Lista 4: Teoria de Vis

Germano Andrade Brandão - 2017080008

19/05/2020

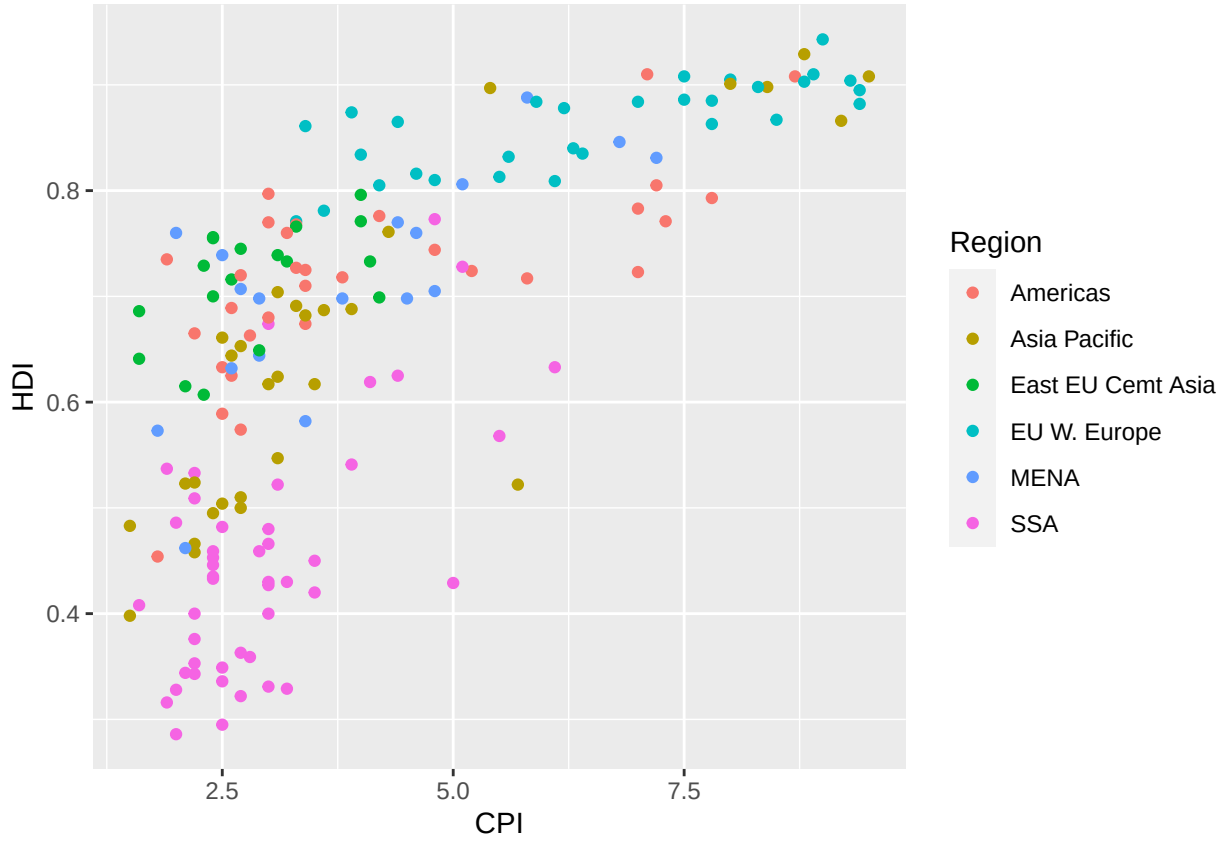
Problema 1: The Economist

Crie o último gráfico do link https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html.
Este é um gráfico que apresenta uma relação entre o IDH x IPC entre diversos países.

Resposta:

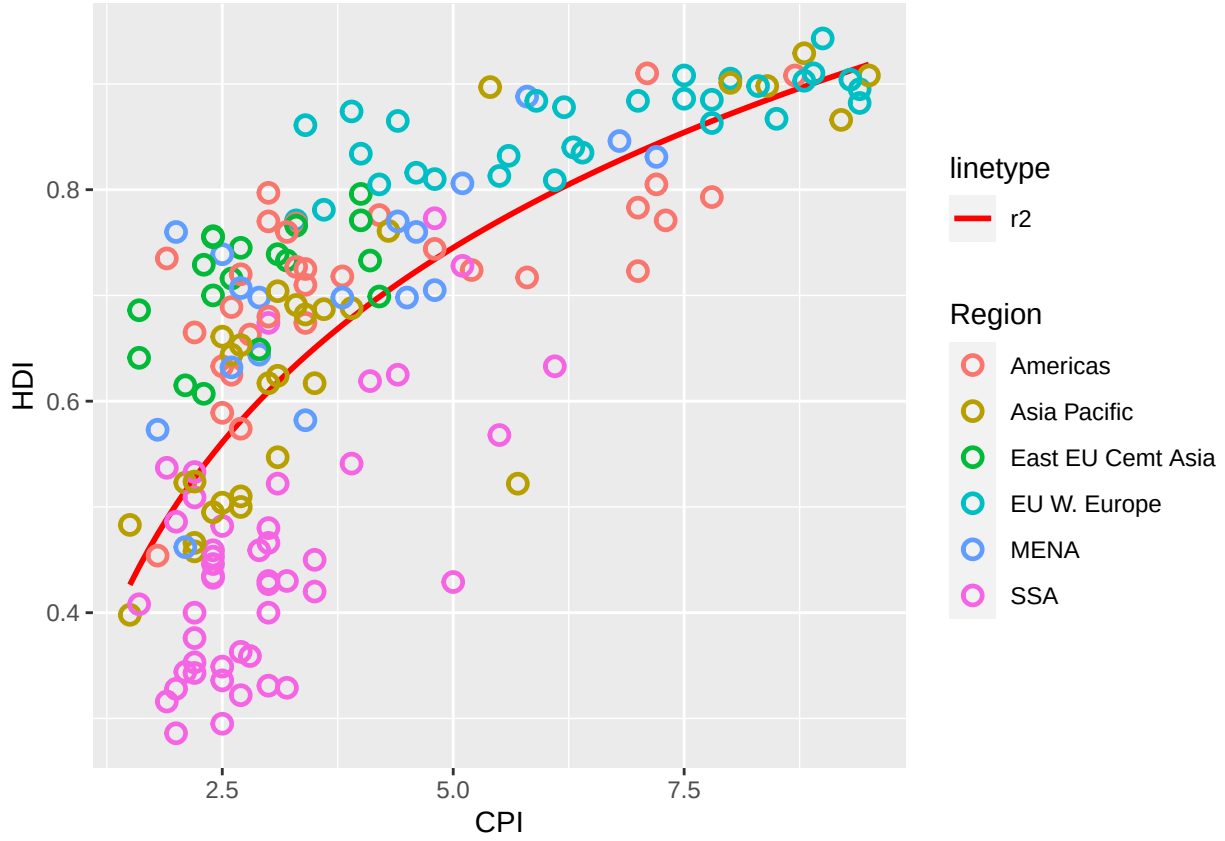
Começando com um simples gráfico de dispersão:

```
dat <- read.csv("./Databases/The Economist/EconomistData.csv")

pc1 <- ggplot(dat, aes(x = CPI, y = HDI, color = Region))

pc1 + geom_point()
```



Adicionando uma linha de tendência:

```r
pc2 <- pc1 +
  geom_smooth(mapping = aes(linetype = "r2"), #Aqui é usado o coeficiente de
              method = "lm",
              formula = y ~ x + log(x), se = FALSE,
              color = "red")

#Nesse ponto, foi alterado a forma dos pontos, para que virassem uma circunferência e
#modificado os seus tamanhos e a espessura das bordas.
(pc3 <- pc2 + geom_point(shape = 1,
                         size = 2.5,
                         stroke = 1.25))
```

