

# Títulos e anotações

Hanna Rodrigues Ferreira

05 maio,2021

Vamos reproduzir os resultados do capítulo 5 do livro **Data Visualization - A practical Introduction**. Os dados utilizados foram retirados deste github.

## Adicionando Títulos e fazendo anotações

Vamos fazer uso da biblioteca a biblioteca `socviz` para fazer uso do datasets mencionados no livro.

```
knitr::opts_chunk$set(echo = TRUE)

library(gapminder)
library(tidyverse)
library(socviz)
```

## Usando pipe para resumir dados

Utilizaremos o dataset `gss_sm` descrito neste site.

Podemos transformar os dados utilizando a biblioteca `dplyr`, contida na `tidyverse`. A seguir temos a tabela de das porcentagens de preferencias religiosas segundo cada região.

```
rel_by_region <- gss_sm %>%
  group_by(bigregion, religion) %>%
  summarize(N = n()) %>%
  mutate(freq = N/sum(N),
         pct = round((freq*100), 0))
```

```
rel_by_region
```

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##   bigregion religion      N   freq  pct
##   <fct>      <fct>   <int> <dbl> <dbl>
## 1 Northeast Protestant  158 0.324   32
## 2 Northeast Catholic   162 0.332   33
## 3 Northeast Jewish     27 0.0553    6
## 4 Northeast None      112 0.230   23
## 5 Northeast Other      28 0.0574    6
## 6 Northeast <NA>        1 0.00205    0
## 7 Midwest   Protestant  325 0.468   47
```

```
## 8 Midwest Catholic 172 0.247 25
## 9 Midwest Jewish 3 0.00432 0
## 10 Midwest None 157 0.226 23
## # ... with 14 more rows
```

O uso de pipes aumenta a legibilidade do código, fazendo a tarefa de checar a sanidade do nosso resultado mais fácil.

Se fizemos tudo na ordem correta, a soma das porcentagens por região deve resultar em 100% (considerando erros de arredondamento). Conforme mostrado a seguir:

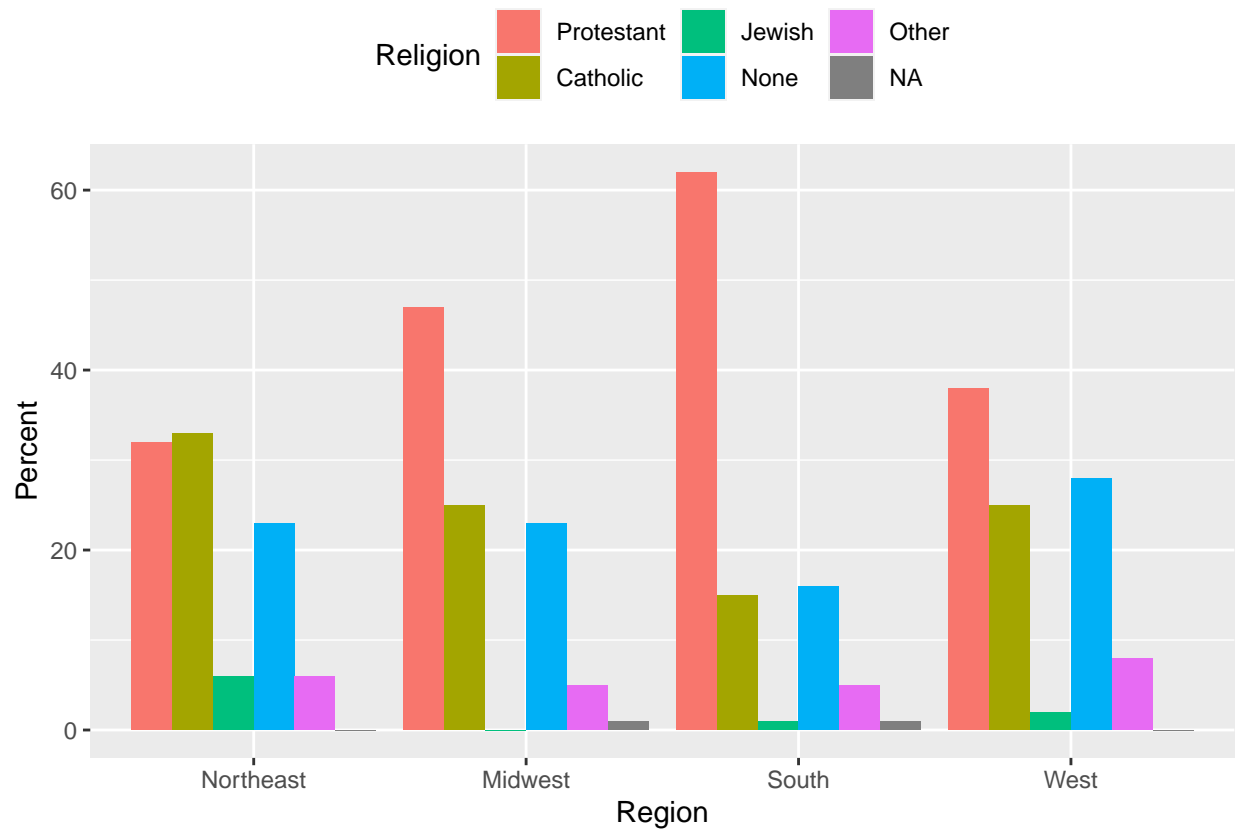
```
rel_by_region %>%
  group_by(bigregion) %>%
  summarize(total = sum(pct))
```

```
## # A tibble: 4 x 2
##   bigregion total
##   <fct>      <dbl>
## 1 Northeast  100
## 2 Midwest   101
## 3 South     100
## 4 West      101
```

Podemos então trabalhar diretamente com as porcentagens e fazer um gráfico de barras das preferências religiosas por região:

```
p <- ggplot(rel_by_region,
  aes(x = bigregion, y = pct, fill = religion) )

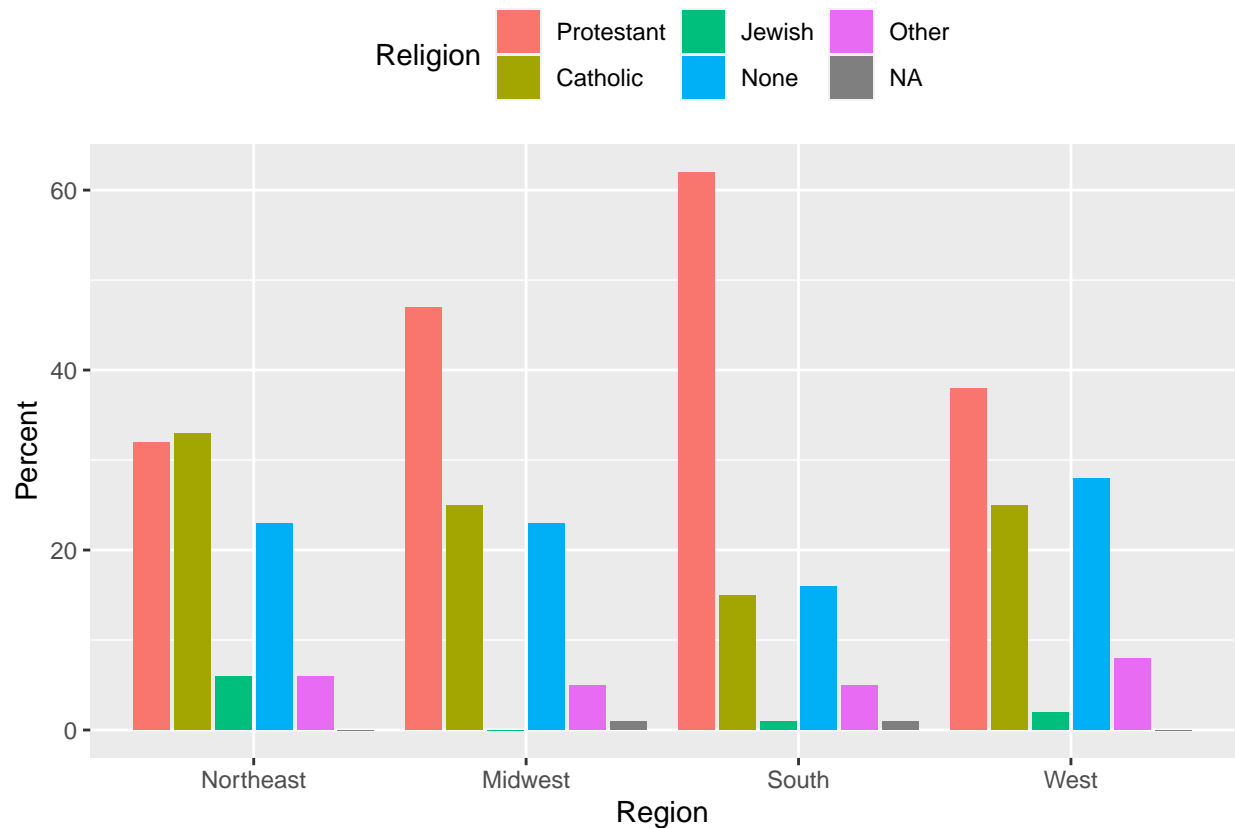
p + geom_col( position = 'dodge') +
  labs(x = 'Region', y = 'Percent', fill = 'Religion') +
  theme(legend.position = 'top')
```



Utilizaremos **dodge2** em vez de **dodge** pois os dados já estão com as proporções computadas.

```
p <- ggplot(rel_by_region,
  aes(x = bigregion, y = pct, fill = religion) )

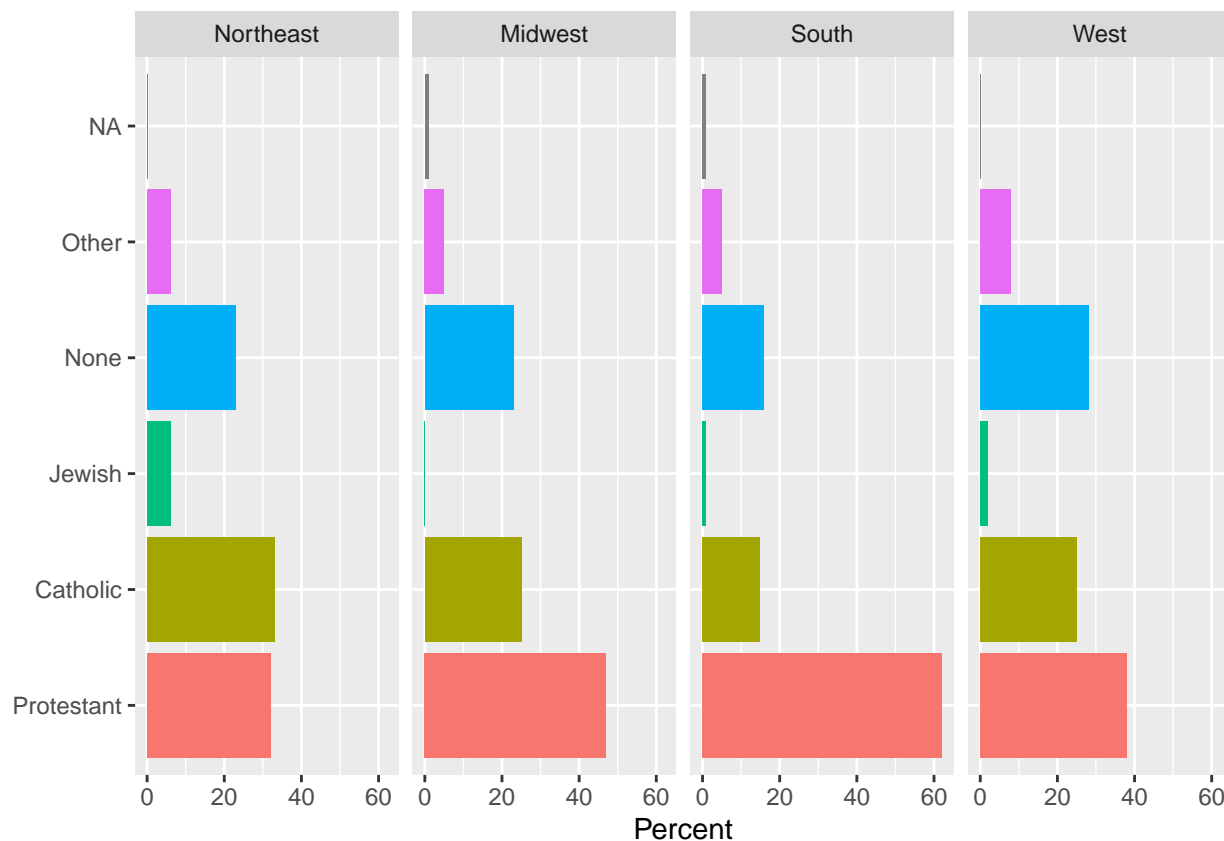
p + geom_col( position = 'dodge2') +
  labs(x = 'Region', y = 'Percent', fill = 'Religion') +
  theme(legend.position = 'top')
```



Ainda sim existem várias barras uma ao lado da outra, uma alternativa é deitar o gráfico de barras através do `coord_flip()`.

```
p <- ggplot(rel_by_region,
            aes(x = religion, y = pct, fill = religion) )

p + geom_col( position = 'dodge2') +
  labs(x = NULL, y = 'Percent', fill = 'Religion') +
  guides(fill = FALSE) +
  coord_flip() +
  facet_grid(~ bigregion)
```



## Variáveis Contínuas agrupadas por Categoria ou Grupo

Utilizaremos o dataset *organdata* descrito neste site

Podemos alternativamente ao `head()`, utilizar a função `sample_n()` para coletar 10 linhas aleatoriamente dos dados. Observe que também selecionamos as primeiras 6 colunas.

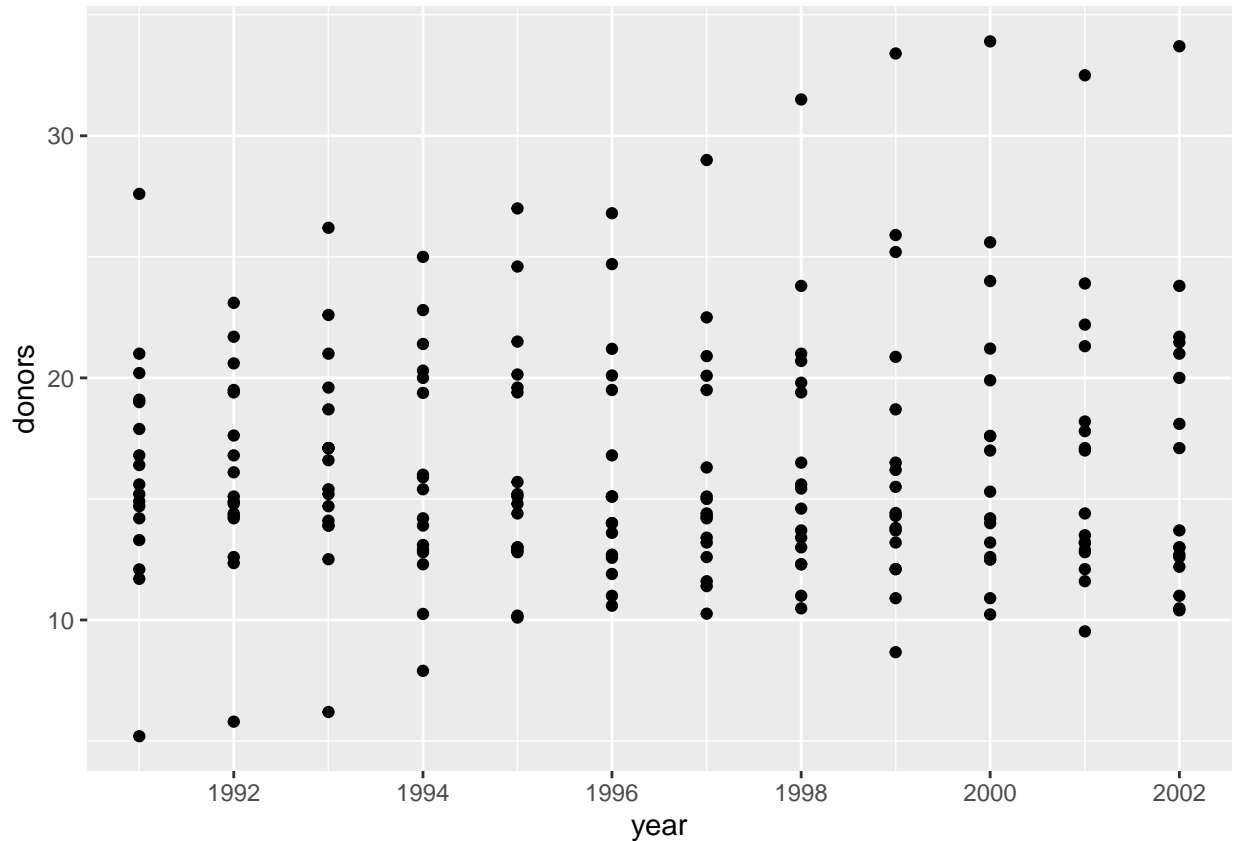
```
organdata %>%
  select(1:6) %>%
  sample_n(size = 10)
```

