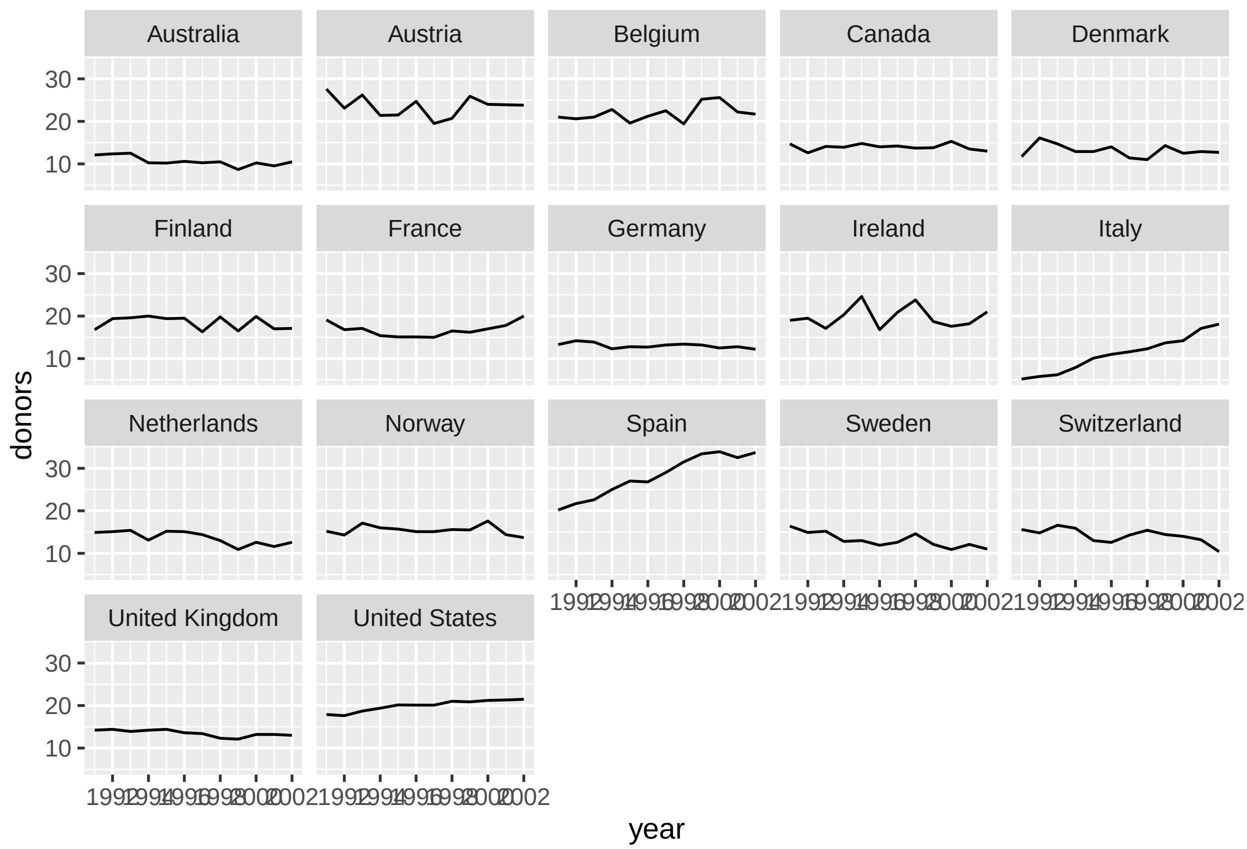
```
## # A tibble: 10 x 6
##    country       year       donors   pop pop_dens   gdp
##    <chr>         <date>      <dbl> <int>    <dbl> <int>
##  1 Germany       2002-01-01   12.2 82489    23.1  25843
##  2 Denmark       1996-01-01   14    5263    12.2  23548
##  3 France        NA           NA      NA    NA       NA
##  4 Finland       NA           NA    4986     1.47 18025
```

```
##   5 Canada        1994-01-01   13.9  29036    0.291 21428
##   6 United States 1992-01-01   17.6 256514    2.66  24411
##   7 Australia     1992-01-01   12.4  17495    0.226 17914
##   8 Germany       1993-01-01   13.9  81156   22.7   19983
##   9 Switzerland   1993-01-01   16.6   6938   16.8   25316
##  10 United States 1991-01-01   17.9 252981    2.63  23443
```

```
#select from col 1 to 6
#random sample for 10 rows
```