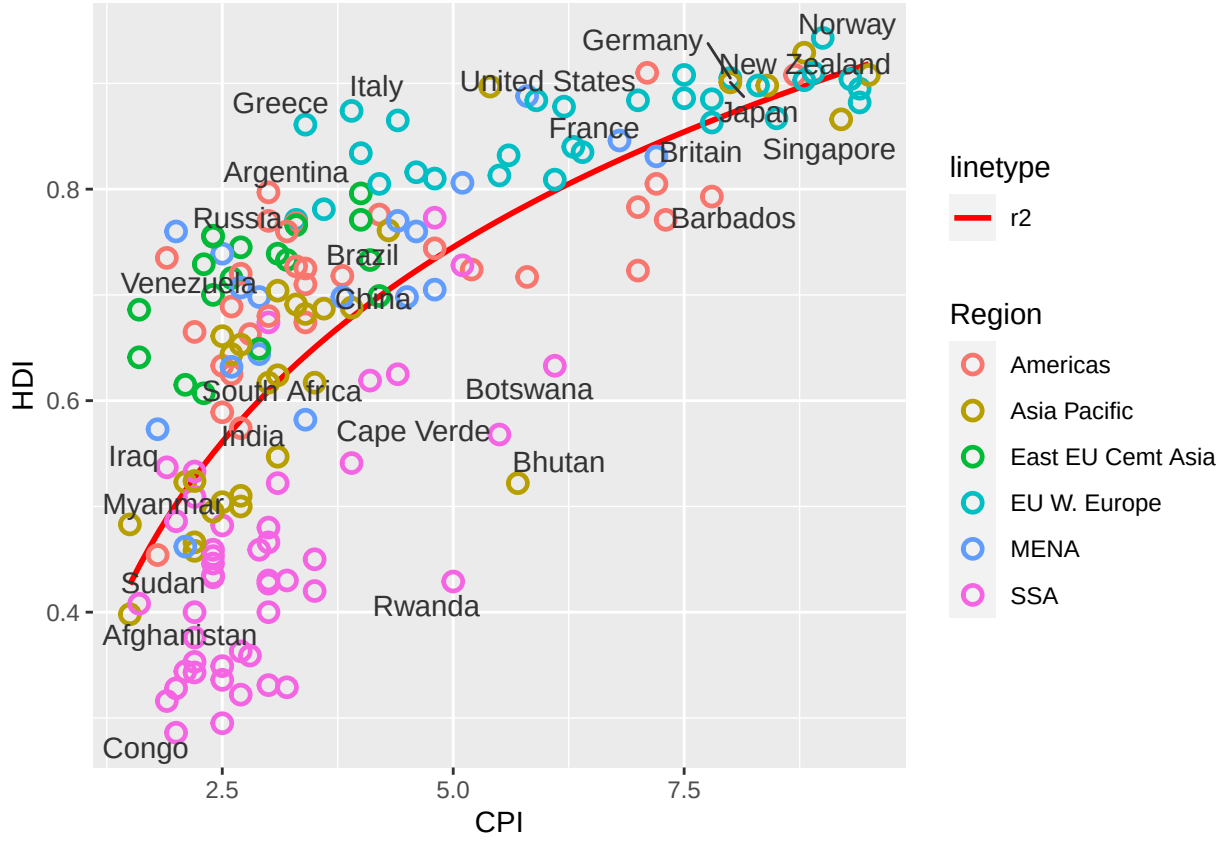


Escolhendo os pontos que devem aparecer no gráfico:

```r
#Agora, baseando-se no gráfico original, foi selecionado os países que vão ser destacados
#frente aos demais, e então foram guardados em uma variável.

pointsToLabel <- c("Russia", "Venezuela", "Iraq", "Myanmar", "Sudan",
                   "Afghanistan", "Congo", "Greece", "Argentina", "Brazil",
                   "India", "Italy", "China", "South Africa", "Spane",
                   "Botswana", "Cape Verde", "Bhutan", "Rwanda", "France",
                   "United States", "Germany", "Britain", "Barbados",
                   "Norway", "Japan", "New Zealand", "Singapore")
```

```r
#Detalhe para a função 'filter' da biblioteca "dplyr" que é usado para adicionar ao
#gráfico apenas os países que estiverem na variável 'pointsToLabel' criada há pouco.
```

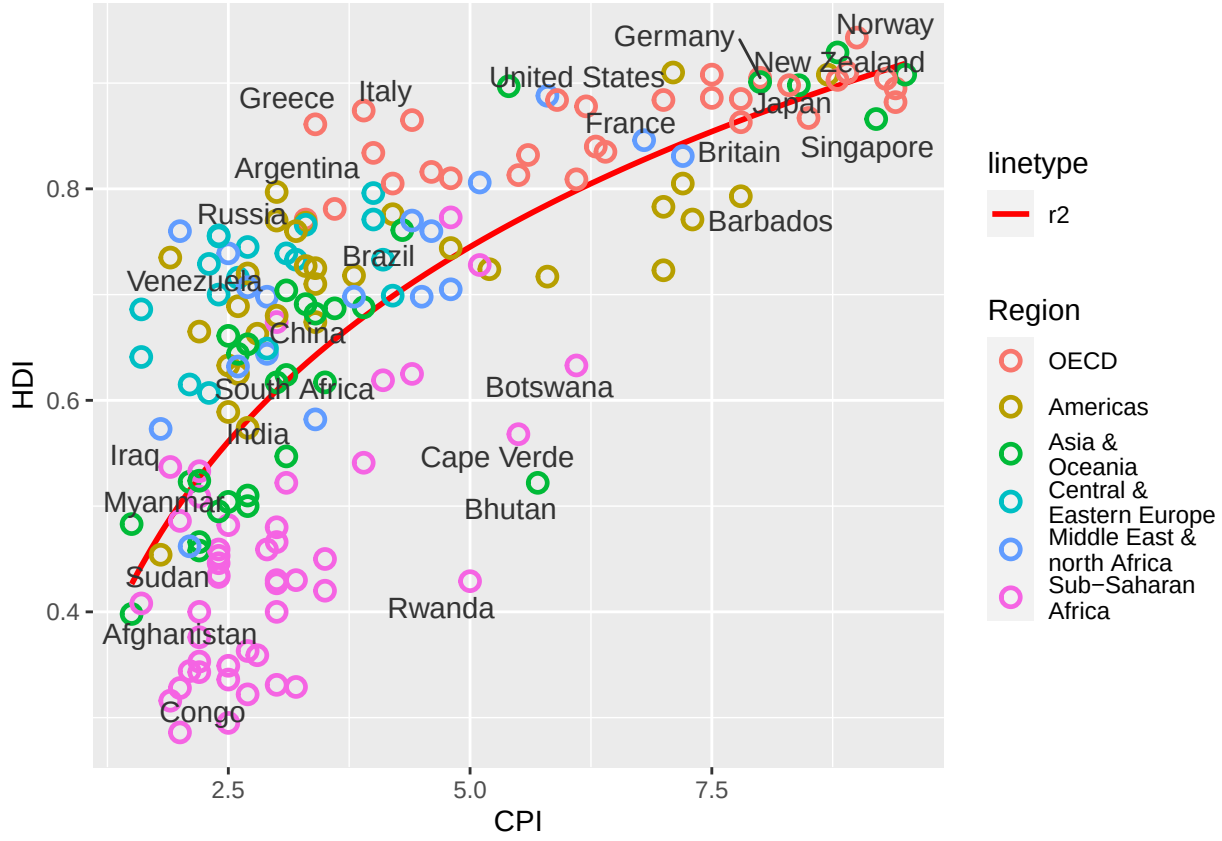
```
#Além disso, foi utilizada nesse ponto a biblioteca "ggrepel" para corrigir a posição
#do nome dos países que estavam gerando conflito no gráfico.
(pc4 <- pc3 +
  geom_text_repel(aes(label = Country),
            color = "gray20",
            data = filter(dat, Country %in% pointsToLabel),
            force = 10))
```



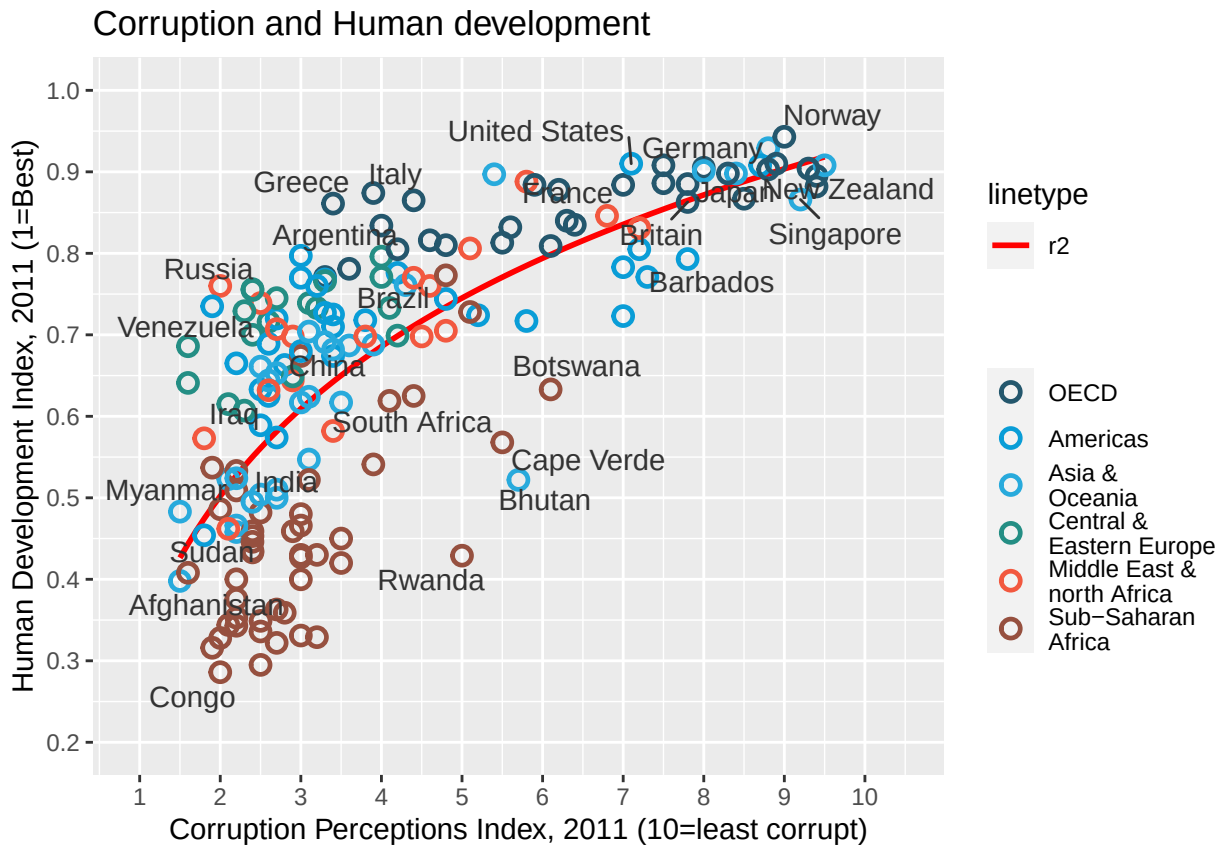
Mudar a ordem e o rótulo da região:

```
dat <- read.csv("./Databases/The Economist/EconomistData.csv")

#Foi alterada dentro da tabela o nome de cada região e o nível de ordem (já que a
#variável região é do tipo 'factor').
dat$Region <- factor(dat$Region,
                     levels = c("EU W. Europe",
                                "Americas",
                                "Asia Pacific",
                                "East EU Cemt Asia",
                                "MENA",
                                "SSA"),
                     labels = c("OECD",
                                "Americas",
                                "Asia &\nOceania",
                                "Central &\nEastern Europe",
                                "Middle East &\nnorth Africa",
                                "Sub-Saharan\nAfrica"))
```