```
## # A tibble: 10 x 6
##   country    year   donors   pop pop_dens   gdp
##   <chr>      <date>   <dbl> <int>   <dbl> <int>
## 1 France    NA        NA      NA      NA      NA
## 2 France    1992-01-01 16.8  57240  10.4  19566
## 3 Netherlands 1995-01-01 15.2  15459  37.2  21723
## 4 Denmark    NA        NA      NA      NA      NA
## 5 Belgium    NA        NA      NA      NA      NA
## 6 Canada     1999-01-01 13.8  30499  0.306 26658
## 7 Canada     1992-01-01 12.6  28377  0.285 19590
## 8 United States 2001-01-01 21.3 285318  2.96 35118
## 9 Austria    1996-01-01 24.7   7959   9.49 23798
## 10 France    1995-01-01 15.1  57844  10.5 21283
```

Vamos plotar a taxa de doação de órgãos por milhão de população, donors, pelo tempo para começar a analisar os dados.

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

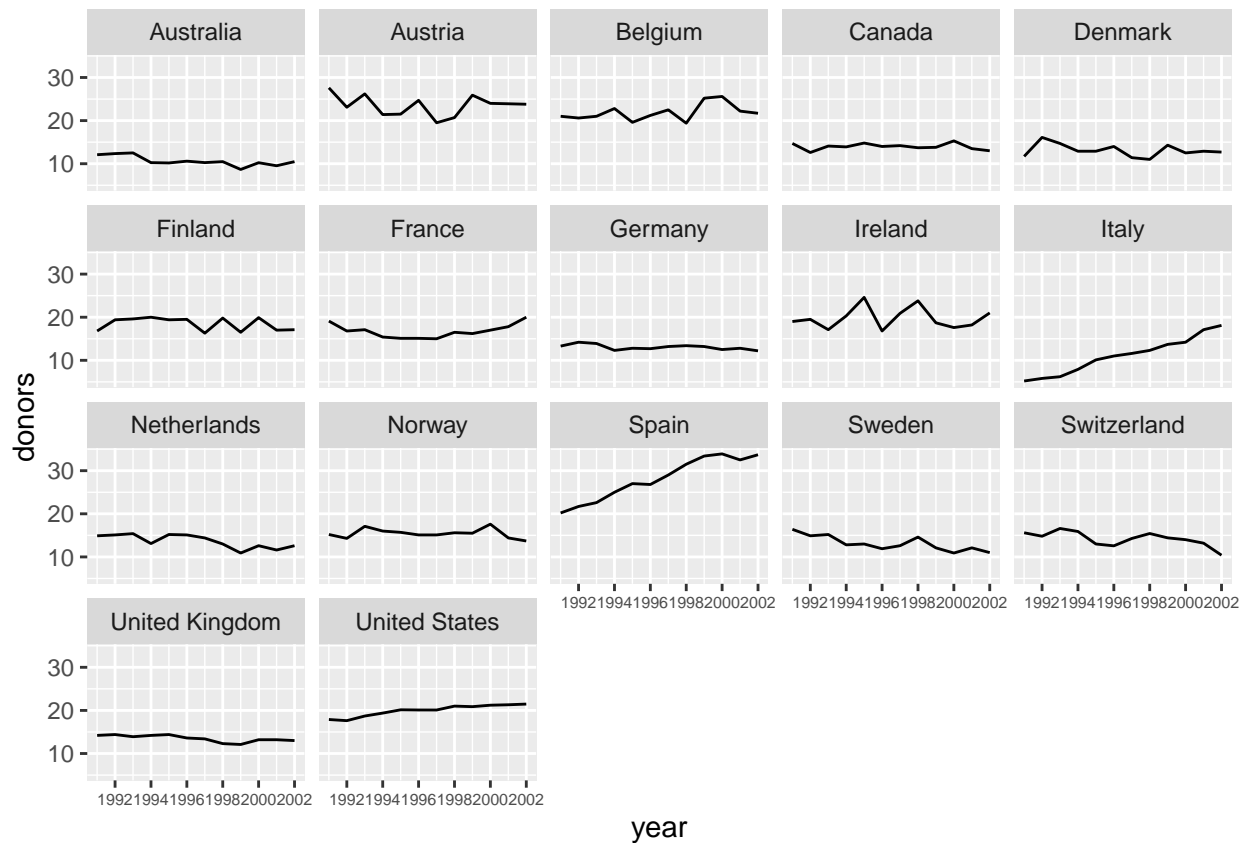
```
## Warning: Removed 34 rows containing missing values (geom_point).
```



Não muito informativo por enquanto. Podemos plotar a série histórica da taxa de doação de cada país utilizando `geom_line()`:

```
p <- ggplot(data = organdata,  
            mapping = aes(x = year, y = donors))  
  
p + geom_line(aes(group = country)) +  
  facet_wrap(~country) +  
  theme(axis.text.x = element_text(size=6))
```

```
## Warning: Removed 34 row(s) containing missing values (geom_path).
```

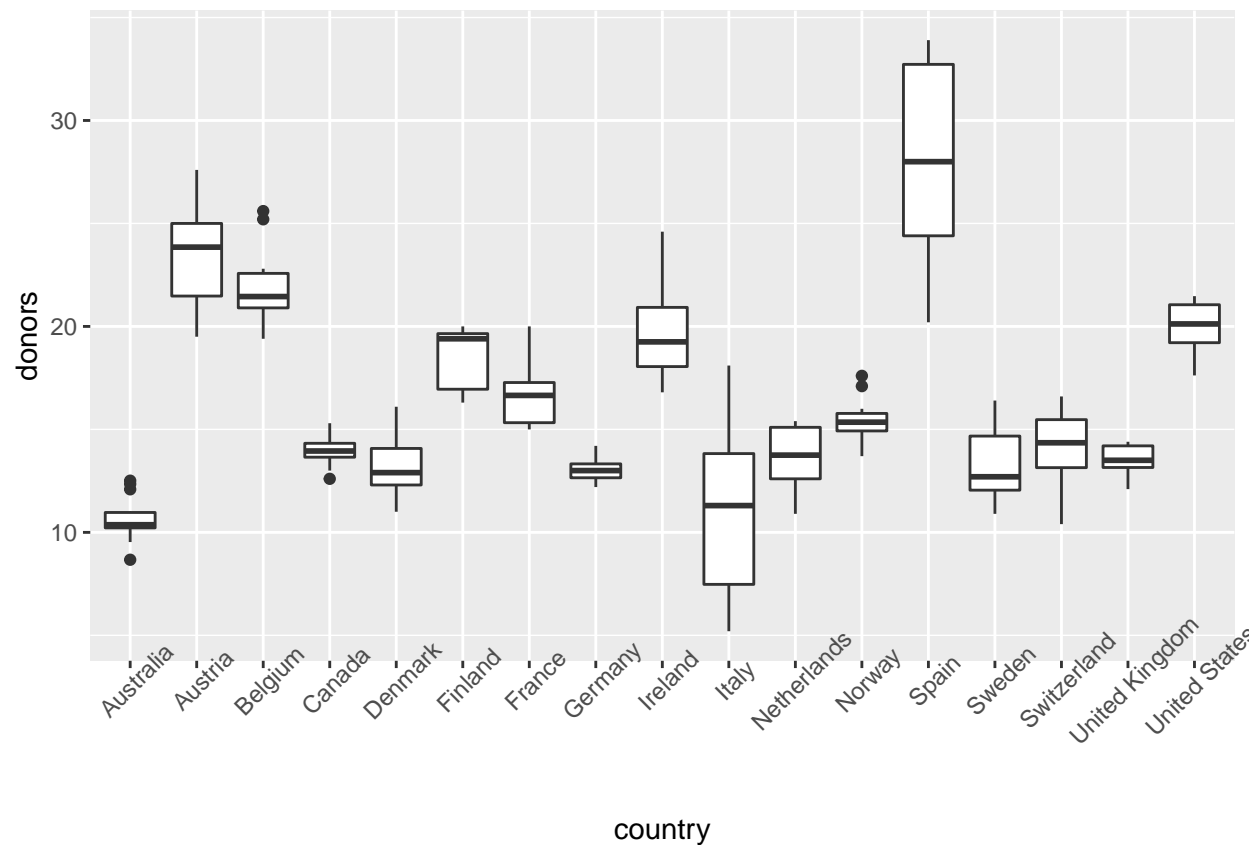


Focando agora não no tempo mas na variação da taxa de doação, plotaremos o boxplot de taxa de doação por país:

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

p + geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45))
```

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



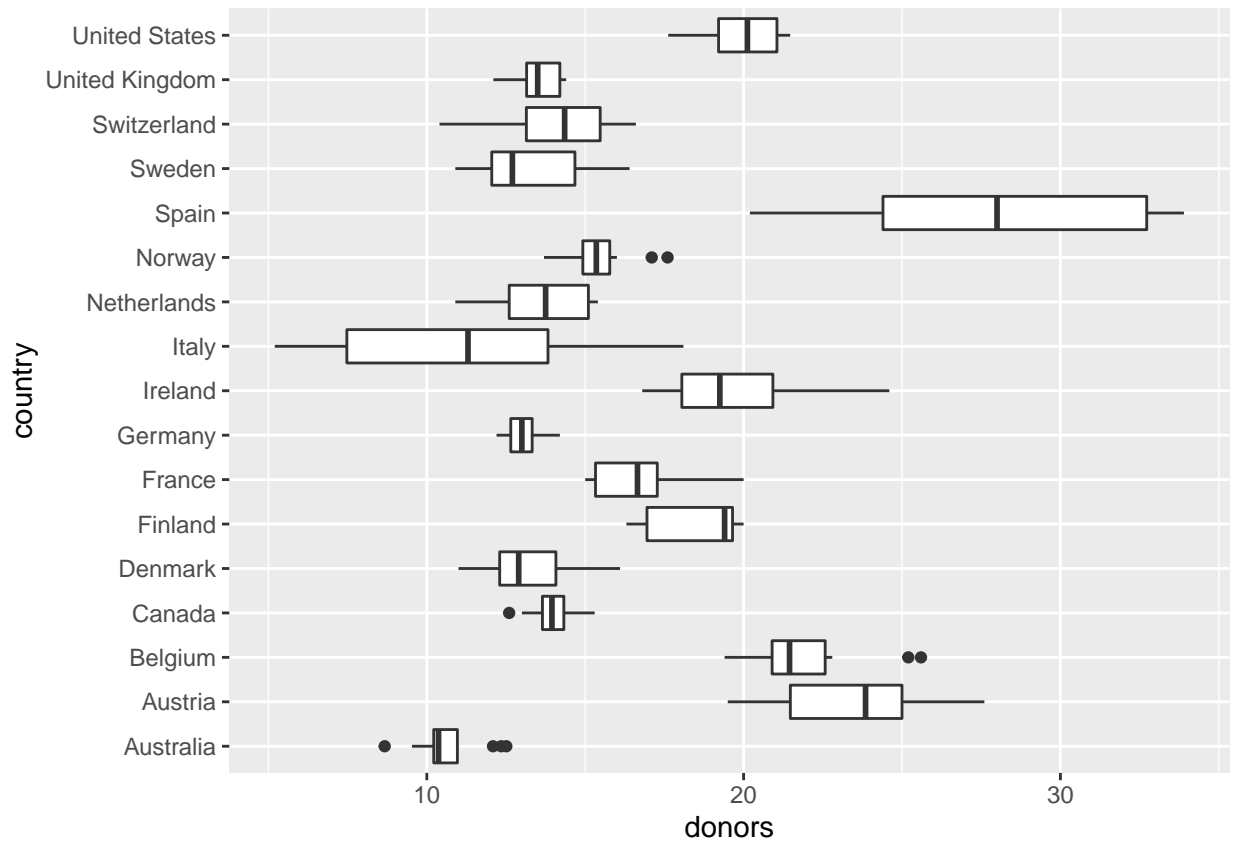
Observe que os países se sobrepõem, uma alternativa é deitar o gráfico usando `coord_flip()`:

```
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).
```



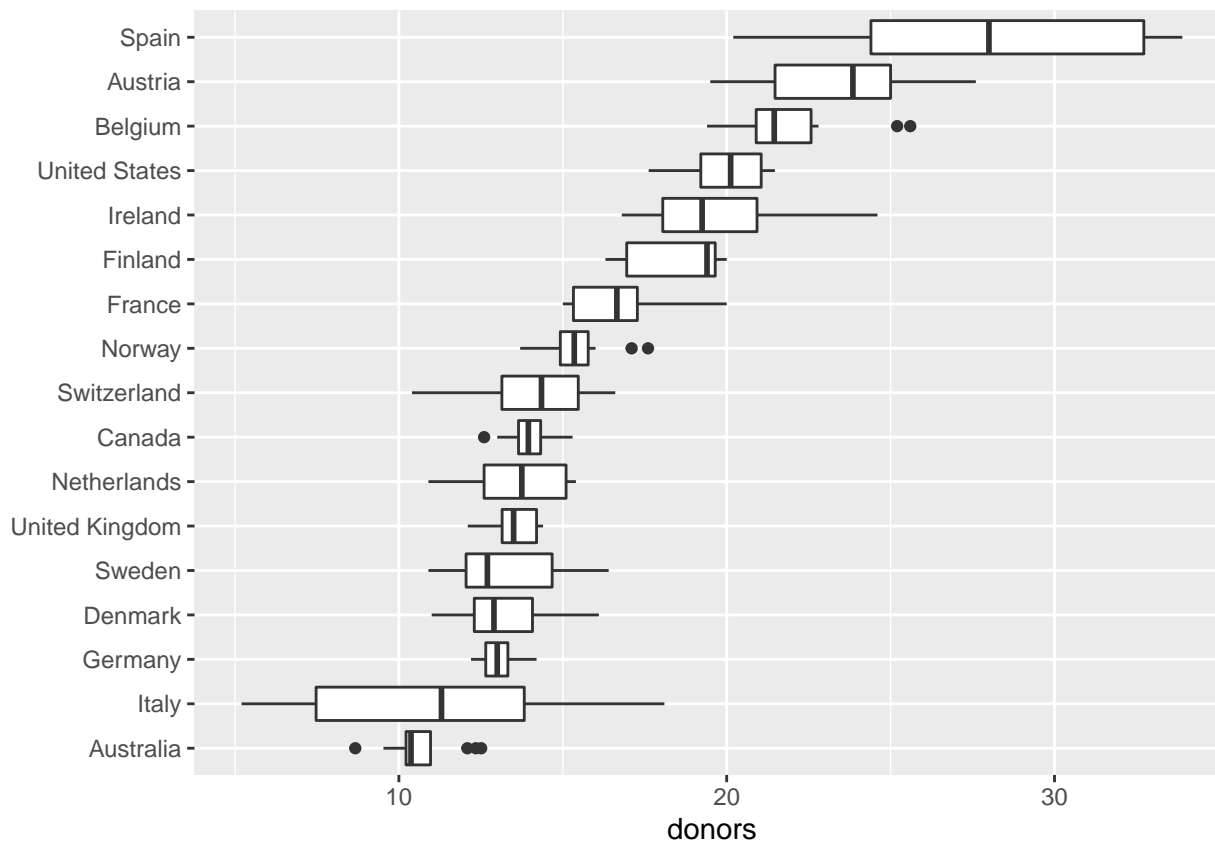


Mesmo mais legível, ele parece a priori pouco informativo, porque os países estão por ordem alfabética. Podemos reordená-los pela taxa de doações médias através da função **reorder()**

