Análise de imagem no Rstudio

Ana, João, Laura, Leonardo e Paulo

20 de novembro de 2019

Formatos de imagens

- TIFF
- BMP
- JPEG
- PNG
- SVG
- GIF
- PDF
- EPS

A importância de análise de imagens

Como os pacotes lêem as imagens?

Pacotes

Oos principais pacotes para manipulação de imagem são:

```
require("BiocManager")
require("EBImage") # JPEG(JPG), PNG E TIFF


require("imager") # JPEG(JPG), PNG E BMP


require("magick")
```

Importação e vizualização de imagens:

- EBImage:

.ima <- readImage("C:/Users/nick_/Downloads/897207.jpg")

.display(ima)

- Imager:

.ima_1 <- load.image("C:/Users/nick_/Downloads/897207.jpg")

.plot(ima_1)

- Magick:

.ima_2 <- image_read("C:/Users/nick_/Downloads/897207.jpg")

.print(ima_2)

Mudar dimensões

```
tigre <- image_read_svg('http://jeroen.github.io/images/tiger.sv
tigre
```

Mudar dimensões

```
tigre2 <- image_read_svg('http://jeroen.github.io/images/tiger.s
                         width = 120) # 120 = width, 120x = heig
tigre2
```

```
tigre_redimensionada <- image_scale(tigre, "120x120")
```

Converter ou salvar em formatos desejados

```
tigre_convertido <- image_convert(tigre, "jpeg")
image_info(tigre_convertido) # Retorna o formato da imagem
```

```
##   format width height colorspace matte filesize density
## 1   JPEG   900    900       sRGB  TRUE        0   72x72
```

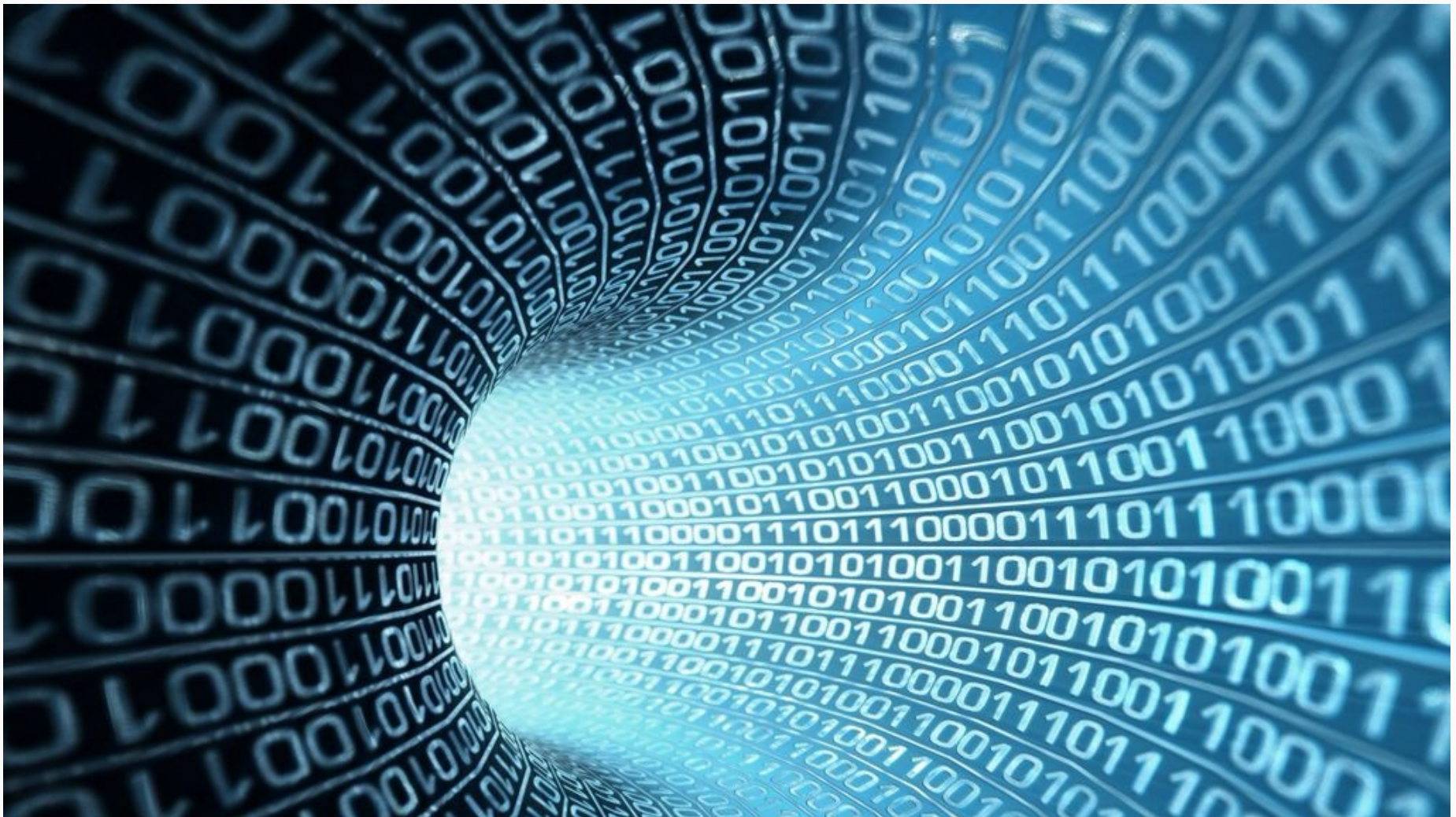
```
image_write(tigre, path = "tiger.png", format = "png")
```

Imagens para manipulação (Frink)

```
frink <- image_read("C:/Users/nick_/OneDrive/Área de Traba
frink
```

Imagens para manipulação (BigData)

```
bigdata <- image_read('C:/Users/nick_/OneDrive/Área de Trab

bigdata
```



Imagens para manipulação (R)

```
logo <- image_read('C:/Users/nick_/OneDrive/Área de Trabalh

logo
```

Girar e modificar

```
image_flop(frink)
```

```
image_flip(frink)
```

```
image_rotate(frink, 45)
```

```
image_crop(frink, "100x150+50")
```

Fonte: https://cran.r-project.org/web/packages/magick/vignettes/intro.html

Alguns tipos de filtros

```
image_blur(frink, 10, 5)
```

```
image_charcoal(frink)
```

```
image_oilpaint(frink)
```

```
image_negate(frink)
```

```
image_modulate(frink, brightness = 80, saturation = 120, hue = 90)
```

```
image_fill(frink, "orange", point = "+100+200", fuzz = 20)
```

Fonte: https://cran.r-project.org/web/packages/magick/vignettes/intro.html

Imagens sobrepostas

```
img <- c(bigdata, logo, frink)
img <- image_scale(img, "300x300")
image_info(img)
```

```
##   format width height colorspace matte filesize density
## 1   JPEG   300    225        sRGB FALSE        0   72x72
## 2    PNG   300    232        sRGB  TRUE        0   72x72
## 3    PNG   148    300        sRGB  TRUE        0   72x72
```