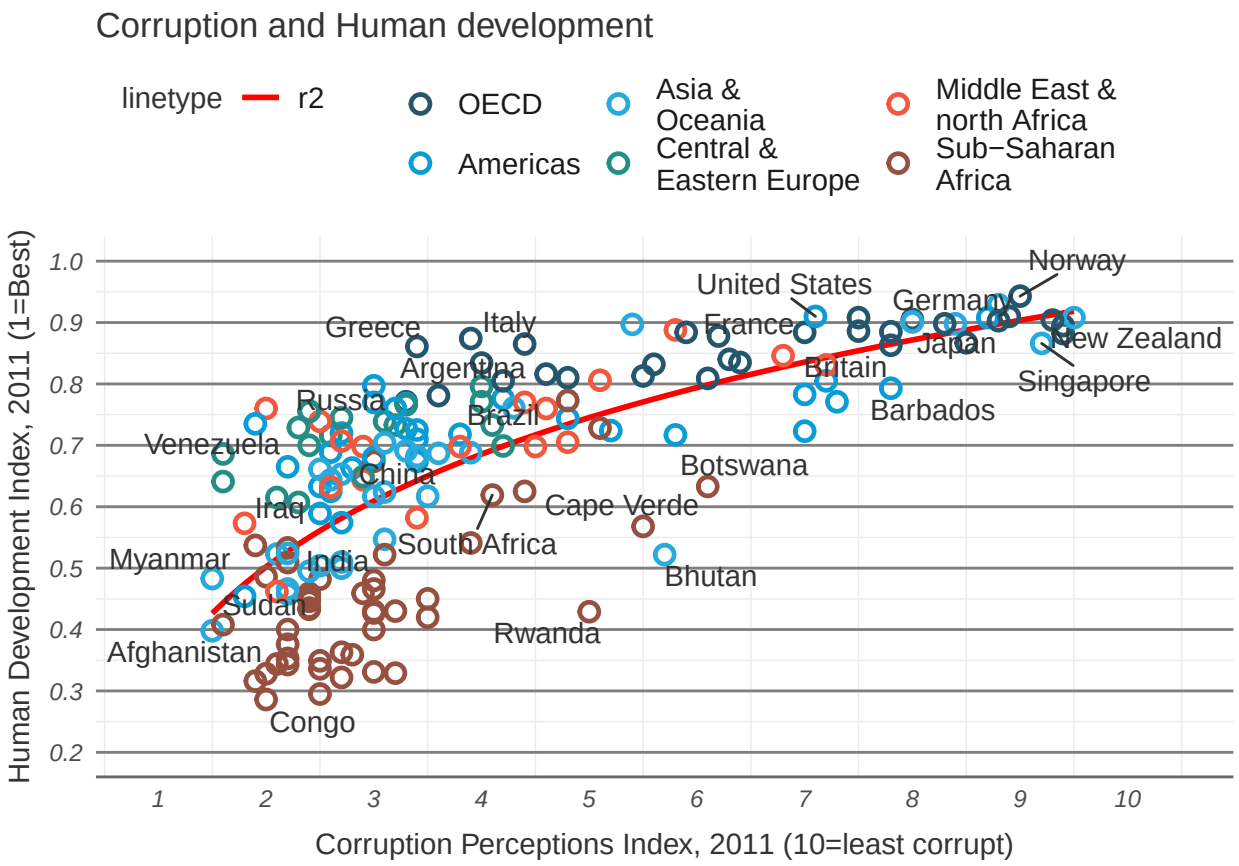
```
#Nesse ponto o que se faz é "atualizar" os dados do gráfico, uma vez que eles foram
#modificados. O 'pc4' havia herdado desde o 'pc1'  o "ggplot(data = dat,...)",
#então foi preciso colocar o 'dat' modificado sem precisar repetir todo o processo.

pc4$data <- dat
pc4
```




```
#Adicionado a biblioteca "grid"
(pc5 <- pc4 +
  scale_x_continuous(name = "Corruption Perceptions Index, 2011 (10=least corrupt)",
                     limits = c(.9, 10.5),
                     breaks = 1:10) +
  scale_y_continuous(name = "Human Development Index, 2011 (1=Best)",
                     limits = c(0.2, 1.0),
                     breaks = seq(0.2, 1.0, by = 0.1)) +
  scale_color_manual(name = "",
                     values = c("#24576D",
                                "#099DD7",
                                "#28AADC",
                                "#248E84",
                                "#F2583F",
                                "#96503F")) +
  ggtitle("Corruption and Human development"))
```




```
(pc6 <- pc5 +
  theme_minimal() + # Começando com um tema minimalista e adicionando o que for preciso
  theme(text = element_text(color = "gray20"),
        legend.position = c("top"), # Posição da legenda no topo à esquerda
        legend.direction = "horizontal",
        legend.justification = 0.1, # Ponto de ancoragem para legend.position.
        legend.text = element_text(size = 11, color = "gray10"),
        axis.text = element_text(face = "italic"),
        axis.title.x = element_text(vjust = -1), # Movendo o título para longe dos eixos
        axis.title.y = element_text(vjust = 2), # Afastar para o eixo
        axis.ticks.y = element_blank(), # element_blank() é como se remove os elementos
        axis.line = element_line(color = "gray40", size = 0.5),
        axis.line.y = element_blank(),
        panel.grid.major = element_line(color = "gray50", size = 0.5),
        panel.grid.major.x = element_blank()
        ))
```



Problema 2: Minard

Reproduza o gráfico de Minard que representa a jornada do lider político Napoleão ao levar seu exército para a Campanha Russa.
O tutorial se encontra em https://www.andrewheiss.com/blog/2017/08/10/exploring-minards-1812-plot-with-ggplot2/.

Resposta:

Minard

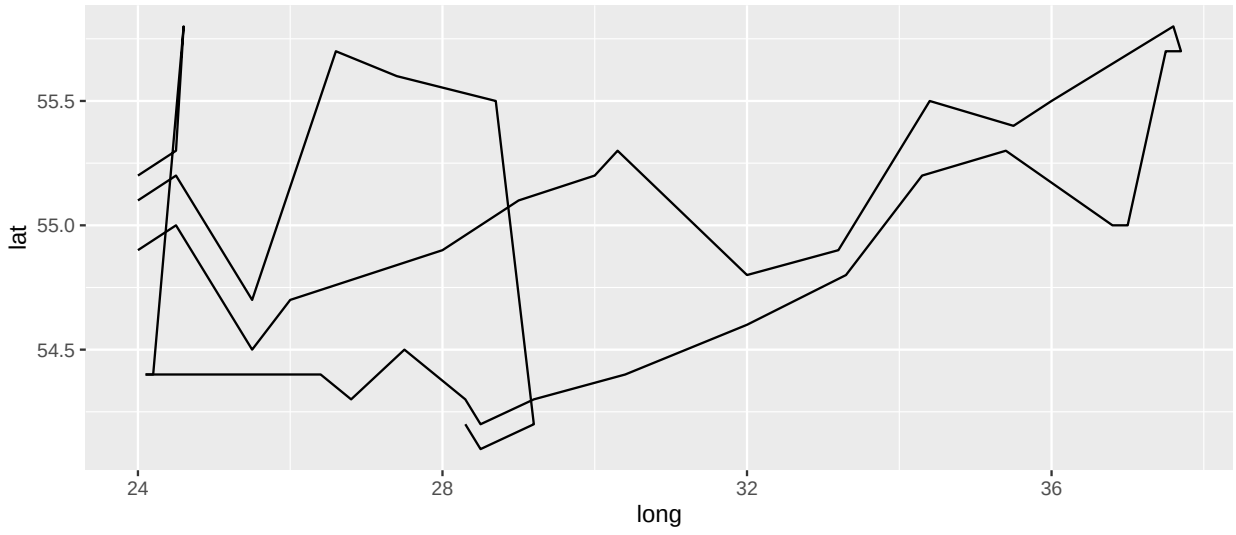
```
cities <- read.table("./Databases/Minard/cities.txt",
                     header = TRUE, stringsAsFactors = FALSE)

troops <- read.table("./Databases/Minard/troops.txt",
                     header = TRUE, stringsAsFactors = FALSE)

#A função 'mutate' foi usada para colocar a data em um formato adequado, sinalizando
#que possui dia, mês e ano (dmy).
temps <- read.table("./Databases/Minard/temps.txt",
                    header = TRUE, stringsAsFactors = FALSE) %>%
    mutate(date = dmy(date))
```

- Uma breve analisada na tabela **troops** nos mostra que ela contém dados geográficos das cidades (latitude e longitude), número de sobreviventes, o grupo que passava por aqueles lugares e a indicação do grupo.