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

p + geom_boxplot() +
  labs(x = NULL) +
  coord_flip()
```

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

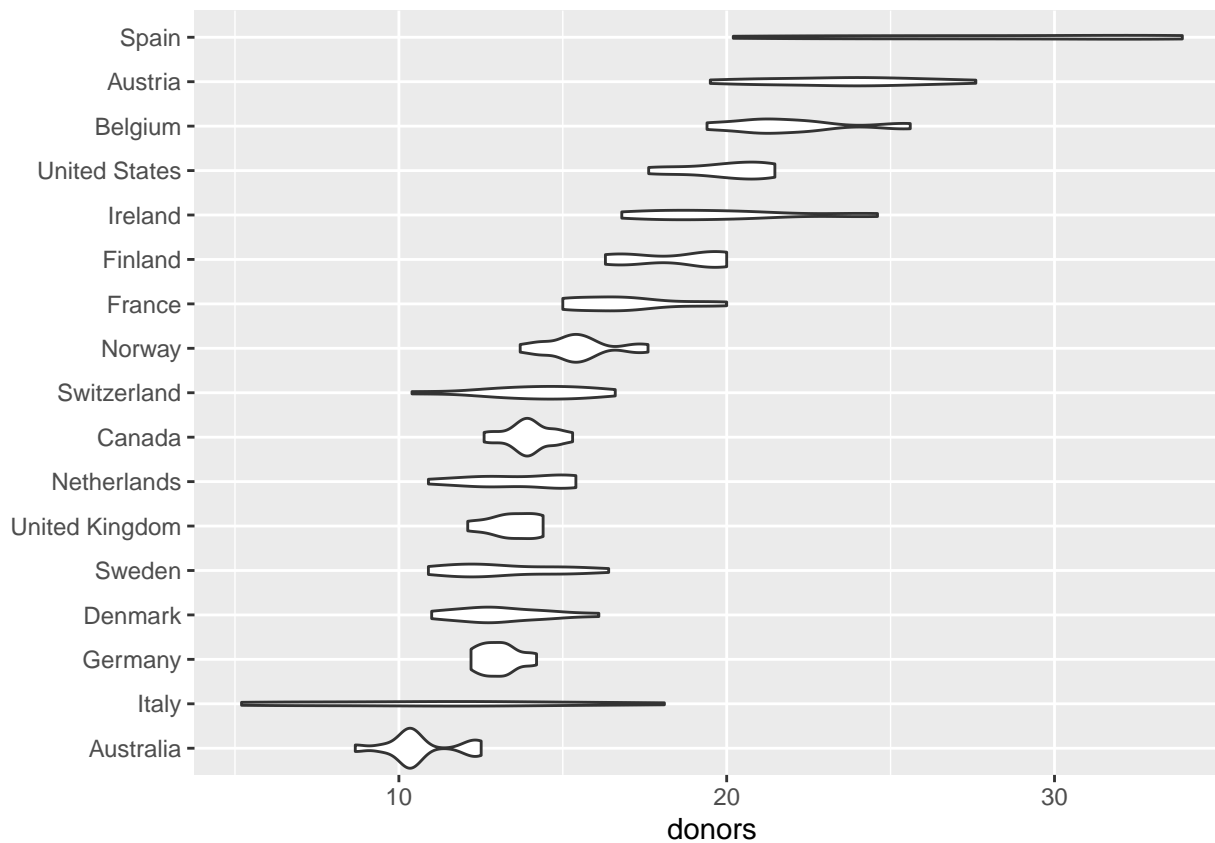


Podemos alternativamente fazer o **Violin plot** através da função `geom_violin()`

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

p + geom_violin() +
  labs(x = NULL) +
  coord_flip()
```

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

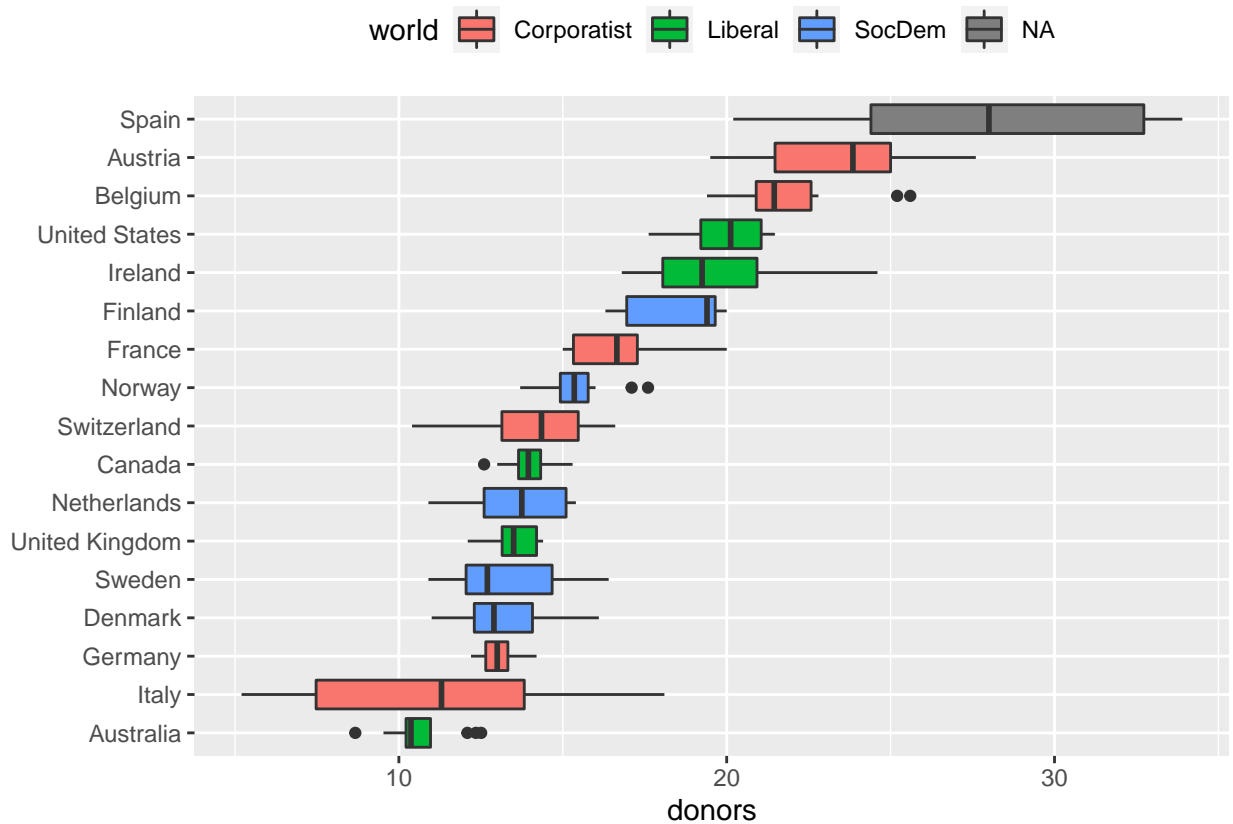


Podemos também colorir os boxplots de acordo com a variável categórica **world**:

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

p + geom_boxplot() +
  labs(x = NULL) +
  coord_flip() +
  theme(legend.position = 'top')
```

## Warning: Removed 34 rows containing non-finite values (stat\_boxplot).



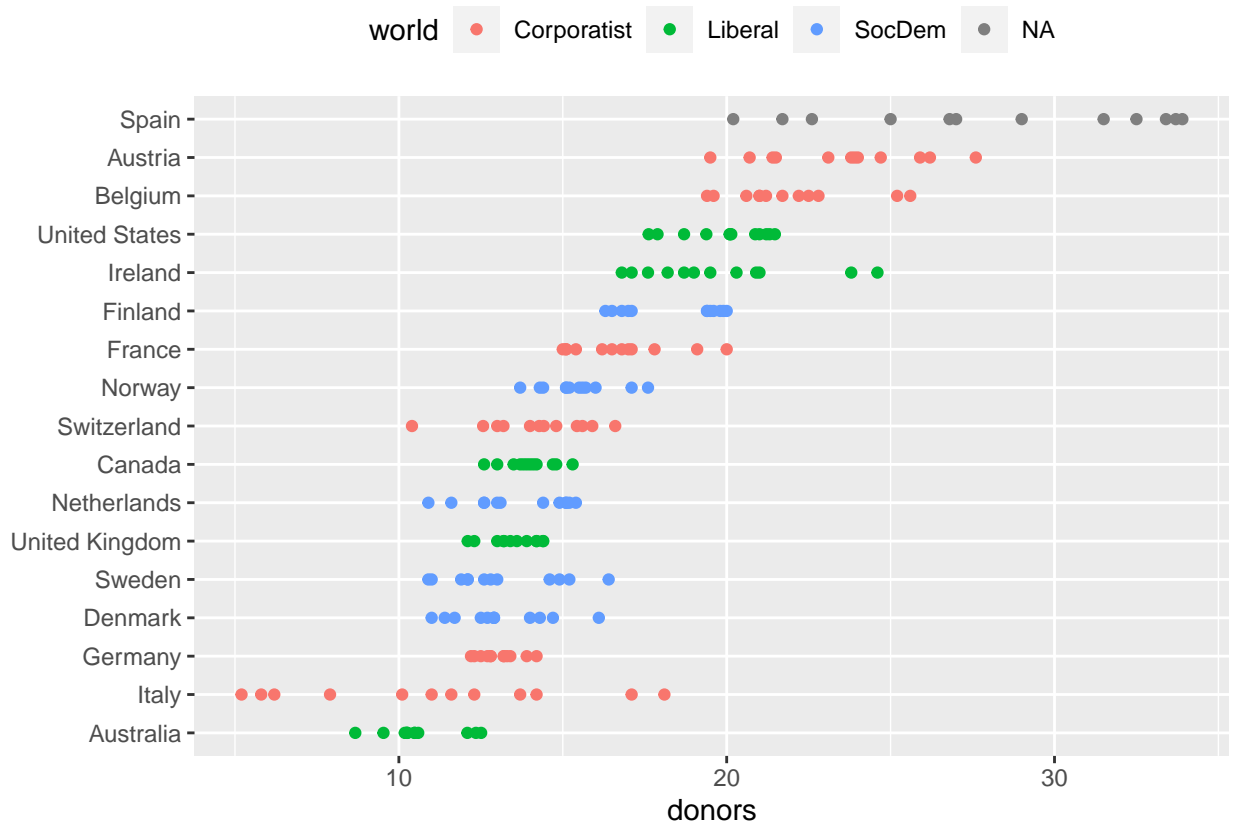
Colocando variáveis categóricas na vertical para comparar as distribuições é um bom recurso para resumir muitos pontos. Entretanto quando a quantidade de pontos por categoria é relativamente pequena, podemos simplesmente plotar as observações.

Vamos utilizar a função `geom_point()` em vez de `geom_boxplot()` fazendo os devidos ajustes de parâmetros:

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

p + geom_point() +
  labs(x = NULL) +
  coord_flip() +
  theme(legend.position = 'top')
```

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

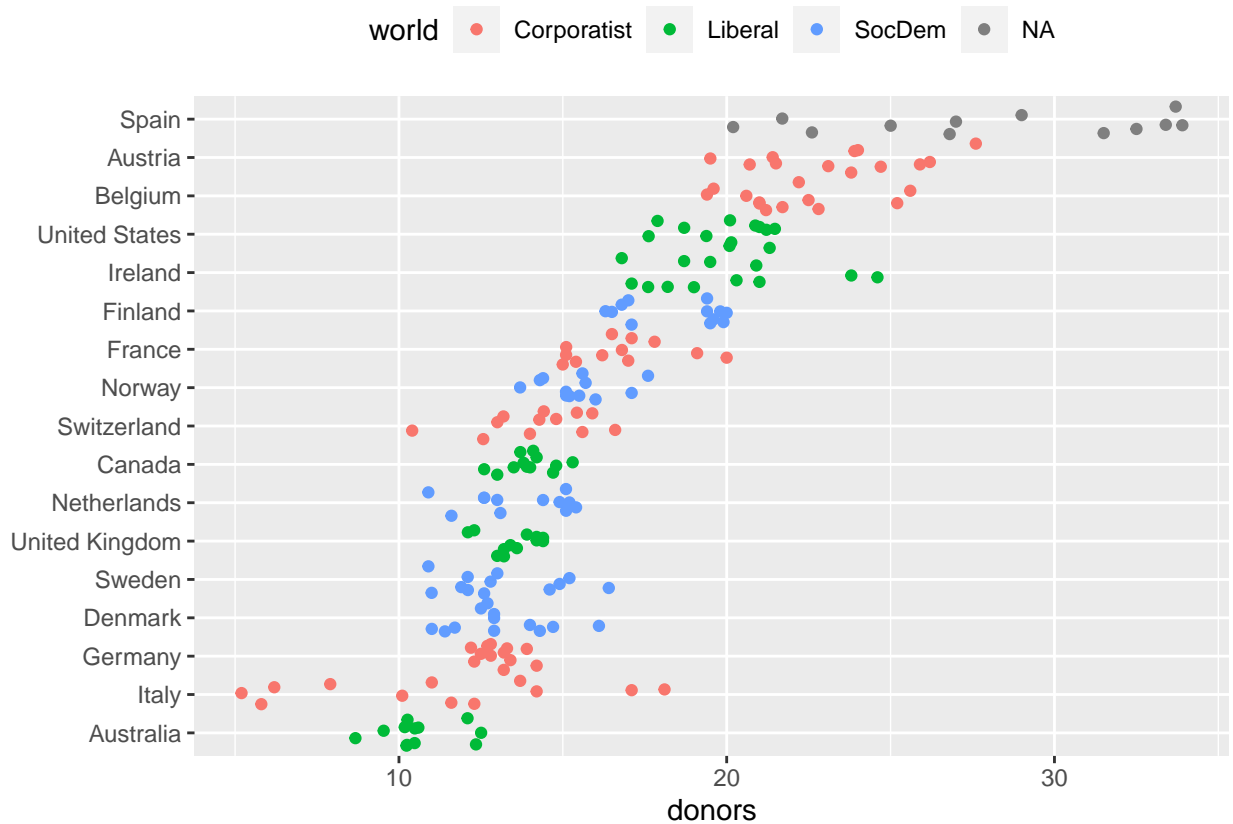


Perceba que desta forma, existe sobreposição de observações, nestes casos podemos perturbar os dados para evitar esta sobreposição através da função `geom_jitter()`:

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

p + geom_jitter() +
  labs(x = NULL) +
  coord_flip() +
  theme(legend.position = 'top')
```

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



Podemos regular o grau de perturbação de dados com o parâmetros **width** e **height**:

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

p + geom_jitter(position = position_jitter(width = 0,
                                           height = 0.5) ) +

  labs(x = NULL) +
  coord_flip() +
  theme(legend.position = 'top')
```

## Warning: Removed 34 rows containing missing values (geom\_point).

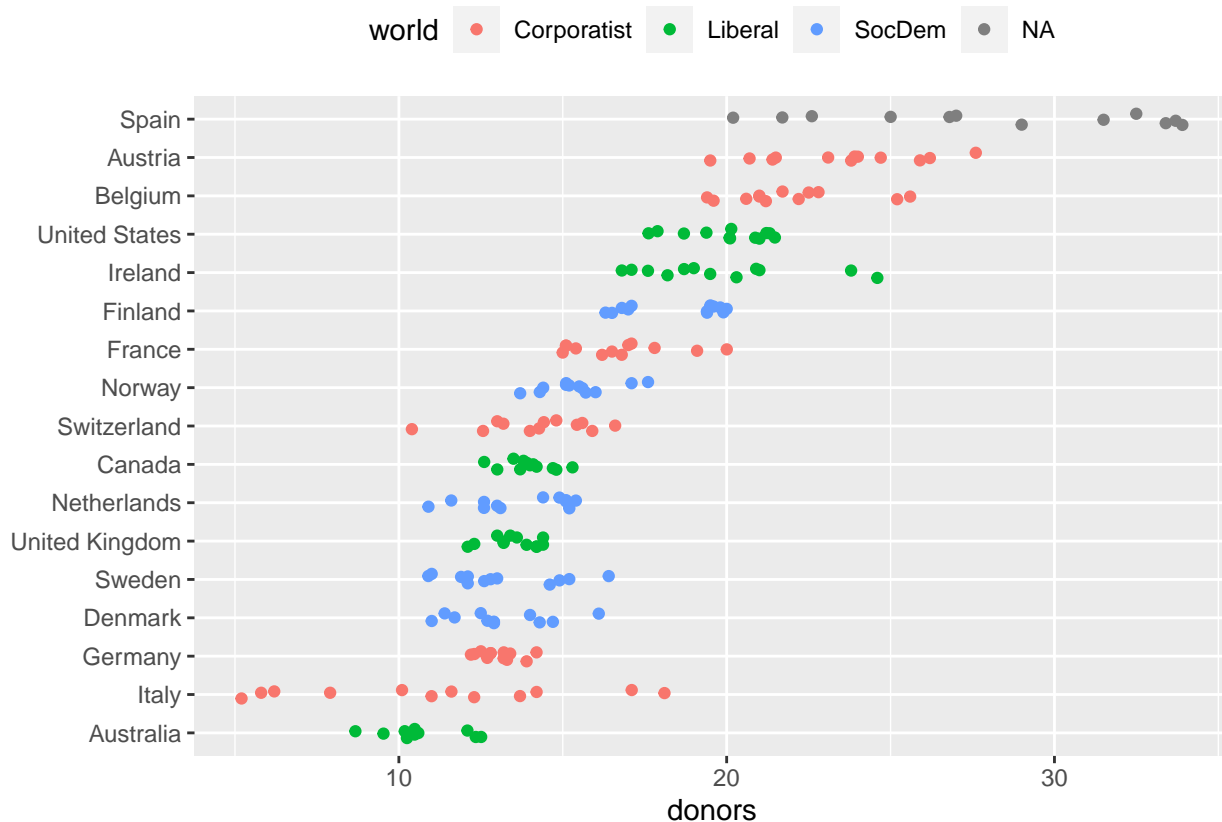


Dado o formato do gráfico, usaremos apenas o **width**, pois o **height** altera o eixo y (não esqueça que deixamos o gráfico), ou seja, altera a distribuição das observações.

