Coleta e Análise de Dados Secundários

Apresentação de slides com todas as aulas

Frederico Bertholini

Introdução

O que são dados secundários?

- "No sentido mais amplo, análise de dados coletados por outra pessoa" (Boslaugh, 2007)
- Uso de dados para responder a uma pergunta diversa da qual originou sua coleta (Vartanian, 2010)
- Em contraste com a análise de dados primários em que o mesmo indivíduo/equipe de pesquisadores desenha, coleta e analisa os dados

O que são dados secundários?

Muitas fontes

- Grandes conjuntos de dados financiados pelo governo
- Registros administrativo
- Suplementos de periódicos
- websites dos autores
- Etc.

O que são dados secundários?

- Disponível para um número aparentemente ilimitado de temas
- Quantitativo ou qualitativo
- Uso restrito ou público
- Direto ou observação indireta

Fontes essenciais de dados secundários

- Portal de dados do Governo Federal http://dados.gov.br/dataset?groups=governo-politica
- IBGE https://www.ibge.gov.br/estatisticas-novoportal/downloads-estatisticas.html
- IPEADATA http://www.ipeadata.gov.br/Default.aspx
- DATASUS http://www2.datasus.gov.br/DATASUS/index.php?area=0205&id=6936
- INEP http://portal.inep.gov.br/web/guest/dados

Internacionais (US)

- Inter-University Consortium for Political and Social Research http://www.icpsr.umich.edu/icpsrweb/ICPSR/access/index.jsp
- Data.gov http://www.data.gov
- National Center for Education Statistics http://nces.ed.gov
- U.S. Census Bureau http://www.census.gov
- Simple Online Data Archive for Population Studies (SodaPop) http://sodapop.pop.psu.edu/data-collections

Vantagens de dados secundários

- Desenho do estudo e coleta de dados já concluídos
- Economiza tempo e dinheiro
- Acesso a dados internacionais e históricos que caso contrário, levariam vários anos e milhões de reais para coletar

Vantagens de dados secundários

- Ideal para uso em exemplos de sala de aula, projetos semestrais, mestrados teses, dissertações
- Normalmente os dados têm qualidade superior
- Estudos financiados pelo governo geralmente envolvem amostras maiores que são mais representativos da população-alvo (maior validade externa)

Vantagens de dados secundários

- A sobreamostra de grupos/comportamentos de baixa prevalência permite maior precisão estatística
- Os conjuntos de dados geralmente têm amplitude considerável (milhares de variáveis)

Desvantagens de dados secundários

Desenho do estudo e coleta de dados já concluídos

- Os dados podem não facilitar uma questão de pesquisa específica
- Informações sobre desenho do estudo e procedimentos de coleta de dados pode ser escassas

Desvantagens de dados secundários

- Os dados podem ter falta de profundidade (quanto maior a largura, mais difícil para medir qualquer construção em profundidade)
- Certos campos ou departamentos (por exemplo, programas experimentais) podem ver menor valor na análise de dados secundários
- Pode exigir conhecimento de estatística/métodos de pesquisa que não são geralmente fornecidos por cursos de graduação ou pós-graduação

Entenda seus dados

Familiarize-se com o estudo e os dados originais!

- Leia todos os manuais
- Para quem os resultados são generalizáveis?

Entenda seus dados

- Como os dados faltantes (perdidos) são tratados?
- Quais são os pesos de análise apropriados?
- Quais variáveis compostas estão disponíveis e como elas são construídas?

Entenda seus dados

- Protocolos de coleta
- Questionários
- Atualizações

Preparo de dados

- Documente TUDO!

1. Transfira ou leia diretamente
2. Lide com missing data
3. Recodifique variáveis
4. Crie novas variáveis

Análise de dados

- Com base na sua questão de pesquisa, identificar análise estatística apropriada
- Selecione o pacote de software que implementará a análise e viabilizará a amostragem complexa
- Examine estatísticas descritivas não ponderadas para identificar erros de codificação e determinar a adequação do tamanho da amostra

Análise de dados

- Identifique pesos
- Identifique método de estimação de variância (e variáveis correspondentes)
- Realize análises de diagnóstico (identificar outliers, não normalidade, etc.)
- Realize análises preliminares e interprete os resultados!