Imagens sobrepostas

```
image_append(image_scale(img, "100"), stack = TRUE)
```

```
image_append(image_scale(img, "x200"))
```

```
image_mosaic(img)
```

```
image_flatten(img, 'Minus')
```

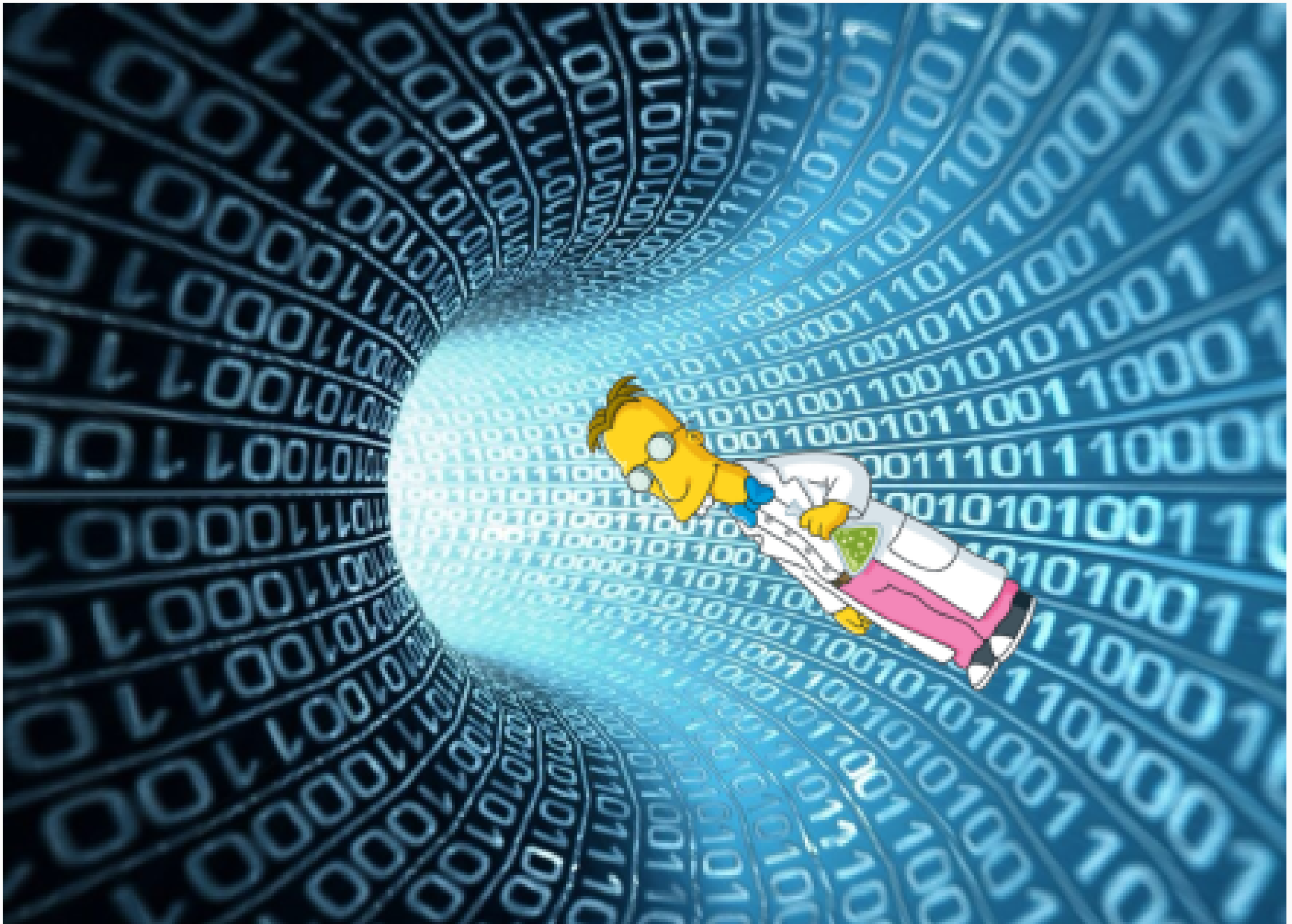
```
image_flatten(img)
```

Fonte: https://cran.r-project.org/web/packages/magick/vignettes/intro.html

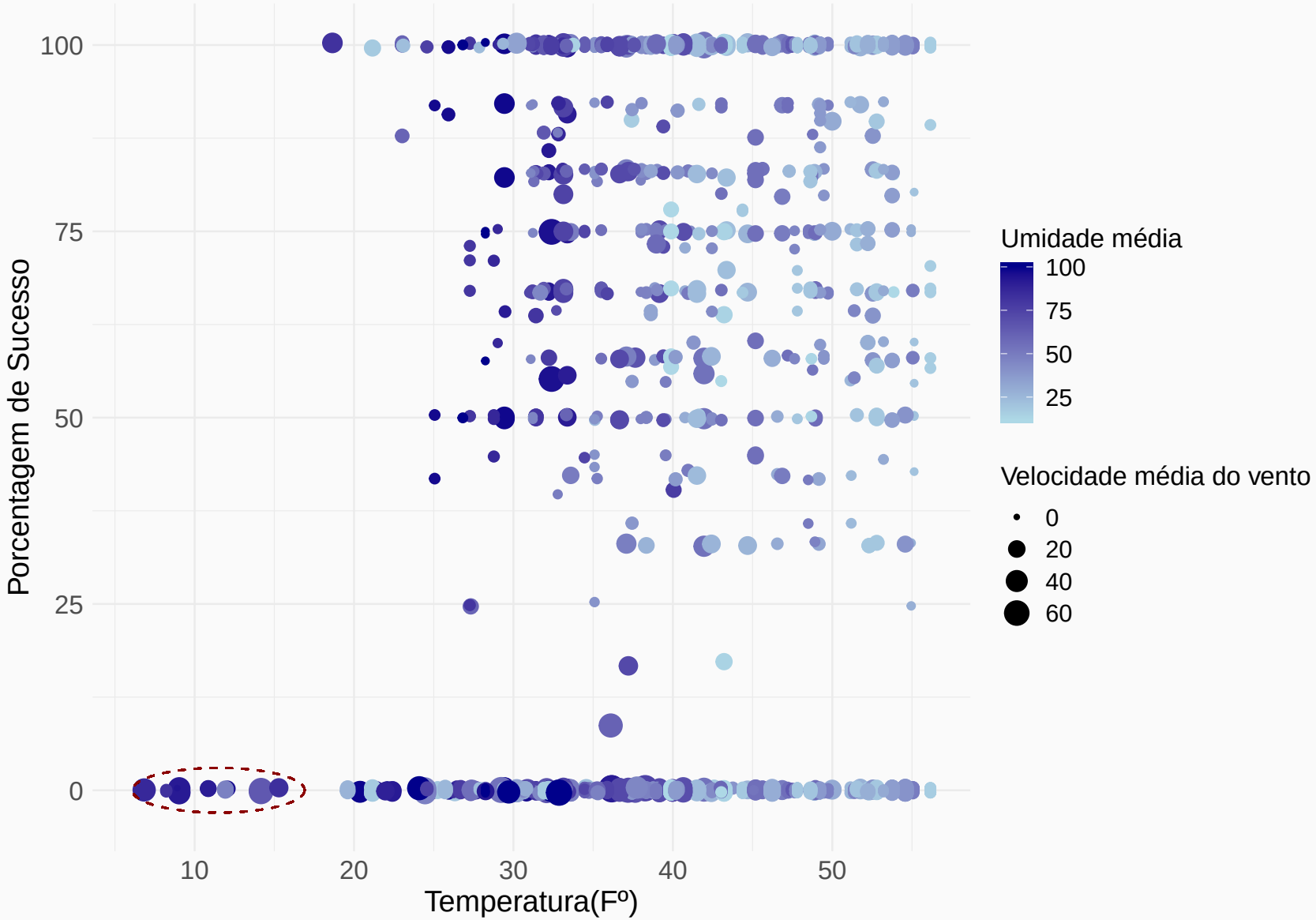
Imagens sobrepostas

```
bigdatafrink <- image_scale(image_rotate(
  image_background(frink, "none"), 300), "x160")
juntos <-image_composite(image_scale(
  bigdata, "x330"), bigdatafrink, offset = "+180+100")
```