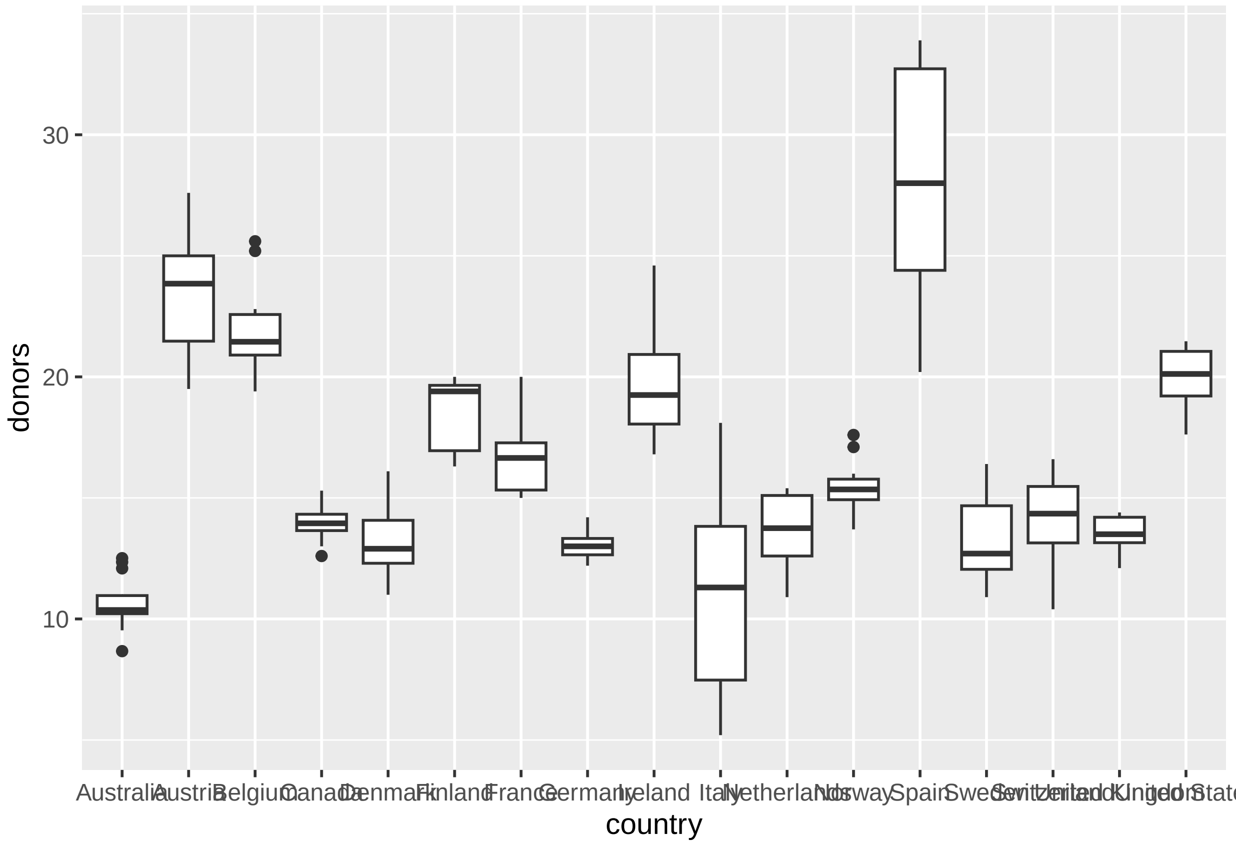
```
#continuous variables by group or category
p <- ggplot(data = organdata, mapping = aes(x = year, y = donors))

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

```
## Warning: Removed 34 rows containing missing values (‘geom_line()‘).
```



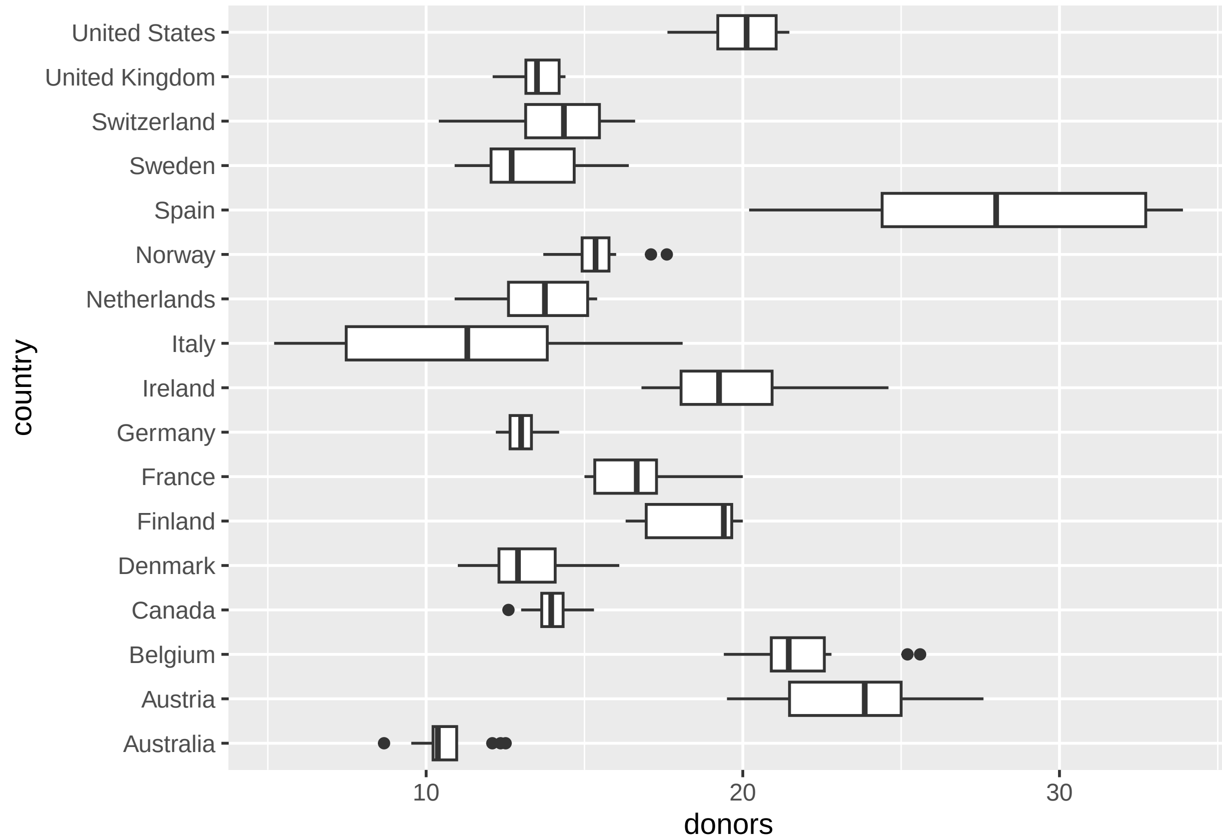

