

International Seminar
on Statistics with R

Visualização de dados
através do pacote

dubois

por Ícaro Bernardes
Maio de 2022



BIT::Analytics
{Past, present and future data-driven}

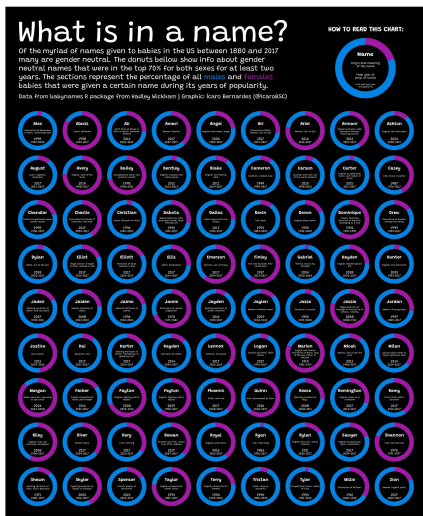
Bacharelado em
Eng. Química
Mestrando em
Eng. Industrial
Co-fundador e
cientista de dados

UFBA

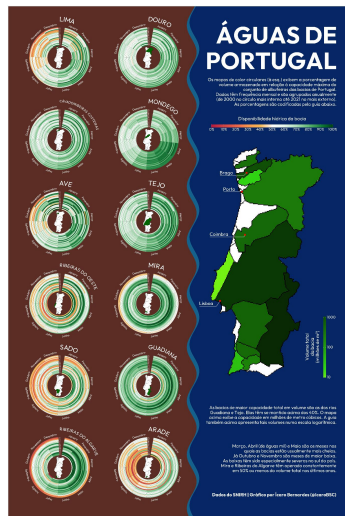
UFBA

BIT::Analytics

Afiliação e trabalhos em dataviz



