Getting Started

*Utilizarei o símbolo > para indicar o input do console e >>> para o output
*É possível salvar os scripts para serem usados depois.

-"The greatest value of a picture is when it forces us to notice what we never expected to see."
- É importante notar a visualização de dados antes do processamento deles, pois nela é possível notar coisas que seriam muito difíceis em dados brutos.

DATA TYPES

-Categoricals (ex. sexo, região), podem ser divididades em ordinals(nivel de pimenta[fraco, médio or quente]) e non-ordinals(ex. Estados).
&
-Numericals, que podem ser divididades em discrete (números redondos, como tamanho populacional) ou continuous (qualquer valor ex. 68.12 inches).

Dicas (linguagem: "*English")*:
**Reserve the therm "ordinal data" for variables belonging to a small number of different groups with each group having many members.*
**Reserve the therm "discrete numerical variables" for variables belonging to a large number of groups with few cases in each group.*

unique()
-mostra os valores únicos de dados
ex.
> *length(unique(y))* //mostra a quantidade de valores únicos

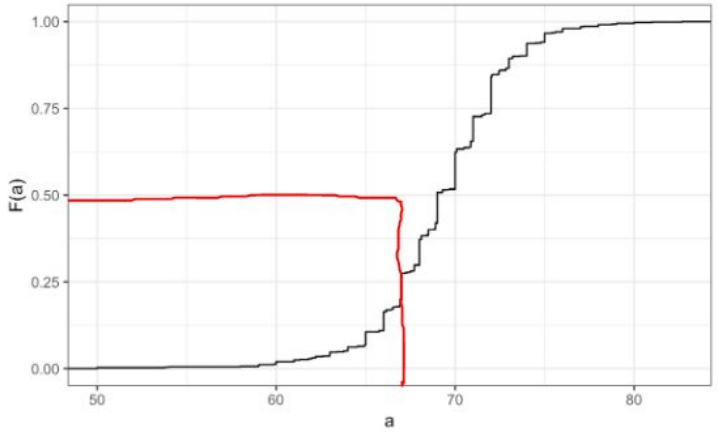
DISTRIBUIÇÕES

-Valor médio (*average*)
    mean()
-Mediana (*median*)
    median()
-Desvio absoluto da mediana (*standard deviation*)
    mad()
    *Outliers tem um efeito menor na mediana que no valor médio.
-Standard Deviation (desvio padrão) //diferença entre o valor médio
    sd()

Cumulative distribution function (CDF)

F(a) = Pr(x<=a)

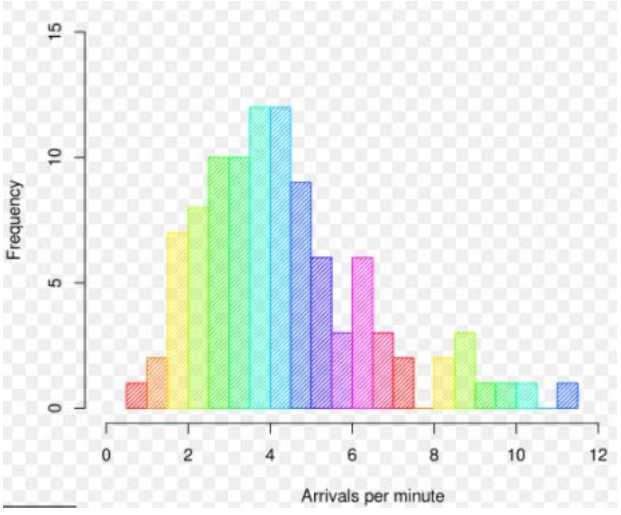
-Define uma distribuição de *numerical data* com uma função definida para reportar a proporação de dados acima de um valor A para todos os números positivos de A.
-A proporção de valores entre duas alturas pode ser dada por F(b) - F(a)




```
> a <- seq(min(my_data), max(my_data), length = 100)     //define uma faixa de valor
> cdf_function <- function(x) {                          //prob. de um valor único
>    mean(my_data <= x)
> }
> cdf_values <- sapply(a, cdf_function)
> plot(a, cdf_values)
```

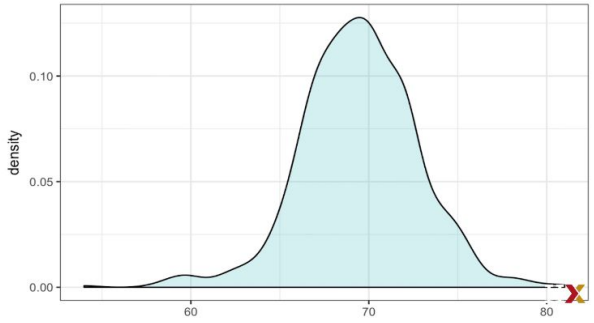
HISTOGRAMS

-Sacrificam um pouco de dados para que se tenha uma visualização mais fácil de ser interpretada.
-Utilizam intervalos

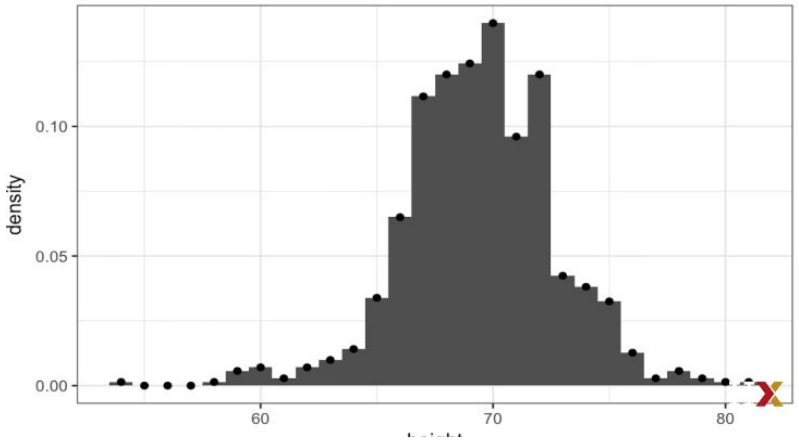


SMOOTH DENSITY PLOTS

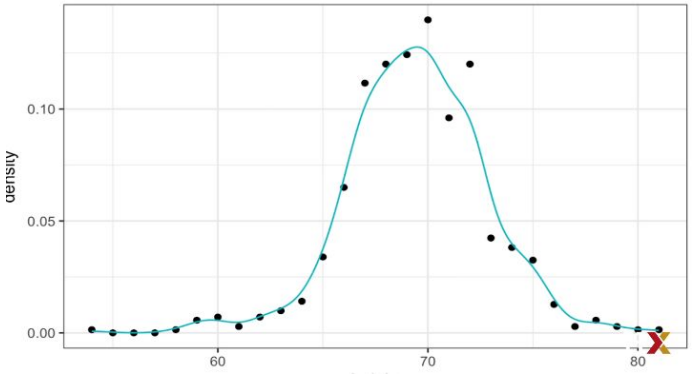
-Similares ao *histograms* mas com estética “melhorada”.
-Gráfico vem de uma grande quantidade de dados não observados.