Imagens sobrepostas



Utilidade em gráficos

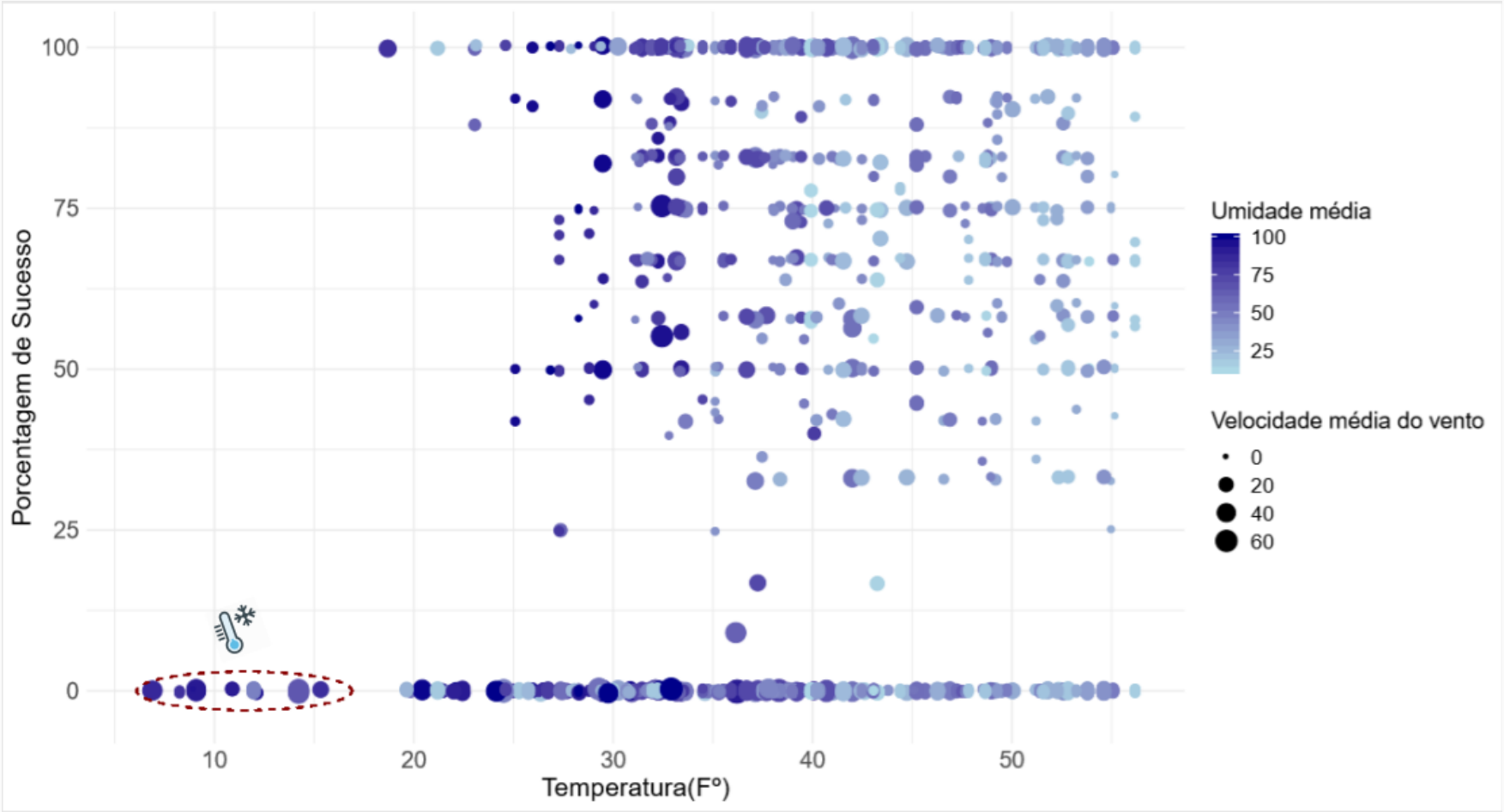


Utilidade para sobreposição de imagens

```
graph <- image_read("C:/Users/nick_/OneDrive/Área de Traba
temp <- image_read("C:/Users/nick_/OneDrive/Área de Trabal
temp_graph <- image_scale(image_rotate(image_background(
  temp, "none"), 340), "x50")
temp_graph
```

```
juntos_2 <-image_composite(image_scale(
  graph, "x600"), temp_graph, offset = "+150+440")
image_write(juntos_2, path = "juntos2.pdf", format = "pdf")
```