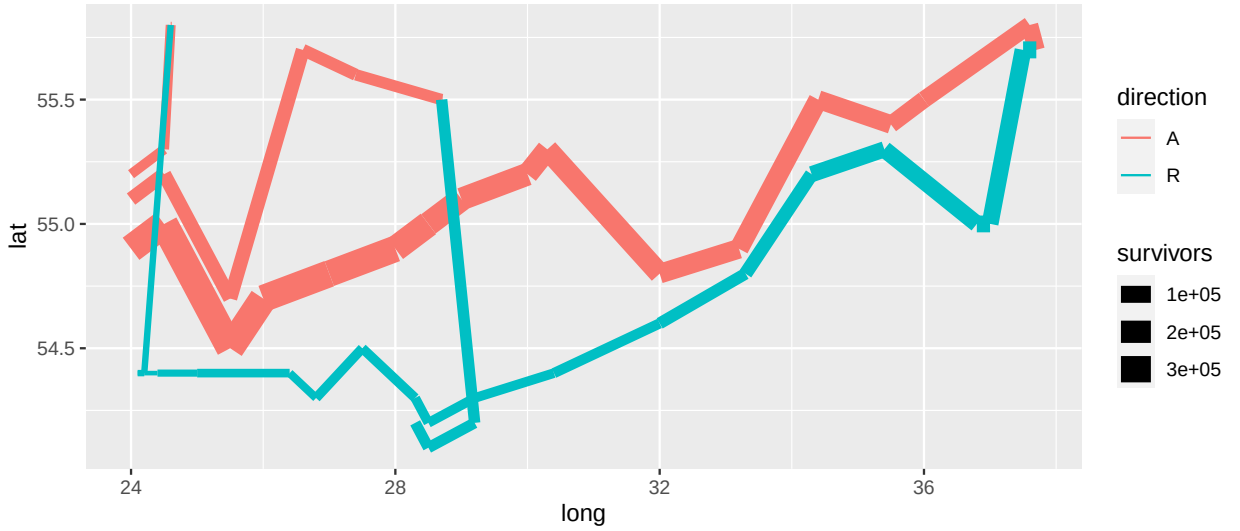
```r
kable(head(troops), align = 'c')
```

long	lat	survivors	direction	group
24.0	54.9	340000	A	1
24.5	55.0	340000	A	1
25.5	54.5	340000	A	1
26.0	54.7	320000	A	1
27.0	54.8	300000	A	1
28.0	54.9	280000	A	1

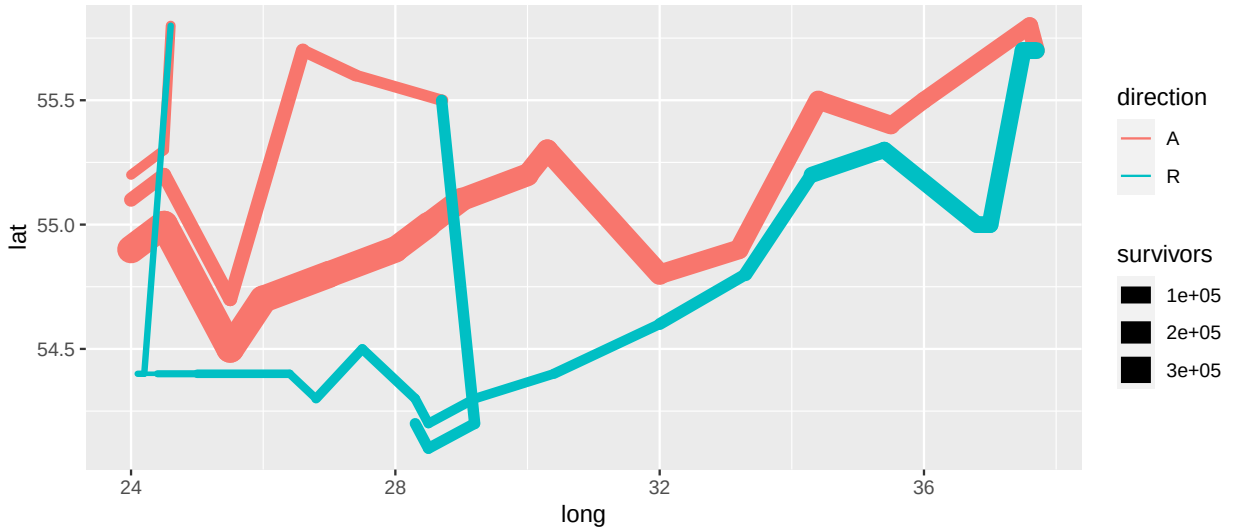
```r
#Apenas mapeando os pontos através de linhas e agrupando pelos diferentes grupos da
#tropa, já obtemos algo parecido com o gráfico final:
ggplot() +
  geom_path(data = troops, aes(x = long, y = lat, group = group))
```



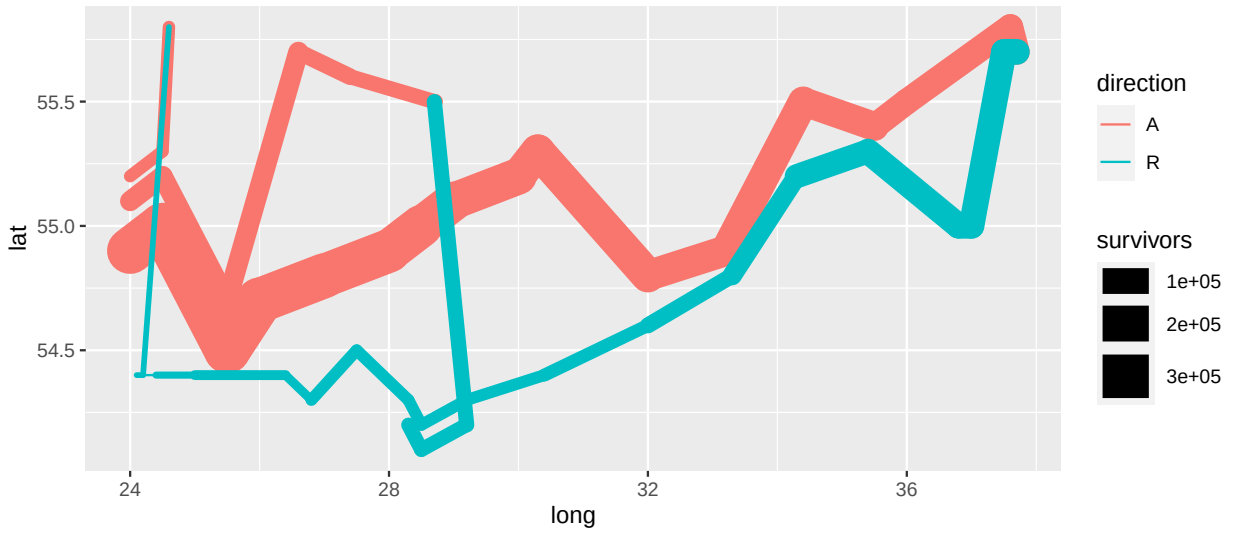

```r
ggplot() +
  geom_path(data = troops, aes(x = long, y = lat, group = group,
                               color = direction, size = survivors))
```




```
#Como o final de linha (lineend) padrão do 'geom_path' é do tipo "butt", aparecem algumas
#falhas em partes das linhas. Para solucionar esse problema, o 'lineend' é alterado:
(p1 <- ggplot() + geom_path(data = troops, aes(x = long, y = lat, group = group,
                                               color = direction, size = survivors),
                            lineend = "round"))
```

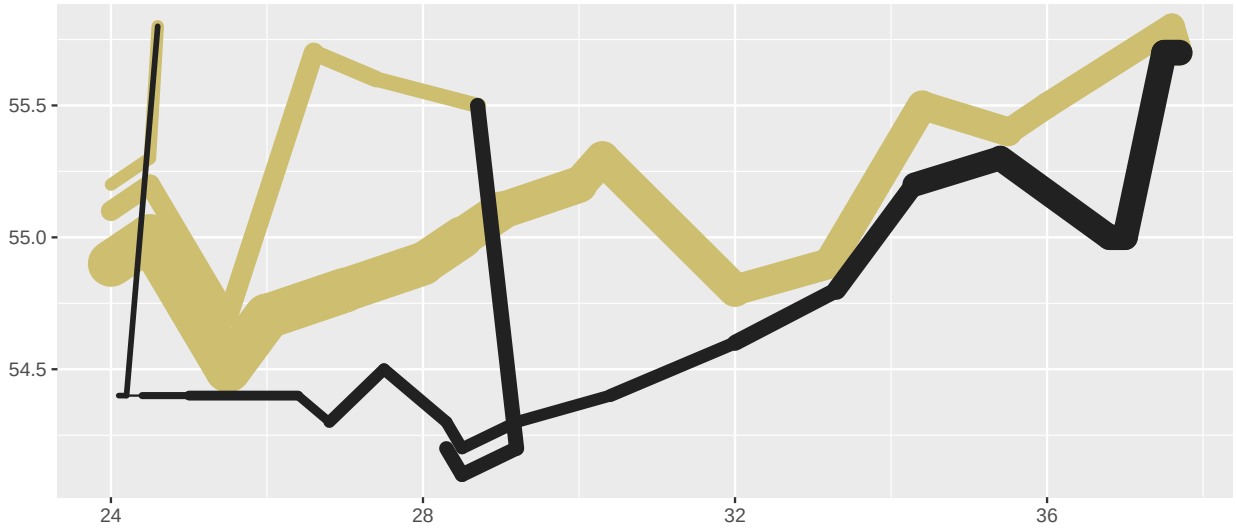



```
#Para aumentar a espessura da linha - que varia de acordo com o número de sobreviventes-,
#altera-se a escala usada para tal:
(p2 <- p1 +
  scale_size(range = c(0.5, 10)))
```