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

p + geom_jitter(position = position_jitter(width = 0.15) ) +
  labs(x = NULL) +
  coord_flip() +
  theme(legend.position = 'top')
```

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



Quando queremos resumir uma variável categórica com apenas um valor por categoria, como a taxa média de doações por país, podemos usar o **Cleveland plot**.

Vamos fazer um pipeline para agregar e resumir nosso dataframe, podemos escolher manualmente as variáveis e usar repetidamente as funções de resumo **mean()** e **std()**:

```
by_country <- organdata %>%
  group_by(consent_law, country) %>%
  summarize(donors_mean = mean(donors, na.rm = TRUE),
            donors_sd = sd(donors, na.rm = TRUE),
            gdp_mean = mean(gdp, na.rm = TRUE),
            health_mean = mean(health, na.rm = TRUE),
            roads_mean = mean(roads, na.rm = TRUE),
            cerebvas_mean = mean(cerebvas, na.rm = TRUE))
```

## 'summarise()' has grouped output by 'consent\_law'. You can override using the '.groups' argument.

```
by_country
```

```
## # A tibble: 17 x 8
## # Groups:   consent_law [2]
##   consent_law country    donors_mean donors_sd gdp_mean health_mean roads_mean
##   <chr>      <chr>      <dbl>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 Informed   Australia    10.6      1.14   22179.   1958.    105.
## 2 Informed   Canada      14.0      0.751  23711.   2272.    109.
## 3 Informed   Denmark     13.1      1.47   23722.   2054.    102.
```



```
## 4 Informed Germany 13.0 0.611 22163. 2349. 113.
## 5 Informed Ireland 19.8 2.48 20824. 1480. 118.
## 6 Informed Netherlands 13.7 1.55 23013. 1993. 76.1
## 7 Informed United Kin~ 13.5 0.775 21359. 1561. 67.9
## 8 Informed United Sta~ 20.0 1.33 29212. 3988. 155.
## 9 Presumed Austria 23.5 2.42 23876. 1875. 150.
## 10 Presumed Belgium 21.9 1.94 22500. 1958. 155.
## 11 Presumed Finland 18.4 1.53 21019. 1615. 93.6
## 12 Presumed France 16.8 1.60 22603. 2160. 156.
## 13 Presumed Italy 11.1 4.28 21554. 1757 122.
## 14 Presumed Norway 15.4 1.11 26448. 2217. 70.0
## 15 Presumed Spain 28.1 4.96 16933 1289. 161.
## 16 Presumed Sweden 13.1 1.75 22415. 1951. 72.3
## 17 Presumed Switzerland 14.2 1.71 27233 2776. 96.4
## # ... with 1 more variable: cerebvas_mean <dbl>
```

Ou alternativamente podemos fazer esta tarefa iterando através de recursos de programação funcional do R:

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

```
## Warning: 'funs()' was deprecated in dplyr 0.8.0.
## 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))
```

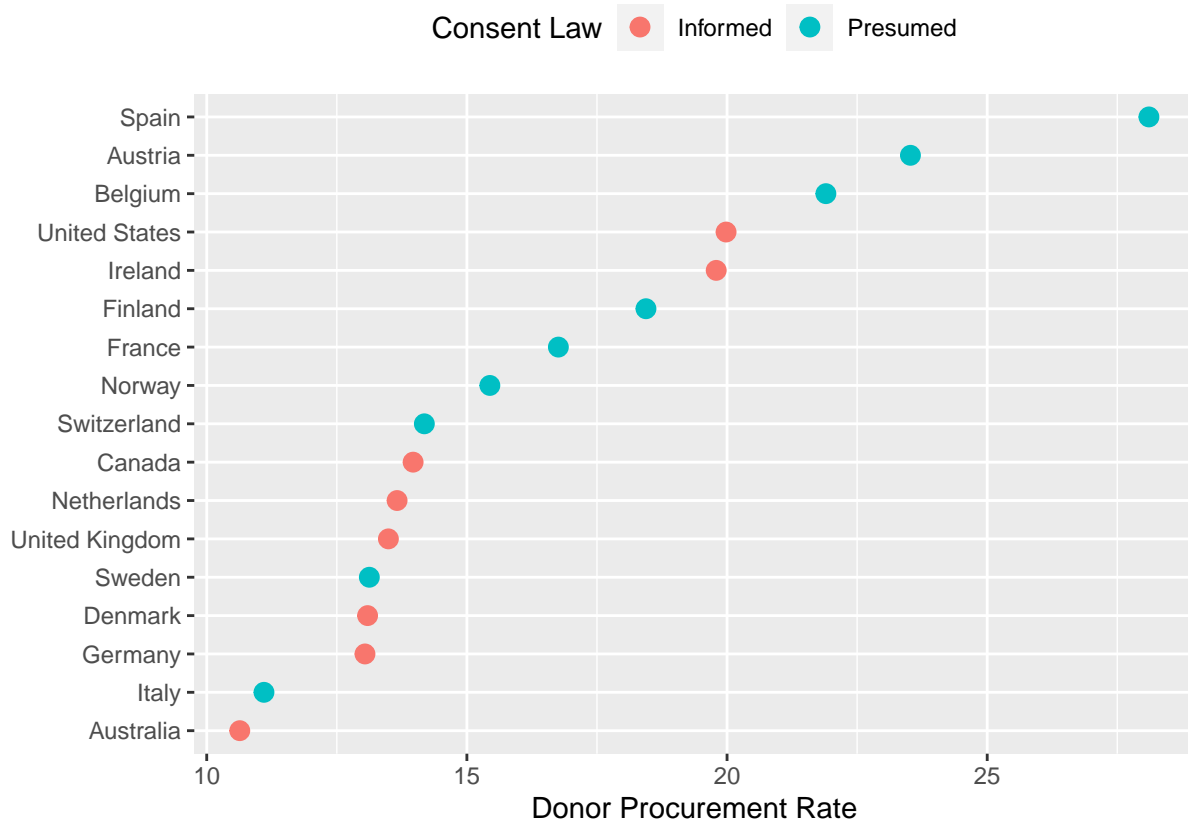
```
by_country

## # A tibble: 17 x 28
##   consent_law country donors_mean pop_mean pop_dens_mean gdp_mean gdp_lag_mean
##   <chr>      <chr>      <dbl>   <dbl>         <dbl>   <dbl>   <dbl>
## 1 Informed Austral~    10.6  18318.         0.237  22179.  21779.
## 2 Informed Canada     14.0  29608.         0.297  23711.  23353.
## 3 Informed Denmark    13.1   5257.        12.2   23722.  23275.
## 4 Informed Germany    13.0  80255.        22.5   22163.  21938.
## 5 Informed Ireland    19.8   3674.         5.23   20824.  20154.
## 6 Informed Netherl~    13.7  15548.        37.4   23013.  22554.
## 7 Informed United ~    13.5  58187.        24.0   21359.  20962.
## 8 Informed United ~    20.0 269330.         2.80   29212.  28699.
## 9 Presumed Austria    23.5   7927.         9.45   23876.  23415.
## 10 Presumed Belgium   21.9  10153.        30.7   22500.  22096.
## 11 Presumed Finland   18.4   5112.         1.51   21019.  20763.
## 12 Presumed France    16.8  58056.        10.5   22603.  22211.
## 13 Presumed Italy      11.1  57360.        19.0   21554.  21195.
```

```
## 14 Presumed      Norway           15.4    4386.           1.35    26448.       25769.
## 15 Presumed      Spain            28.1   39666.           7.84    16933.       16584.
## 16 Presumed      Sweden           13.1    8789.           1.95    22415.       22094
## 17 Presumed      Switzer~        14.2    7037.           17.0    27233.       26931.
## # ... with 21 more variables: health_mean <dbl>, health_lag_mean <dbl>,
## #   pubhealth_mean <dbl>, roads_mean <dbl>, cerebvas_mean <dbl>,
## #   assault_mean <dbl>, external_mean <dbl>, txp_pop_mean <dbl>,
## #   donors_sd <dbl>, pop_sd <dbl>, pop_dens_sd <dbl>, gdp_sd <dbl>,
## #   gdp_lag_sd <dbl>, health_sd <dbl>, health_lag_sd <dbl>, pubhealth_sd <dbl>,
## #   roads_sd <dbl>, cerebvas_sd <dbl>, assault_sd <dbl>, external_sd <dbl>,
## #   txp_pop_sd <dbl>
```

Com os dados resumidos por país, podemos fazer um **Cleveland dotplot** com os pontos coloridos conforme a variável categórica **consent\_law**.

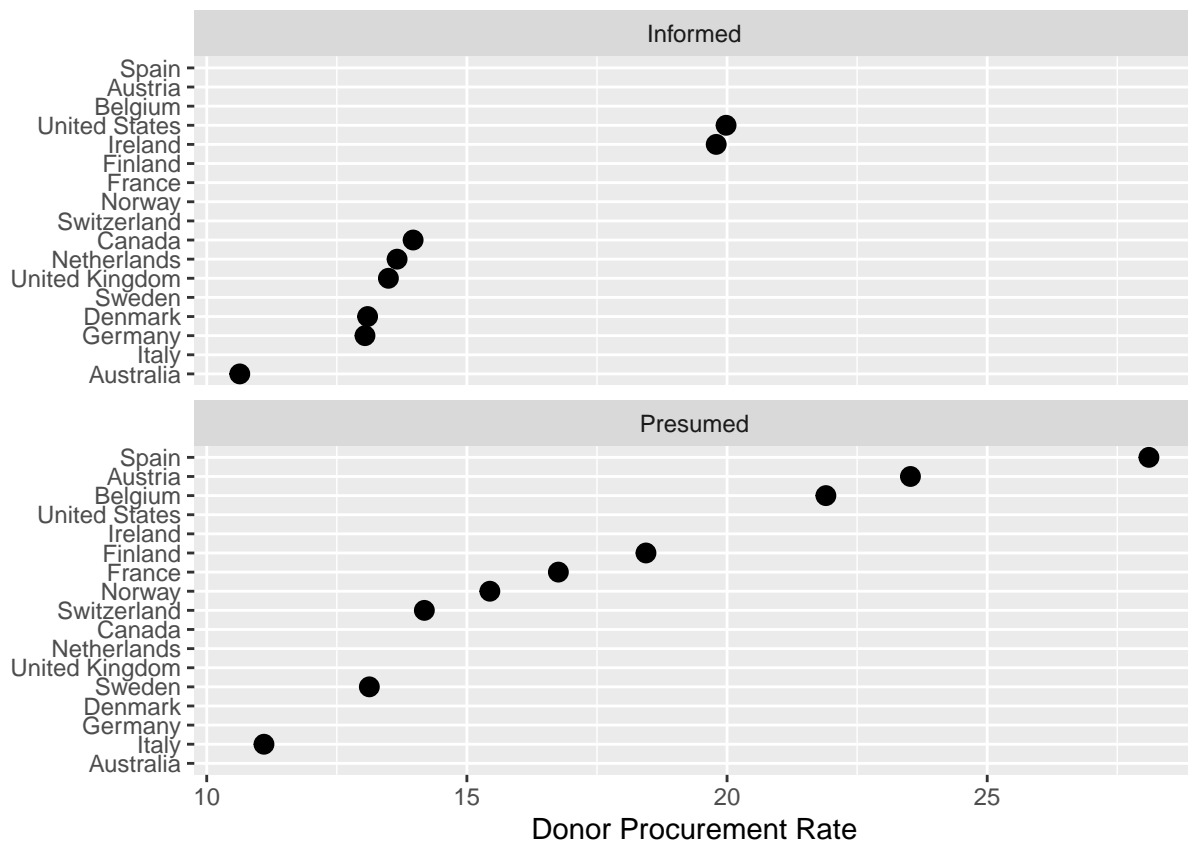
```
p <- ggplot(data = by_country,
           mapping = aes(x = donors_mean,
                        y = reorder(country, donors_mean),
                        color = consent_law))
p + geom_point(size=3) +
  labs(x = "Donor Procurement Rate",
       y = "", color = "Consent Law") +
  theme(legend.position="top")
```



Alternativamente podemos plotar separadamente pela categoria **consent\_law** em vez de colorir. Colocaremos os gráficos empilhados, **ncol = 1**, para melhor compará-los dado que são da mesma escala.

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

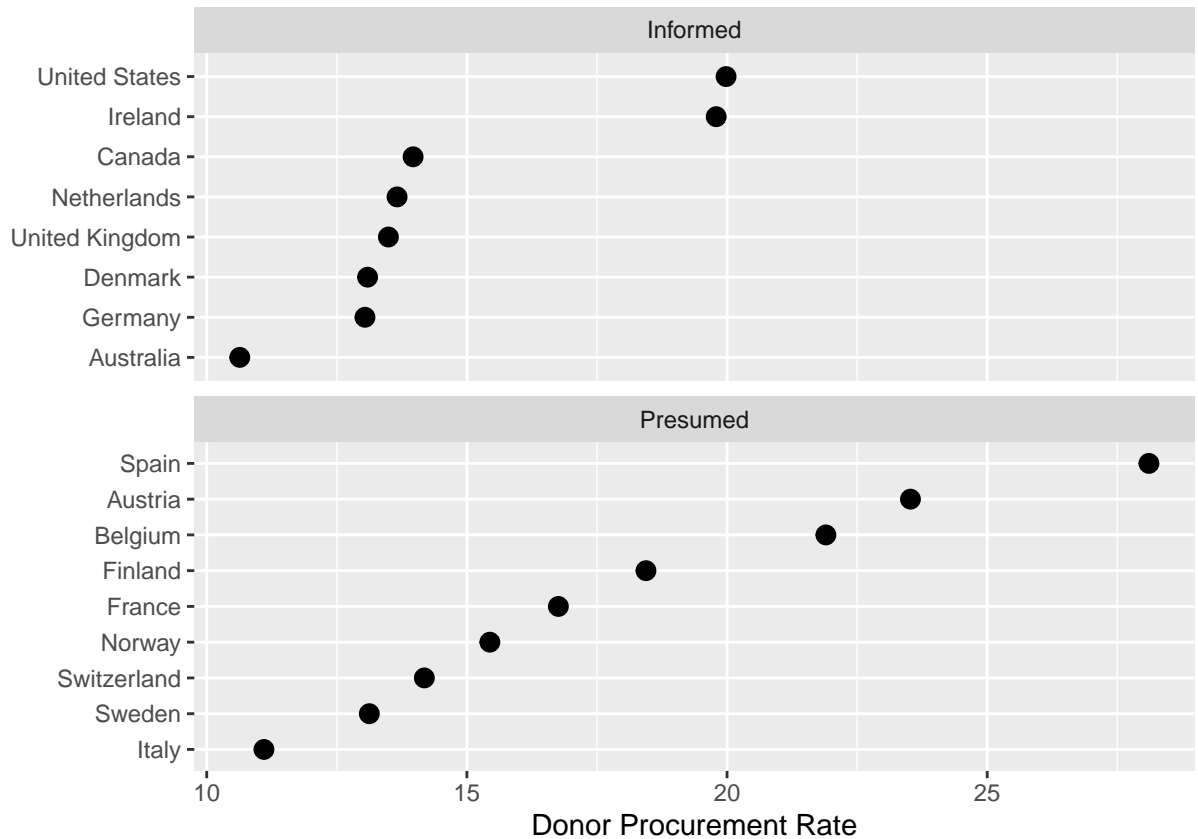
p + geom_point(size=3) +
  facet_wrap(~ consent_law, ncol = 1) +
  labs(x = "Donor Procurement Rate",
       y = "")
```



Note que pelo eixo y ser categórico, ele vai plotar o nome todos os países nos 2 plots, então para evitar isso colocaremos `scales = "free_y"`.