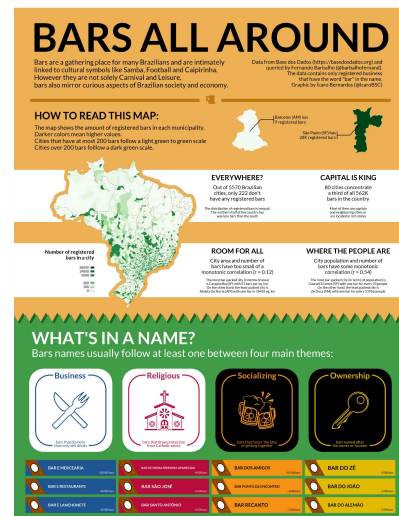
TidyTuesday



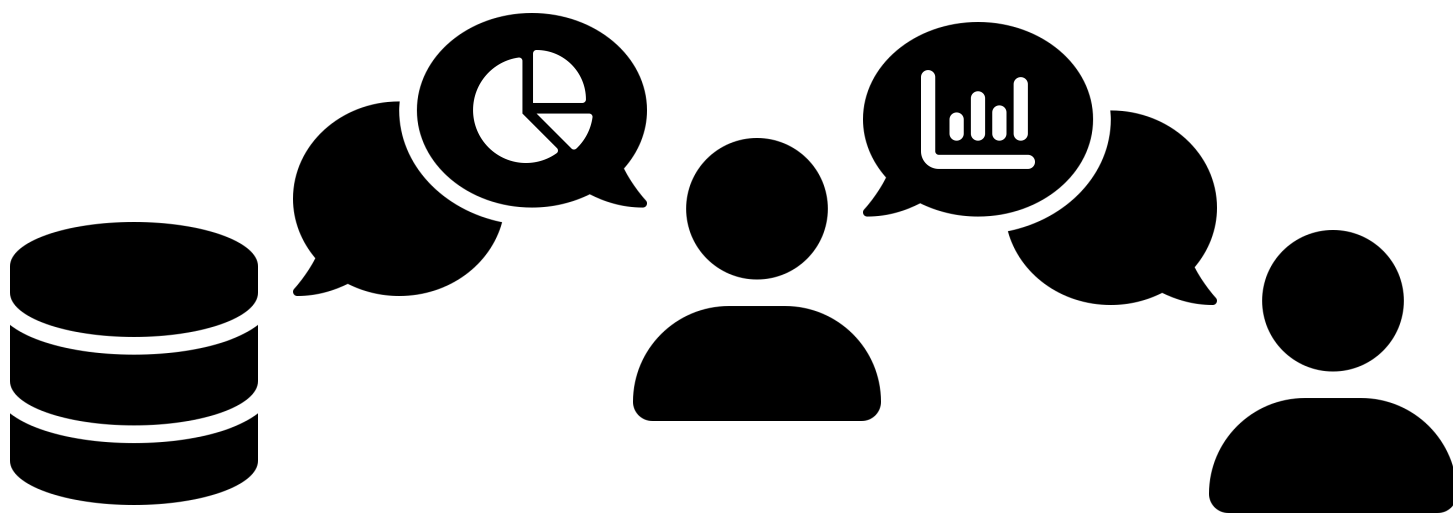
30DayChartChallenge



DuBoisChallenge



Extras

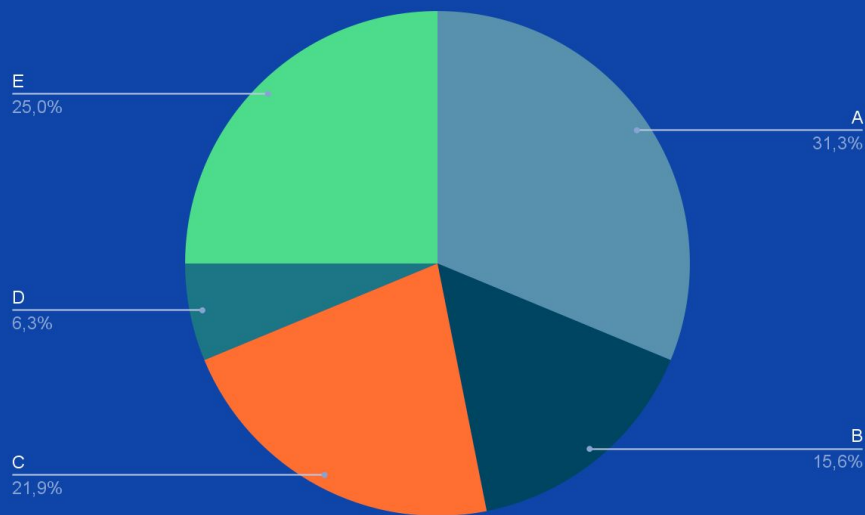


Ícones do Font Awesome

Produzir uma
visualização
para dados é

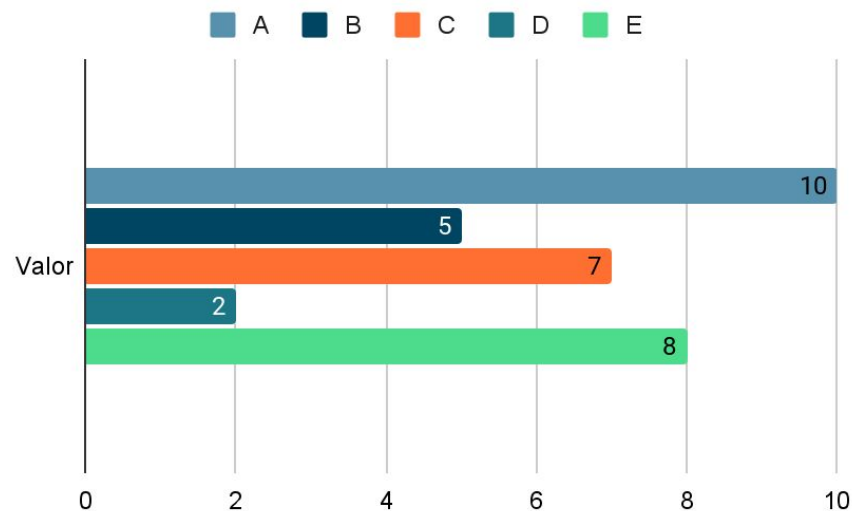
comunicar

Das escolhas feitas derivam



mal-entendidos

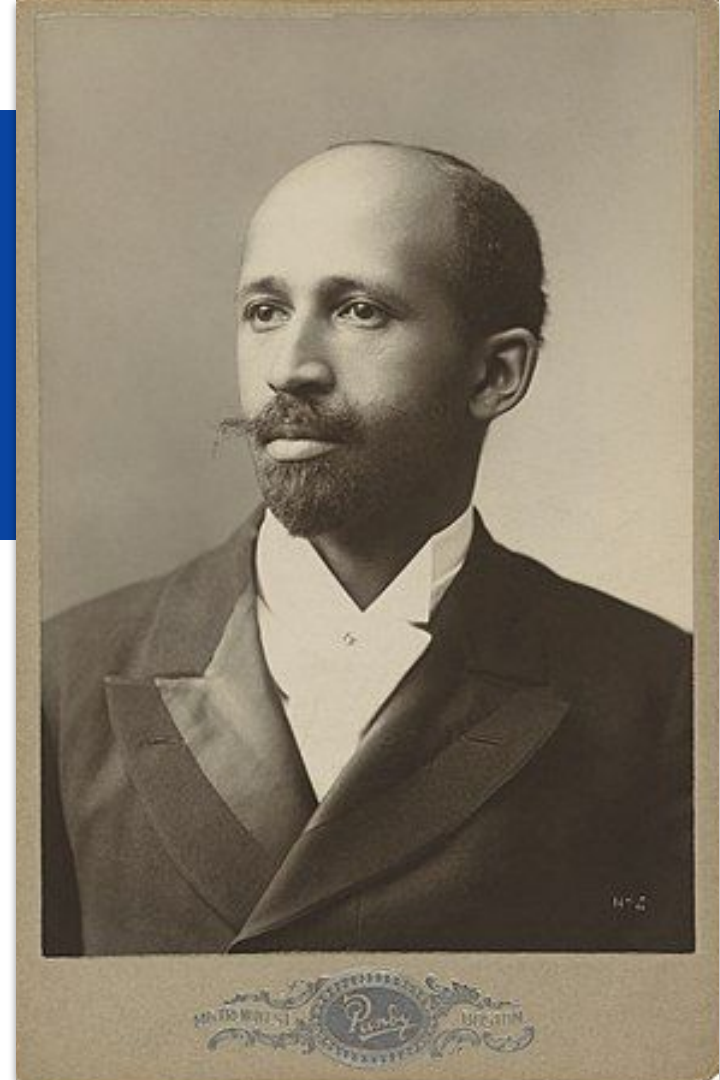
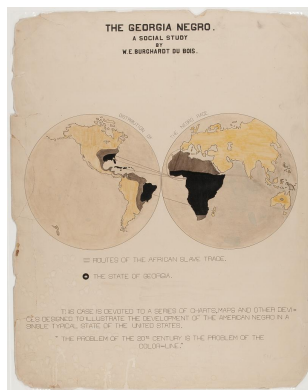