Utilidade para sobreposição de imagens

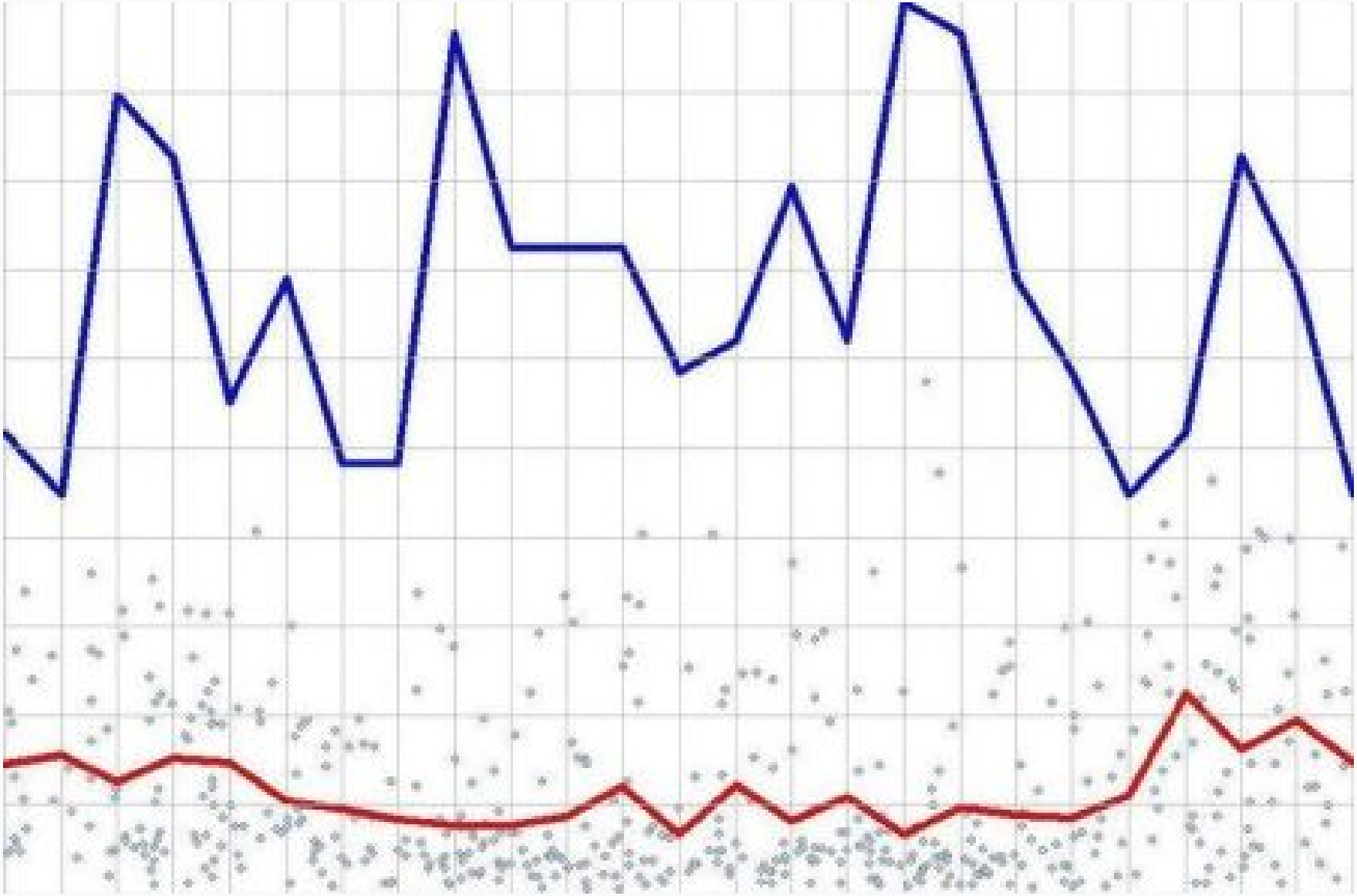


Anotações em imagens

```
frink_anot <- image_annotate(frink, "Aqui", size = 25,
                             color = "red",
                             boxcolor = "black",
                             degrees = 30,
                             location = "+150+310")

frink_anot <- image_scale(frink_anot, "x350")

image_write(frink_anot, path = "frink_anot.png", format = '
```

Anotações em imagens

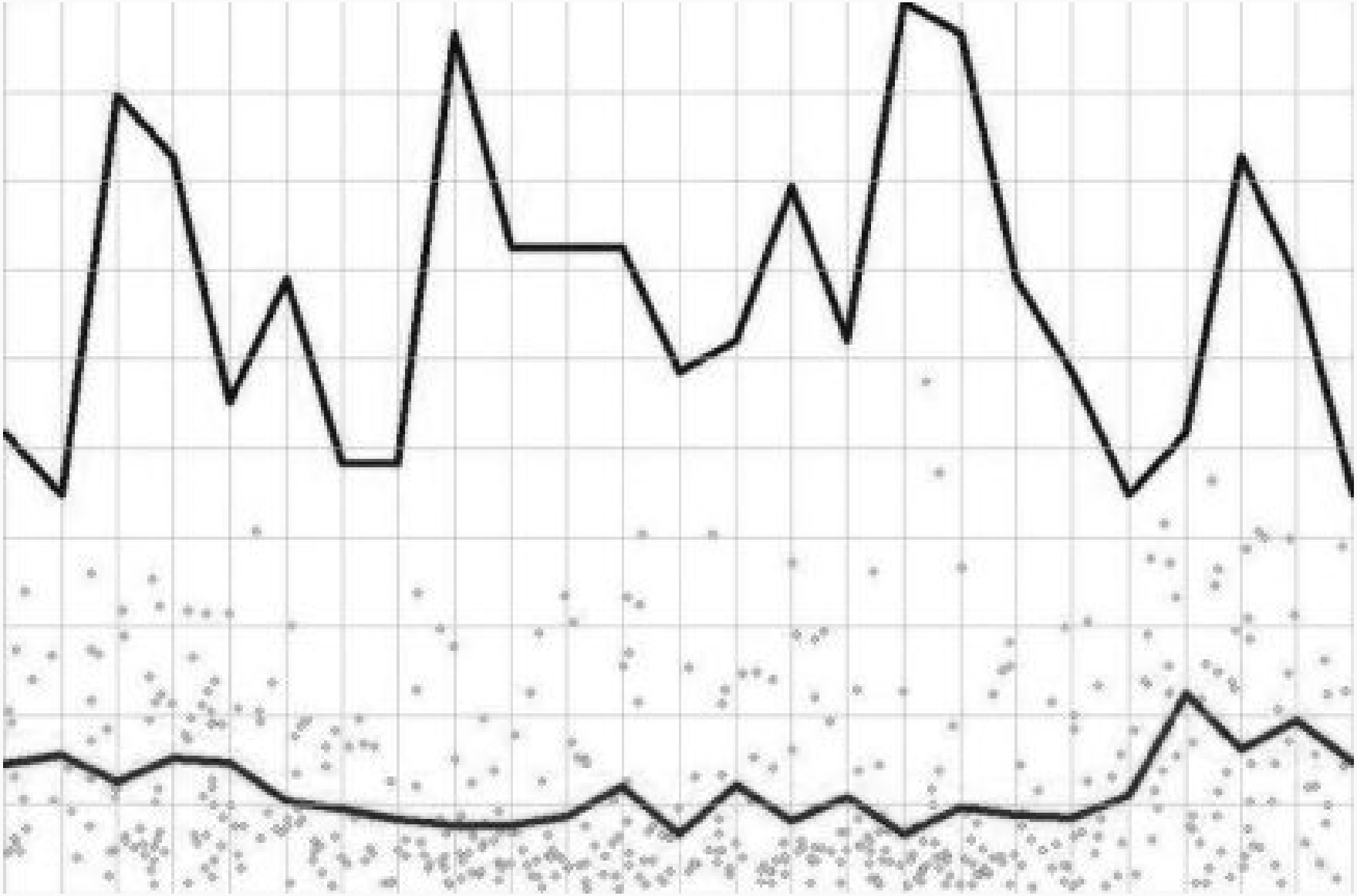
Como retirar pontos de um gráfico sem seu código?



Como retirar pontos de um gráfico sem seu código?

```
library(tidyverse)

im <- image_read("C:/Users/nick_/OneDrive/Área de Trabalho/

im_proc <- im %>%

    image_channel("saturation")

image_write(im_proc, path = "IMAGENS/grafico_ponto1.png", f
```

Como retirar pontos de um gráfico sem seu código?

