Dataset de Imagens

Daniel Amaral

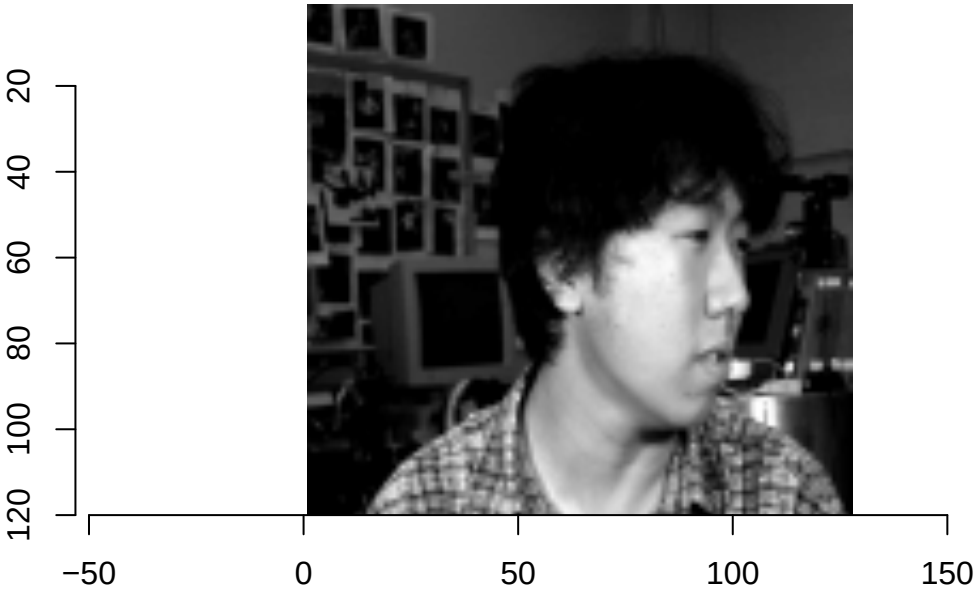
08/09/2021

Ler Imagem

Lendo com `magick`, por causa da compatibilidade com a extensão `.pgm`, e depois convertendo para o formato CImg, que é mais fácil de manipular, com o pacote `imager`.

```
read_image <- function(path) {
  img <- magick::image_read(path)
  img <- imager::magick2cimg(img)
  return(img)
}

img <- read_image('../an2i/an2i_left_angry_open.pgm')
plot(img)
```



Matriz para Vetor

A ideia aqui é pegar a matriz de intensidades da imagem e converter em um vetor unidimensional. Como o `as.vector` transforma a imagem para vetor coluna por coluna, coloquei como input a matriz transposta, uma gambiarra pra ele concatenar linha por linha.

```r
flatten <- function(img) {
  return(matrix(t(as.matrix(img)), nrow = 1))
}
```

Exemplo básico:

```r
mat_example <- matrix(1:9, 3, 3, byrow = T)
mat_example
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

convertendo para vetor

```r
mat_example_fl <- flatten(mat_example)
mat_example_fl
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
```

O mesmo processo é possivel com a imagem:

```r
img_matrix <- as.matrix(img) # Matriz de intensidades
img_flatten <- flatten(img_matrix)
dim(img_flatten) # Vetor 1D
```

```
## [1]     1 15360
```

Leitura de todas as imagens

extrair a pasta `faces` do arquivo e acessar pelo R:

```r
paths <- fs::dir_ls(path = '../../faces',
                    glob = '*.pgm', recurse = TRUE)
head(paths)
```

```
## ../../faces/an2i/an2i_left_angry_open.pgm
## ../../faces/an2i/an2i_left_angry_open_2.pgm
## ../../faces/an2i/an2i_left_angry_open_4.pgm
## ../../faces/an2i/an2i_left_angry_sunglasses.pgm
## ../../faces/an2i/an2i_left_angry_sunglasses_2.pgm
## ../../faces/an2i/an2i_left_angry_sunglasses_4.pgm
```

Criando o dataset (sem nada de imagens ainda...)

O dataset abaixo tem todas as informações da imagem, no entanto, não dá pra colocar os vetores das imagens ainda pq tem diferentes tipos de escala.