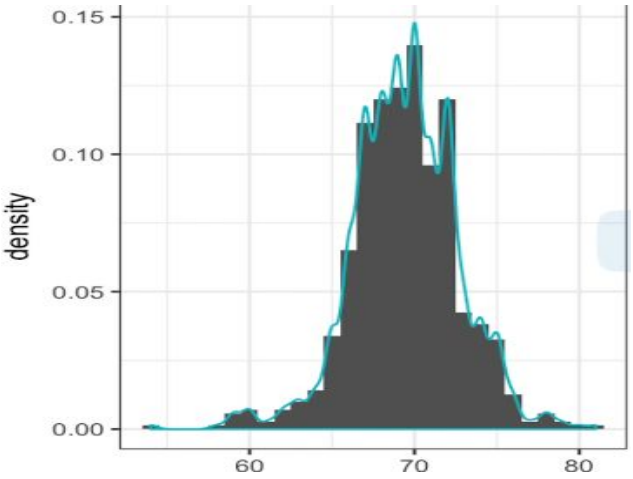
(I) Para fazê-lo, marca-se os pontos de um histogram:



(II) E traça-se as curvas:



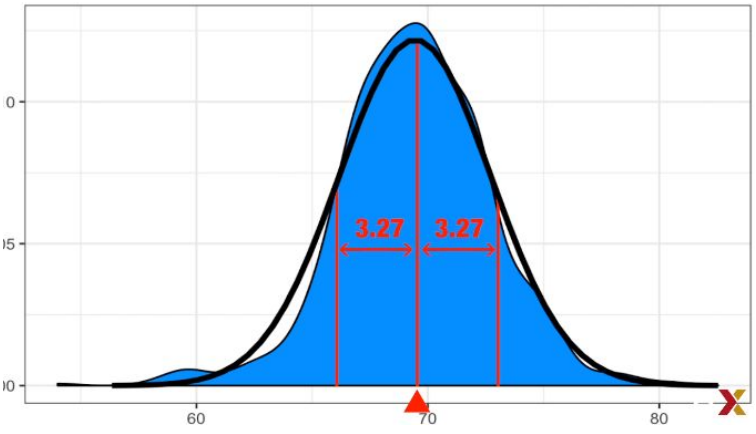
*Obs: Este seria um gráfico mais denteado e não “*smooth*”



NORMAL DISTRIBUTION

$$\Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi}s} e^{-\frac{1}{2}\left(\frac{x-m}{s}\right)^2} dx$$

Sendo “m” para mean e “s” para standard deviation
Sendo z = (x-m)/u
*Quando z=0, a distribuição normal está no máximo (na média m)
*A distribuição normal “z-scores” ou standard normal distribution acontece quando m=0 e s=1.



$$z = (x - \text{average})/\text{SD}$$

Para obter standard units em R:

scale()
> z <- scale(x)
> mean(abs(z) < 2) //Números de z's menores que 2 e maiores que -2
>>> 0.95 //95% dos dados

THE NORMAL CDF AND PNORM

-A distribuição cumulativa para a distribuição normal pode ser feita em R por:
pnorm()
pnorm(a, avg, s)
*onde a é o valor, avg a média e s o desvio padrão.
Ou seja, para saber a probabilidade de um valor (ex. 70.5) ser selecionado sem utilizar todo o dataset, temos:
> library(tidyverse)
> library(dslabs)
> data(heights)
> x <- heights %>% filter(sex=="Male") %>% pull(height)
> 1 - pnorm(70.5, mean(x), sd(x)) //sd = standard deviation
*lembrando que a aproximação normal não é uma boa prática para números que não apresentam um inteiro nele (ex. 0.2216)

QUANTILES

-Divide um dataset em intervalos, a q% (*quantile*) é o valor em que as observações do dataset são iguais ou menores que esse valor.
quantile(data, q)

PERCENTILES

São os *quantiles* que dividem o dataset em 100 intervalos de 1%.
ex.
> p <- seq(0.01, 1.00, 0.01) //sequencia de 0.01 a 1.00 aumentado-se em 0.01
> quantiles(data, p)

ex2.
> library(dslabs)
> data(heights)
> summary(heights$height)
> p <- seq(0.01, 0.99, 0.01)
> percentiles <- quantile(heights$height, p)

> percentiles[names(percentiles) == "25%"]
> percentiles[names(percentiles) == "75%"]

QUARTILE

Dividem o dataset em 4 partes de 25% (25, 50, 75).
Sendo:

- 25: primeiro quartil
- 50: médio quartil
- 75: terceiro quartil

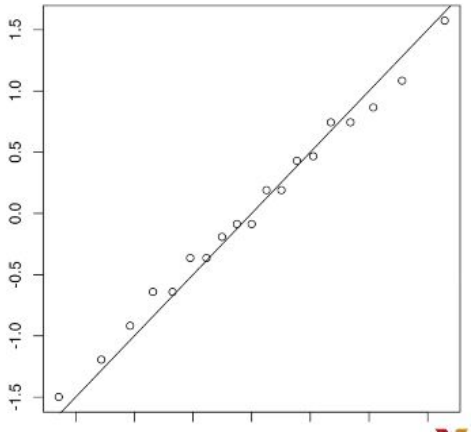
-A função summary() retorna o minimo, quartis e maximo de um vetor.

QNORM

-A função qnorm() nos da o valor teórico que um quartil tem de probabilidade de “p” apresentar um valor igual ou menor que o valor dado com a média “mu” e o desvio padrão “sigma”.
qnorm(p, mu, sigma)
-Por padrão, mu = 0 e sigma = 1. //qnorm(p)
*qnorm() e pnorm() são funções inversas:
pnorm(-1.96) ≈ 0.025
qnorm(0.025) ≈ −1.96

QUANTILE-QUANTILE PLOTS

-Ou q-q plots
> observed_quantiles <- quantile(z, p)
> theoretical_quantiles <- qnorm(p)
> plot(theoretical_quantiles, observed_quantiles)
> abline(0,1)



GGPLOT

GGPLOT2

-Bom pacote para iniciantes;

-Trabalha com tabela de dados com linhas sendo obersevações e colunas como variáveis.

> library(tidyverse)

-"COLA" para utilizar o ggplot2
https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

GRAPH COMPONENTS

-Há algumas importantes a se notar:

- A tabela de US murders está resumida (**data component**).
- O gráfico é um scatter plot (**geometry component**).
- O mapeamento x e y, cores e texto (**aesthetic component**).
- Está em escala de log (**scale component)**.
- Labels, título, legenda etc.

CREATING A NEW PLOT

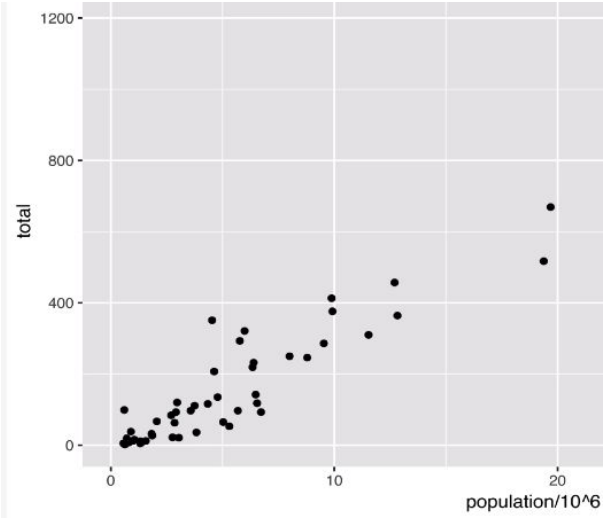
-O primeiro passo é criar um ggplot object.
ex.
> ggplot(data = murders)
ex2.
> muders %>% ggplot()

-Associar o gráfico a um objeto.
> p <- ggplot(data = murders)
> class(p)
>>> ggplot
> p //plot será exibido