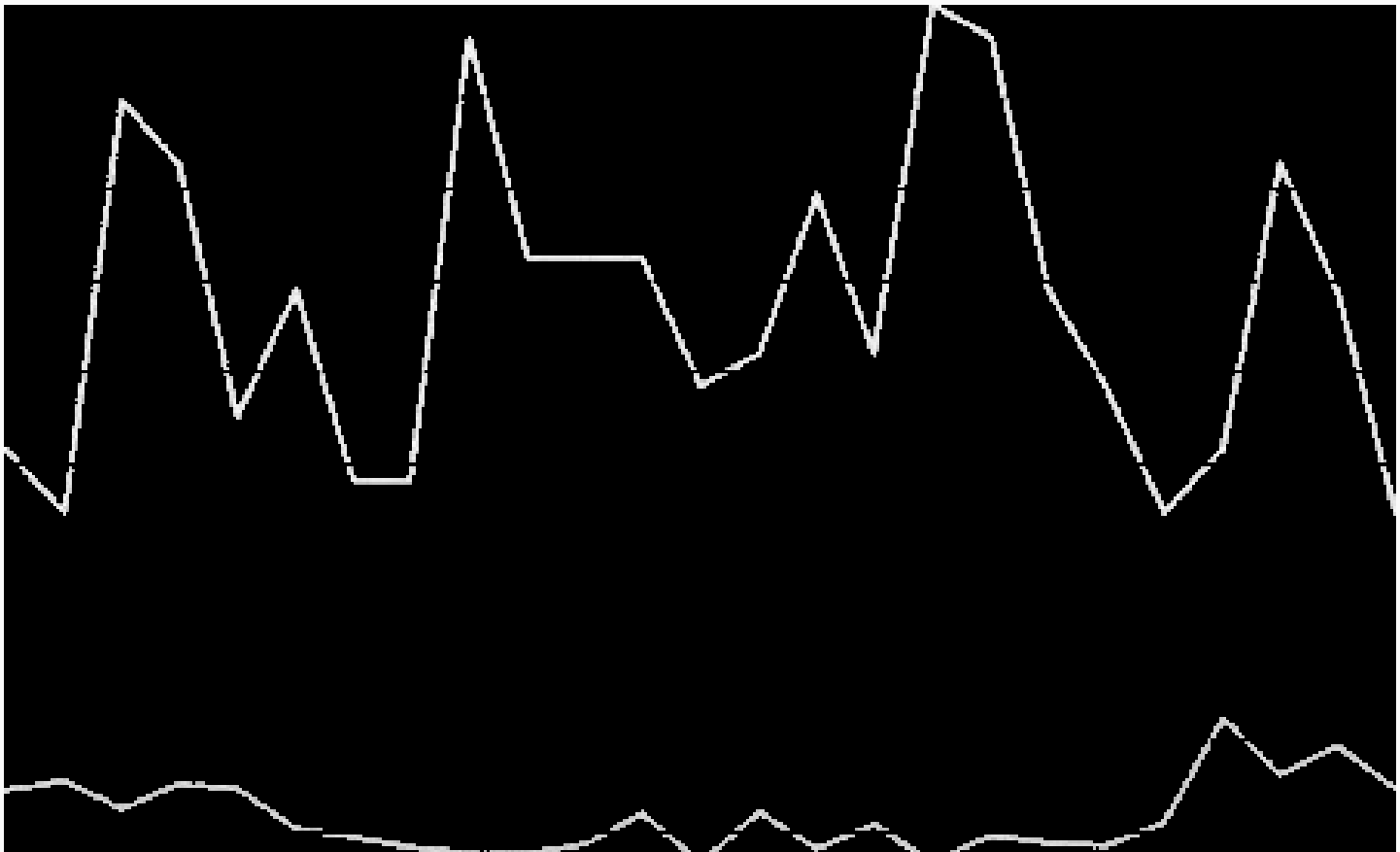


Como retirar pontos de um gráfico sem seu código?

```
im_proc2 <- im_proc %>%
    image_threshold("white", "30%")
image_write(im_proc2, path = "IMAGENS/grafico_ponto2.png",
            format = "png")
```

Como retirar pontos de um gráfico sem seu código?

```
im_proc3 <- im_proc2 %>%
    image_negate()
image_write(im_proc3, path = "IMAGENS/grafico_ponto3.png",
            format = "png")
```



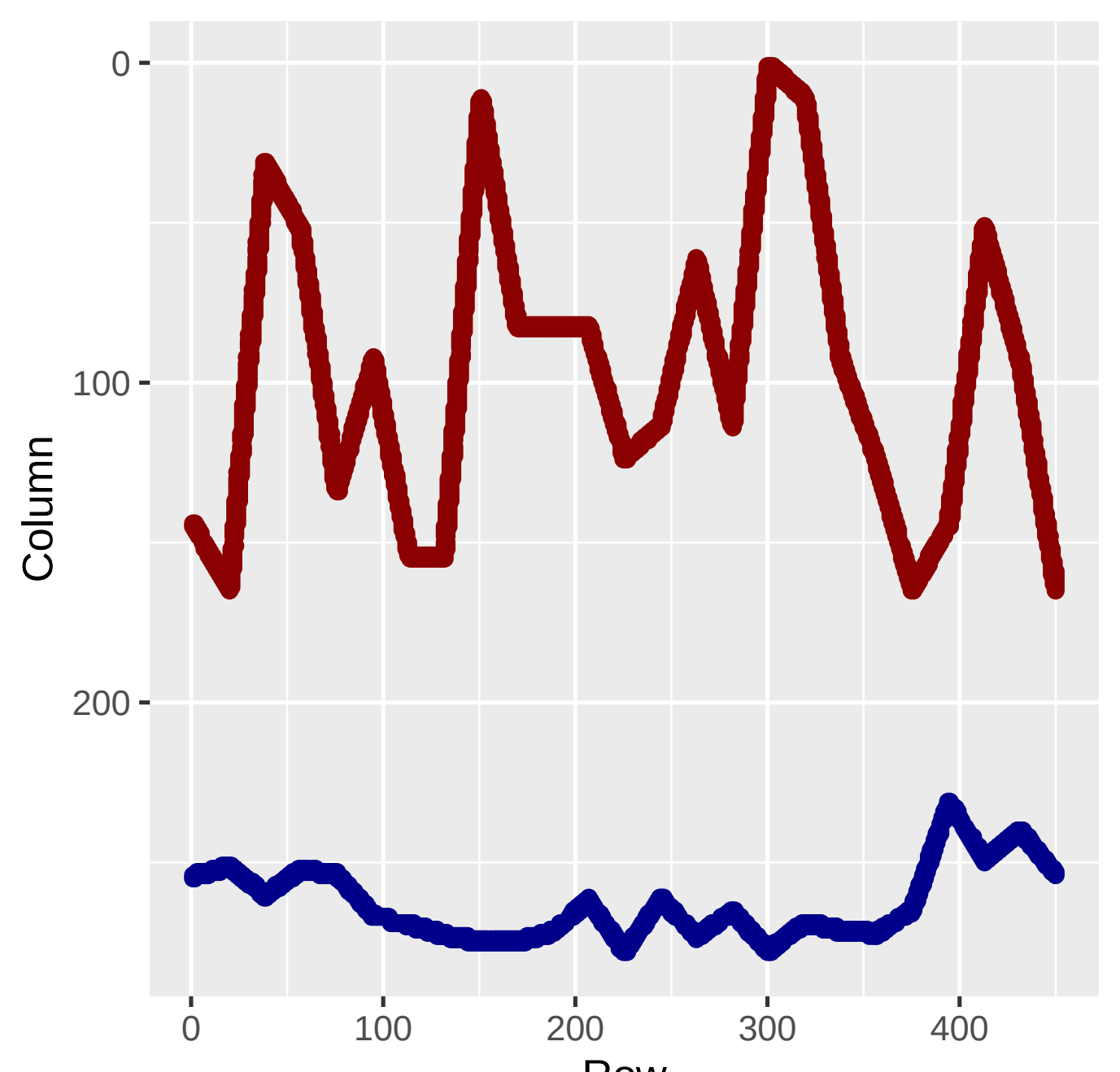
Como retirar pontos de um gráfico sem seu código?

```
require(tidyverse)
dat <- image_data(im_proc3)[1,,] %>%
    as.data.frame() %>%
    mutate(Row = 1:nrow(.)) %>%
    select(Row, everything()) %>%
    mutate_all(as.character) %>%
    gather(key = Column, value = value, 2:ncol(.)) %>%
    mutate(Column = as.numeric(gsub("V", "", Column)),
           Row = as.numeric(Row),
           value = ifelse(value == "00", NA, 1)) %>%
    filter(!is.na(value))
```