```
#create the geom_boxplot()
p <- ggplot(data = organdata, mapping = aes(x = country, y = donors))

p + geom_boxplot()
```

```
## Warning: Removed 34 rows containing non-finite values (‘stat_boxplot()‘).
```



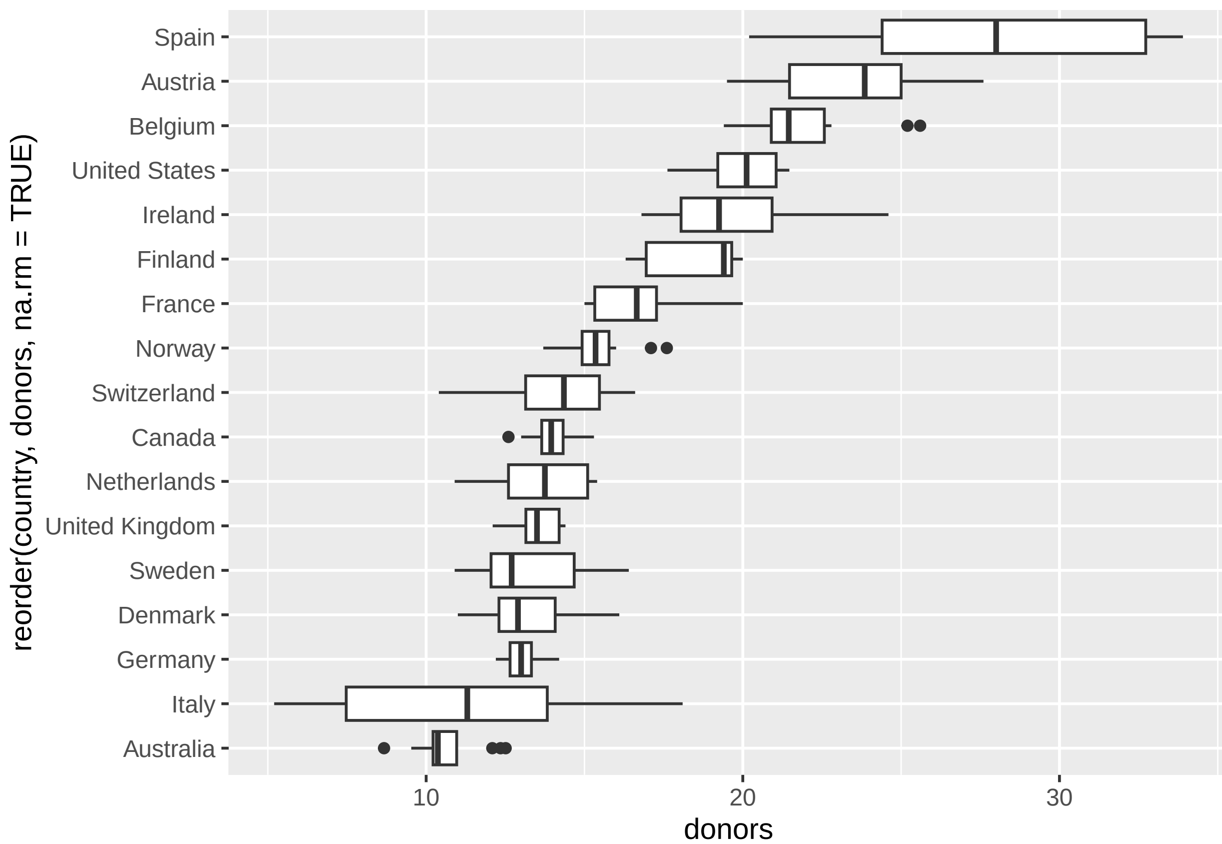

```r
#improving the boxplot by adding coord_flip()
p <- ggplot(data = organdata, mapping = aes(x = country, y = donors))

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

```
## Warning: Removed 34 rows containing non-finite values (‘stat_boxplot()‘).
```




```r
#improving the boxplot by reordering
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, donors, na.rm = TRUE), y = donors))

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