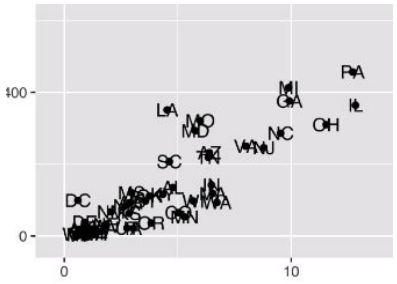
LAYERS

+
ex.
DATA %>% ggplot() + LAYER 1 + LAYER 2 + … + LAYER n
ex.
geom_point() //geom_"NomeQueLembreOVisual"
-Para saber o que se usará no geom_"NAME"() :
> ?geom_point
>>> x y alpha colour

ex. de scatterplot
> murders %>% ggplot() +
geom_point(aes(x = population/10^6, y = total))



geom_label() //adiciona o label com um retângulo
geom_text() //adiciona o texto
> p + geom_point(aes(population/10^6, total)) +
geom_text(aes(population/10^6, total, label = abb))



TINKERING

-Ler a help file para os geom_"name" para saber como customizá-los;
-Além disso, pode-se ver os argumentos de gg plot com:
args(ggplot)

SCALE, LABELS AND COLOURS

scale_x_continuous

```
> p + geom_point(size = 3) +
>     geom_text(nudge_x = 0.05) +          //nudge separa um pouco o ponto do nome
>     scale_x_continuous(trans = "log10") +
>     scale_y_continuous(trans = "log10")
```

ou

```
> p + geom_point(size = 3) +
>     geom_text(nudge_x = 0.075) +
>     scale_x_log10() +
>     scale_y_log10()
```

xlab()
ylab()
ggtitle()

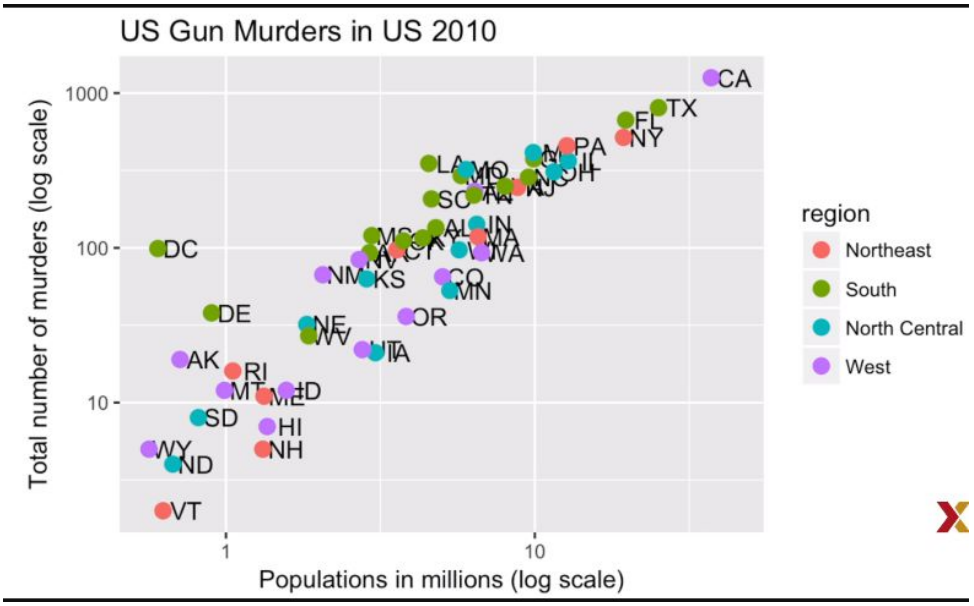
```
> p + geom_point(size = 3) +
>     geom_text(nudge_x = 0.075) +
>     scale_x_log10() +Tx
>     scale_y_log10() +
>     xlab("Population in millions (log scale)") +          //label eixo x
>     ylab("Total number of murders (log scale)") +       //label eixo y
>     ggtitle("US Gun Murders in 2010")                        //titulo
```

color / col

```
# redefine p to be everything except the points layer
> p <- murders %>%
>     ggplot(aes(population/10^6, total, label = abb)) +
>     geom_text(nudge_x = 0.075) +
```

```
>    scale_x_log10() +
>    scale_y_log10() +
>    xlab("Population in millions (log scale)") +
>    ylab("Total number of murders (log scale)") +
>    ggtitle("US Gun Murders in 2010")

# make all points blue
> p + geom_point(size = 3, color = "blue")

# color points by region
> p + geom_point(aes(col = region), size = 3)
```

-ggplot automaticamente irá colocar as legendas para as regiões



geom_abline()

```
# define average murder rate
> r <- murders %>%
>    summarize(rate = sum(total) / sum(population) * 10^6) %>%
>    pull(rate)

# basic line with average murder rate for the country
>  p + geom_point(aes(col = region), size = 3) +
>    geom_abline(intercept = log10(r))    # slope is default of 1

# change line to dashed and dark grey, line under points
>  p +
```

```
>     geom_abline(intercept = log10(r), lty = 2, color = >"darkgrey") +
>     geom_point(aes(col = region), size = 3)
```

legend title

```
> p <- scale_color_discrete(name = “Region”)
```

ADD-ON PACKAGES

library(ggthemes)

-O estilo do gráfico ggplot pode ser alterado pela função:

theme()

- que conta por exemplo, com theme_economist
- p + theme_economist()

library(ggrepel)

- + geom_text_repel()

SUMMARIZING WITH DPLYR

SUMMARIZE

-”Troca o contéudo da variável pela função (ex. mean(height)) em seguida”.

```
> library(tidyverse)
> library(dslabs)
> data(heights)
> s <- heights %>%
>     filter(sex == "Male") %>%
>     summarize(average = mean(height), standard_deviation = sd(height))

# access average and standard deviation from summary table
> s$average
> s$standard_deviation

# compute median, min and max
> heights %>%
>     filter(sex == "Male") %>%
>     summarize(median = median(height),
>               minimum = min(height),
>               maximum = max(height))
```

DOT PLACEHOLDER

- Para acessar um dado que está em *piped* por um *pipe character*, usamos um *dot*:

> us_murder_rate %>% .$rate
>>> 3.03

GROUP BY

- ”Agrupe por”

> heights %>% group_by(sex)

ex2.
compute median murder rate in 4 regions of country
> murders <- murders %>%
*> mutate(murder_rate = total/population * 100000)*
> murders %>%
> group_by(region) %>%
> summarize(median_rate = median(murder_rate))

SORTING DATA TABLES

arrange()

- “Ordena um dataset com base em um parâmetro”

ex.
Ordenar estados por população:
> murders %>% arrange(population) %>% head()
por ordem decrescente:
> murders %>% arrange(desc(population)) %>% head()

ex.
> muders %>% arrange(region, murder_rate) %>% head()

```
> murders %>% arrange(region, murder_rate) %>% head()
          state abb    region population total murder_rate
1       Vermont  VT Northeast     625741     2       0.320
2 New Hampshire  NH Northeast    1316470     5       0.380
3         Maine  ME Northeast    1328361    11       0.828
4  Rhode Island  RI Northeast    1052567    16       1.520
5 Massachusetts  MA Northeast    6547629   118       1.802
6      New York  NY Northeast   19378102   517       2.668
```

top_n()

- Similar ao head(), podendo-se escolher os parâmetros.

ex.10 estados com os maiores índices de homicídios
> murders %>% top_n(10, murder_rate)
ou
sem especificar colunas:
> muders %>% arrange(desc(murder_rate)) %>% top_n(10)

GAPMINDER

-"Most people have misconceptions about world health and economics, which can be addressed by considering real data".