O Universo tidyverse

Manifesto tidyverse

O tidyverse, também chamado por muitos de hadleyverse, é um conjunto de pacotes que, por compartilharem esses princípios do manifesto tidy, podem ser utilizados naturalmente em conjunto. Pode-se dizer que existe o R antes do tidyverse e o R depois do tidyverse.

Os princípios fundamentais do tidyverse são:

- Reutilizar estruturas de dados existentes.
- Organizar funções simples usando o pipe.
- Aderir à programação funcional.
- Projetado para ser usado por seres humanos.

Manifesto tidy

- Tidy Tools Manifesto https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html
- Tidy data vignette https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html
- Tidy Data paper http://vita.had.co.nz/papers/tidy-data.pdf
- Conjunto de pacotes https://www.tidyverse.org/packages/

Usando o pipe - O operador %>%

O operador %>% (pipe) foi uma das grandes revoluções recentes do R, tornando a leitura de códigos mais lógica, fácil e compreensível.

```
library(tidyverse)
library(magrittr)
```

Ideia

A ideia do operador %>% (pipe) é bem simples: usar o valor resultante da expressão do lado esquerdo como primeiro argumento da função do lado direito.

- As duas linhas abaixo são equivalentes.

```
f(x, y)
```

```
x %>% f(y)
```

E se aumentarmos o código?

Vamos calcular a raiz quadrada da soma dos valores de 1 a 4.

Primeiro, sem o pipe.

```
sqrt(sum(x))
```

```
## [1] 3.162278
```

Agora com o pipe.

```
x %>%
  sum %>%
  sqrt
```

```
## [1] 3.162278
```

E se realmente tivermos muitas funções aninhadas?

A utilização do pipe transforma um código confuso e difícil de ser lido em algo *simples e intuitivo*.

Receita de bolo - sem pipe

Tente entender o que é preciso fazer.

```
esfrie(
  asse(
    coloque(
      bata(
        acrescente(
          recipiente(rep("farinha", 2), "água",
                     "fermento", "leite", "óleo"),
          "farinha", até = "macio"),
        duração = "3min"),
      lugar = "forma", tipo = "grande",
      untada = TRUE), duração = "50min"),
  "geladeira", "20min")
```

Receita de bolo - com pipe

Desistiu? Agora veja como fica escrevendo com o %>%:

```
recipiente(rep("farinha", 2), "água", "fermento", "leite",
  acrescente("farinha", até = "macio") %>%
  bata(duração = "3min") %>%
  coloque(lugar = "forma", tipo = "grande", untada = TRUE)
  asse(duração = "50min") %>%
  esfrie("geladeira", "20min")
```

Exercícios pipe

Exercício

1. Reescreva a expressão abaixo utilizando o %>%.

```
round(mean(divide_by(sum(1:10),3)),digits = 1)
```

Resolução

Exercício

```
2 %>%
  add(2) %>%
  c(6, NA) %>%
  mean(na.rm = T) %>%
  equals(5)
```

Resolução

Importação no tidyverse

Importação com `readr`, `readxl`, `haven` e `DBI`

No tidyverse, geralmente

- Funções `read_<formato>` servem para ler um arquivo no formato `<formato>`
- Funções `write_<formato>` servem para escrever num arquivo com o formato `<formato>`

Arquivos de texto

- `csv`, `tsv`, `txt`, ...
- Para esses aqui, usar o pacote `readr`
- Você também pode experimentar o `data.table::fread`

‘readr’ para textos

Exemplo:

```
read_csv("data/import/mtcars.csv")

data.table::fread("data/import/mtcars.csv")
```

Arquivos binários

- `.RData`, `.rds`, `.feather`, `.fst`
- `.dta` (Stata), `.sas7bdat` (SAS), `.sav` (SPSS)
- Ler com `readr`, `haven`, `feather`, `fst`.