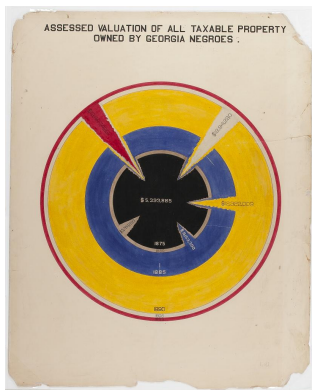
derivam



ou compreensão

Du Bois

Sociólogo americano e
gênio do *dataviz*



Entendendo a função `dubois::db_area`

0. Argumentos da função (linhas 33-154)

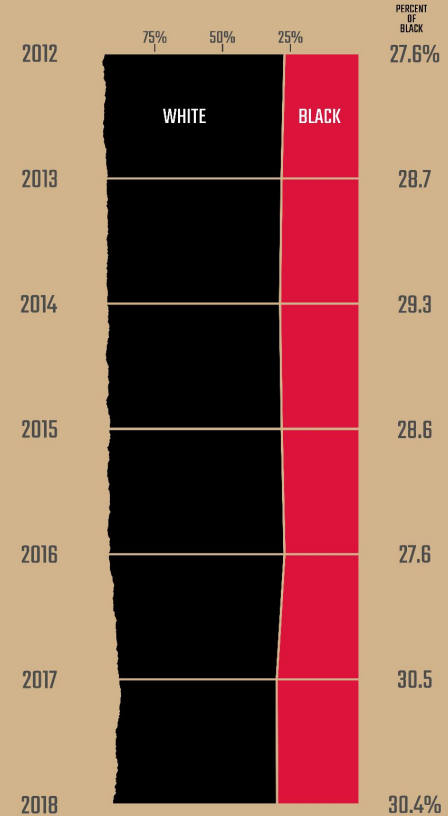
1. Uso de `{showtext}` (linhas 156-170)

2. Manejo dos dados (linhas 172-302)

3. Produção do gráfico (linhas 304-374)

PARTICIPATION IN MANAGERIAL POSITIONS BY RACE IN BRAZIL.