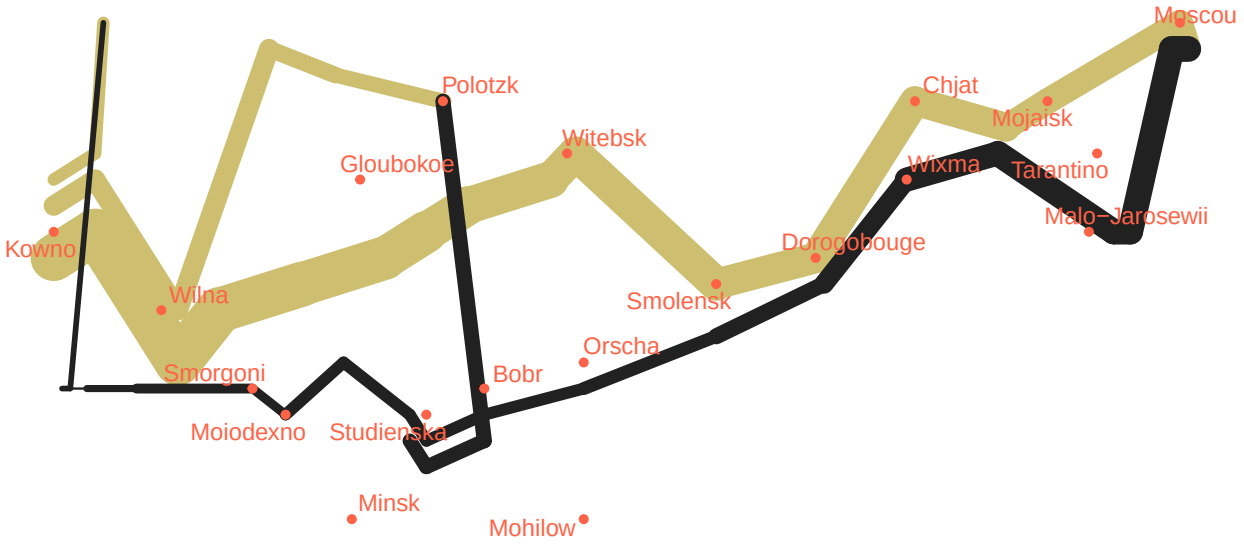


- Agora que o gráfico está se assemelhando mais ainda do original, desconsiderando a cor, vamos tratar de pequenos detalhes para que fique ainda mais parecido.

```
(p3 <- p2 +
  scale_colour_manual(values = c("lightgoldenrod3", "gray13")) +
  labs(x = NULL, y = NULL) +
  guides(color = FALSE, size = FALSE))
```



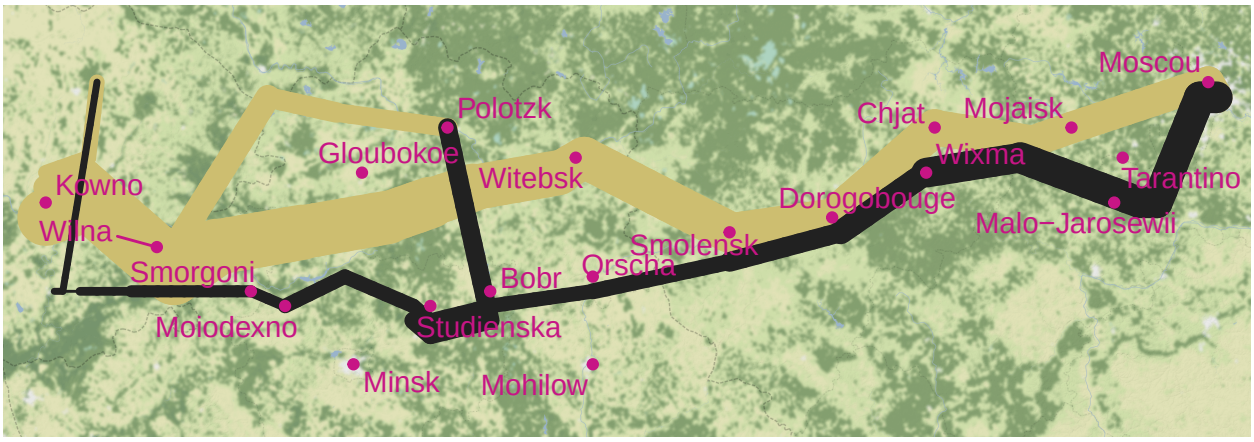

```
#Vamos adicionar as cidades e seus respectivos nomes:
(p4 <- p3 +
  geom_point(data = cities,
             aes(x = long, y = lat),
             color = "tomato") +
  geom_text_repel(data = cities,
                  aes(x = long, y = lat, label = city),
                  color = "tomato") +
   theme_void())
```



- Agora, por curiosidade, vamos utilizar os recursos da biblioteca "ggmap" para colocar nosso gráfico em um mapa real. Isso porque o nosso gráfico contém as coordenadas geográficas reais das cidades.

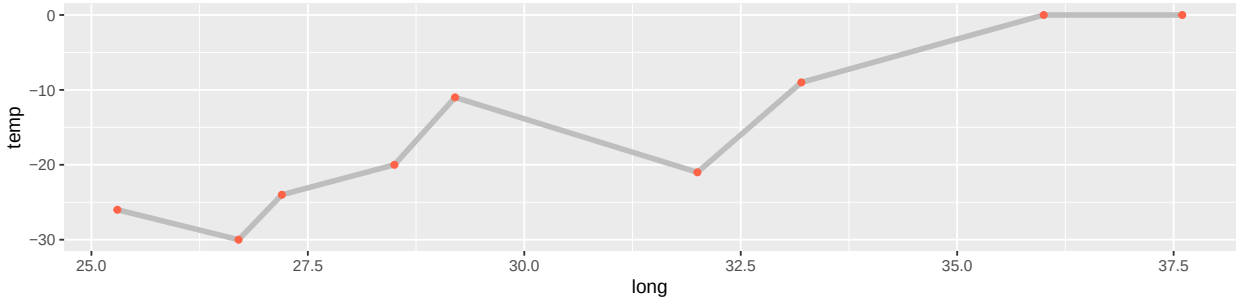
```
march.1812.ne.europe <- c(left = 23.5, bottom = 53.4, right = 38.1, top = 56.3)

march.1812.ne.europe.map <- get_stamenmap(bbox = march.1812.ne.europe,
                                          zoom = 8,
                                          maptype = "terrain-background",
                                          where = "cache")
```

```
(march.1812.plot <- ggmap(march.1812.ne.europe.map) +
   geom_path(data = troops,
             aes(x = long, y = lat, group = group,
                 color = direction, size = survivors),
             lineend = "round") +
   scale_size(range = c(0.5, 10)) +
   scale_colour_manual(values = c("lightgoldenrod3", "gray13")) +
   guides(color = FALSE, size = FALSE) +
   theme_nothing() +
   geom_point(data = cities,
              aes(x = long, y = lat),
              color = "mediumvioletred") +
   geom_text_repel(data = cities,
                   aes(x = long, y = lat, label = city),
                   color = "mediumvioletred"))
```



Vamos adicionar as temperaturas ao gráfico.

```
(t1 <- ggplot(temps, aes(long, temp)) +
    geom_path(color="grey", size=1.5) +
    geom_point(size=1.5, color = "tomato"))
```



- Aqui, vamos utilizar a função 'mutate' da biblioteca "dplyr" para adicionar uma coluna chamada 'label' à base de dados das temperaturas, que vai conter as temperaturas na forma que devem aparecer no gráfico;

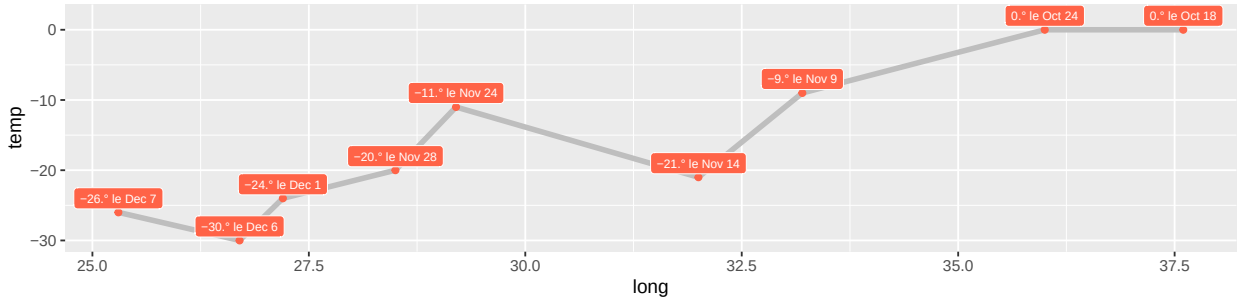
```
temps <- mutate(temps, label = paste0(temp, ".° le ", month, " ", day))
```