```
## Warning: Removed 34 rows containing non-finite values (`stat_boxplot()`).
```

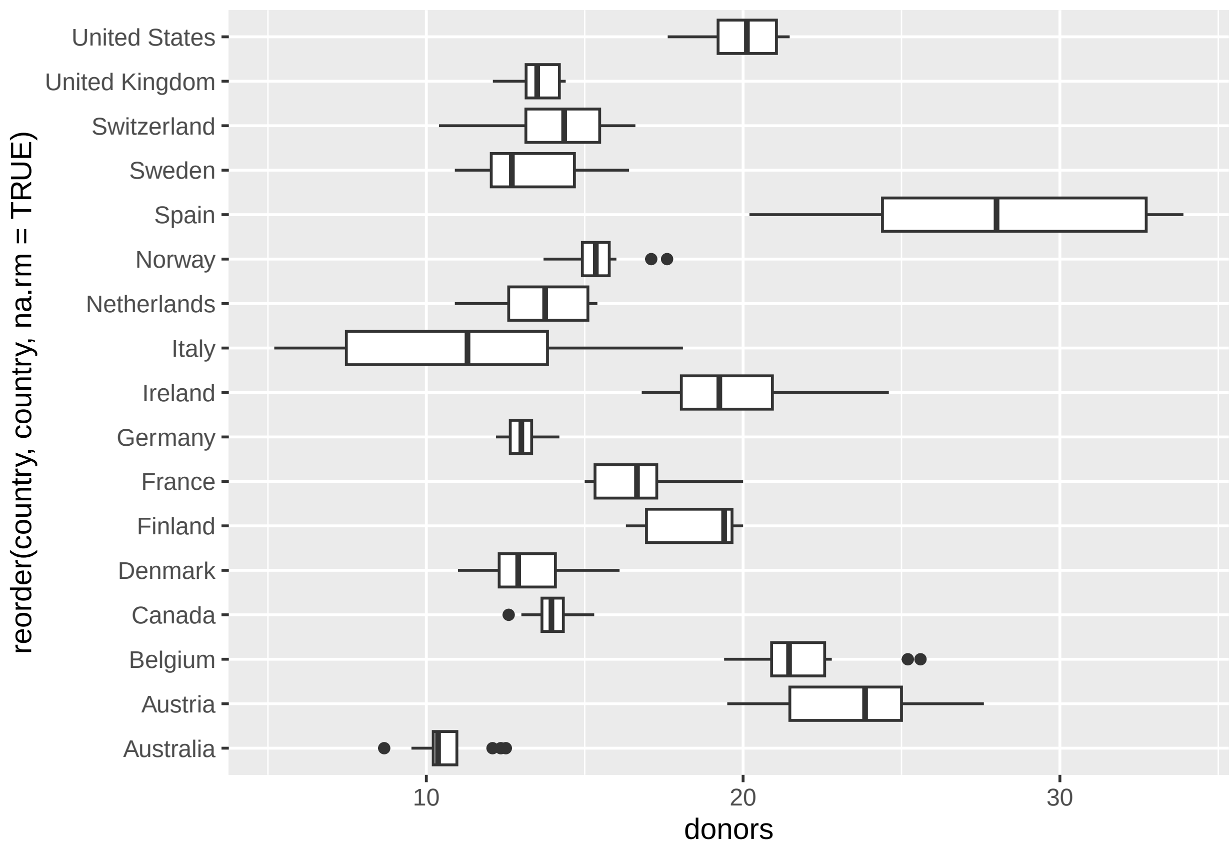



```
#improving the boxplot by reordering alphabettically
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, country, na.rm = TRUE), y = donors))

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

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

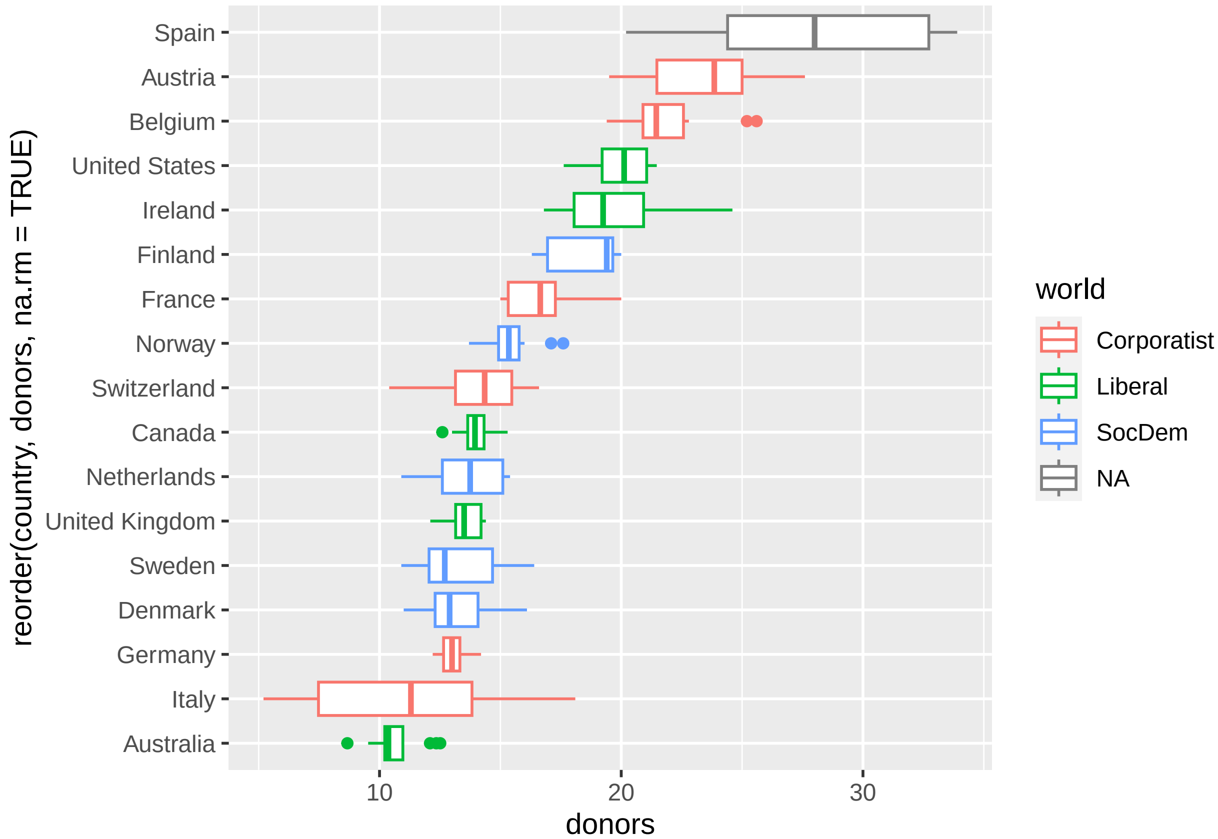
```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning: Removed 34 rows containing non-finite values (‘stat_boxplot()‘).
```