Exemplo:

```
read_rds("data/import/mtcars.rds")
```

Bancos de dados

- `MySQL`, `SQL Server`, `PostgreSQL`, `SQLite`, ...
- `Spark`, `MongoDB`, `Hive`, ...
- Utilizar pacotes `DBI` e `odbc`

Pacotes `dplyr` e `tidyr`

Conjunto de dados

Vamos trabalhar com a base decisoes, que contém decisões do Tribunal de Justiça de São Paulo

```
decisoes <- read_rds("CADS2018/Exercícios/dados/decisoes.rd
glimpse(decisoes)
```

```
## Observations: 11,731
## Variables: 9
## $ id_decisao     <chr> "11094999", "11093733", "1109367
## $ n_processo     <chr> "0057003-20.2017.8.26.0000", "005
## $ classe_assunto <chr> "Habeas Corpus / Homicídio Simple
## $ municipio      <chr> "Cosmópolis", "São Paulo", "Ribei
## $ camara         <chr> "3ª Câmara de Direito Criminal",
## $ data_decisao   <chr> "19/12/2017", "19/12/2017", "19/1
## $ data_registro  <chr> "19/12/2017", "19/12/2017", "19/1
## $ juiz           <chr> "Luiz Antonio Cardoso", "Luiz Ant
## $ txt_decisao    <chr> NA, NA, NA, "Execução Penal -  Co
```

Características do `dplyr`

- A utilização é facilitada com o emprego do operador %>%
- No primeiro argumento colocamos o `data.frame` ou o `tibble`, e nos outros argumentos colocamos o que queremos fazer.

As cinco funções principais do `dplyr`

- `select`: selecionar colunas
- `filter`: filtrar linhas
- `mutate`: criar colunas
- `summarise`: sumarizar colunas
- `arrange`: ordenar linhas

select

`select`

- Utilizar `starts_with(x)`, `contains(x)`, `matches(x)`, `one_of(x)`, etc.
- Possível colocar nomes, índices, e intervalos de variáveis com `:`.

Em ação

```
decisoes %>%
  select(id_decisao, n_processo, municipio, juiz)
```

```
## # A tibble: 11,731 x 4
##    id_decisao n_processo                    municipio
##    <chr>      <chr>                         <chr>
##  1 11094999   0057003-20.2017.8.26.0000 Cosmópolis
##  2 11093733   0052762-03.2017.8.26.0000 São Paulo
##  3 11093677   0055169-79.2017.8.26.0000 Ribeirão Preto
##  4 11093270   9000580-82.2017.8.26.0032 Araçatuba
##  5 11093374   0052938-79.2017.8.26.0000 São Paulo
##  6 11093320   9000723-79.2017.8.26.0482 Presidente Prude
##  7 11091506   0003276-86.2015.8.26.0075 Bertioga
##  8 11093326   9000298-11.2017.8.26.0625 Taubaté
##  9 11092475   0004653-39.2015.8.26.0028 Aparecida
## 10 11093773   2221930-66.2017.8.26.0000 Jandira
## # ... with 11,721 more rows
```

Em ação

```
decisoes %>%
  select(classe_assunto:id_decisao, juiz)
```

```
## # A tibble: 11,731 x 4
##    classe_assunto                          n_processo          id_
##    <chr>                                   <chr>               <ch
##  1 Habeas Corpus / Homicídio Simpl~ 0057003-20.2017~ 110
##  2 Habeas Corpus / Roubo            0052762-03.2017~ 110
##  3 Habeas Corpus / DIREITO PENAL    0055169-79.2017~ 110
##  4 Agravo de Execução Penal / Pena~ 9000580-82.2017~ 110
##  5 Mandado de Segurança / Crimes d~ 0052938-79.2017~ 110
##  6 Agravo de Execução Penal / Pena~ 9000723-79.2017~ 110
##  7 Apelação / Tráfico de Drogas e ~ 0003276-86.2015~ 110
##  8 Agravo de Execução Penal / Livr~ 9000298-11.2017~ 110
##  9 Apelação / Tráfico de Drogas e ~ 0004653-39.2015~ 110
## 10 Habeas Corpus / Furto Qualifica~ 2221930-66.2017~ 110
## # ... with 11,721 more rows
```