- Vejamos o resultado;

```
kable(select(head(temps),temp, date, label),
      col.names = c("Temperatura", "Data", "label"),
      align = 'c')
```

Temperatura	Data	label
0	1812-10-18	0.° le Oct 18
0	1812-10-24	0.° le Oct 24
-9	1812-11-09	-9.° le Nov 9
-21	1812-11-14	-21.° le Nov 14
-11	1812-11-24	-11.° le Nov 24
-20	1812-11-28	-20.° le Nov 28

```
(t2 <- t1 +
  geom_label(aes(label=temps$label),
             color = "white",
             fill = "tomato",
             size=2.6, nudge_y = 2))
```




```
(t3 <- t2 +
  coord_cartesian(xlim = c(24, 38), ylim = c(-35, 5)) +
  labs(x = NULL, y="° Celsius") +
  scale_y_continuous(position = "right") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid.minor.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        panel.border = element_blank()))
```

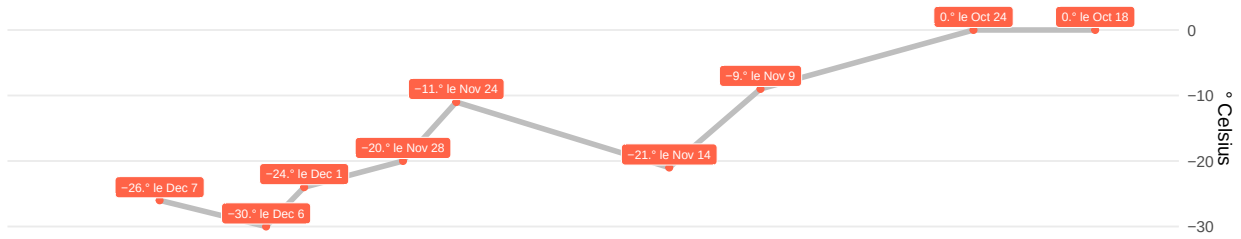
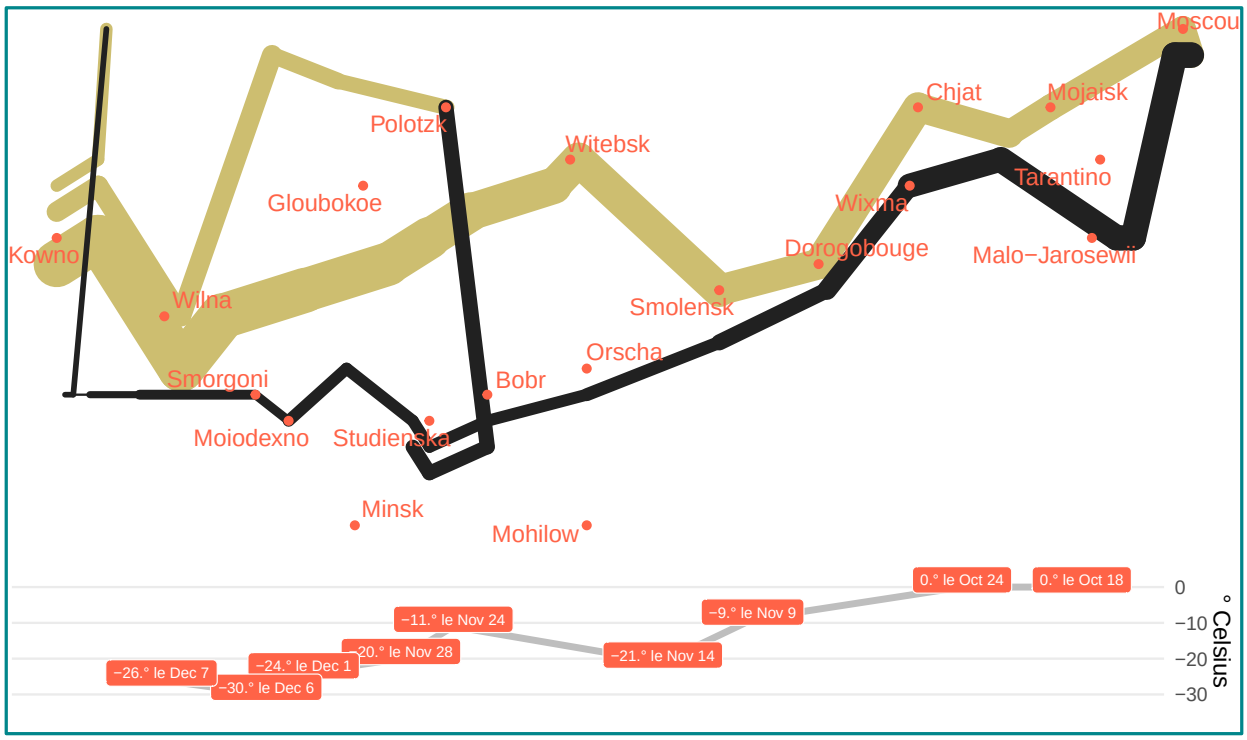


Gráfico final

```
grid.arrange(p4, t3, nrow=2, heights=c(3.5, 1.2))
grid.rect(width = .99, height = .99,
          gp = gpar(lwd = 2, col = "turquoise4", fill = NA))
```



Problema 3: Gapminder

Faça a animação presente em https://towardsdatascience.com/how-to-build-animated-charts-like-hans-rosling-doing-it-all-in-r-570efc6ba382 seguindo os passos lá descritos.

Resposta:

Gapminder

Logo de início, a biblioteca 'xlsx', indicada para realizar a importação dos dados disponibilizados, não funcionou de maneira alguma, mesmo depois de pesquisas e tentativas de solucionar o problema. Então, a solução encontrada por mim foi transformar as planilhas em csv e usar a função 'read.csv()' do R para seguir. Mas como os dados vieram tendo como separador decimal a vírgula, a função necessária foi a 'read.csv2()' e isso resolveu. $ $

Então vamos carregar os arquivos csv contendo a população, fertilidade, e expectativa de vida dos países ao longo dos anos.

```
population_csv <- read.csv2("./Databases/Gapminder/indicator gapminder population.CSV",
                            sep = ";", stringsAsFactors = FALSE)

fertility_csv <- read.csv2("./Databases/Gapminder/indicator undata total_fertility.CSV",
                           sep = ";", stringsAsFactors = FALSE)

lifeexp_csv <- read.csv2("./Databases/Gapminder/indicator life_expectancy_at_birth.CSV",
                         sep = ";", stringsAsFactors = FALSE)
```

```
#Vamos criar uma variável que contenha apenas os anos de 1962 a 2015
myvars <- paste("X", 1962:2015, sep="")

#Agora vamos usar nossa variável para selecionar apenas as colunas correspondentes
```

```r
#aos anos que a gente quer.
population <- population_csv %>% select(Total.population, all_of(myvars))

fertility <- fertility_csv %>% select(Total.fertility.rate, all_of(myvars))

lifeexp <- lifeexp_csv %>% select(Life.expectancy, all_of(myvars))
```

```r
#Renomeando a primeira coluna de todas as tabelas para "Country".
colnames(population)[1] <- "Country"
colnames(fertility)[1] <- "Country"
colnames(lifeexp)[1] <- "Country"

#Temos que remover as linhas vazias que foram criadas. Para isso, vamos manter apenas
#as primeiras 275 linhas de duas tabelas que precisam disso.
lifeexp <- lifeexp[1:275,]
population <- population[1:275,]
```

```r
#Vamos usar uma função da biblioteca "reshape" para transformar as colunas dos anos
#em linhas após o nome de cada país.
population_m <- melt(population, id=c("Country"))
lifeexp_m <- melt(lifeexp, id=c("Country"))
fertility_m <- melt(fertility, id=c("Country"))
```

```r
#Vamos nomear as nossas novas colunas criadas, que agora funcionam como indicador.
colnames(population_m)[3] <- "pop"
colnames(lifeexp_m)[3] <- "life"
colnames(fertility_m)[3] <- "fert"

#Transformando as três tabelas em apenas uma, com a função 'merge' juntando duas por vez.
mydf <- merge(lifeexp_m, fertility_m, by=c("Country","variable"), header =T)
mydf <- merge(mydf, population_m, by=c("Country","variable"), header =T)
```

```r
#Para poder mapear cada país por continente usando cores diferentes para identificarmos
#cada continente, vamos usar a biblioteca "gapminder":
continent <- gapminder %>% group_by(continent, country) %>% distinct(country, continent)

continent <- data.frame(lapply(continent, as.character), stringsAsFactors=FALSE)

colnames(continent)[1] <- "Country"

#Vamos usar o "dplyr" para filtrar nossa tabela 'mydf' e fazer com que ela contenha
#apenas os países que existem na tabela 'continent' do "gapminder":
mydf_filter <- mydf %>% filter(Country %in% unique(continent$Country))

#E então vamos juntar as tabelas 'continent' e 'mydf_filter' para que adicionemos
#os países aos respectivos continentes a que pertencem:
mydf_filter <- merge(mydf_filter, continent, by=c("Country"), header =T)
```

```r
#Por fim, vamos fazer um último trabalho de limpeza, como por exemplo remoção de
#valores coagidos como NA, transformar variáveis do tipo "Factor", etc..

#Eliminando os NA's
mydf_filter[is.na(mydf_filter)] <- 0

#Alterando a coluna 'variable' de modo a remover a letra "X" antes do ano e depois
```

```r
#transformá-lo em um número inteiro.
mydf_filter$variable <- as.integer(as.character(gsub("X", "", mydf_filter$variable)))

#Mudar seu nome para "year".
colnames(mydf_filter)[colnames(mydf_filter)=="variable"] <- "year"

#Ao invés de pegarmos a população total, vamos trabalhar com a população por milhões de
#habitantes e apenas uam casa decimal:
mydf_filter$pop <- round(as.numeric(as.character(mydf_filter$pop))/1000000, 1)

#Demonstração de como ficou a tabela final:
kable(head(mydf_filter[805:810,]),
      col.names = c("País", "Ano", "Expectativa de Vida", "Taxa de Fertilidade",
                    "População (em milhões)", "Continente"),
      align = 'c')
```