load and inspect gapminder data
> library(dslabs)
> data(gapminder)
> head(gapminder)

compare infant mortality in Sri Lanka and Turkey
> gapminder %>%
> filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%
> select(country, infant_mortality)

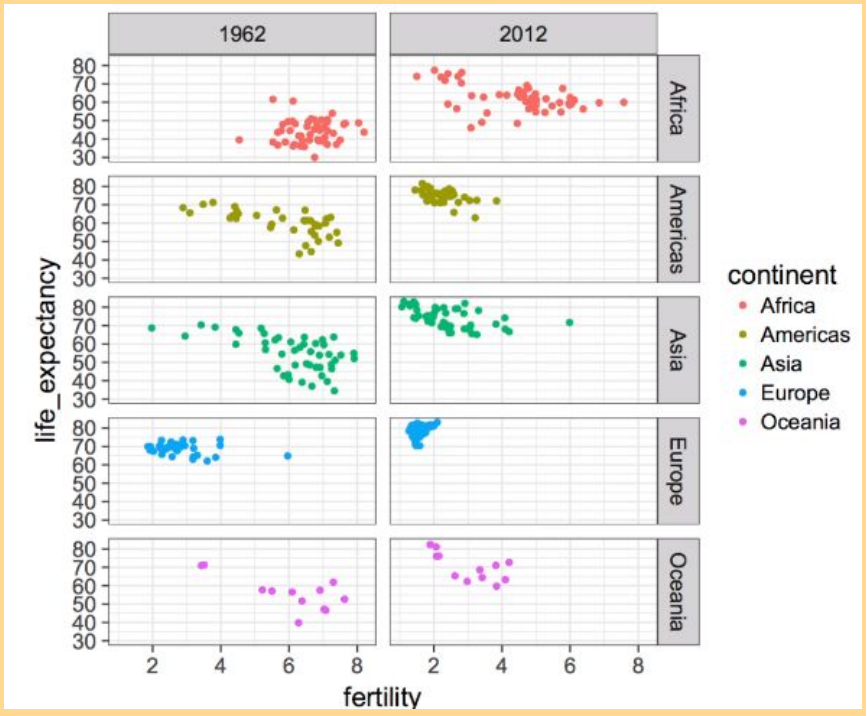
FACETING VARIABLES

-Ao usar *facets*, a comparação dos gráficos fica melhor devido ao mesmo *range* de comparação (y e x) entre eles.

facet_grid()

- "Útil para se comparar gráficos."

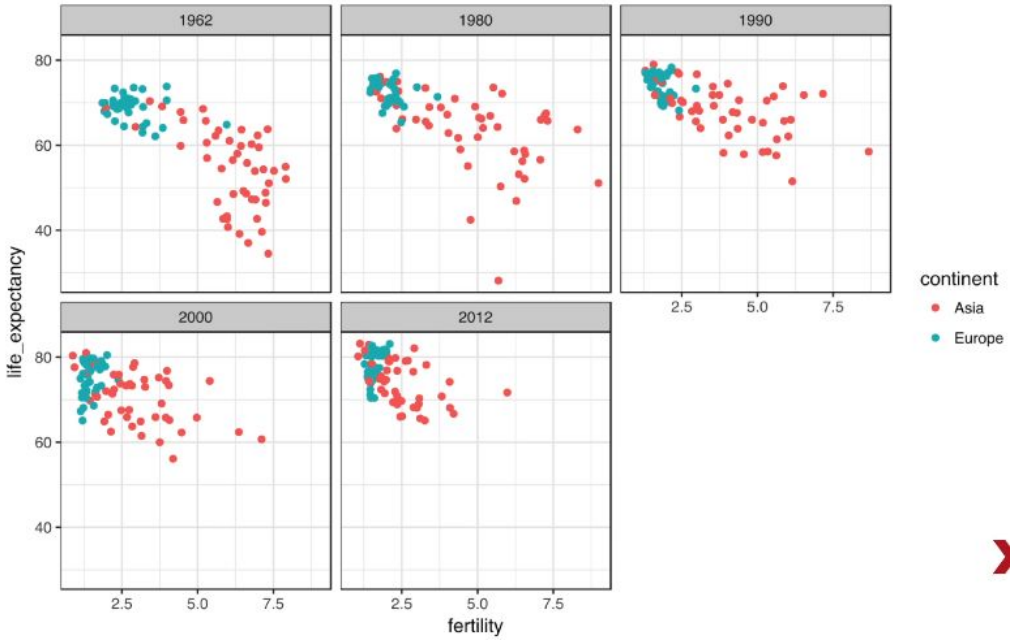
facet by continent and year
> filter(gapminder, year %in% c(1962, 2012)) %>%
> ggplot(aes(fertility, life_expectancy, col = continent)) +
> geom_point() +
> facet_grid(continent ~ year) //no caso de apenas uma variável usa-se .~year



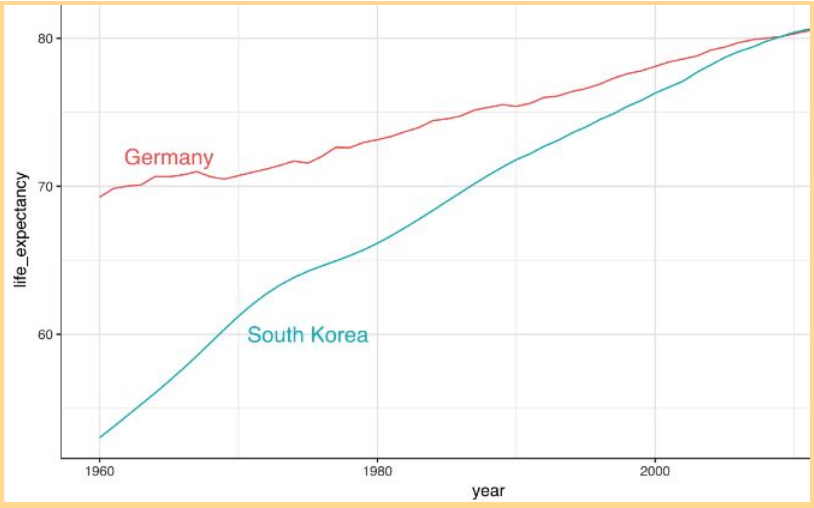
facet_wrap()

facet by year, plots wrapped onto multiple rows

```
> years <- c(1962, 1980, 1990, 2000, 2012)
> continents <- c("Europe", "Asia")
> gapminder %>%
>     filter(year %in% years & continent %in% continents) %>%
>     ggplot(aes(fertility, life_expectancy, col = continent)) +
>     geom_point() +
>     facet_wrap(~year)
```



TIME SERIES PLOT



life expectancy time series - lines colored by country and labeled, no legend
> labels <- data.frame(country = countries, x = c(1975, 1965), y = c(60, 72))
> gapminder %>% filter(country %in% countries) %>%
> ggplot(aes(year, life_expectancy, col = country)) +
> geom_line() +
> geom_text(data = labels, aes(x, y, label = country), size = 5) +
> theme(legend.position = "none")

STRATIGY & BOXPLOT

reorder()