INSPIRED BY: W.E.B. DU BOIS | DATA FROM: IBGE | GRAPHIC BY: ICARD BERNARDES



IN THE SERIES, USUALLY WHITES OCCUPY SLIGHTLY LESS GENERAL WORK POSITIONS.
HOWEVER WHITES OCCUPY WAY MORE MANAGERIAL POSITIONS THAN BLACKS

```
# Carrega o pacote e outros auxiliares
```

```
library(dubois)
library(readr)
library(dplyr)
library(tidyr)
```

```
# Busca os dados sobre trabalhadores
# em cargos de comando contido no pacote
data <- dubois::managers
```

```
# Faz pequenas manipulações nos dados
```

```
data <- data %>%
  dplyr::select(race, year, pct_bosses_total) %>%
  tidyr::pivot_wider(names_from = "race",
                     values_from = "pct_bosses_total")
```

```
# Define título, subtítulo e mensagem
```

```
title <- "PARTICIPATION IN MANAGERIAL POSITIONS BY RACE IN BRAZIL."
subtitle <- "INSPIRED BY: W.E.B. DU BOIS | DATA FROM: IBGE | GRAPHIC
BY: ICARO BERNARDES"
message <- "IN THE SERIES, USUALLY WHITES OCCUPY SLIGHTLY LESS
GENERAL WORK POSITIONS. HOWEVER WHITES OCCUPY WAY MORE MANAGERIAL
POSITIONS THAN BLACKS"
```

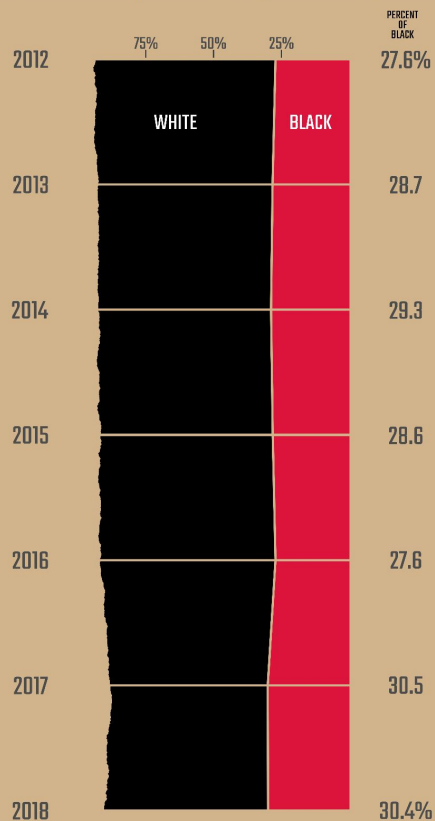
```
# Faz uso da função. A figura é
```

```
# salva no working directory
```

```
dubois::db_area(data = data, order = "year",
               cat1 = "black", cat2 = "white",
               limits = c(-3,4), filename = "managers.png",
               title = title, subtitle = subtitle,
               message = message)
```

PARTICIPATION IN MANAGERIAL POSITIONS BY RACE IN BRAZIL.

INSPIRED BY: W.E.B. DU BOIS | DATA FROM: IBGE | GRAPHIC BY: ICARO BERNARDES



IN THE SERIES, USUALLY WHITES OCCUPY SLIGHTLY LESS GENERAL WORK POSITIONS.
HOWEVER WHITES OCCUPY WAY MORE MANAGERIAL POSITIONS THAN BLACKS

Entendendo a função `dubois::db_area`

0. Argumentos da função (linhas 63-154)

1. Uso de `{showtext}` (linhas 156-170)

2. Manejo dos dados (linhas 172-302)

3. Produção do gráfico (linhas 304-374)



0. Verificação inicial dos argumentos

Converte limits a numérico (evita casos onde o número tem classe inteira)

```
limits = as.numeric(limits) %>% stats::na.exclude()
```