	País	Ano	Expectativa de Vida	Taxa de Fertilidade	População (em milhões)	Continente
805	Brazil	2010	73.6	1.84	198.6	Americas
806	Brazil	2011	73.8	1.82	200.5	Americas
807	Brazil	2012	74.0	1.81	202.4	Americas
808	Brazil	2013	74.1	1.80	204.3	Americas
809	Brazil	2014	74.3	1.79	206.1	Americas
810	Brazil	2015	74.4	1.78	207.8	Americas

E, finalmente, o Gráfico!

```r
# Criar um tema global para o nosso 'plot'
theme_set(theme_grey() +
            theme(legend.box.background = element_rect(),
                  legend.box.margin = margin(6, 6, 6, 6)))


# O nosso gráfico vai conter os anos como cada frame da animação, limitando o eixo y
#(da expectativa de vida) de 30 anos até 100, para termos uma boa visualização.
p <- ggplot(mydf_filter, aes(fert, life, size = pop, color = continent, frame = year)) +
  labs(x="Fertility Rate",
       y = "Life expectancy at birth (years)",

####------------------------Pequena contribuição-------------------------------------####
       caption =
  "(Based on data from Hans Rosling - gapminder.com)\nReproduced by Germano Andrade",
 #-------------------------------------------------------------------------------------#
       color = 'Continent', size = "Population (millions)") +
  ylim(30,100) +
  geom_point() +
  scale_color_brewer(type = 'div', palette = 'Spectral') +
  # gganimate code
  ggtitle("Year: {frame_time}") +
  transition_time(year) +
  ease_aes("linear") +
  enter_fade() +
  exit_fade()
```

```r
# animate
#animate(p, width = 450, height = 450)

# save as a GIF
#anim_save("Gapminder.gif")
```

```r
# Criar um gráfico para o plotly
p <- ggplot(mydf_filter, aes(fert, life, size = pop, color = continent, frame = year)) +
  geom_point() +
  ylim(30,100) +
  labs(x="Fertility Rate",
       y = "Life expectancy at birth (years)",
       color = 'Continent',
       size = "Population (millions)") +
  scale_color_brewer(type = 'div',
                     palette = 'Spectral')
# Gerar a Visualização e a saída em HTML para ser salva;
ggp <- ggplotly(p, height = 900, width = 900) %>%
  animation_opts(frame = 100,
                 easing = "linear",
                 redraw = FALSE)
ggp
```

```r
htmlwidgets::saveWidget(ggp, "Gapminder - Plotly.html")
```

Problema 4: Atirei o pau no gráfico

Assista o vídeo https://www.youtube.com/watch?v=CJkzf4IZRuk em que o autor realiza um gráfico em Excel. Seguindo os mesmos passos, faça esse gráfico em R.

Resposta:

Atirei o pau no gráfico

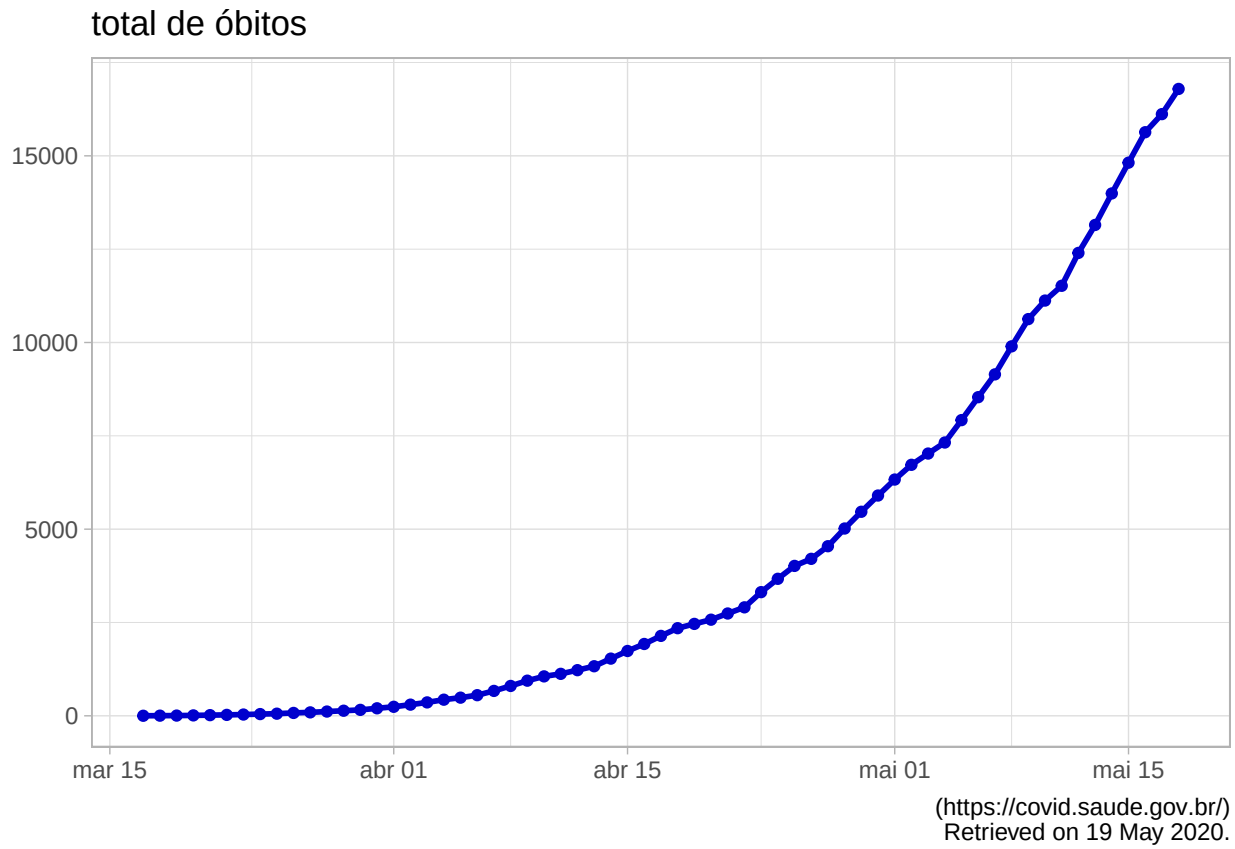
```r
path = "./Databases/Atirei o Pau no Gráfico/HIST_PAINEL_COVIDBR_18mai2020.CSV"
tab_covid <- read.csv(path,
                      sep = ";",stringsAsFactors = FALSE)
```

```r
# Alterando a coluna 'data' para que o R reconheça como uma data:
tab_covid$data <- as.Date(tab_covid$data, "%d/%m/%y")
```