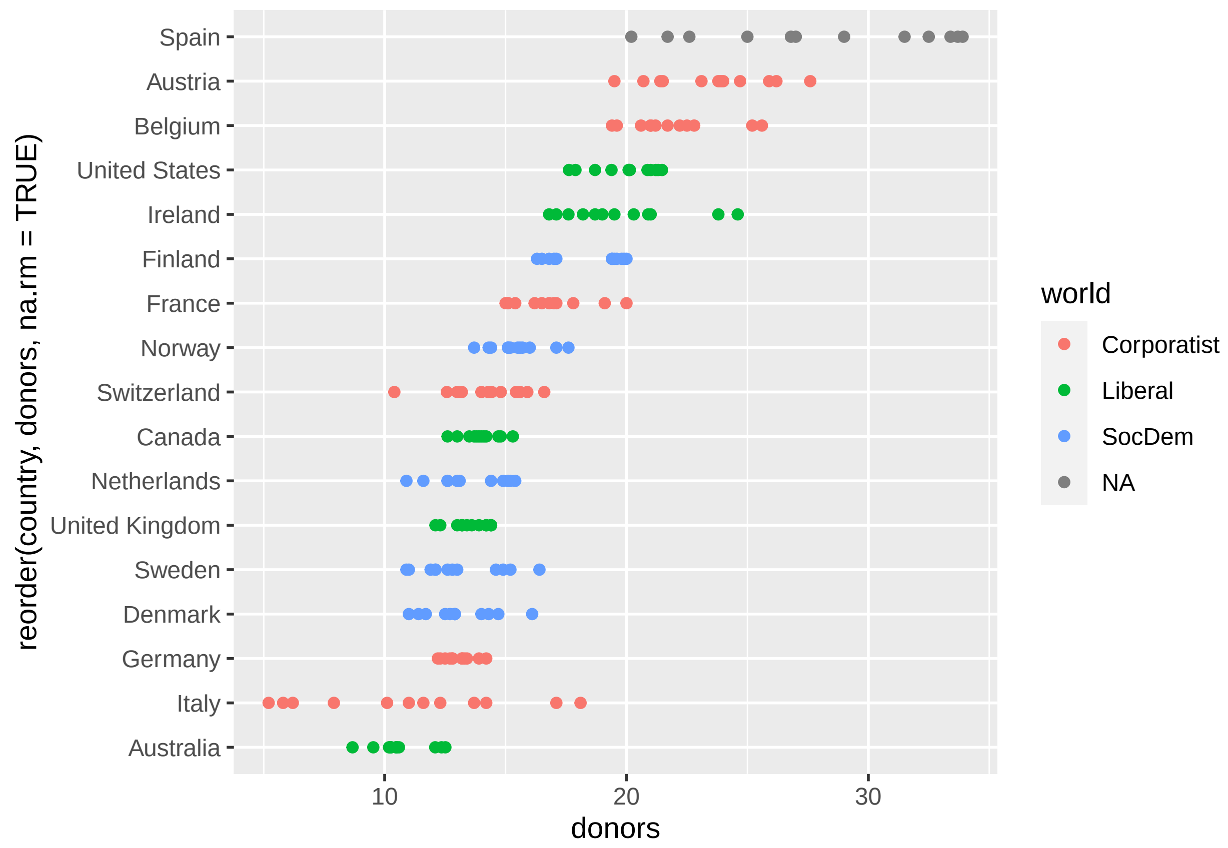

```
#improving the boxplot by coloring the world variable
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, donors, na.rm = TRUE), y = donors, colo

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