Em ação

```
decisoes %>%
  select(id_decisao, starts_with('data_'))
```

```
## # A tibble: 11,731 x 3
##    id_decisao data_decisao data_registro
##    <chr>      <chr>        <chr>
##  1 11094999   19/12/2017   19/12/2017
##  2 11093733   19/12/2017   19/12/2017
##  3 11093677   19/12/2017   19/12/2017
##  4 11093270   14/12/2017   19/12/2017
##  5 11093374   14/12/2017   19/12/2017
##  6 11093320   14/12/2017   19/12/2017
##  7 11091506   14/12/2017   19/12/2017
##  8 11093326   14/12/2017   19/12/2017
##  9 11092475   14/12/2017   19/12/2017
## 10 11093773   19/12/2017   19/12/2017
## # ... with 11,721 more rows
```

Exercício

- selecione as colunas que acabam com “cisao”.

Resolução

```
decisoes %>%
  select(ends_with("cisao"))
```

```
## # A tibble: 11,731 x 3
##    id_decisao data_decisao txt_decisao
##    <chr>      <chr>        <chr>
##  1 11094999   19/12/2017   <NA>
##  2 11093733   19/12/2017   <NA>
##  3 11093677   19/12/2017   <NA>
##  4 11093270   14/12/2017   "Execução Penal -  Comutação
##  5 11093374   14/12/2017   "Mandado de segurança - Impet
##  6 11093320   14/12/2017   "Execução Penal - Apuração de
##  7 11091506   14/12/2017   "Tráfico de entorpecentes - A
##  8 11093326   14/12/2017   "Execução Penal - Pedido de I
##  9 11092475   14/12/2017   "Tráfico de entorpecentes -
## 10 11093773   19/12/2017   <NA>
## # ... with 11,721 more rows
```

Exercício

- tire as colunas de texto = ‘txt_decisao’ e classe/assunto = ‘classe_assunto’.
  - Dica: veja os exemplos de `?select` em `Drop variables ...`

Resolução

```
decisoes %>%
  select(-classe_assunto, -txt_decisao)
```

```
## # A tibble: 11,731 x 7
##    id_decisao n_processo municipio camara data_decisao c
##    <chr>      <chr>      <chr>     <chr>  <chr>         <
##  1 11094999   0057003-2~ Cosmópol~ 3ª Câ~ 19/12/2017   1
##  2 11093733   0052762-0~ São Paulo 3ª Câ~ 19/12/2017   1
##  3 11093677   0055169-7~ Ribeirão~ 3ª Câ~ 19/12/2017   1
##  4 11093270   9000580-8~ Araçatuba 8ª Câ~ 14/12/2017   1
##  5 11093374   0052938-7~ São Paulo 8ª Câ~ 14/12/2017   1
##  6 11093320   9000723-7~ Presiden~ 8ª Câ~ 14/12/2017   1
##  7 11091506   0003276-8~ Bertioga  8ª Câ~ 14/12/2017   1
##  8 11093326   9000298-1~ Taubaté   8ª Câ~ 14/12/2017   1
##  9 11092475   0004653-3~ Aparecida 8ª Câ~ 14/12/2017   1
## 10 11093773   2221930-6~ Jandira   3ª Câ~ 19/12/2017   1
## # ... with 11,721 more rows
```

filter

`filter`

- Use , ou & para “e” e | para “ou”.
- Condições separadas por vírgulas é o mesmo que separar por &.

filter em ação

```
decisoes %>%
  select(n_processo, id_decisao, municipio, juiz) %>%
  filter(municipio == 'São Paulo')
```

```
## # A tibble: 2,446 x 4
##    n_processo                id_decisao municipio juiz
##    <chr>                     <chr>      <chr>     <chr>
##  1 0052762-03.2017.8.26.0000 11093733   São Paulo Luiz A
##  2 0052938-79.2017.8.26.0000 11093374   São Paulo Grassi
##  3 2214049-38.2017.8.26.0000 11093604   São Paulo Luiz A
##  4 2227499-48.2017.8.26.0000 11093642   São Paulo Luiz A
##  5 9002384-31.2017.8.26.0050 11093376   São Paulo Grassi
##  6 0021158-39.2015.8.26.0050 11091508   São Paulo Grassi
##  7 7005375-26.2015.8.26.0198 11091668   São Paulo Grassi
##  8 9002039-65.2017.8.26.0050 11094451   São Paulo Grassi
##  9 2203993-43.2017.8.26.0000 11094449   São Paulo Grassi
## 10 0099423-21.2016.8.26.0050 11091474   São Paulo Grassi
## # ... with 2,436 more rows
```