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

p + geom_point(size=3) +
  facet_wrap(~ consent_law, scales = "free_y", ncol = 1) +
  labs(x = "Donor Procurement Rate",
       y = "")
```

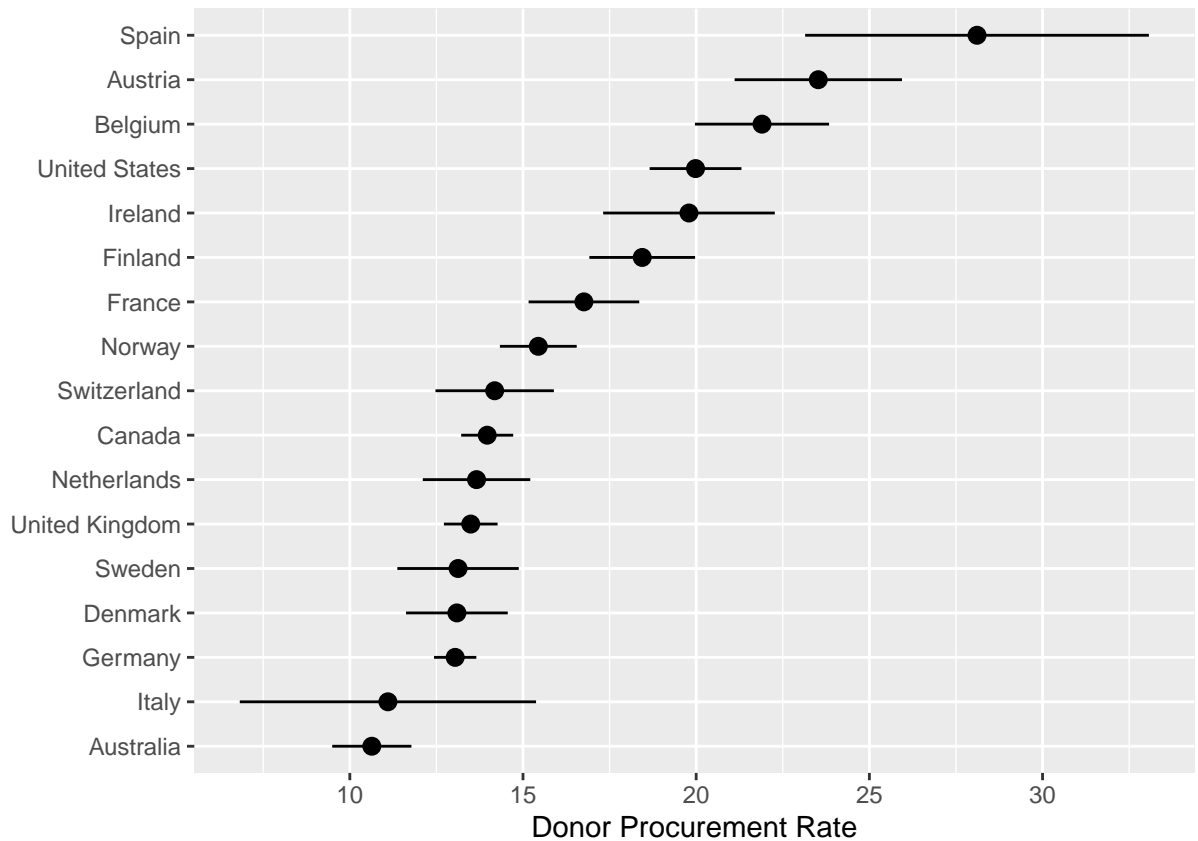


**Cleveland plots** é um ótimo recurso de resumir resultados de dados que possuem erros de range. Para estender este tipo de plot e incluir medidas de variância ou erro utilizamos o `geom_pointrange`.

Este precisa do range em que se encontram os dados, `ymin` e `ymax`:

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

p + geom_pointrange(mapping = aes(ymin = donors_mean - donors_sd,
                                  ymax = donors_mean + donors_sd)) +
  labs(x= "", y= "Donor Procurement Rate") +
  coord_flip()
```

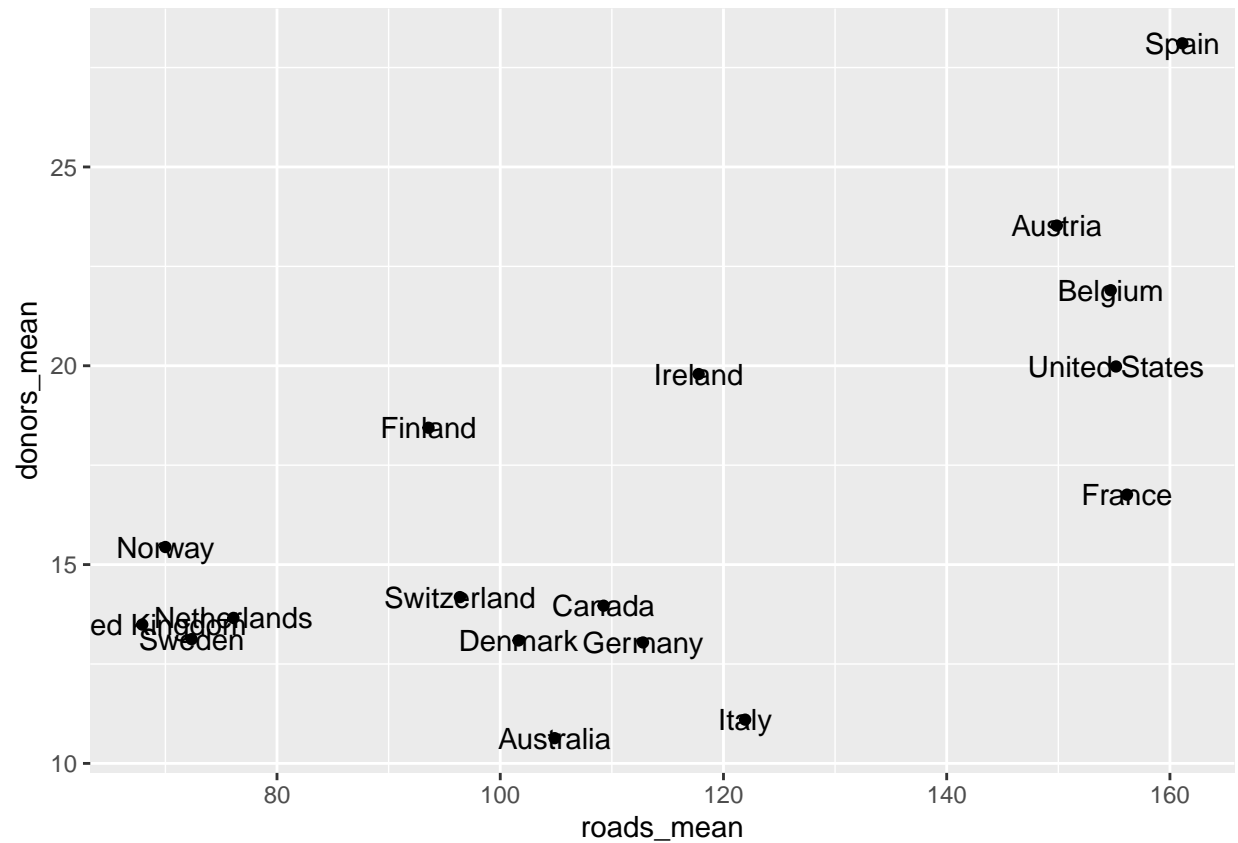


## Adicionando texto

Pode ser útil as vezes plotar os labels dos pontos diretamente, podemos fazer isso através da função `geom_text()`

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

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

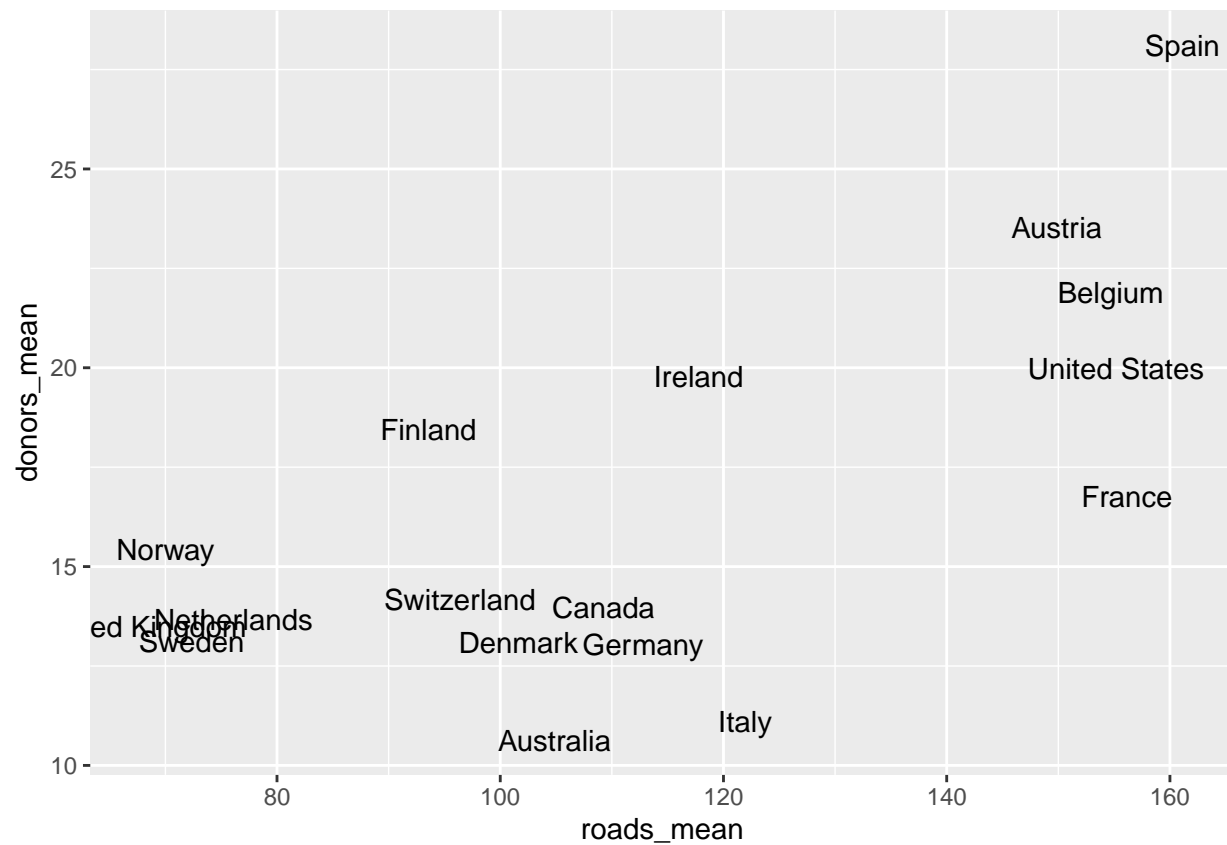


Como estamos utilizando o mesmo **mapping** para os pontos e labels, eles ficam sobrepostos.

Para corrigir isso podemos retirar os pontos e manter os labels:

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

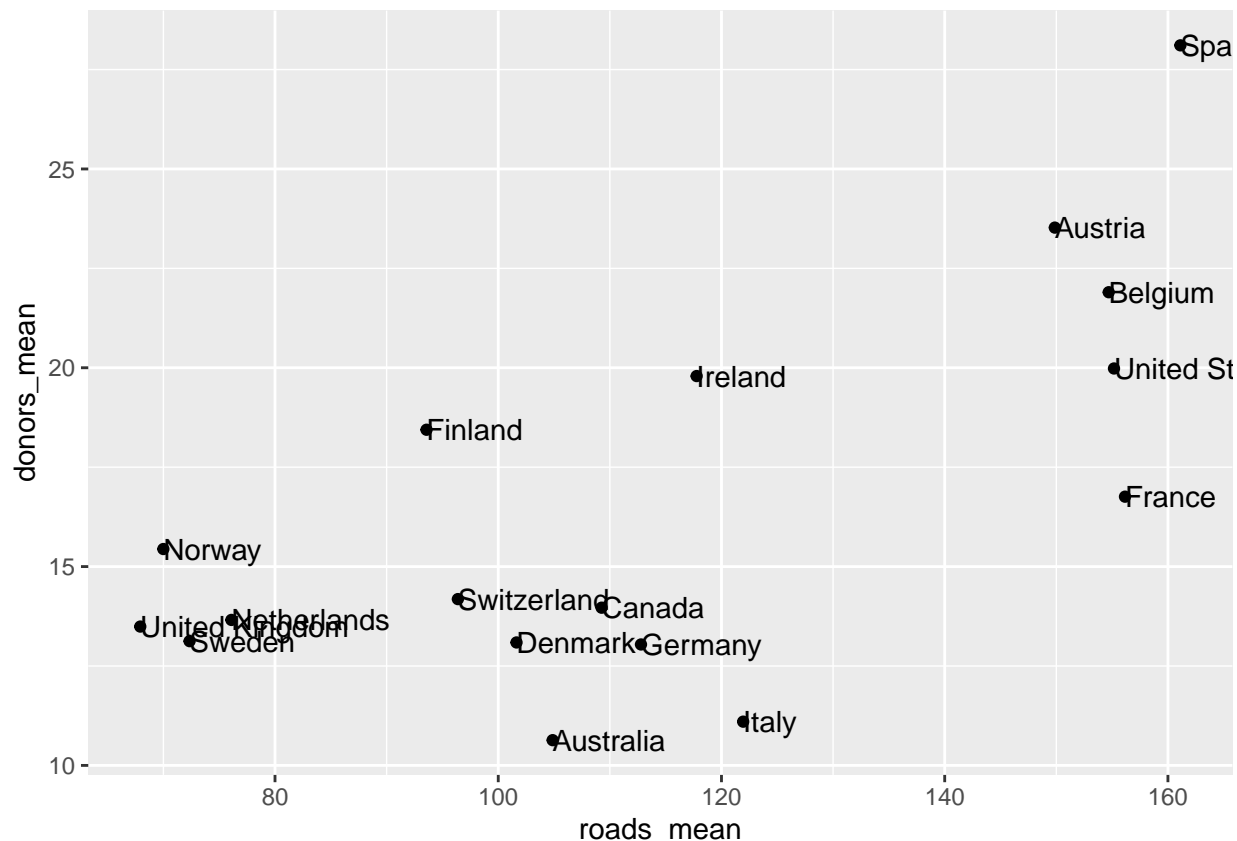
p + geom_text(mapping = aes(label = country))
```



Ou alternativamente é ajustar o label, utilizando `hjust = 0` para deslocar para a esquerda e `hjust = 1` para a direita.

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

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



Perceba que por mais que modifiquemos o ajuste do label, não parece uma abordagem robusta. Em vez de usar `geom_text()` vamos usar as funções `geom_text_repel()` e `geom_label_repel()` da biblioteca `ggrepel`.