Como retirar pontos de um gráfico sem seu código?

```
require(ggplot2)
grafico_final <-ggplot(data = dat,
                       aes(x = Row,
                           y = Column,
                           colour = (Column < 200))) +
    geom_point() +
    scale_y_continuous(trans = "reverse") +
    scale_colour_manual(values = c( "blue4","red4")) +
    theme(legend.position = "off")+
  ggsave("grafico_final.pdf", width = 4, height = 4)
```

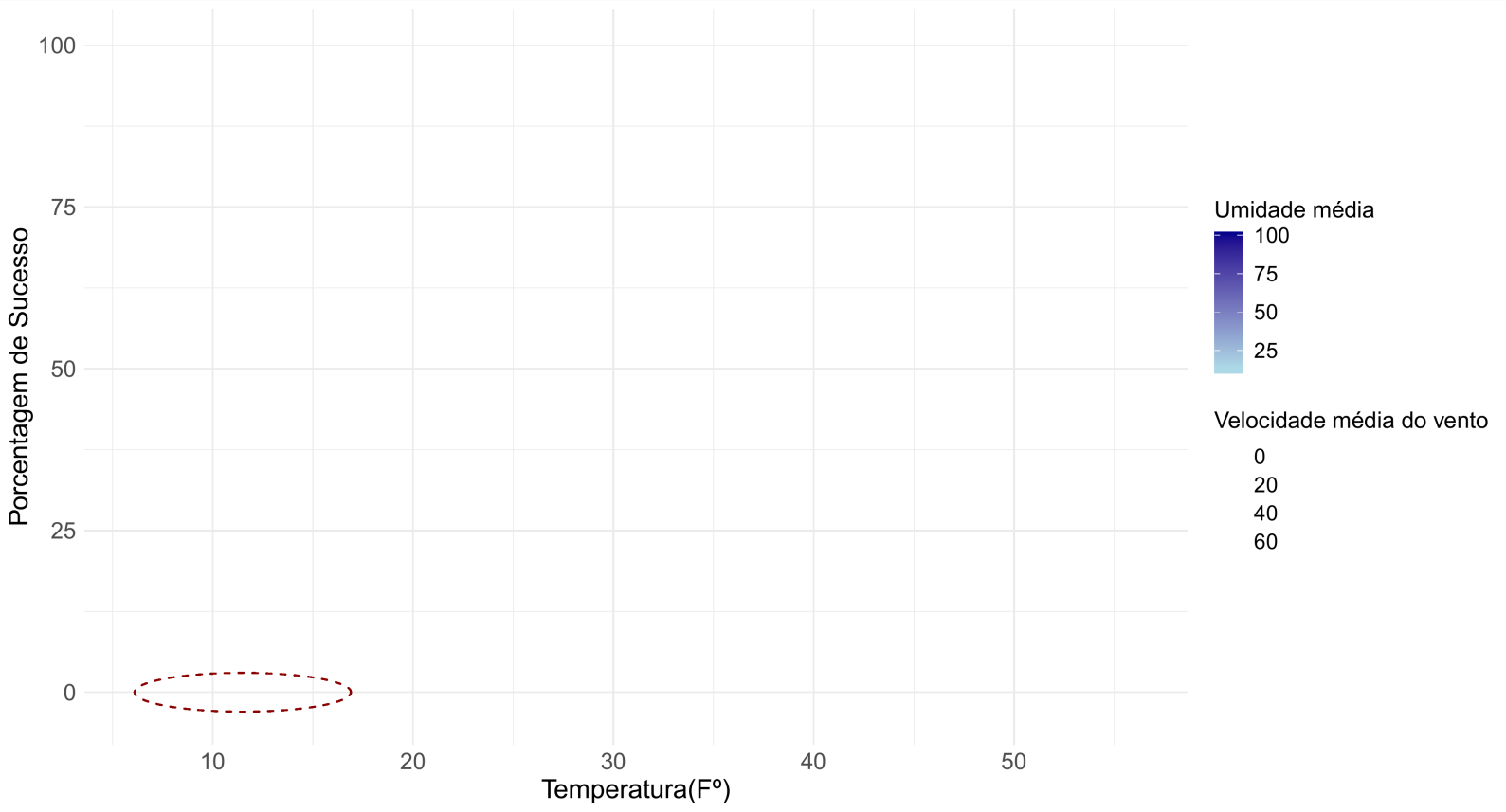
Como retirar pontos de um gráfico sem seu código?



Fraqueza na leitura de PDF

```
require(pdftools)

tempo <- image_read_pdf("C:/Users/nick_/OneDrive/Área de Tr

image_write(tempo, path = "tempo.pdf", format = "pdf")
```

Fraqueza na leitura de PDF



Gifs

```
earth <- image_read("https://jeroen.github.io/images/earth.
  image_scale("250x") %>%
  image_quantize()

length(earth)
```

```
## [1] 44
```

Como montar um GIF

1º - Importe as imagens:

⇒ im_1 <- image_read("C:/Users/nick_/Downloads/im_1.jpg")

⇒ im_n <- image_read("C:/Users/nick_/Downloads/im_n.jpg")

2º - Junte as imagens e redimensione:

⇒ img <- c(im_1, ... , im_n)

⇒ img <- image_scale(img, "300x300")

3º - Argumentos:

⇒ image_animate(img)

Salvar o GIF na máquina

```
library(gifski)


image_write_gif(img, path = "grafico.gif")
 # É possivel adicionar um delay
image_write_gif(img, path = "grafico_delay.gif",
                delay = 1/6)
```