```
da_faces <- paths |>
  # cria o dataframe com os paths
  tibble::tibble() |>
  # só o nome do arquivo
  dplyr::mutate(paths_aux = stringr::str_remove_all(paths, '.*/')) |>
  # o separate aqui separa a string em cada "_"
  tidyr::separate(col = paths_aux,
                  into = c('user_id', 'head_pos',
                           'facial_exp', 'eye_state', 'scale'),
                  sep = "_") |>
  # corrigindo a variável eye_state e scale
  dplyr::mutate(eye_state = stringr::str_remove(eye_state, '.pgm'),
                scale = stringr::str_remove(scale, '.pgm'),
                scale = as.integer(scale),
                scale = ifelse(is.na(scale), 1, scale))
```

Aqui é mais complicadinho... Basicamente, **filtrando por um certo tipo de escala**, vou criar uma pipeline para percorrer o dataset acima, ler cada imagem (função que criei `read_image`) e concatenar o resultado (imagem como vetor).

obs: demora um pouquinho... 30 sec - 1 min...

```
da_faces_flatten <- da_faces |>
  dplyr::filter(scale == 1) |> # filtrando pela resolução full
  # extração da variável de paths
  dplyr::pull(paths) |>
  # percorrendo cada img, lendo, vetorizando e rbind
  purrr::map_dfr(function(path) {
    img <- read_image(path)
    img_vec <- flatten(img)
    return(as.data.frame(img_vec))
  }, .id = 'paths') |>
  # concatenando com as informações do da_faces
  dplyr::inner_join(da_faces, by = c("paths" = "paths")) |>
  # convertendo pra tibble, facilitar a manipulação
  tibble::as_tibble() |>
  # deixando bonitinho a organização das variáveis
  dplyr::relocate(c('user_id', 'head_pos', 'facial_exp',
                    'eye_state', 'scale'), .before = V1)
```

Bom, agora o dataset ta prontinho pra trabalhar...

```
head(da_faces_flatten)
```

```
## # A tibble: 6 x 15,366
##   paths      user_id head_pos facial_exp eye_state scale    V1    V2    V3    V4
##   <fs::path> <chr>   <chr>    <chr>      <chr>     <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ../../fac~ an2i    left     angry      open          1 0.133 0.184 0.184 0.102
## 2 ../../fac~ an2i    left     angry      sunglass~     1 0     0     0     0.102
## 3 ../../fac~ an2i    left     happy      open          1 0.133 0.192 0.184 0.110
## 4 ../../fac~ an2i    left     happy      sunglass~     1 0.141 0.204 0.196 0.110
## 5 ../../fac~ an2i    left     neutral    open          1 0.192 0.153 0.133 0.122
## 6 ../../fac~ an2i    left     neutral    sunglass~     1 0.145 0.212 0.204 0.114
```

```
## # ... with 15,356 more variables: V5 <dbl>, V6 <dbl>, V7 <dbl>, V8 <dbl>,
## #   V9 <dbl>, V10 <dbl>, V11 <dbl>, V12 <dbl>, V13 <dbl>, V14 <dbl>, V15 <dbl>,
## #   V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>,
## #   V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>,
## #   V28 <dbl>, V29 <dbl>, V30 <dbl>, V31 <dbl>, V32 <dbl>, V33 <dbl>,
## #   V34 <dbl>, V35 <dbl>, V36 <dbl>, V37 <dbl>, V38 <dbl>, V39 <dbl>,
## #   V40 <dbl>, V41 <dbl>, V42 <dbl>, V43 <dbl>, V44 <dbl>, V45 <dbl>, ...
```

Com as seguintes dimensões:

```
dim(da_faces_flatten)
```

```
## [1]   624 15366
```

5 variáveis (removendo o paths, senão é 6) de informações gerais da imagem + as variáveis de intensidades dos pixels (imagem vetorizada).