```
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.0.5
```

Vamos utilizar o dataset `elections_historic` descrito neste site

```
elections_historic %>% select(2:7)
```

```
## # A tibble: 49 x 6
##   year winner      win_party ec_pct popular_pct popular_margin
##   <int> <chr>      <chr>      <dbl>      <dbl>          <dbl>
## 1 1824 John Quincy Adams D.-R.      0.322      0.309        -0.104
## 2 1828 Andrew Jackson  Dem.      0.682      0.559         0.122
## 3 1832 Andrew Jackson  Dem.      0.766      0.547         0.178
## 4 1836 Martin Van Buren Dem.      0.578      0.508         0.142
## 5 1840 William Henry Harrison Whig      0.796      0.529         0.0605
## 6 1844 James Polk      Dem.      0.618      0.495         0.0145
## 7 1848 Zachary Taylor  Whig      0.562      0.473         0.0479
## 8 1852 Franklin Pierce Dem.      0.858      0.508         0.0695
## 9 1856 James Buchanan  Dem.      0.588      0.453         0.122
## 10 1860 Abraham Lincoln Rep.      0.594      0.396         0.101
## # ... with 39 more rows
```



A figura a seguir plota cada eleição presidencial Americana desde 1824, primeiro ano que foi computado o voto popular. Em que os eixos são a parcela de votos no vencedor e a parcela de votos do vencedor que são populares.

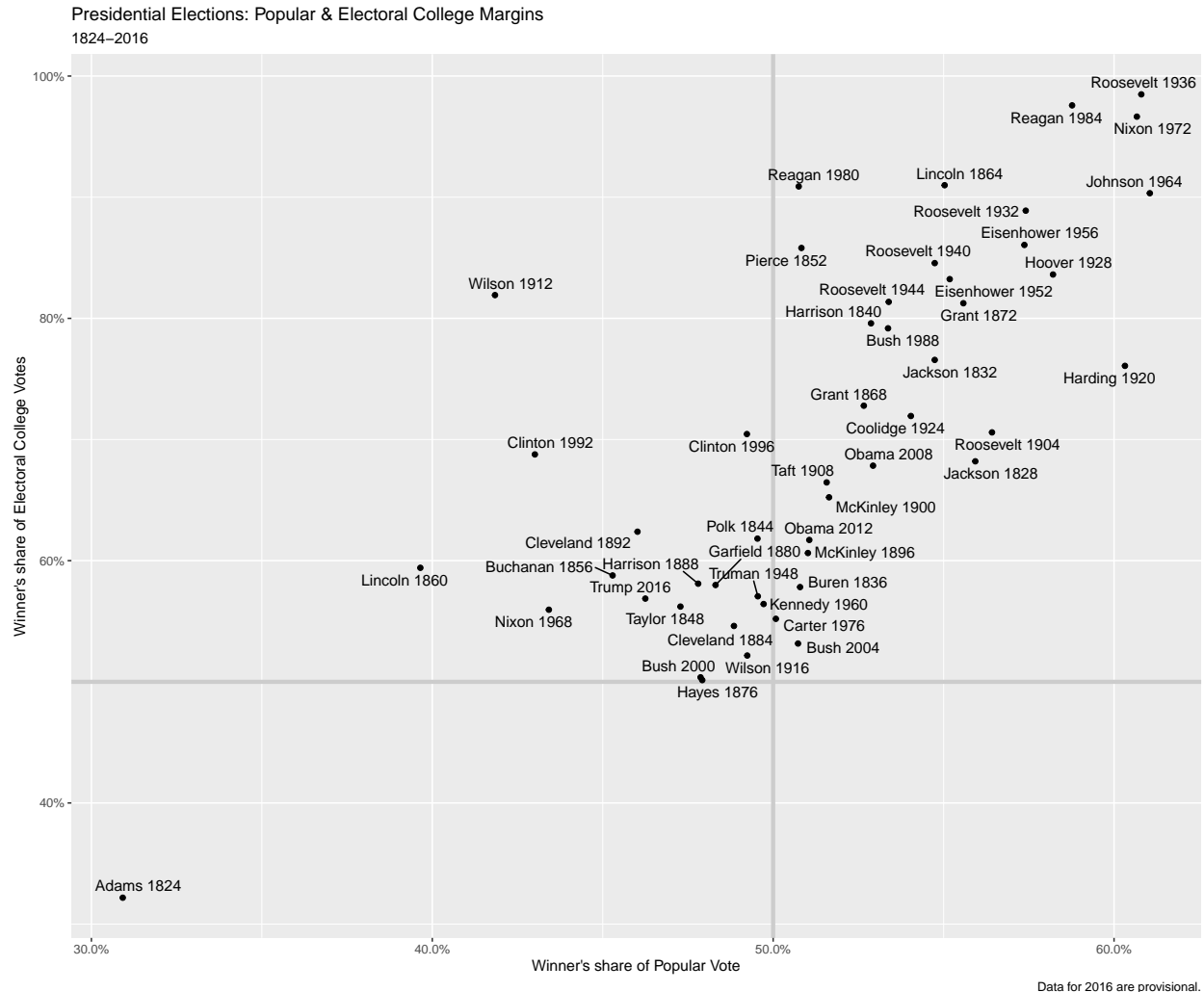
Estando interessados em certas presidências, vamos adicionar labels aos pontos. Porém para evitar sobreposição, como discutido anteriormente, vamos utilizar a função `geom_text_repel()`.

```
p_title <- "Presidential Elections: Popular & Electoral College Margins"
p_subtitle <- "1824-2016"
p_caption <- "Data for 2016 are provisional."

x_label <- "Winner's share of Popular Vote"
y_label <- "Winner's share of Electoral College Votes"

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 = x_label,
       y = y_label,
       title = p_title,
       subtitle = p_subtitle,
       caption = p_caption)
```



Como os dados dos eixos estão em proporção, de 0 a 1, utilizamos a função `scale_x_continuous()` e `scale_y_continuous()` para transformá-los em porcentagens.

Adicionaremos retas que marcam 50% de votos no vencedor e 50% de votos do vencedor que são populares para melhor analisar o distanciamento de cada ponto destas retas.

## Texto condicional

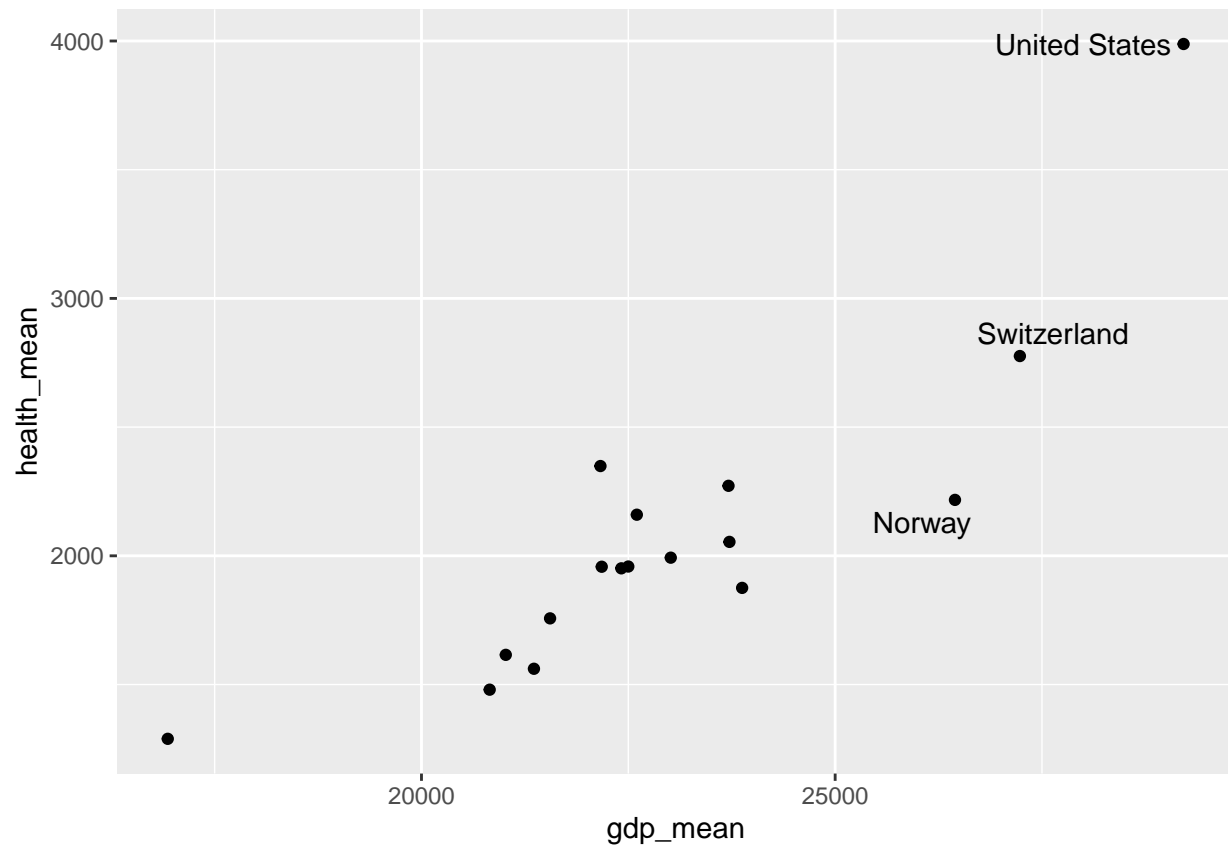
Perceba que as vezes não é uma boa ideia nomear todos os pontos do plot, mas sim escolher alguns de interesse para adicionar seus labels. Podemos fazer isso ainda com as mesmas funções.

Basta que passemos um dataframe apenas com os pontos selecionados para a função `geom_text_repel()` através da ferramenta `subset()`. Como podemos ver a seguir:

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

p + geom_point() +
  geom_text_repel(data = subset(by_country,
```

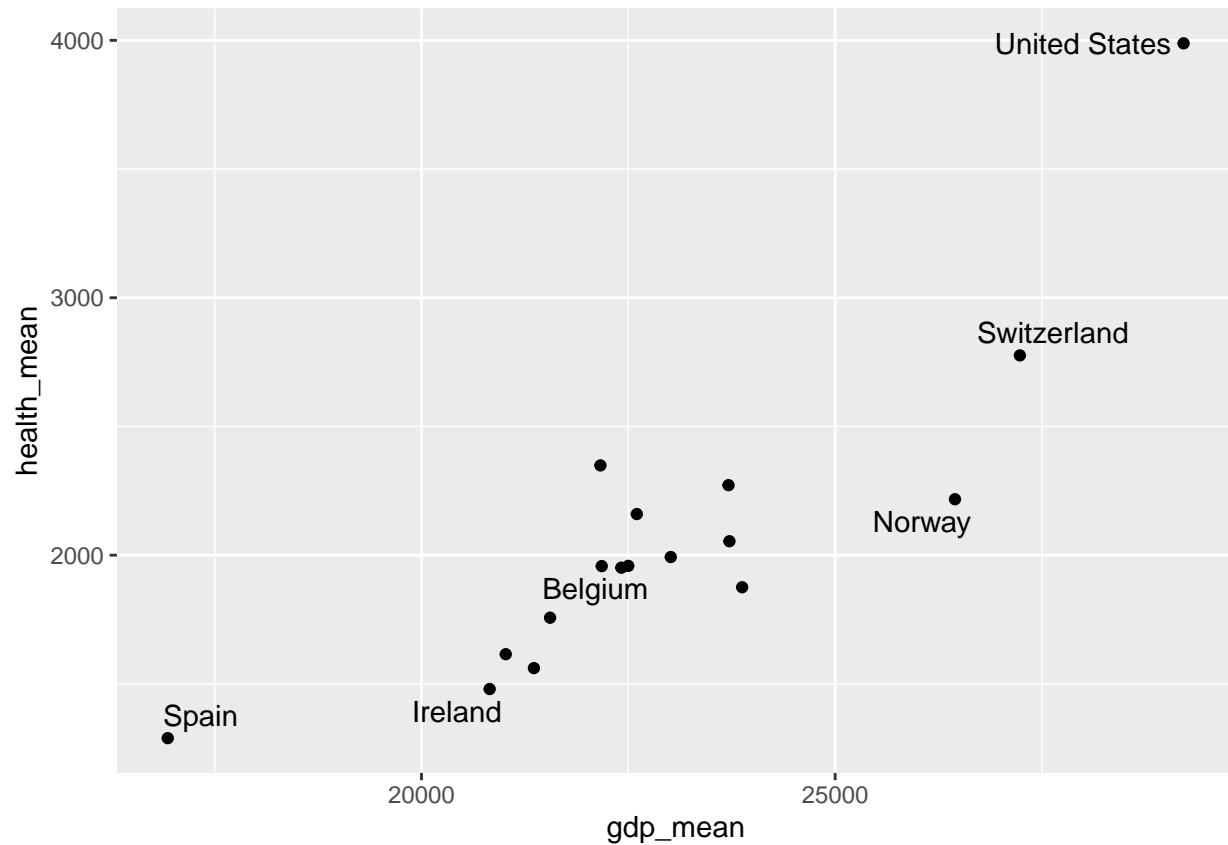
```
gdp_mean > 25000),
mapping = aes(label = country))
```



Neste caso utilizamos um critério apenas, para mais critérios é de modo similar:

```
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 |
                                health_mean < 1500 |
                                country %in% "Belgium"),
mapping = aes(label = country))
```



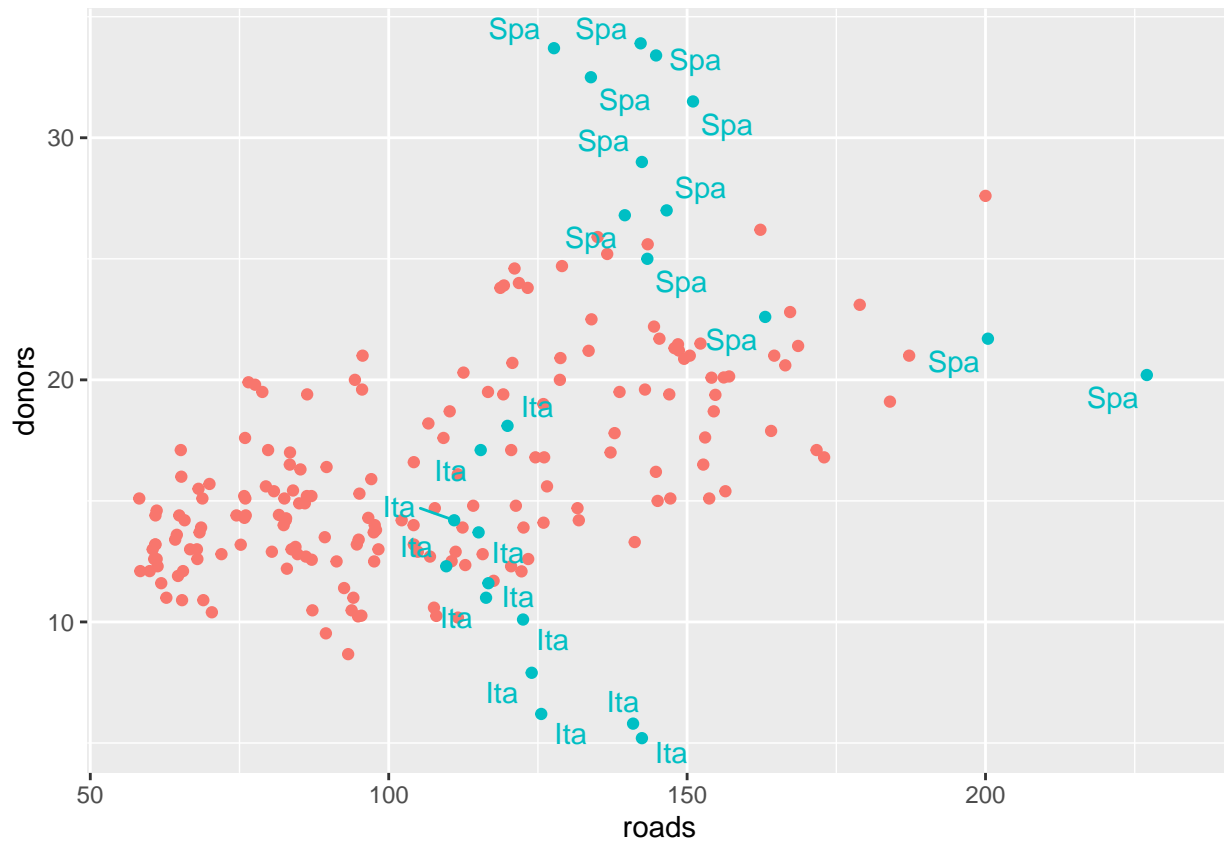
Alternativamente, podemos selecionar pontos específicos criando uma variável auxiliar para essa tarefa, no caso abaixo criamos a coluna **ind**.

```
organdata$ind <- organdata$ccode %in%
  c("Ita", "Spa") & organdata$year > 1998

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

p + geom_point() +
  geom_text_repel(data = subset(organdata, ind),
    mapping = aes(label = ccode)) +
  guides(label = FALSE,
    color = FALSE)
```

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



Utilizamos a coluna auxiliar tanto no filtro de labels quanto no parâmetro **color**. Além disso, para suprimir a legenda faz-se uso da função **guides()**.