Confirma a classe dos argumentos

```
verify_class_fun <- function(arg, class) {  
  sym = rlang::sym(arg)  
  if (!(class %in% class(rlang::eval_tidy(sym)))) {  
    stop(glue::glue("{arg} has to be a {class} or similar"), call. = FALSE)  
  }  
}  
  
verify_class_data <- tibble::tibble(  
  arg = c("data", "order", "cat1", "cat2",  
          "dpi", "seed", "res_step", "limits",  
          "names", "title", "subtitle", "message", "path", "filename"),  
  class = c("data.frame", rep("character",3), rep("numeric",4), rep("character",6))  
)  
  
purrr::pwalk(verify_class_data, ~verify_class_fun(.x, .y))
```

Confirma que as variáveis estão realmente presentes em data

```
verify_varnames_fun <- function(name, var) {  
  status = var %in% colnames(data)  
  if (!status) {  
    stop(glue::glue("Needed variable absent: {name}"), call. = FALSE)  
  }  
}  
  
verify_varnames_data <- tibble::tibble(  
  name = c("order", "cat1", "cat2"),  
  var = c(order, cat1, cat2)  
)  
  
purrr::pwalk(verify_varnames_data, ~verify_varnames_fun(.x, .y))
```

Confirma que ambas categorias são numéricas

```
verify_varnum_fun <- function(name, var) {  
  sym = rlang::sym(var)  
  status = data %>%  
    dplyr::select("num" = !!sym) %>%  
    dplyr::pull(num) %>%  
    is.numeric()  
  if (!status) {  
    stop(glue::glue("Variable has to be numeric: {name}"), call. = FALSE)  
  }  
}  
  
verify_varnum_data <- tibble::tibble(  
  name = c("cat1", "cat2"),  
  var = c(cat1, cat2)  
)  
  
purrr::pwalk(verify_varnum_data, ~verify_varnum_fun(.x, .y))
```

Alerta o usuário em caso de dpi e seed não-inteiras

```
verify_whole_fun <- function(name, var) {  
  tol = .Machine$double.eps^0.5  
  result = abs(var - round(var)) < tol  
  if (!result) {  
    warning(glue::glue("{name} was not an integer"), call. = FALSE)  
  }  
}  
  
verify_whole_data <- tibble::tibble(  
  name = c("dpi", "seed"),  
  var = c(dpi, seed)  
)  
  
purrr::pwalk(verify_whole_data, ~verify_whole_fun(.x, .y))
```

```
## Confirma que limits e names tem o comprimento correto
```

```
verify_length_fun <- function(name, var) {  
  amount = var %>% length()  
  if (amount != 2) {  
    stop(glue::glue("{name} must be a vector with two items"), call. = FALSE)  
  }  
}  
  
verify_length_data <- tibble::tibble(  
  name = c("limits", "names"),  
  var = list(limits, names)  
)  
  
purrr::pwalk(verify_length_data, ~verify_length_fun(.x, .y))
```

```
## Confirma que os limites inferior e superior estão na ordem correta
```

```
if (limits[1] > limits[2]) {  
  stop("limits must be given in the following order: inferior, superior", call. = FALSE)  
}
```

```
## Confirma que filename tem uma extensão válida para uso por parte de ggplot2::ggsave
```

```
file_exts <- c(".eps", ".ps", ".tex", ".pdf", ".jpeg",  
  ".tiff", ".png", ".bmp", ".svg", ".wmf")  
  
if (stringr::str_detect(filename, paste0(file_exts, collapse = "|"), negate = TRUE)) {  
  stop(glue::glue("filename must contain one of these extensions: {paste0(file_exts, collapse = ', ')}"), call. =  
FALSE)  
}
```

Entendendo a função `dubois::db_area`

0. Argumentos da função (linhas 63-154)

1. Uso de `{showtext}` (linhas 156-170)

2. Manejo dos dados (linhas 172-302)

3. Produção do gráfico (linhas 304-374)



```
# 1. Cuida da impressão do texto na imagem
## Verifica se a fonte "Teko" está disponível para ser
## usada por showtext e baixa ela se estiver ausente
if (!("Teko" %in% sysfonts::font_families())) {
  sysfonts::font_add_google(name = "Teko")
}

## Define a resolução de textos impressos por showtext nos dispositivos gráficos
showtext::showtext_opts(dpi = dpi)

## Ativa o controle da impressão do texto por parte de showtext.
## ATENÇÃO! showtext tem problemas na interalção com alguns plots
## (notadamente do pacote circlize), então é melhor criar eles
## primeiro e depois usar esse pacote
showtext::showtext_auto()
```

