Importação e Tratamento

Ricardo Theodoro

OBSCOOP/USP

Base de dados

- Questionário respondido por vocês e disponível no E-disciplinas com o nome:

`SurvAlun - EstProb I.xlsx`

- Recomendo que abram o R e rodem os códigos enquanto acompanham estes slides
- Ficará mais fácil para verem o que está acontecendo

Pacotes utilizados

A primeira vez é preciso instalar no computador:

```
install.packages(c("dplyr", "janitor", "lubridate", "readxl", "stringr"))
```

Depois, é preciso carregar sempre que for usar:

```
library(dplyr)
library(janitor)
library(lubridate)
library(readxl)
library(stringr)
```

Importando a base

Lembrando do formato `nome_pacote::funcao()`

```
survAluno <- readxl::read_xlsx("Aula-RCC0219/data_input/SurvAlun - EstProb
```

Resumo das informações:

```
dplyr::glimpse(survAluno)
```

Limpando nomes

- Podemos observar que os nomes estão muito compridos e não seguem um padrão
- Teremos que consertar na mão
- Se os nomes fossem curtos, poderiamos utilizar a função `janitor::clean_names()`

```
names(survAluno) <- c(
  "data_hora", "altura", "n_calcado", "peso", "cirurgia", "estetica",
  "estetica_outros", "cor", "oculos", "sexo", "estado_civil",
  "n_pessoas_moradia", "entidade_estudantil", "tipo_entidade_estudantil"
) # Colocar o nome de todas as variáveis aqui
```

Verificando novamente:

```
dplyr::glimpse(survAluno)
```

Selecionando colunas

Existem duas formas de selecionar variáveis de um `data.frame`

Primeira, com pacote base do R

```
survAluno[, 1:25]
survAluno[, c("bandejao","ano_ingresso_usp")]
```

Segunda, com o pacote `dplyr`

```
survAluno |>
   dplyr::select(1:25)

servAluno |>
   dplyr::select(bandejao:ano_ingresso_usp)
```

Filtrando valores

- Verificando informações de trabalho apenas daqueles que trabalham com contabilidade

```
survAluno |>
   dplyr::filter(trabalho_contabil == "Sim") |>
   dplyr::group_by(horas_trabalho_semana) |>
   dplyr::count() |>
   janitor::adorn_totals()
```

Agrupando valores de colunas

- Comparando informações sobre aqueles que trabalham com contabilidade ou não

```
survAluno |>
   dplyr::group_by(trabalho_contabil, horas_trabalho_semana) |>
   dplyr::count()
```

Separando colunas de data

```r
# Separando colunas
survAluno <- survAluno |>
  dplyr::mutate(
    data = lubridate::ymd(as.Date(data_hora)),
    hora = stringr::str_sub(data_hora, start = 12),
    hora = lubridate::hms(hora)
  ) |>
  dplyr::select(data_hora, data, hora, everything())

max(survAluno$data_hora) - min(survAluno$data_hora)
max(survAluno$data) - min(survAluno$data)
max(lubridate::hour(survAluno$hora)) - min(lubridate::hour(survAluno$hora))
```

Resumo de respostas

Estatística descritiva das variáveis numéricas

```
library(tidyselect)
survAluno |>
  dplyr::select(where(is.double)) |>
  summary()
```

Resumo das respostas categóricas

```
survAluno |>
  dplyr::group_by(animais_domesticos) |>
  dplyr::count() |>
  dplyr::arrange(n) |>
  janitor::adorn_totals()
```

Criando função

Criando função para verificar resumo das respostas categóricas de uma vez

```
resumo_categorica <- function(base, variavel) {
  base |>
    dplyr::group_by({{ variavel }}) |>
    dplyr::count() |>
    dplyr::arrange(n) |>
    janitor::adorn_totals()
}
```

Criando uma iteração para todas as colunas de uma vez

```
for (i in 1:ncol(survAluno)) {
  resumo_categorica(survAluno, survAluno[, i]) |>
    print()
}
```

Padronizado as respostas

```
survAluno <- survAluno |>
  dplyr::mutate(
    estetica_outros = ifelse(is.na(estetica_outros) |
      estetica_outros == "nenhum" |
      estetica_outros == "nada",
    "Nenhum",
    estetica_outros
    ),
    cidade_empresa = tolower(cidade_empresa),
    cidade_empresa = abjutils::rm_accent(cidade_empresa),
    cidade_empresa = stringr::str_replace_all(cidade_empresa, "[[:punct:]]
    cidade_empresa = dplyr::case_when(
      stringr::str_detect(cidade_empresa, "nao") |
        cidade_empresa == "" ~ "nenhuma",
      cidade_empresa == "rp" ~ "ribeirao preto",
      TRUE ~ as.character(cidade_empresa)
```