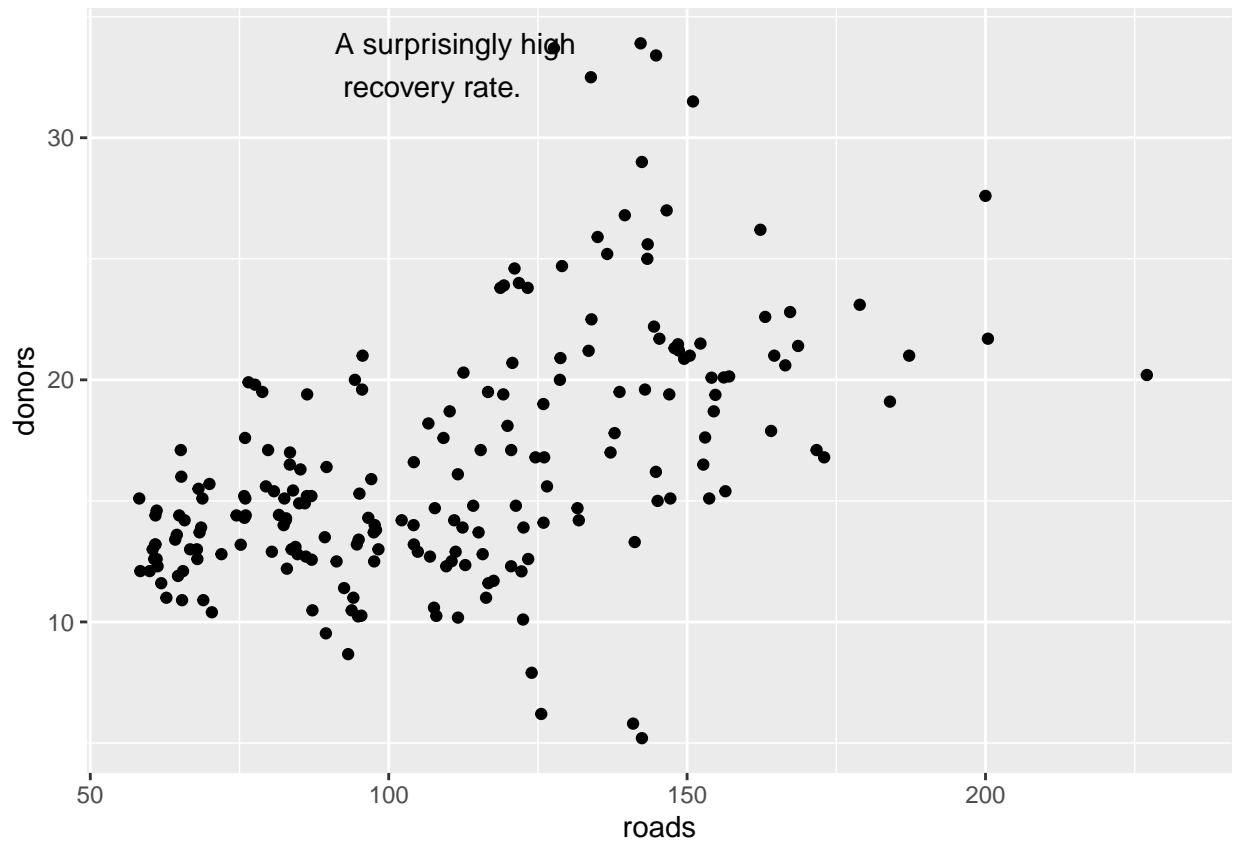
## Escrevendo e desenhando no Gráfico

Por vezes se faz necessário fazer anotações diretamente no gráfico para fazer alguma observação. Utilizaremos a função **annotate()** e diremos para ela utilizar os argumentos da **geom\_text()** através do parâmetro **geom**. Como pode-se ver no exemplo a seguir:

```
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)
```

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

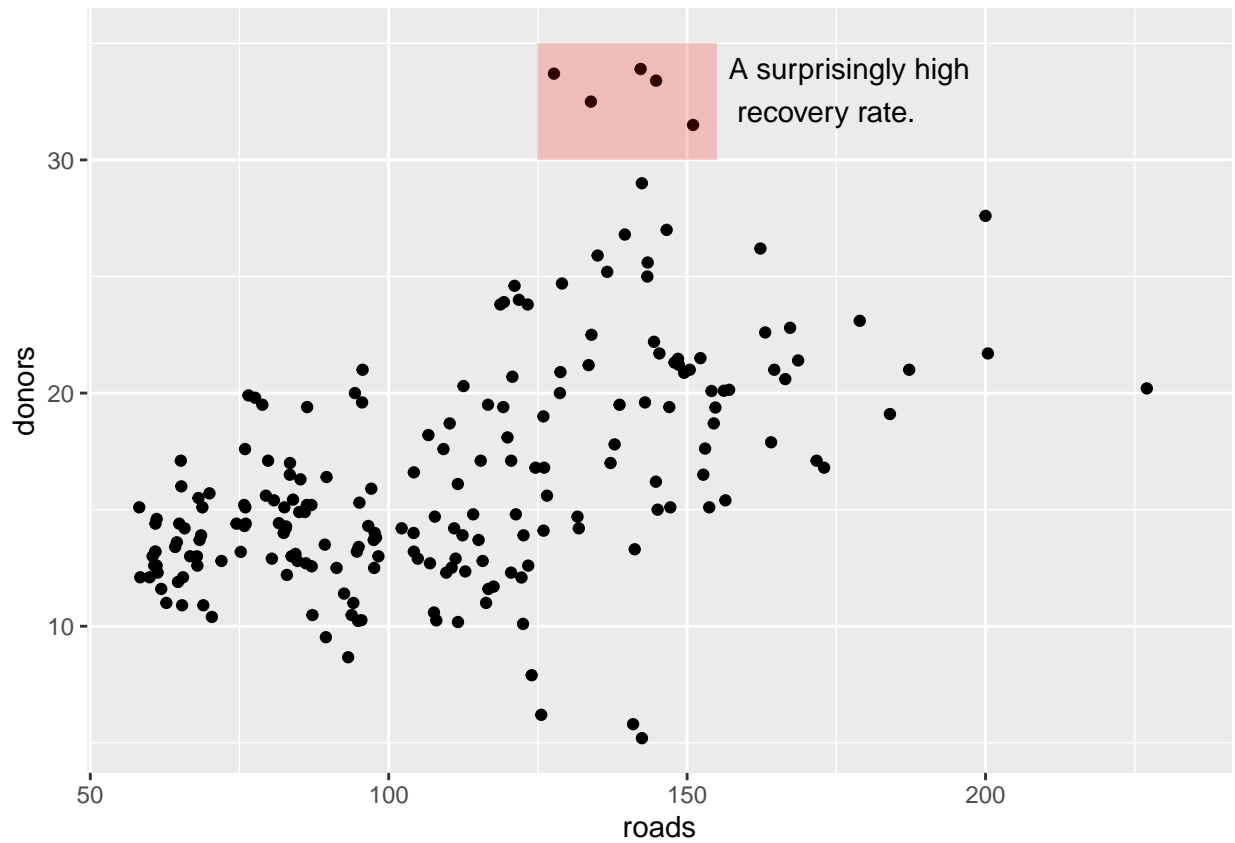


Podemos fazer uso dos outros geoms, e desenhar retângulos, retas e setas. Vamos adicionar um retângulo para resaltar a quais pontos nossa observação se refere:

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

p + geom_point() +
  annotate(geom = "rect",
          xmin = 125, xmax = 155,
          ymin = 30, ymax = 35,
          fill = "red", alpha = 0.2) +
  annotate(geom = "text",
          x = 157, y = 33,
          label = "A surprisingly high \n recovery rate.",
          hjust = 0)
```

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

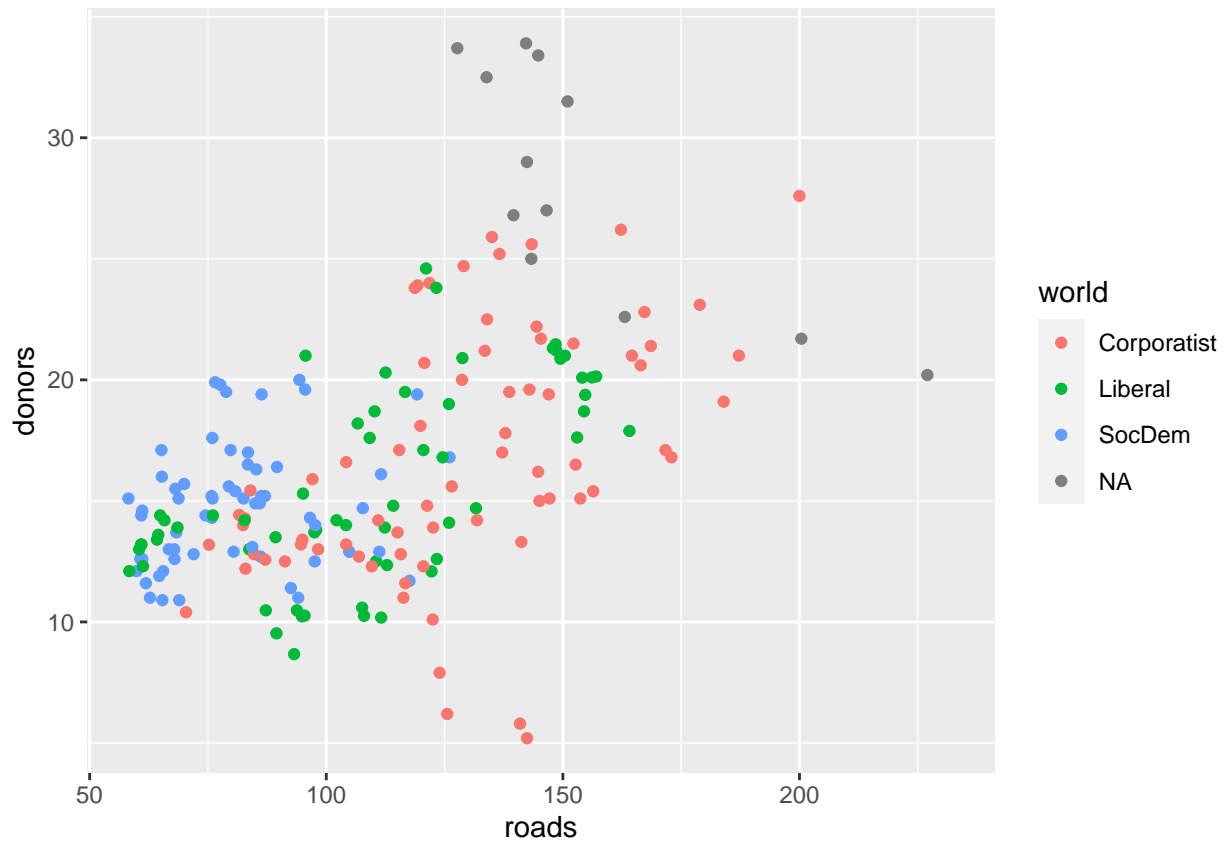


## Escalas, guias e Temas

A seguir temos um gráfico simples de dispersão:

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

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



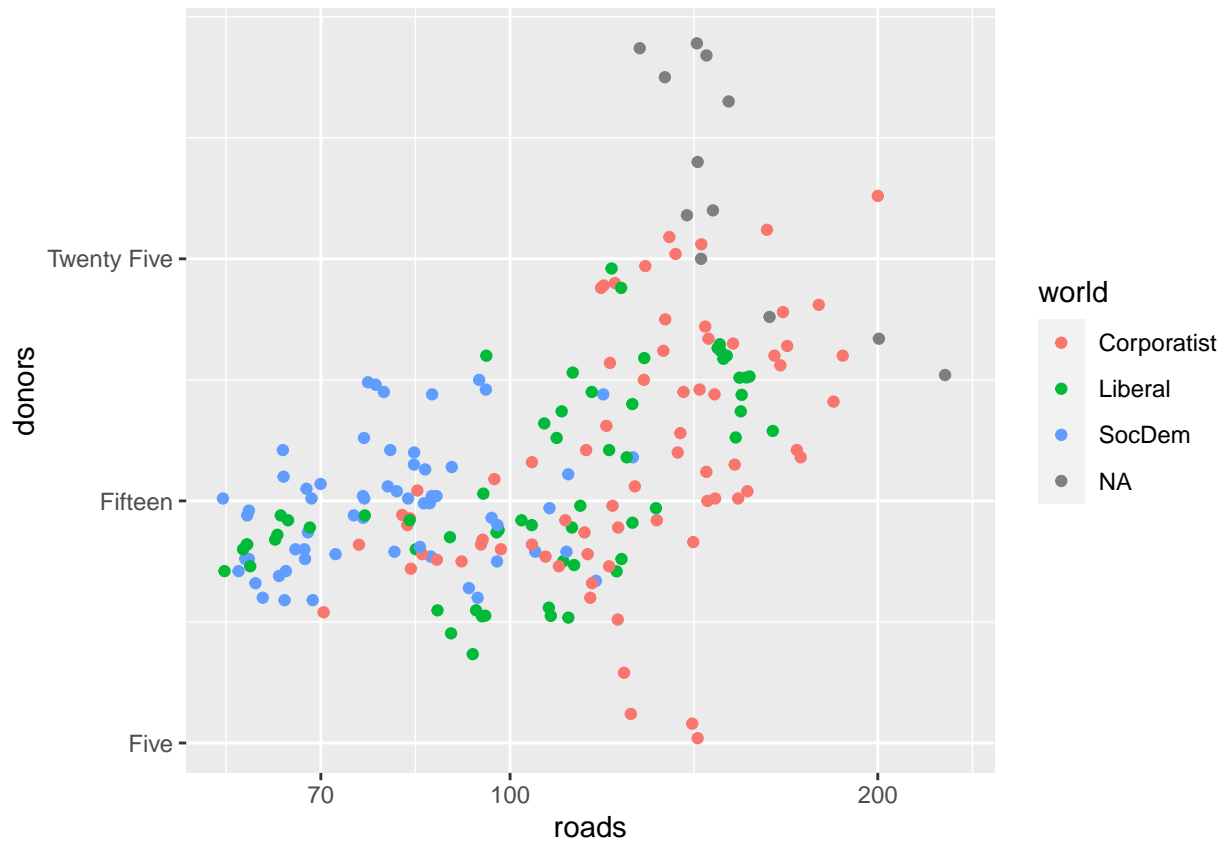
Podemos mudar a escala do eixo através de uma função do tipo `scale_mapping_kind()`, além de mudar as escalas podemos definir as posições e os labels para as marcações do eixo. Como podemos ver a seguir:

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

p + geom_point() +
  scale_x_log10() +
  scale_y_continuous(breaks = c(5, 15, 25),
                     labels = c("Five",
                                "Fifteen",
                                "Twenty Five"))
```

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

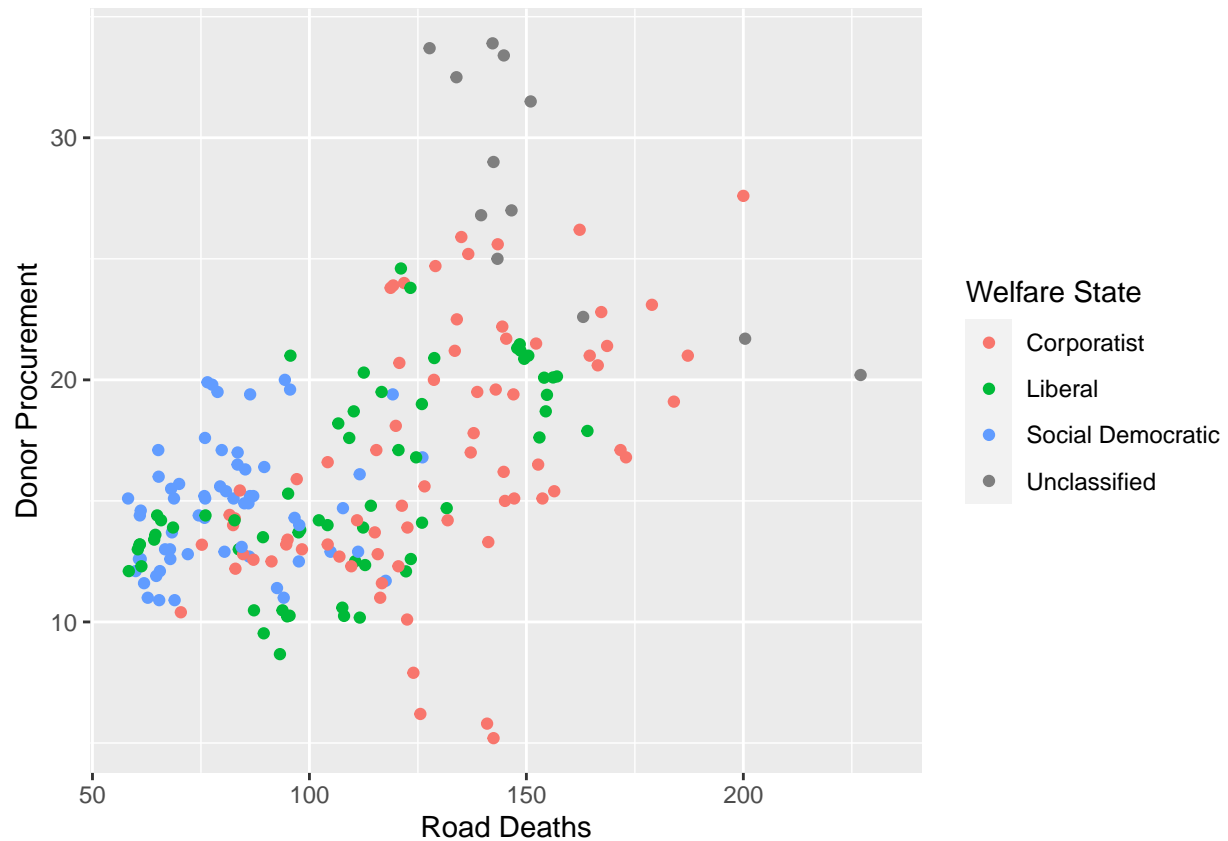




Porém, se quisermos mudar os labels da legenda ou dos eixos em si, faremos isso utilizando a função **labs()** :

```
p <- ggplot(data = organdata,
            mapping = aes(x = roads,
                          y = donors,
                          color = world))
p + geom_point() +
  scale_color_discrete(labels = c("Corporatist",
                                  "Liberal",
                                  "Social Democratic",
                                  "Unclassified")) +
  labs(x = "Road Deaths",
       y = "Donor Procurement",
       color = "Welfare State")
```