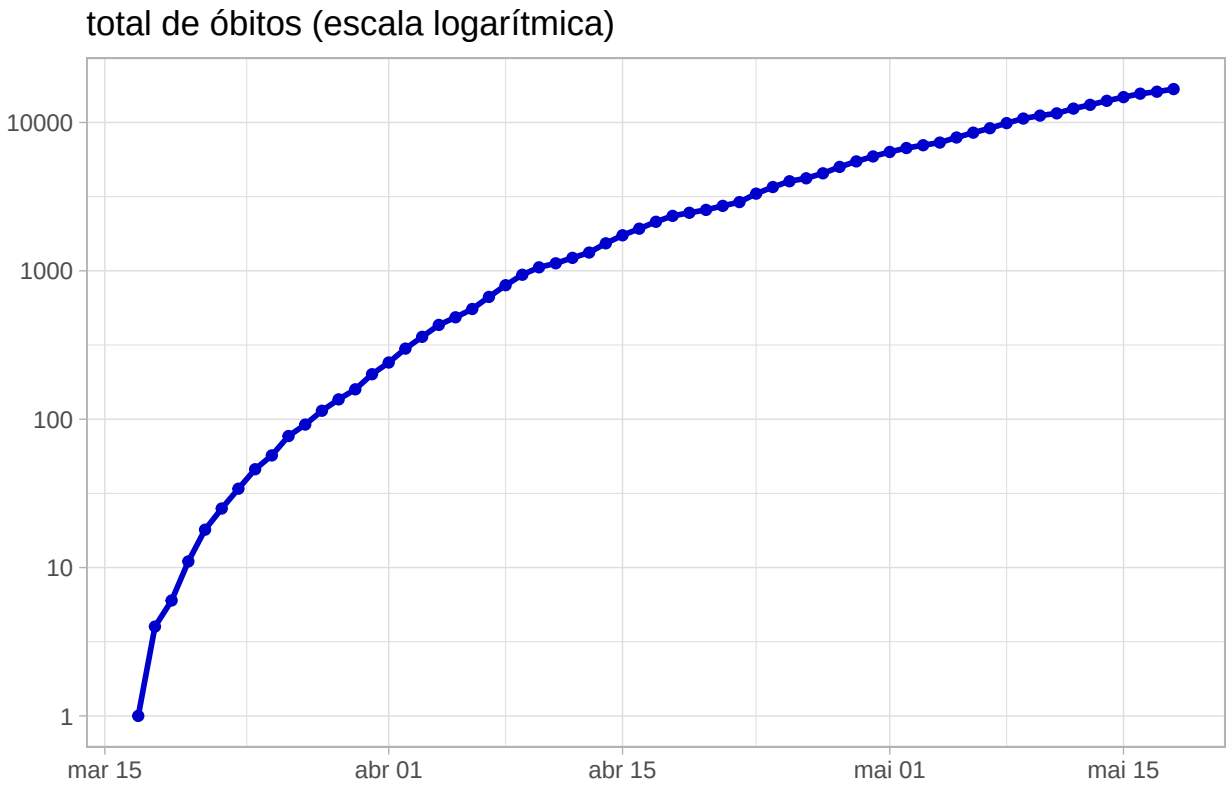
```r
# Filtrando os óbitos acumulados por data até o dia 18/05/2020:
Sars_CoV_2 <- tab_covid %>%
  group_by(data, obitosAcumulado) %>%
  distinct(data, obitosAcumulado)

Sars_CoV_2 <- Sars_CoV_2[21:83,]
```

```r
(c1 <- ggplot(Sars_CoV_2, aes(x = data, y = obitosAcumulado)) +
  geom_point(color = "mediumblue") +
  geom_path(color = "mediumblue", size = 1) +
  theme_light() +
  labs(title = "total de óbitos",
```

```
       x = NULL, y = NULL,
       caption = "(https://covid.saude.gov.br/)\nRetrieved on 19 May 2020."))
```




```
(c1 +
  scale_y_continuous(trans = 'log10') +
  labs(title = "total de óbitos (escala logarítmica)"))
```



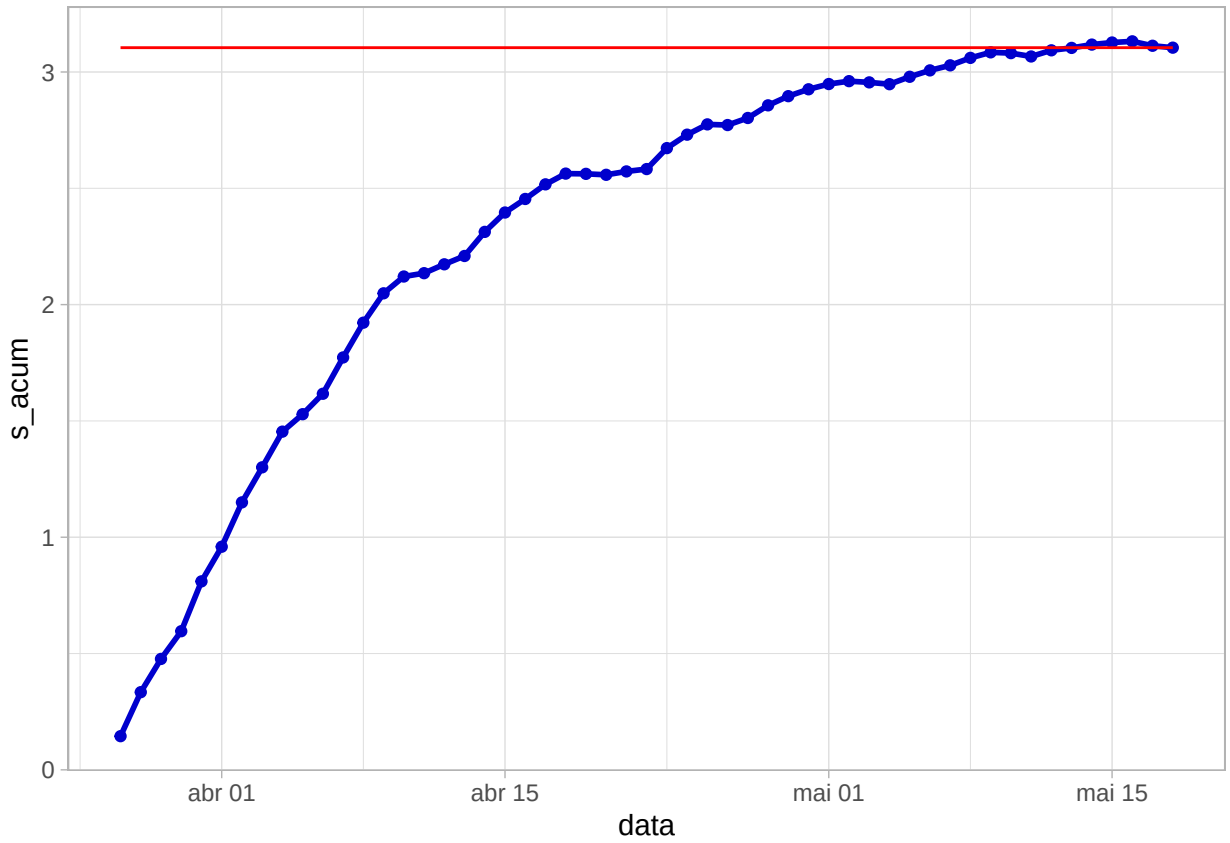
(https://covid.saude.gov.br/)
Retrieved on 19 May 2020.

```
# taxa de crescimento = valor no dia/valor no dia anterior
dias <- 63
taxa <- vector(length = dias)# Criando um vetor vazio;
taxa[1] <- 0
for (i in 1:(dias-1)){
  taxa[i+1] <- (Sars_CoV_2$obitosAcumulado[i+1]/Sars_CoV_2$obitosAcumulado[i])
}

# Adicionando a coluna da taxa de crescimento
Sars_CoV_2 <- data.frame(data = Sars_CoV_2$data,
                         obitosAcumulado = Sars_CoV_2$obitosAcumulado,
                         taxa = taxa)

#Criando uma base que contenha apenas os dados a partir do dia 27/03/2020, porque é
#mais interessante observar a taxa de crescimento nesse período:
Sel_Sars <- Sars_CoV_2[11:63,]
```

```
ay <- list(
  tickfont = list(color = "red"),
  overlaying = "y",
  side = "right",
  title = "Taxa"
)
fig <- plot_ly()
fig <- fig %>% add_lines(data = Sel_Sars,
                         x = Sel_Sars$data,
```

```
                          y = Sel_Sars$obitosAcumulado,
                          yaxis = "y1",
                          name = "Óbitos acumulados")

fig <- fig %>% add_lines(x = Sel_Sars$data,
                          y = Sel_Sars$taxa,
                          name = "Taxa de crescimento",
                          yaxis = "y2")

fig <- fig %>% layout(
    title = "total de óbitos",
    yaxis = list(type = "log", title = "Escala logarítmica"),
    yaxis2 = ay,
    xaxis = list(title=NULL)
  )

fig
```

```
#Calculando a soma acumulada:
dias <- 53
media <- 1.05
s_acum <- vector(length = dias)# Criando um vetor vazio;
s_acum[1] <- Sel_Sars$taxa[1]-media
for (i in 1:(dias-1)){
  s_acum[i+1] <- (Sel_Sars$taxa[i+1]-media+s_acum[i])
}
#s_acum

#Sel_Sars <- data.frame(data = Sel_Sars$data,
                        # obitosAcumulado = Sel_Sars$obitosAcumulado,
                         #taxa = Sel_Sars$taxa,
                         #soma = s_acum)

ggplot(Sel_Sars, aes(x = data, y = s_acum)) +
  geom_point(color = "mediumblue") +
  geom_path(color = "mediumblue", size = 1) +
  geom_line(aes(y = s_acum[53]), col = "red") +
  theme_light()
```




```
data <-  c("19/05/2020", "20/05/2020", "21/05/2020", "22/05/2020", "23/05/2020",
           "24/05/2020", "25/05/2020", "26/05/2020", "27/05/2020",
           "28/05/2020", "29/05/2020", "30/05/2020", "31/05/2020")
taxa_1 <- rep(media, 13)
```

```
m <- 13
casos <- vector(length = m)# Criando um vetor vazio;
casos[1] <- Sel_Sars$obitosAcumulado[53]*media
for (i in 1:(m-1)){
  casos[i+1] <- (casos[i]*media)
}

casos <- round(casos,0)
```

```
Previ <- data.frame("data" = data,
                    "obitosAcumulado" = casos,
                    "taxa" = taxa_1, stringsAsFactors = FALSE)

Previ$data <- as.Date(Previ$data, "%d/%m/%y")

Previs <- rbind(Sel_Sars, Previ)
```

```
ggplot(Previs, aes(x = data, y = obitosAcumulado)) +
  geom_point(color = "mediumblue") +
  geom_path(color = "mediumblue", size = 1) +
  theme_light() +
  labs(title = "total de óbitos",
```

```
      x = NULL, y = NULL,
      caption = "(https://covid.saude.gov.br/)\nRetrieved on 19 May 2020.") +
scale_y_continuous(trans = 'log10') +
labs(title = "Previsão de Óbitos (escala logarítmica)")
```