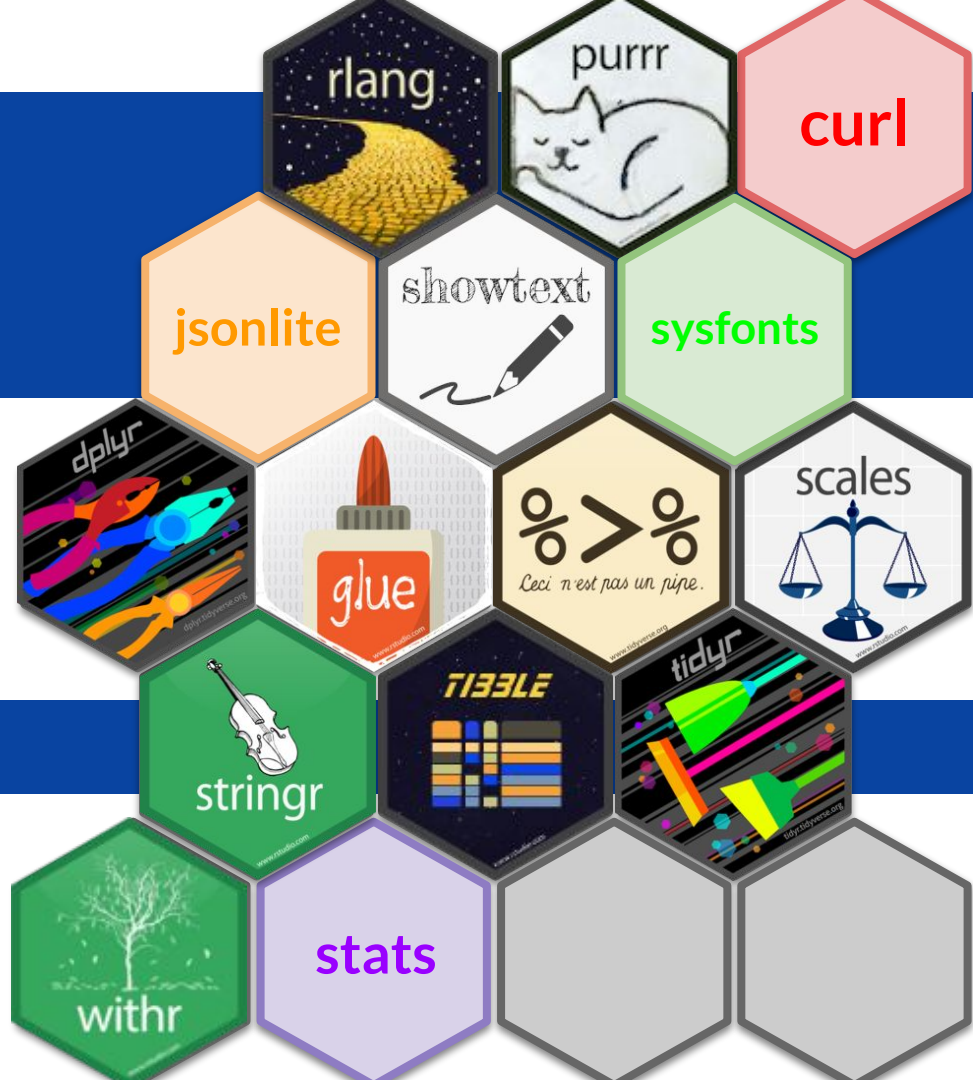
Entendendo a função `dubois::db_area`

0. Argumentos da função (linhas 63-154)

1. Uso de `{showtext}` (linhas 156-170)

2. Manejo dos dados (linhas 172-302)

3. Produção do gráfico (linhas 304-374)



2. Maneja os dados

Elimina linhas com dados ausentes

```
data <- data %>%  
  dplyr::filter(dplyr::if_all(.fns = ~ !is.na(.)))
```