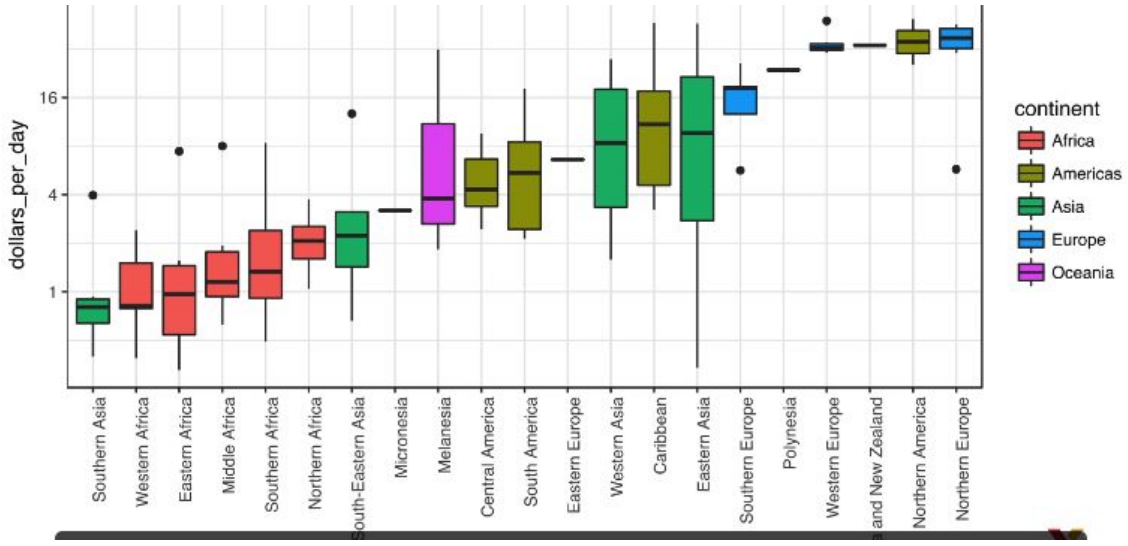
- “Reordena por”.

reorder by median income and color by continent
> p <- gapminder %>%
> filter(year == past_year & !is.na(gdp)) %>%
> mutate(region = reorder(region, dollars_per_day, FUN = median)) %>%
 # reorder
> ggplot(aes(region, dollars_per_day, fill = continent)) + # color by continent
> geom_boxplot() +
> theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
> xlab("")

log2 scale y-axis
p + scale_y_continuous(trans = "log2")

add data points

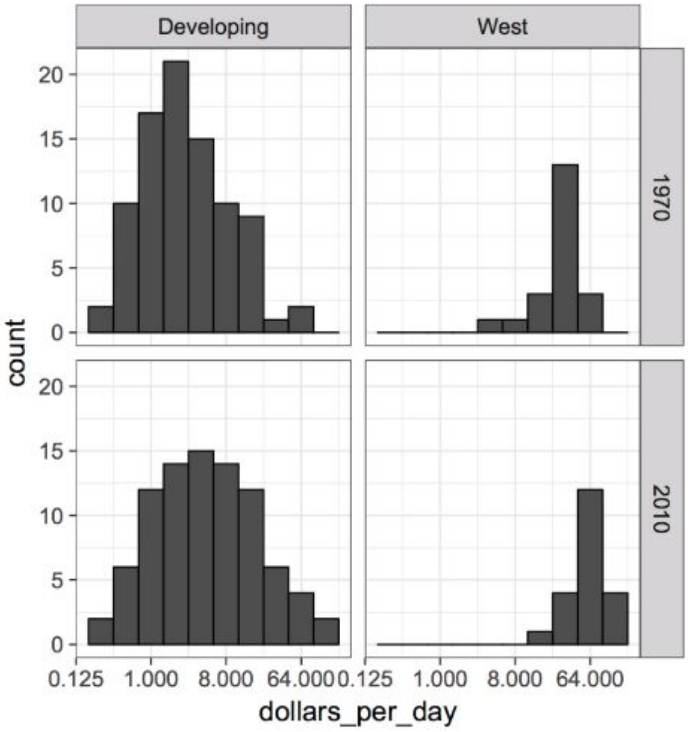
p + scale_y_continuous(trans = "log2") + geom_point(show.legend = FALSE)



COMPARING DISTRIBUTIONS

-Podemos usar o ifelse dentro de um mutate

-Função intersect() //intersecta duas tabelas e retorna os iguais



define countries that have data available in both years

> country_list_1 <- gapminder %>%

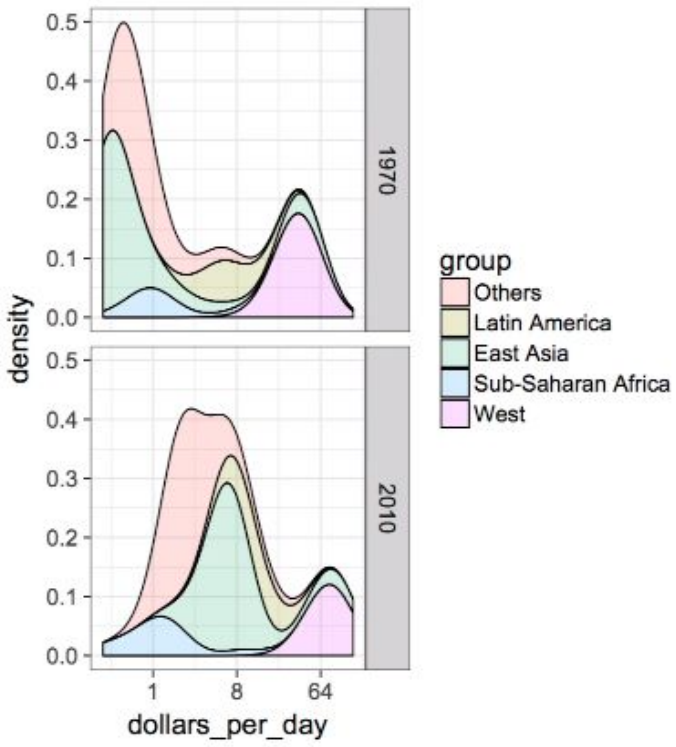
> filter(year == past_year & !is.na(dollars_per_day)) %>% .$country

```
>    country_list_2 <- gapminder %>%
>    filter(year == present_year & !is.na(dollars_per_day)) %>% .$country
>   country_list <- intersect(country_list_1, country_list_2)

# make histogram including only countries with data available in both years
gapminder %>%
>    filter(year %in% c(past_year, present_year) & country %in% country_list) %>%

# keep only selected countries
>    mutate(group = ifelse(region %in% west, "West", "Developing")) %>%
>    ggplot(aes(dollars_per_day)) +
>    geom_histogram(binwidth = 1, color = "black") +
>    scale_x_continuous(trans = "log2") +
>    facet_grid(year ~ group)
```

DENSITY PLOTS

-geom_density()
-case_when()
-dot dot // ..variableName..
-position = “stack” //gráficos menores sobrepõem os maiores
-weight()

```
# weighted stacked density plot
> gapminder %>%
>    filter(year %in% c(past_year, present_year) & country %in% country_list) %>%
>    group_by(year) %>%
>    mutate(weight = population/sum(population*2)) %>%
>    ungroup() %>%
>    ggplot(aes(dollars_per_day, fill = group, weight = weight)) +
>    scale_x_continuous(trans = "log2") +
>    geom_density(alpha = 0.2, bw = 0.75, position = "stack") + facet_grid(year ~ .)
```



odds = p/(1-p)