ANA

```
# Download da imagem
file="http://ereaderbackgrounds.com/movies/bw/Frankenstein.jpg"
download.file(file, destfile = "frankenstein.jpg", mode = 'wb')
# Lê e converte para a escala de cinza
load.image("C:/Users/nick_/Downloads/Frankenstein.jpg") %>%
  grayscale() -> x
```

ANA

```
# Isso é apenas para definir os limites dos frames
x %>%
  as.data.frame() %>%
  group_by() %>%
  summarize(xmin=min(x), xmax=max(x), ymin=min(y), ymax=max(y))
  as.vector()->rw
```

ANA

```
# Filtra a imagem e converte para preto e branco
x %>%
  threshold("45%") %>%
  as.cimg() %>%
  as.data.frame() -> df
```

ANA

```
# Função para calcular e plotar o diagrama de Voronoi, depende
# do tamanho da amostra
doPlot = function(n)
{
  #V Diagrama de Voronoi
  df %>%
    sample_n(n, weight=(1-value)) %>%
    select(x,y) %>%
    deldir(rw=rw, sort=TRUE) %>%
    .$dirsgs -> data
```

```
 # Isso é apenas para adicionar alguns alfas nas linhas, depende
# da longitude
  data %>%
    mutate(long=sqrt((x1-x2)^2+(y1-y2)^2),
           alpha=findInterval(long,
                              quantile(long,
                                       probs = seq(0,
                                                   1,
                                                   length.out =
                                       )
                              )/21)-> data
```

ANA

```
data %>%
    ggplot(aes(alpha=(1-alpha))) +
    geom_segment(aes(x = x1, y = y1, xend = x2, yend = y2),
                 color="black", lwd=1) +
    scale_x_continuous(expand=c(0,0))+
    scale_y_continuous(expand=c(0,0), trans=reverse_trans())+
    theme(legend.position  = "none",
          panel.background = element_rect(fill="white"),
          axis.ticks       = element_blank(),
          panel.grid       = element_blank(),
          axis.title       = element_blank(),
          axis.text        = element_blank())->plot
  return(plot)}
```

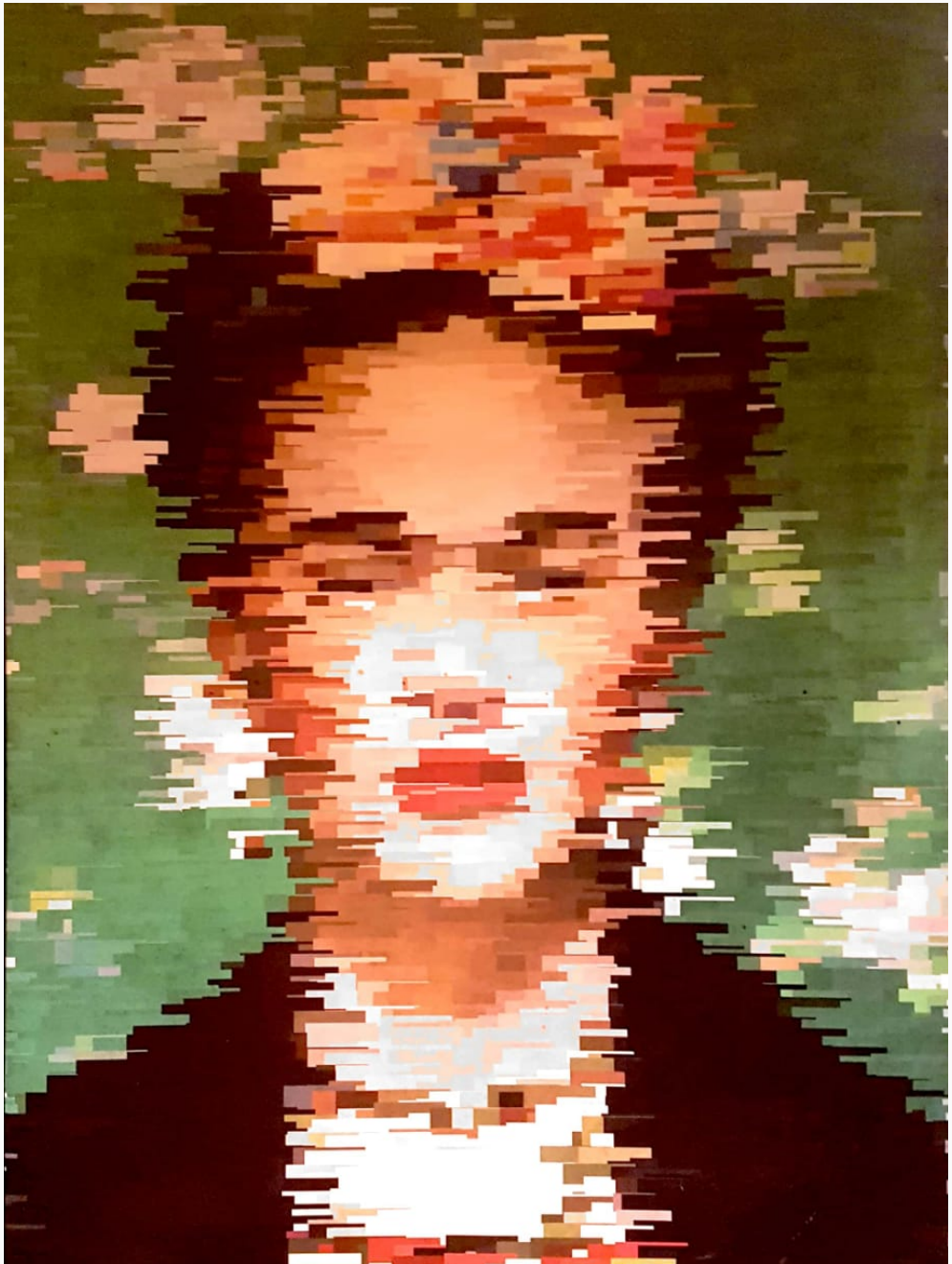
ANA

```
# Eu chamei a função anterior e salvei o resultado do plot em
# formato jpeg
i= 500
name=paste0("frankie",i,".jpeg")
jpeg(name, width = 600, height = 800, units = "px",
     quality = 100)
doPlot(i)
dev.off()
```

ANA

```
# Assim que todas as imagens são salvas, eu posso criar o GIF
library(magick)
frames=c()
images=list.files(pattern="jpeg")
for (i in length(images):1)
{
  x=image_read(images[i])
  x=image_scale(x, "300")
  c(x, frames) -> frames
}
animation=image_animate(frames, fps = 2)
image_write(animation, "Frankenstein.gif")
print(animation)
```

Filtro geométrico



Filtro geométrico

Por meio do pacote “imager” é possível estilizar uma imagem e deixá-la pixelizada:

```
library(imager)
foto <- load.image("C:/Users/nick_/Downloads/foto.jpg")


foto2<- foto %>%  resize(size_x = 80, size_y = 80,
                         interpolation_type = 1L)
suppressMessages(suppressWarnings(library(imager)))
foto2 <- rowMeans(foto2, dims = 2)
```