```
## Warning: Removed 34 rows containing non-finite values (`stat_boxplot()`).
```



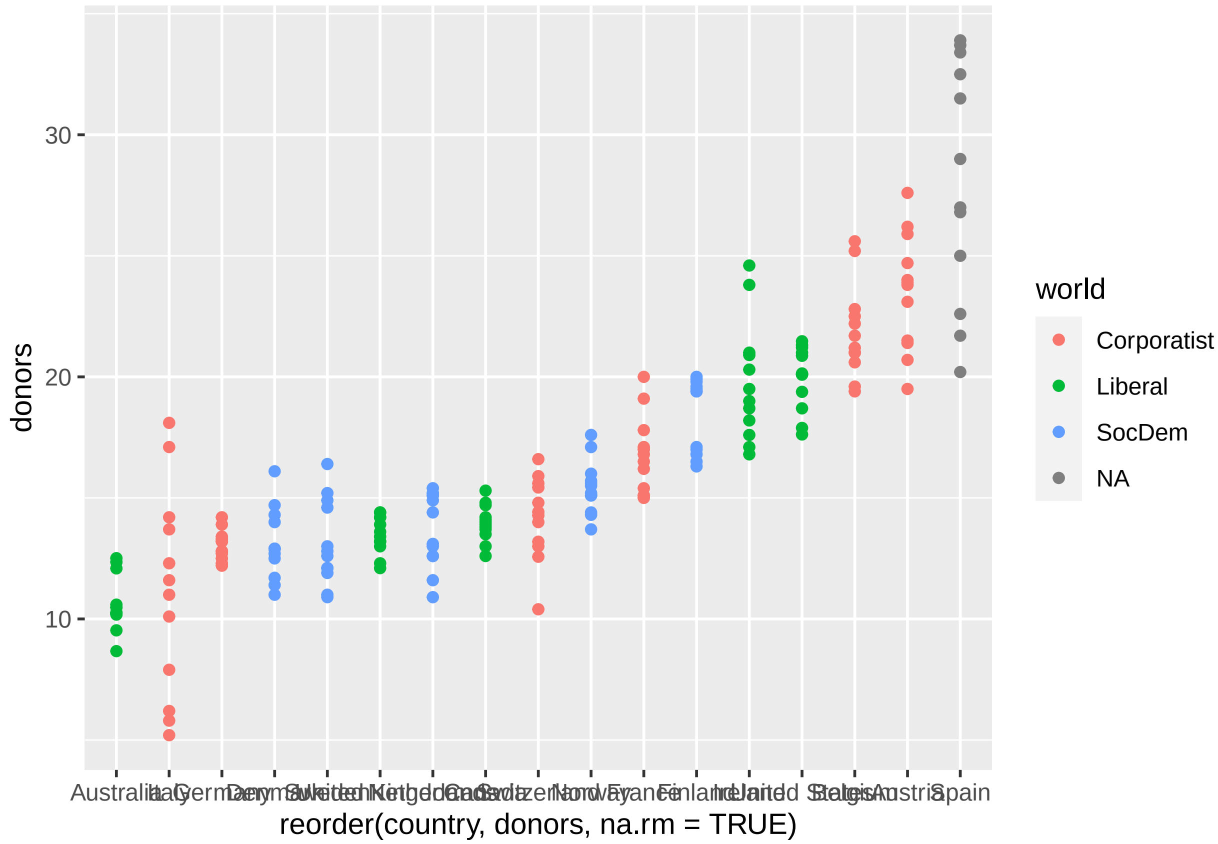

```
#improving the boxplot by changing into a point plot
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, donors, na.rm = TRUE), y = donors, colo

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

```
## Warning: Removed 34 rows containing missing values (`geom_point()`).
```




```
#improving the boxplot by changing into a point plot and remove coord_flip()
p <- ggplot(data = organdata, mapping = aes(x = reorder(country, donors, na.rm = TRUE), y = donors, col

p + geom_point()
```

```
## Warning: Removed 34 rows containing missing values (`geom_point()`).
```




Summarizing the data

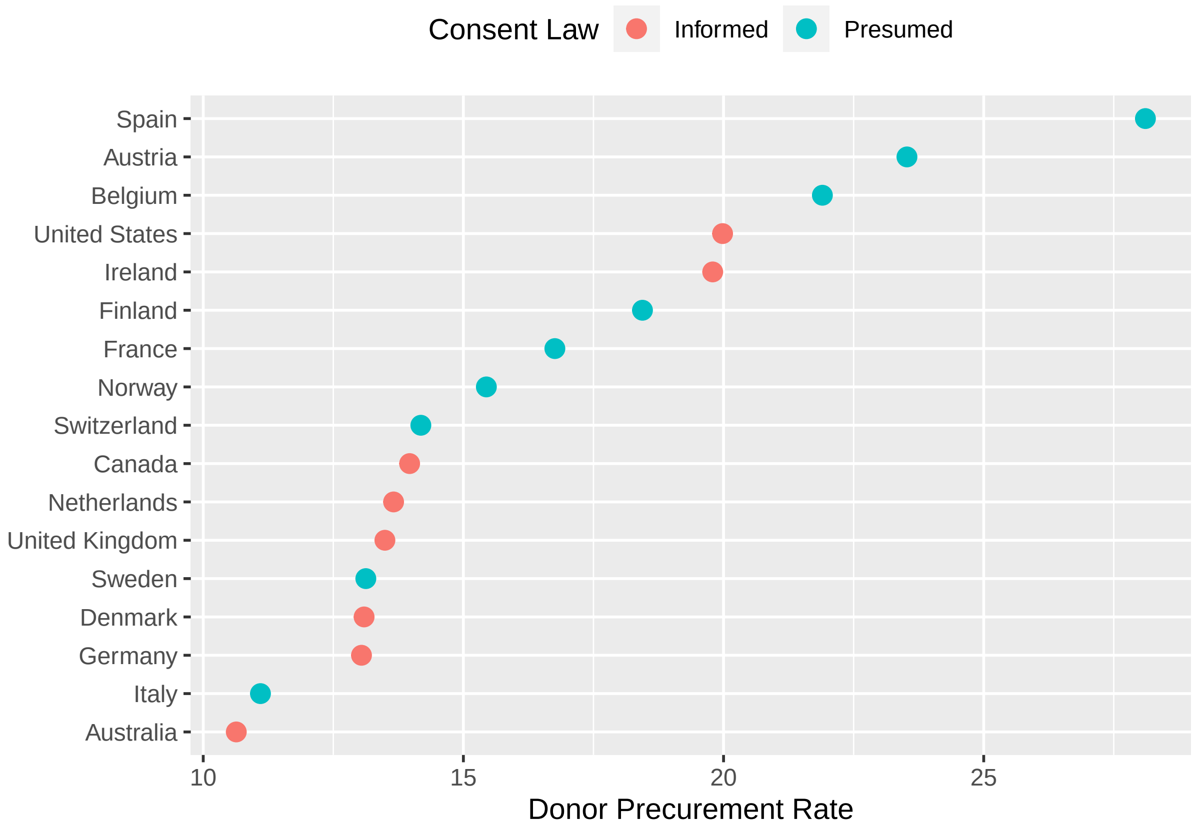
```
by_country <- organdata |> group_by(consent_law, country) |> summarize_if(is.numeric, funs(mean, sd), na
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```
#if our col is a numeric, it will calculate the mean and SD
#exclude NA data by using na.rm = TRUE
```