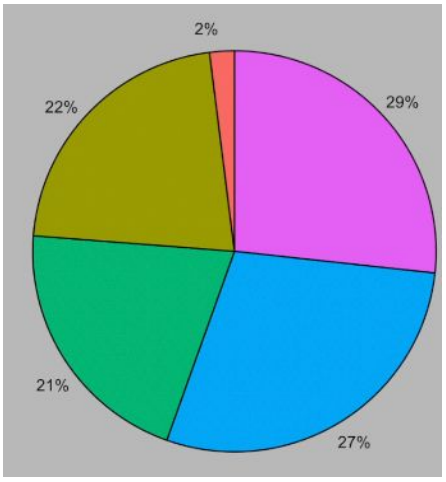
INTRODUTION DO DATA VISUALIZATION PRINCIPLES

ENCODING DATA USING VISUAL CUES

-position
-aligned lengths
-angles
-area
-brightness
-color hue

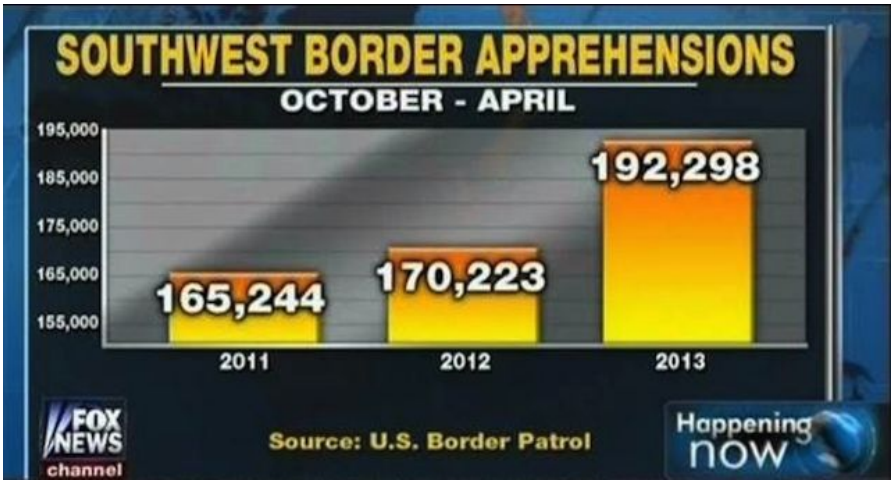
-O gráfico de pizza ou donnuts não são uma boa escolha, uma vez que humanos tem dificuldades em recolher dados visuais usando apenas ângulos ou áreas. Caso seja necessário usar gráficos, é melhor usarmos posições e tamanhos, como num gráfico de barras, usando também escalas corretas entre outros artifícios.

-Dica para utilizar o gráfico de pizza; utilize porcentagens com números sobre as partes, ex:



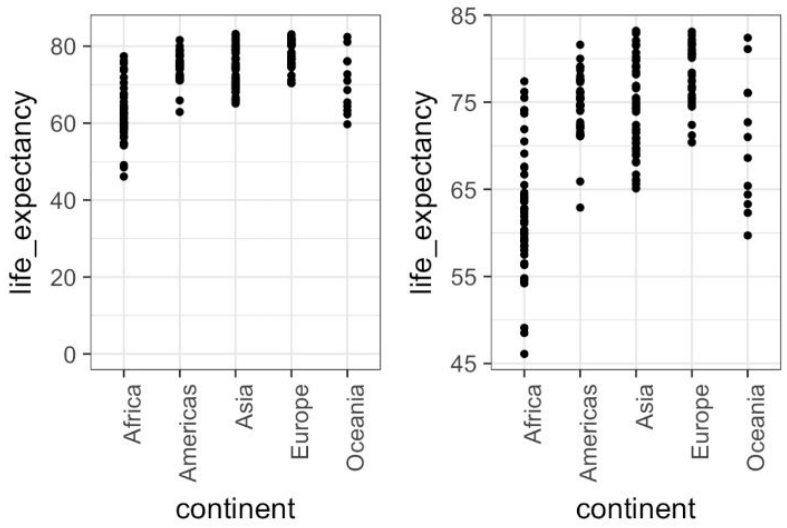
KNOW WHEN TO INCLUDE ZERO

-Algumas vezes, como na política por exemplo, não se utiliza o 0 como escala inicial para gráficos de barra para que se tenha um exagero na diferença gráfica, ex:



*Apesar de se ter uma apârencia que as apreensões aumentaram quase o triplo de 2011 para 2013, teve-se um aumento de apenas 16%.

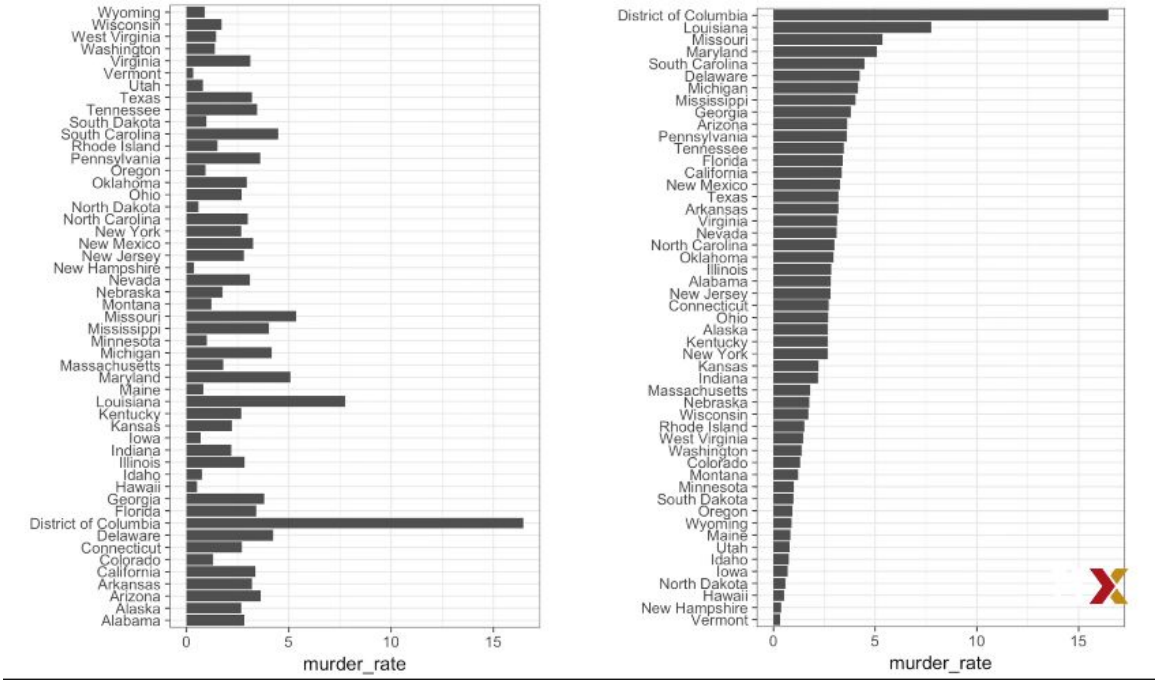
-Porém, em um gráfico onde se quer ver a diferença de um mesmo país por exemplo, pode-se diminuir a escala começando de outro valo a fim de que se tenha um “zoom” nas partes que se quer observar.



- **do not distort quantities**

ordene por um valor com significado

-Nesse caso por ex, nota-se que fica muito melhor a visualização ordenando por “murder rate” do que pela ordem alfabética.