Filtro geométrico

```
foto2 %>%
  apply(1, rev) %>%
  t() %>%
  image(col = grey.colors(256), axes = FALSE)
```

Keras


TensorFlow

Anaconda

Reconhecimento de imagem - Aviões e carros

Reconhecimento de imagem - Pacotes

```
install.packages("tfestimators")

install_tensorflow()

devtools::install_github("rstudio/keras")

devtools::install_github("rstudio/tensorflow")

devtools::install_github("rstudio/keras")

reticulate::py_discover_config()

reticulate::use_condaenv("r-tensorflow")

reticulate::py_config()
```

Reconhecimento de imagem - Leitura

```
library(EBImage)

library(keras)

library(kerasR)

library(kerasformula)

setwd('C:\\Users\\Matheus\\Desktop\\Nova pasta\\diretorio F

pics <- c('p1.jpg', 'p2.jpg', 'p3.jpg', 'p4.jpg', 'p5.jpg',
          'p6.jpg','c1.jpg', 'c2.jpg', 'c3.jpg', 'c4.jpg',
          'c5.jpg','c6.jpg')

mypic <- list()

for (i in 1:12) {mypic[[i]] <- readImage(pics[i])}
```

Reconhecimento de imagem - Análise manual

```
print(mypic[[1]])

display(mypic[[12]])

summary(mypic[[1]])

hist(mypic[[2]])
```

Reconhecimento de imagem - Redimensionamento

```
for (i in 1:12) {mypic[[i]] <- resize(mypic[[i]],28, 28)}

for (i in 1:12) {mypic[[i]] <- array_reshape(mypic[[i]],
                                             c(28,28,3))}
```

Reconhecimneto de imagem - Remodelagem

```
trainx <- NULL
for (i in 1:5) {trainx <- rbind(trainx, mypic[[i]])}
for (i in 7:11) {trainx <- rbind(trainx, mypic[[i]])}
str(trainx)
testx <- rbind(mypic[[6]], mypic[[12]])
trainy <- c(0,0,0,0,0,1,1,1,1,1)
testy <- c(0, 1)
```

Reconhecimneto de imagem - Criação de labels

```
trainLabels <- to_categorical(trainy)

testLabels <- to_categorical(testy)
```

Reconhecimento de imagem - Modelo

```
model <- keras_model_sequential()

model %>%
        layer_dense(units = 256, activation = 'relu',
                    input_shape = c(2352)) %>%
        layer_dense(units = 128, activation = 'relu') %>%
        layer_dense(units = 2, activation = 'softmax')

summary(model)
```

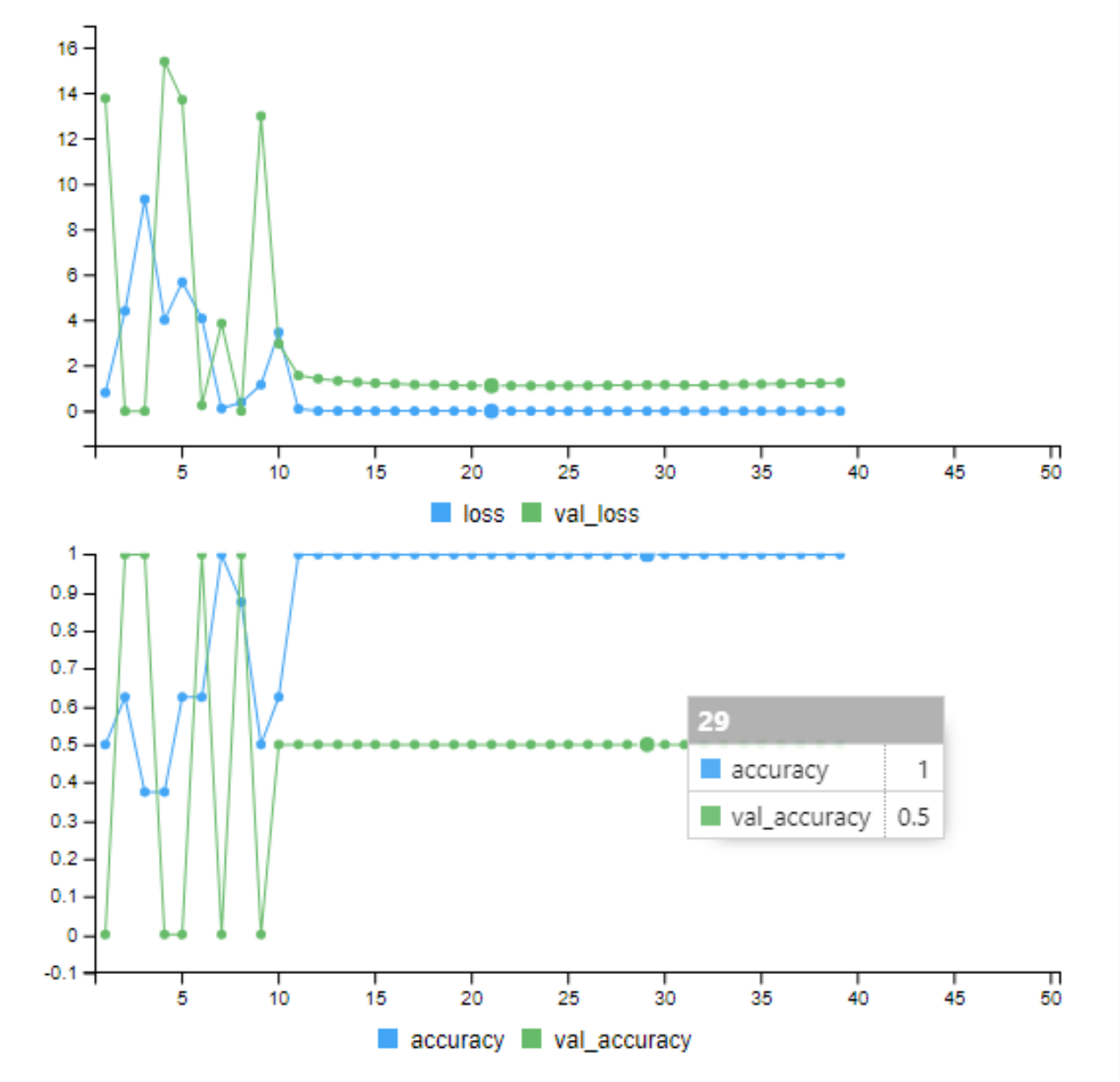
Reconhecimento de imagem - Compilação

```
model %>%
          compile(loss = 'binary_crossentropy',
                  optimizer = optimizer_rmsprop(),
                  metrics = 'accuracy')
```

Reconhecimento de imagem - Ajustando/ Treinando o modelo

```
history <- model %>%
          fit(trainx,
              trainLabels,
              epochs = 500,
              batch_size = 32,
              validation_split = 0.2)
```

Reconhecimento de imagem - Ajustando/ Treinando o modelo



Reconhecimento de imagem - Avaliação e previsão

```
model %>% evaluate(testx, testLabels)

pred <- model %>% predict_classes(trainx)

table(Predicted = pred, Actual = trainy)

prob <- model %>% predict_proba(trainx)

cbind(prob, Prected = pred, Actual= trainy)
```