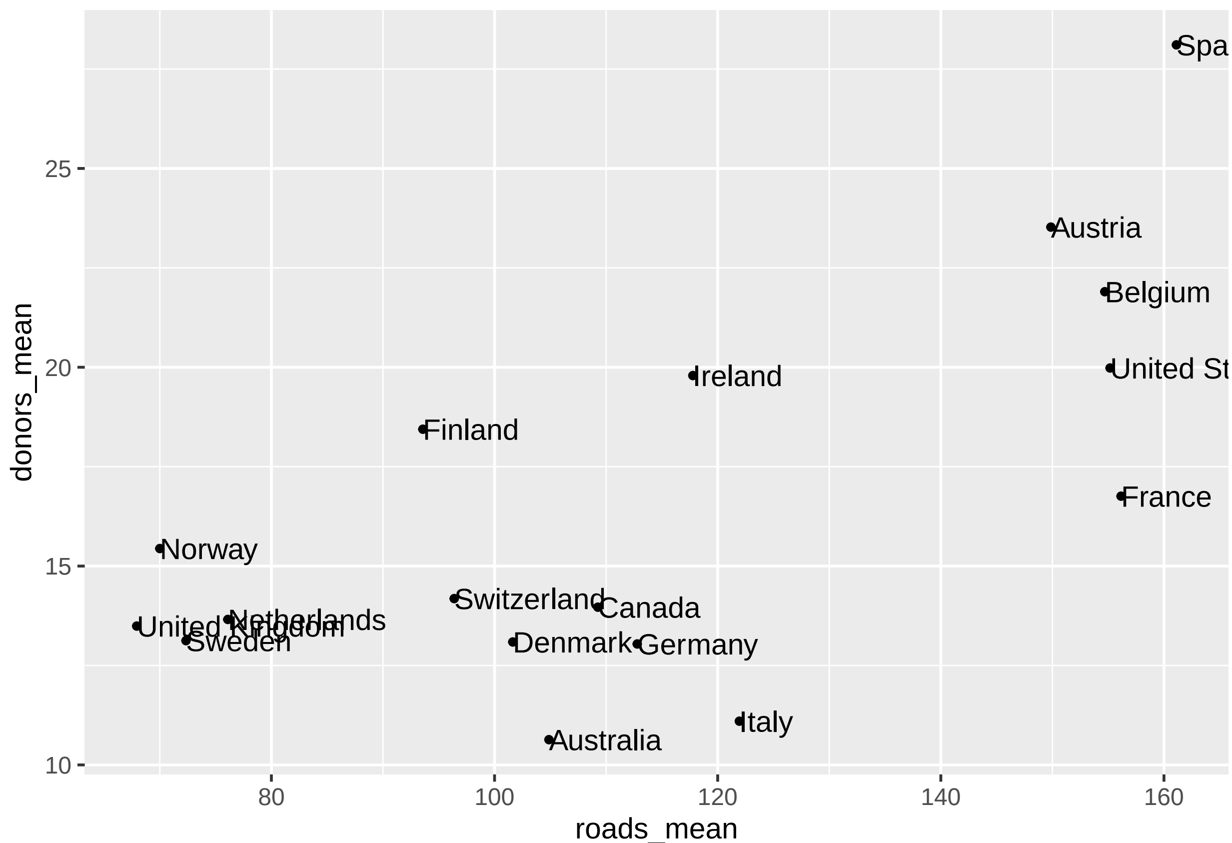
**Categorical variables with a single point** Now that we have summarized our data, we can make our Cleveland dotplot using geom_point() We will also - Colour our results by the consent law for each country - Move our legend to the top of our plot