SHOW THE DATA

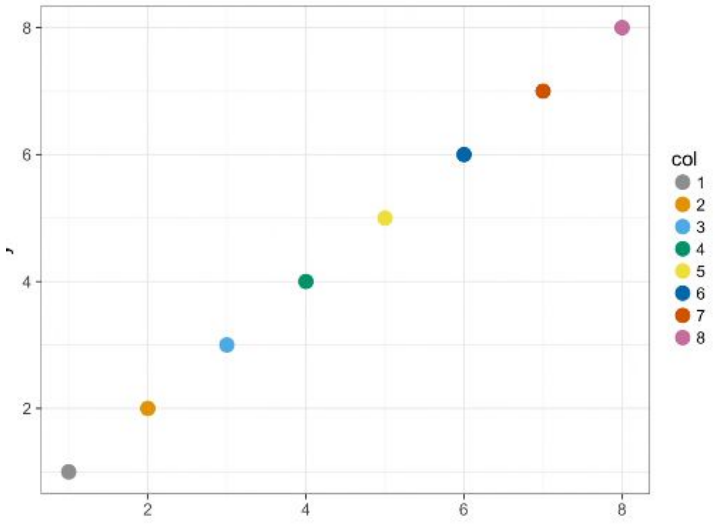
- standard erros != standard deviations

OMMON AXE's

- Gráfico de Barra são úteis para 1 único valor, mas não para distribuições.

PALETA DE CORES PARA DALTÔNICOS

```
> color_blind_friendly_cols <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
> "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

> p1 <- data.frame(x = 1:8, y = 1:8, col = as.character(1:8)) %>%
>   ggplot(aes(x, y, color = col)) +
>   geom_point(size = 5)
> p1 + scale_color_manual(values = color_blind_friendly_cols)
```



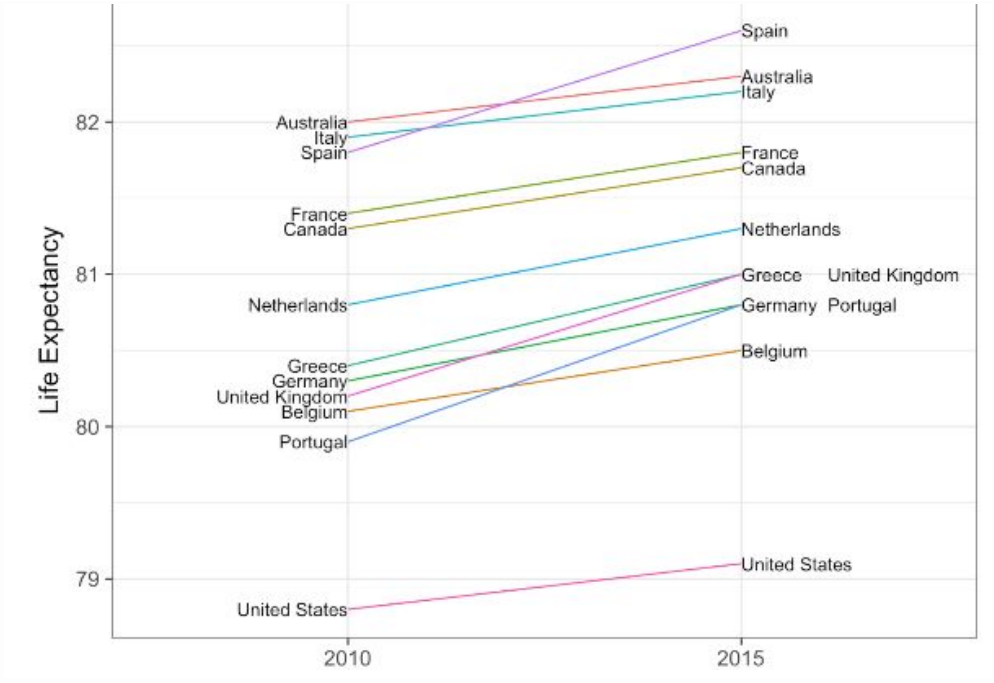
SLOPE CHARTS

-Útil ao comparar variáveis do mesmo tipo, em tempos diferentes e com um número pequeno de comparações.
Ex.

- Comparar expectativa de vida entre 2010 e 2015

```
> library(tidyverse)
> library(dslabs)
> data(gapminder)

> west <- c("Western Europe", "Northern Europe", "Southern Europe", "Northern
> America", "Australia and New Zealand")

> -dat <- gapminder %>%
```

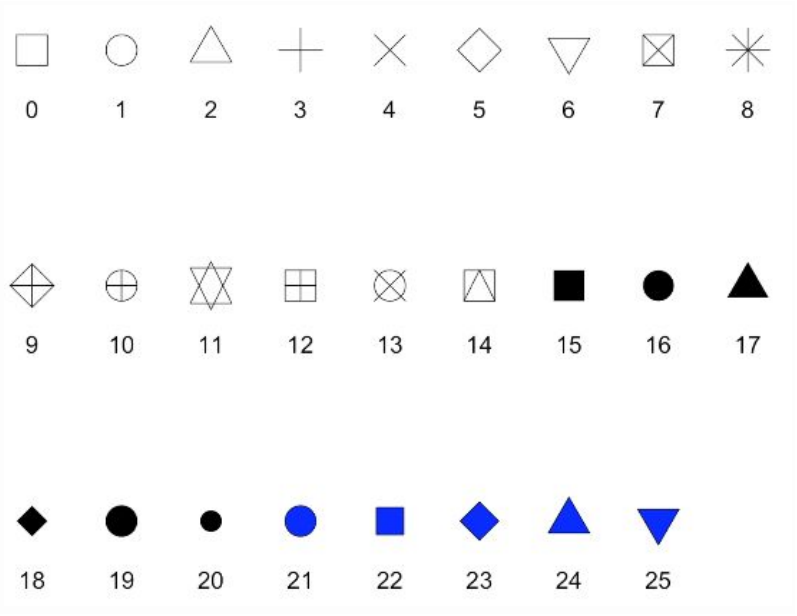
```
>  filter(year %in% c(2010, 2015) & region %in% west & !is.na(life_expectancy) &
>  population > 10^7)

> dat %>%
>    mutate(location = ifelse(year == 2010, 1, 2),
>           location = ifelse(year == 2015 & country %in% c("United Kingdom",
> "Portugal"),
>                             location + 0.22, location),
>           hjust = ifelse(year == 2010, 1, 0)) %>%
>    mutate(year = as.factor(year)) %>%
>    ggplot(aes(year, life_expectancy, group = country)) +
>    geom_line(aes(color = country), show.legend = FALSE) +
>    geom_text(aes(x = location, label = country, hjust = hjust), show.legend =
>  FALSE) +
>    xlab("") +
>    ylab("Life Expectancy")
```



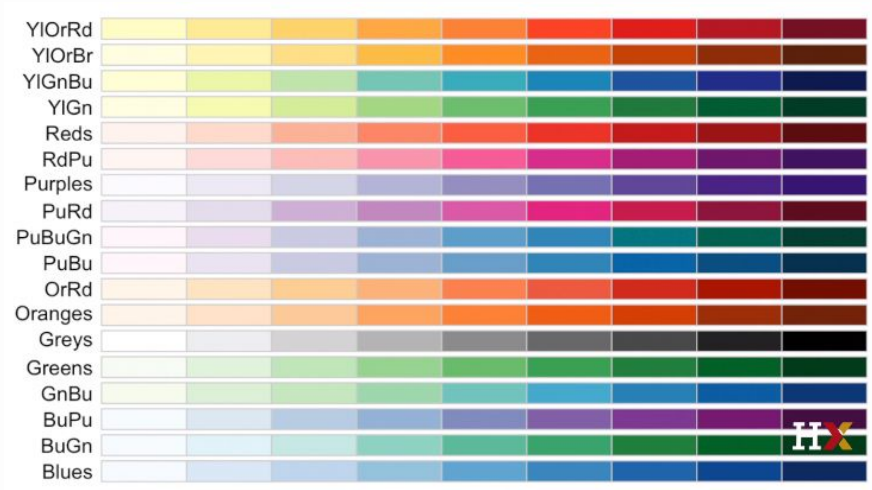
*Apesar de se uma visualização por ângulo, temos mais informações para ajudar no entendimento, como altura no eixo y, posição e eixo x.