Obtém o número total de observações

```
n_obs <- dim(data)[1]
```

Mantém apenas os dados da ordem das observações e os

valores das duas categorias. Também as renomeia

```
data <- data %>%  
  dplyr::select(order, cat1, cat2) %>%  
  dplyr::rename(  
    "order" = order,  
    "cat1" = cat1,  
    "cat2" = cat2  
  )
```

```
## Verifica se a soma do par de categorias é igual a 100%.
## Caso não seja, converte as categorias a porcentagens que o são.
```

```
check <- data %>%
  dplyr::mutate(
    pair = cat1 + cat2,
    pair == 100
  ) %>%
  dplyr::summarise(pair = sum(pair)) %>%
  dplyr::mutate(pair = (pair == n_obs)) %>%
  dplyr::pull(pair)
```

```
if (!check) {
  data <- data %>%
    dplyr::mutate(
      total = cat1 + cat2,
      cat1 = 100 * cat1 / total,
      cat2 = 100 * cat2 / total
    ) %>%
    dplyr::select(-total)
}
```

```
## Garante que a variável de ordem é ordenada. Se é um character,
## converte a factor e toma os níveis na ordem que eles aparecem
```

```
if (!is.numeric(data$order)) {
  if (!is.factor(data$order)) {
    data <- data %>%
      dplyr::mutate(order = factor(order, levels = unique(order)))
  }
}
data <- data %>% dplyr::arrange(order)
```



```
## Cria os rótulos para a categoria destacada
```

```
highlight <- data %>%  
  dplyr::mutate(  
    cat1 = round(cat1, digits = 1),  
    cat1 = ifelse(dplyr::row_number(order) == 1L | dplyr::row_number(order) == dplyr::n(),  
                  paste0(cat1, "%"),  
                  cat1  
  )  
  ) %>%  
  dplyr::pull(cat1)
```

```
## Obtém o primeiro e último itens entre as
```

```
## categorias da ordem e converte elas a números
```

```
ord1 <- data %>%  
  dplyr::slice(1L) %>%  
  dplyr::mutate(order = as.numeric(order)) %>%  
  dplyr::pull(order)  
ord2 <- data %>%  
  dplyr::slice(dplyr::n()) %>%  
  dplyr::mutate(order = as.numeric(order)) %>%  
  dplyr::pull(order)
```

```
## Calcula uma área à esquerda do gráfico
## usando algo como um bounded random walk
withr::local_seed(seed)
lft_area <- tibble::tibble(
  y = seq(ord1, ord2, res_step * (ord2 - ord1))
) %>%
  dplyr::rowwise() %>%
  dplyr::mutate(r = stats::rnorm(n = 1)) %>%
  dplyr::ungroup() %>%
  dplyr::mutate(
    z = cumsum(r),
    z = scales::rescale(z, to = c(limits[1], limits[2]))
  ) %>%
  dplyr::mutate(
    x = z + 2 * mean(data$cat1) / 3 + mean(data$cat2),
    x = ifelse(x > 100, 100, x)
  )
```

```
## Define os nomes das categorias
```

```
categ_names <- data %>%  
  dplyr::slice(1L:2L) %>%  
  dplyr::summarise(  
    x1 = mean(cat1) / 2,  
    x2 = mean(cat2) / 2 + mean(cat1),  
    y = mean(as.numeric(order))  
  ) %>%  
  tidyr::pivot_longer(  
    cols = c("x1", "x2"),  
    names_to = "varname",  
    values_to = "x"  
  ) %>%  
  dplyr::select(-varname) %>%  
  dplyr::mutate(label = toupper(names))
```

```
## Define o título, título do eixo secundário e mensagem do gráfico
```

```
title <- paste0(toupper(title), "<br><span style='font-size:60px;'>", toupper(subtitle), "</span>")  
sectitle <- paste0("PERCENT<br>OF<br>", toupper(names[1]))  
message <- toupper(message) %>%  
  stringr::str_wrap() %>%  
  stringr::str_replace_all(pattern = "\n", "<br>")
```

```
## Define algumas constantes de layout
```

```
lnhgt <- 0.8
```

```
bgcolor <- "#d2b48c"
```

```
l_marg <- 750 - 10 * max(stringr::str_length(data$order))
```

```
lb_sz <- 60 - 2 * max(stringr::str_length(data$order))
```

```
if (lb_sz < 30) {
```

```
  lb_sz <- 30
```

```
}
```

```
## Reorganiza os dados
```

```
data <- data %>%
```

```
  tidyr::pivot_longer(
```

```
    cols = c("cat1", "cat2"),
```

```
    names_to = "categ",
```

```
    values_to = "pct"
```

```
)
```

```
## Cria uma nova variável numérica com base na ordem
```

```
data <- data %>%
```

```
  dplyr::mutate(num_order = as.numeric(order))
```