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

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




##Adding text labels to data points

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

p + geom_point(size = 1) + geom_text(mapping = aes(label = country), hjust = 0)
```



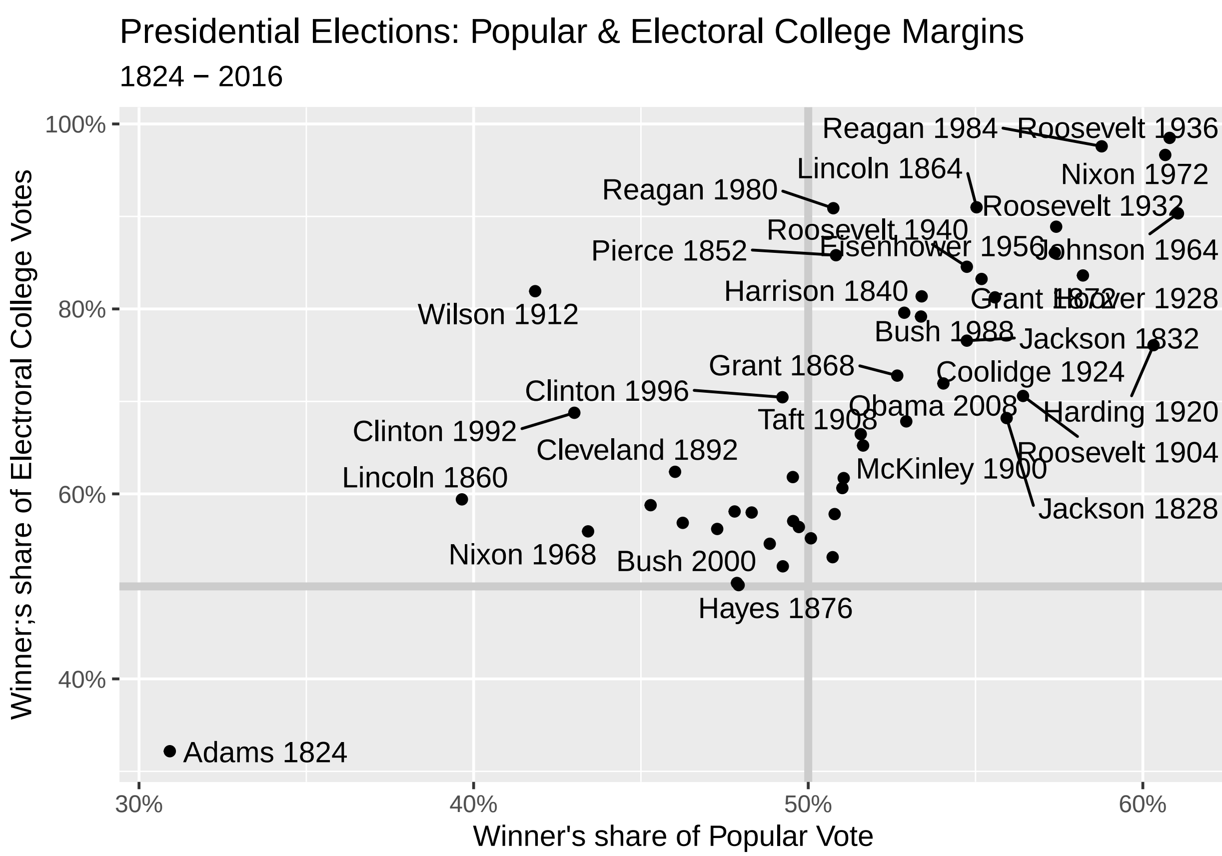

#load the library ggrepel

```
library(ggrepel)

elections_historic
```

```
## # A tibble: 49 x 19
##    election  year winner    win_p~1 ec_pct popul~2 popul~3  votes margin runne~4
##       <int> <int> <chr>     <chr>    <dbl>   <dbl>   <dbl>  <int>  <int> <chr>
##  1       10  1824 John Qui~ D.-R.    0.322   0.309 -0.104  1.13e5 -38221 Andrew~
##  2       11  1828 Andrew J~ Dem.     0.682   0.559  0.122  6.43e5 140839 John Q~
##  3       12  1832 Andrew J~ Dem.     0.766   0.547  0.178  7.03e5 228628 Henry ~
##  4       13  1836 Martin V~ Dem.     0.578   0.508  0.142  7.63e5 213384 Willia~
##  5       14  1840 William ~ Whig     0.796   0.529  0.0605 1.28e6 145938 Martin~
##  6       15  1844 James Po~ Dem.     0.618   0.495  0.0145 1.34e6  39413 Henry ~
##  7       16  1848 Zachary ~ Whig     0.562   0.473  0.0479 1.36e6 137882 Lewis ~
##  8       17  1852 Franklin~ Dem.     0.858   0.508  0.0695 1.61e6 219525 Winfie~
##  9       18  1856 James Bu~ Dem.     0.588   0.453  0.122  1.84e6 494472 John F~
## 10       19  1860 Abraham ~ Rep.     0.594   0.396  0.101  1.86e6 474049 John B~
## # ... with 39 more rows, 9 more variables: ru_part <chr>, turnout_pct <dbl>,
## #   winner_lname <chr>, winner_label <chr>, ru_lname <chr>, ru_label <chr>,
## #   two_term <lgl>, ec_votes <dbl>, ec_denom <dbl>, and abbreviated variable
## #   names 1: win_party, 2: popular_pct, 3: popular_margin, 4: runner_up
```

```r
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") +
  #We add reference lines so we can see the 50% threshold of votes on each axis.by geom_hline
  geom_vline(xintercept = 0.5, size = 1.4, color = "gray80") +
  geom_point() +
  #geom_text_repel() will ensure that our data labels do not overlap.
  geom_text_repel() +
  #Vote shares are stored as proportions rather than percents, so we adjust the labels of the scales.
  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 Electroral College Votes", title ="
```

```
## Warning: ggrepel: 17 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




Reference