SHAPE ARGUMENT

shape()



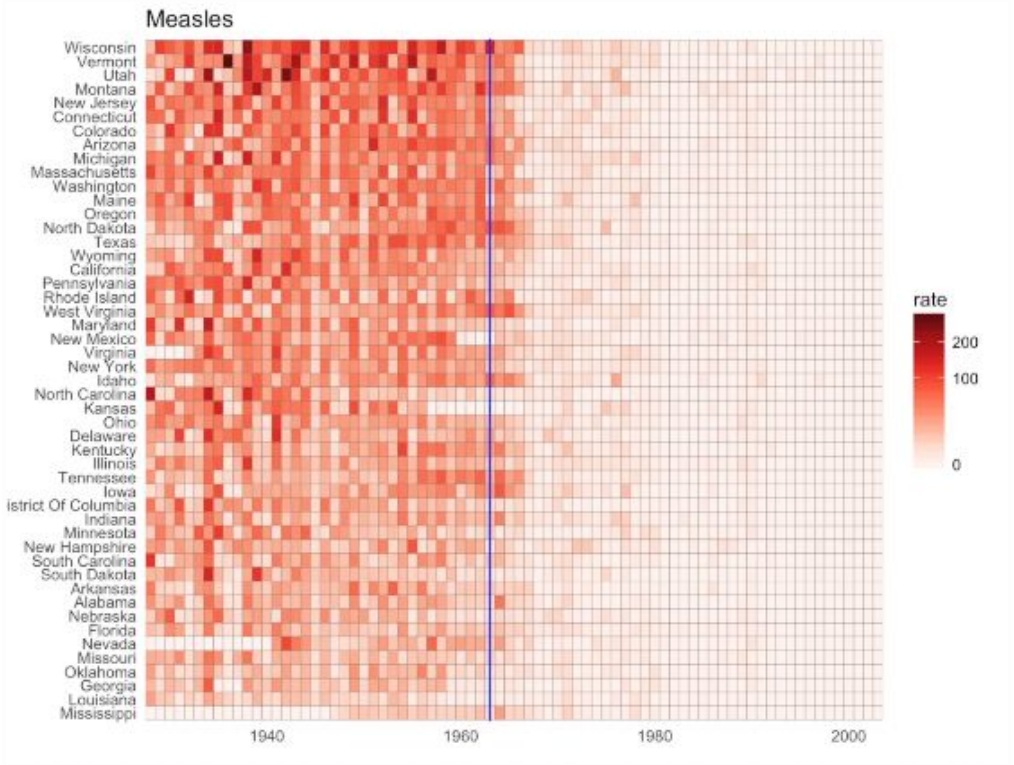
> library(RColorBrewer)
> display.brewer.all(type="seq")



> library(RColorBrewer)
> display.brewer.all(type="div")

tile plot of disease rate by state and year

```
> dat %>% ggplot(aes(year, state, fill=rate)) +
>   geom_tile(color = "grey50") +
>   scale_x_continuous(expand = c(0,0)) +
>  scale_fill_gradientn(colors = RColorBrewer::brewer.pal(9, "Reds"), trans = "sqrt")
> +
>   geom_vline(xintercept = 1963, col = "blue") +
>   theme_minimal() + theme(panel.grid = element_blank()) +
>   ggtitle(the_disease) +
>   ylab("") +
>   xlab("")
```



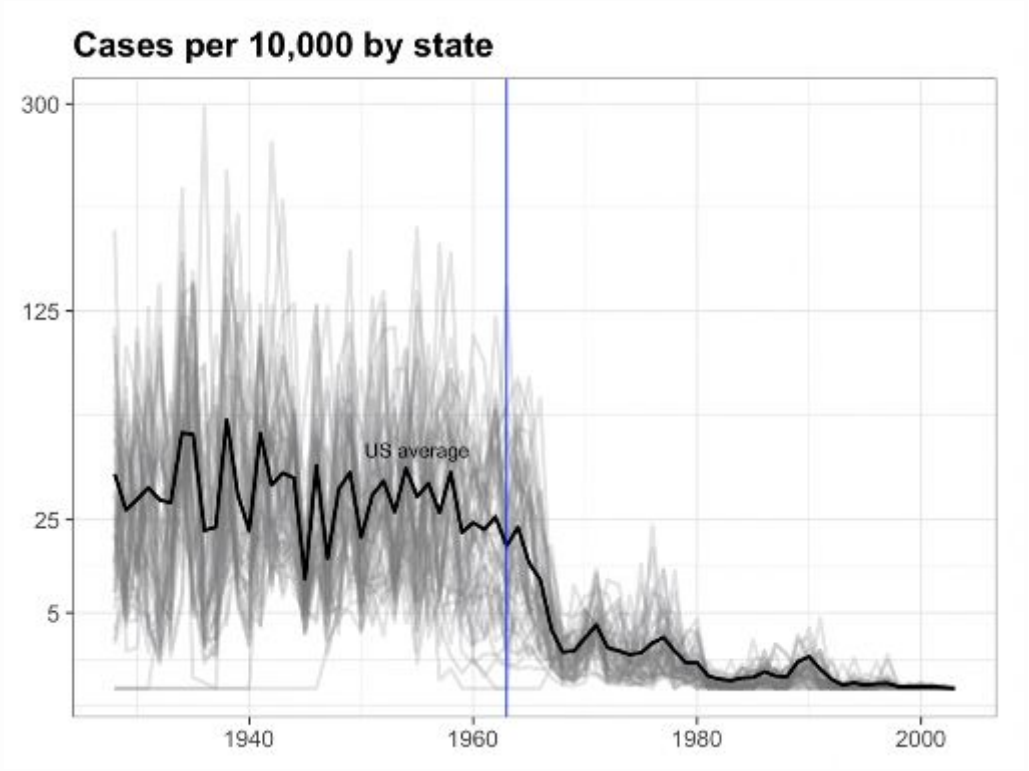
compute US average measles rate by year

```
> avg <- us_contagious_diseases %>%
```

```
>   filter(disease == the_disease) %>% group_by(year) %>%
>   summarize(us_rate = sum(count, na.rm = TRUE)/sum(population, na.rm =
TRUE)*10000)
```

//na.rm ignora os valores NA's

```
# make line plot of measles rate by year by state
> dat %>%
>   filter(!is.na(rate)) %>%
>   ggplot() +
>   geom_line(aes(year, rate, group = state), color = "grey50",
>       show.legend = FALSE, alpha = 0.2, size = 1) +
>   geom_line(mapping = aes(year, us_rate), data = avg, size = 1, col = "black") +
>   scale_y_continuous(trans = "sqrt", breaks = c(5, 25, 125, 300)) +
>   ggtitle("Cases per 10,000 by state") +
>   xlab("") +
>   ylab("") +
>   geom_text(data = data.frame(x = 1955, y = 50),
>       mapping = aes(x, y, label = "US average"), color = "black") +
>   geom_vline(xintercept = 1963, col = "blue")
```



Avoid Pseudo and Gratuitous 3D Plots

Avoid Too Many Significant Digits