Entendendo a função `dubois::db_area`

0. Argumentos da função (linhas 63-154)

1. Uso de `{showtext}` (linhas 156-170)

2. Manejo dos dados (linhas 172-302)

3. Produção do gráfico (linhas 304-374)



3. Produção do gráfico

Cria o gráfico

```
p <- data %>%
```

```
  ggplot2::ggplot() +
```

Insere o par de áreas

```
  ggplot2::geom_area(ggplot2::aes(x = pct, y = num_order, fill = categ),  
    orientation = "y", size = 4, color = bgcolor,  
    position = ggplot2::position_stack(reverse = TRUE)
```

```
) +
```

Insere o efeito de "rasgo" à esquerda

```
  ggplot2::geom_ribbon(ggplot2::aes(xmin = x, xmax = 100, y = y),  
    fill = bgcolor, data = lft_area
```

```
) +
```

Insere o nome das categorias

```
  ggplot2::geom_text(ggplot2::aes(x = x, y = y, label = label),  
    color = "white",  
    size = 15, family = "Teko", data = categ_names
```

```
) +
```

Insere títulos e mensagem

```
  ggplot2::labs(title = title, subtitle = sectitle, caption = message, x = NULL, y = NULL) +
```

Reverte a escalas dos eixos e controla elementos delas

```
ggplot2::scale_x_reverse(  
  expand = ggplot2::expansion(0, 10), breaks = seq(25, 75, 25),  
  label = scales::label_percent(scale = 1), position = "top"  
) +  
ggplot2::scale_y_reverse(  
  expand = ggplot2::expansion(0, 0.01),  
  breaks = unique(data$num_order),  
  labels = unique(data$order),  
  sec.axis = ggplot2::dup_axis(  
    name = NULL,  
    breaks = unique(data$num_order),  
    labels = highlight  
  )  
) +
```

Aplica cores definidas para as áreas

```
ggplot2::scale_fill_manual(  
  values = c("#dc143c", "black"),  
  guide = "none"  
) +
```

Customiza elementos do gráfico

```
ggplot2::theme(  
  text = ggplot2::element_text(family = "Teko"),  
  plot.margin = ggplot2::margin(t = 60, r = 400, b = 50, l = 400, unit = "pt"),  
  plot.background = ggplot2::element_rect(fill = bgcolor, color = NA),  
  plot.title = ggtext::element_textbox_simple(  
    size = 80, halign = 0.5, valign = 0.5, width = 3, lineheight = lnhgt,  
    margin = ggplot2::margin(t = 0, r = 0, b = 20, l = 0, unit = "pt")  
  ),  
  plot.subtitle = ggtext::element_textbox_simple(  
    size = 23, halign = 0.5, valign = 0.5, width = 1.5, lineheight = lnhgt,  
    padding = ggplot2::margin(t = 0, r = 0, b = -10, l = l_marg, unit = "pt")  
  ),  
  plot.caption = ggtext::element_textbox_simple(  
    size = 45, fill = "#654321", color = bgcolor,  
    halign = 0.5, valign = 0.5, width = 3,  
    padding = ggplot2::margin(t = 40, r = 0, b = 40, l = 0, unit = "pt"),  
    margin = ggplot2::margin(t = 80, r = 0, b = 20, l = 0, unit = "pt")  
  ),  
  panel.background = ggplot2::element_blank(),  
  panel.grid.minor = ggplot2::element_blank(),  
  panel.grid.major.x = ggplot2::element_blank(),  
  panel.grid.major.y = ggplot2::element_line(color = bgcolor, size = 2),  
  panel.ontop = TRUE,  
  axis.text.x = ggplot2::element_text(size = 35),  
  axis.text.y = ggplot2::element_text(size = lb_sz),  
  axis.text.y.right = ggplot2::element_text(hjust = 0.5),  
  axis.ticks.length.x = ggplot2::unit(15, "pt"),  
  axis.ticks.x = ggplot2::element_line(size = 1),  
  axis.ticks.y = ggplot2::element_blank()  
)
```



```
## Salva o gráfico
file <- paste0(path, "/", filename)
ggplot2::ggsave(file,
  plot = p, dpi = dpi,
  width = 22, height = 28
)
```

Muito obrigado!



@IcaroBSC



<https://github.com/IcaroBernardes>



icaro@bitanalytics.dev.br



<https://www.linkedin.com/in/icarobsc>