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

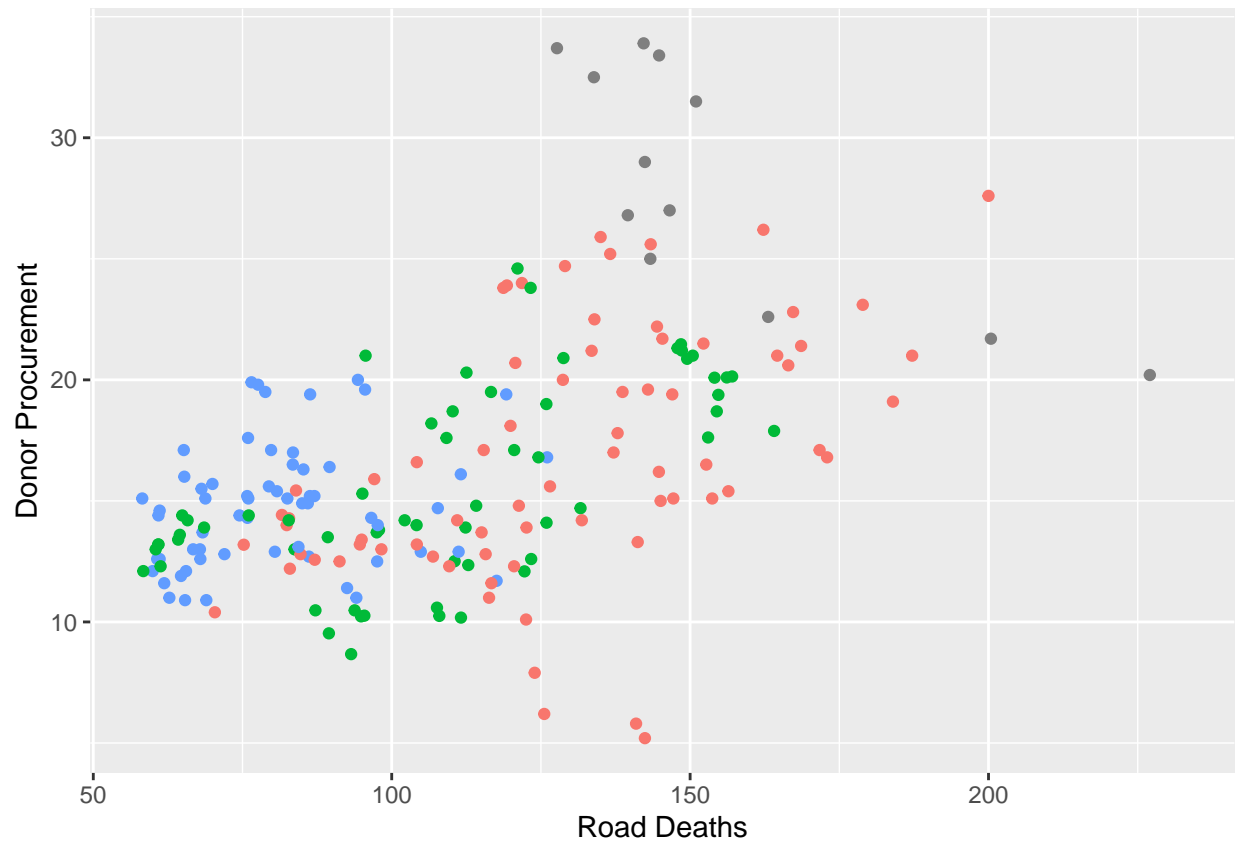


Por fim, para o caso de não quisermos a legenda, podemos utilizar o `guides()`.

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

p + geom_point() +
  labs(x = "Road Deaths",
       y = "Donor Procurement") +
  guides(color = FALSE)
```

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



## Revisitando um Gráfico

Vamos agora aplicar o que foi discutido no capítulo revisitando um gráfico do trabalho “Casas para Alugar” :

```
library(ggExtra)

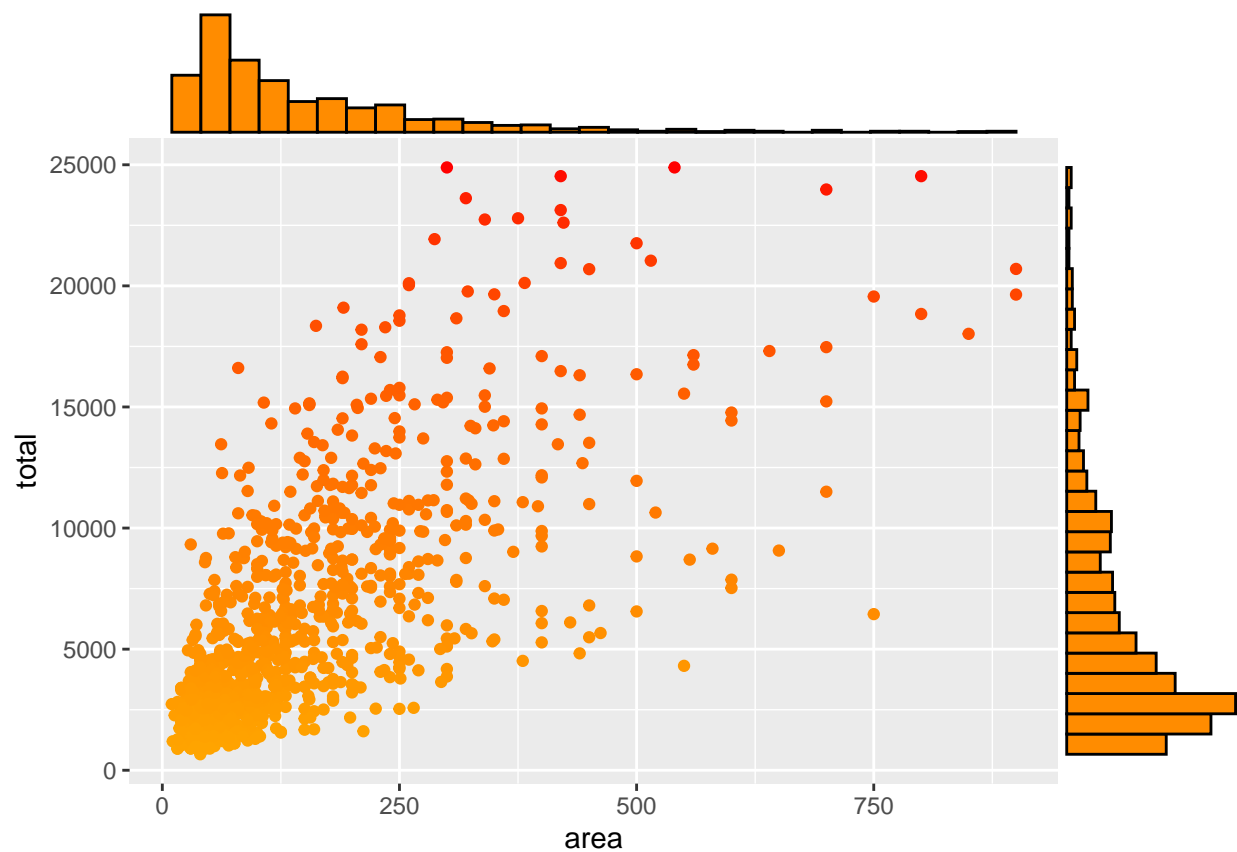
Tabela <- read.csv('../Moradia/Dados limpos')
```

Trata-se de um gráfico de dispersão da metragem do imóvel pelo valor total para se alugar o imóvel:

```
graf <- Tabela %>%
  filter( !is.na(Tabela) ) %>%
  ggplot() +
  geom_point(mapping = aes(x = area, y = total, color = total)) +
  scale_color_gradient(low = "orange", high = "red") +
  theme(legend.position="none")

graf <- ggMarginal(graf, type="histogram", fill = "darkorange")

show(graf)
```

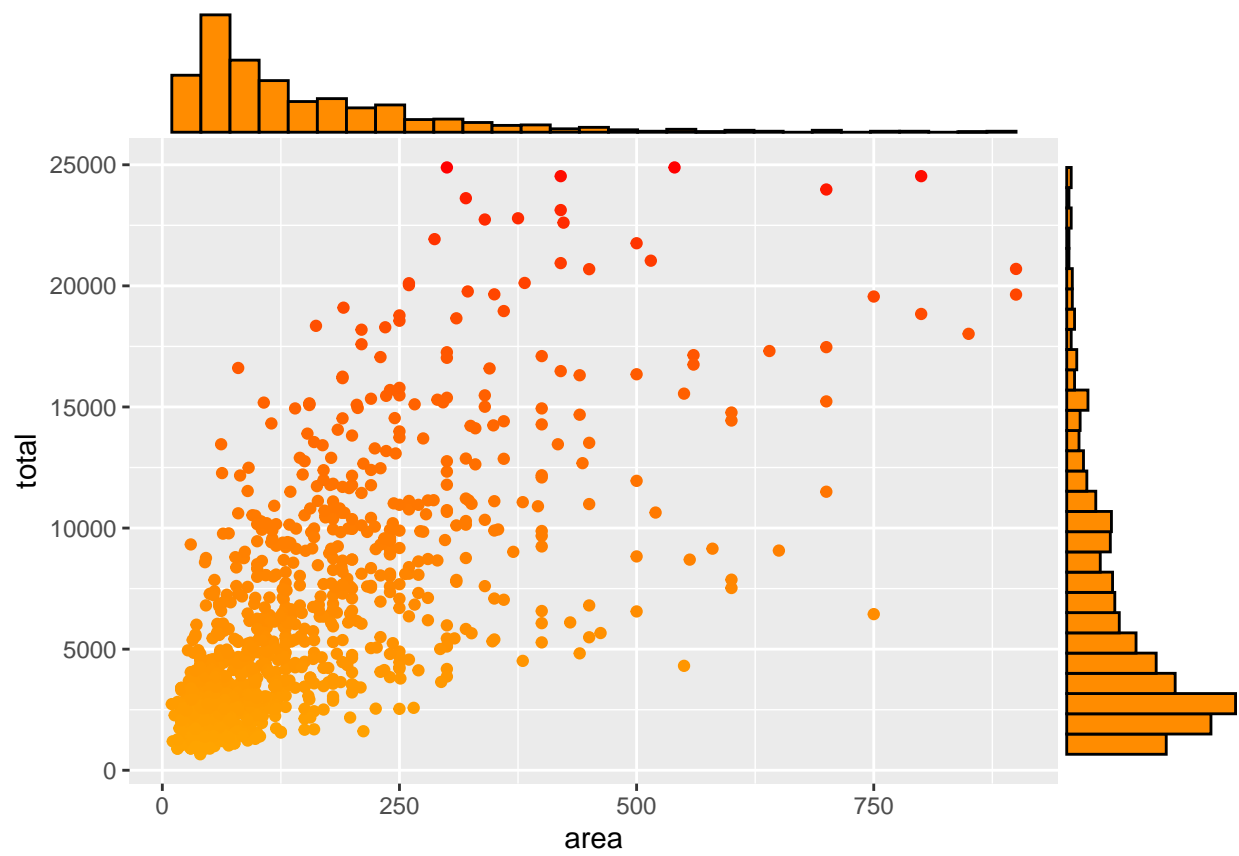


Podemos omitir a legenda de modo mais elegante utilizando a função **guides()**,

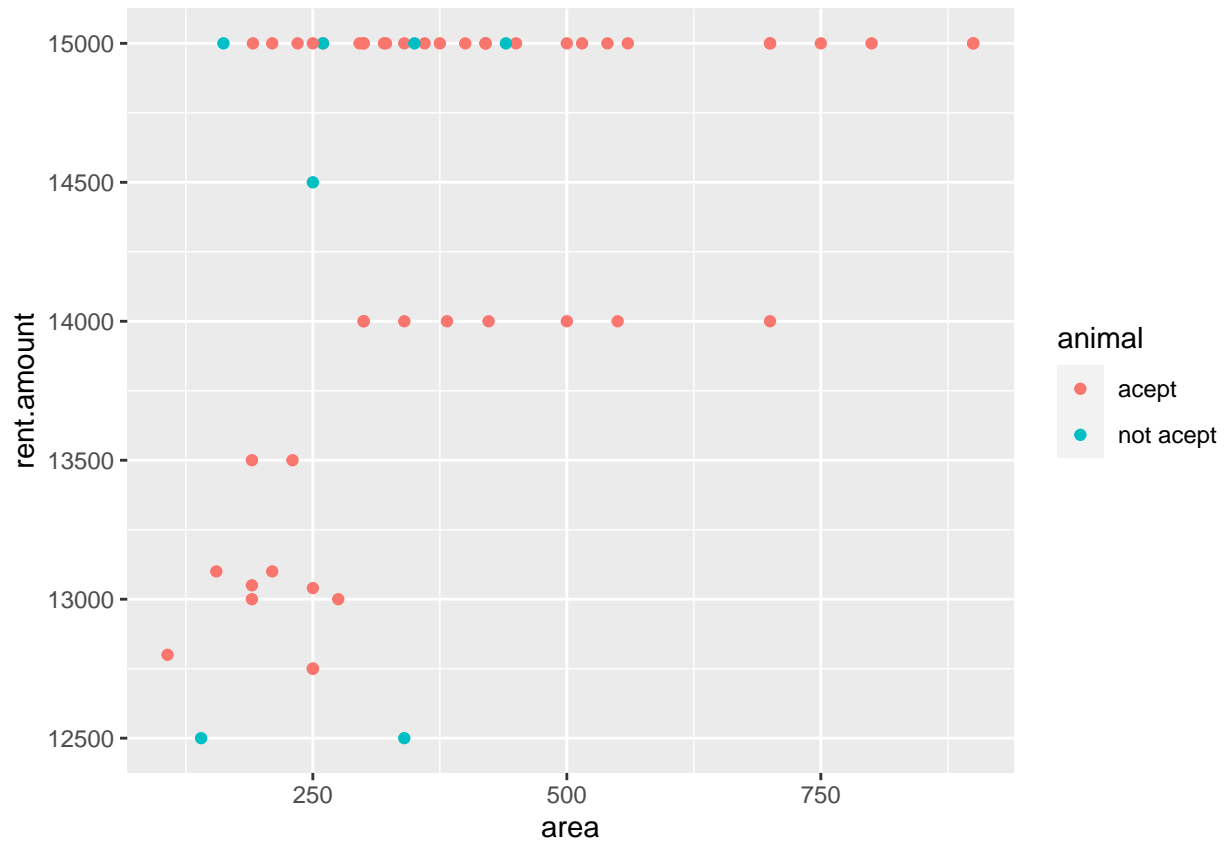
```
graf <- Tabela %>%
  filter( !is.na(Tabela) ) %>%
  ggplot() +
  geom_point(mapping = aes(x = area, y = total, color = total)) +
  scale_color_gradient(low = "orange", high = "red") +
  guides(color = FALSE)

graf <- ggMarginal(graf, type="histogram", fill = "darkorange")

show(graf)
```



```
Tabela %>% filter(rent.amount > 12000) %>%
  ggplot(aes(x = area, y = rent.amount, colour = animal)) + geom_point()
```



Adicionei desenhos e um texto no gráfico diretamente utilizando **annotate()**:

```
Tabela %>% filter(rent.amount > 12000) %>%
  ggplot(aes(x = area, y = rent.amount, colour = animal)) +
  geom_point() +
  annotate(geom = "rect",
    xmin = 125, xmax = 910,
    ymin = 15100, ymax = 14900,
    fill = "yellow", alpha = 0.2) +
  annotate(geom = "rect",
    xmin = 250, xmax = 750,
    ymin = 14100, ymax = 13900,
    fill = "yellow", alpha = 0.2) +
  annotate(geom = "text",
    x = 350, y = 14500,
    label = "Observe as faixas de valores \n invariantes à metragem do imóvel",
    hjust = 0)
```