Dica: usar %in%

```
library(lubridate) # para trabalhar com as datas
#`day(dmy(data_decisao))` pega o dia da decisão.
```

```
decisoes %>%
  select(id_decisao, municipio, data_decisao, juiz) %>%
  # municipio igual a campinas ou jaú, OU dia da decisão m
  filter(municipio %in% c('Campinas', 'Jaú') | day(dmy(data
```

```
## # A tibble: 3,352 x 4
##    id_decisao municipio data_decisao juiz
##    <chr>      <chr>     <chr>        <chr>
##  1 11093272   Campinas  14/12/2017   Grassi Neto
##  2 11093359   Campinas  07/12/2017   Grassi Neto
##  3 11088333   Campinas  14/12/2017   Grassi Neto
##  4 11093018   Jaú       28/11/2017   Ivan Sartori
##  5 11089105   Jaú       14/12/2017   Ricardo Tucunduva
##  6 11089111   Campinas  14/12/2017   Ricardo Tucunduva
##  7 11091386   Santos    27/11/2017   Ivo de Almeida
```

Mais ação

```
decisoes %>%
  select(juiz) %>%
  # filtra juízes que têm `Z` ou `z` no nome
  filter(str_detect(juiz, regex("z", ignore_case = TRUE)))
  # conta e ordena os juizes em ordem decrescente
  count(juiz, sort = TRUE) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   juiz                          n
##   <chr>                     <int>
## 1 Gilberto Ferreira da Cruz   237
## 2 Diniz Fernando              198
## 3 Sérgio Mazina Martins       173
## 4 Luiz Antonio Cardoso        163
## 5 Rachid Vaz de Almeida       150
```

Obs

A função `str_detect()` retorna `TRUE` se um elemento do vetor de textos é compatível com uma *expressão regular*. Estudaremos o pacote `stringr` e as funções `str_*` em outra aula.

Exercício

- filtre apenas casos em que `id_decisao` não é `NA`

Resolução

```
decisoes %>%
  filter(is.na(id_decisao))
```

```
## # A tibble: 65 x 9
##    id_decisao n_processo classe_assunto municipio camara
##    <chr>      <chr>      <chr>           <chr>     <chr>
##  1 <NA>       <NA>       <NA>            <NA>      <NA>
##  2 <NA>       <NA>       <NA>            <NA>      <NA>
##  3 <NA>       <NA>       <NA>            <NA>      <NA>
##  4 <NA>       <NA>       <NA>            <NA>      <NA>
##  5 <NA>       <NA>       <NA>            <NA>      <NA>
##  6 <NA>       <NA>       <NA>            <NA>      <NA>
##  7 <NA>       <NA>       <NA>            <NA>      <NA>
##  8 <NA>       <NA>       <NA>            <NA>      <NA>
##  9 <NA>       <NA>       <NA>            <NA>      <NA>
## 10 <NA>       <NA>       <NA>            <NA>      <NA>
## # ... with 55 more rows, and 3 more variables: data_regi
## #   juiz <chr>, txt_decisao <chr>
```

Exercício

- filtre todas as decisões de 2018.

– Dica: função `lubridate::year()`

Resolução

```
decisoes %>%
  filter(year(dmy(data_decisao)) == 2018)
```

```
## # A tibble: 314 x 9
##    id_decisao n_processo  classe_assunto    municipio ca
##    <chr>      <chr>       <chr>             <chr>     <c
##  1 11107242   0009617-63~ Apelação / Roubo~ São Paulo 2ª
##  2 11107425   2227593-93~ Habeas Corpus / ~ Iepê      2ª
##  3 11107492   0076977-24~ Embargos de Decl~ São Paulo 2ª
##  4 11107361   0012191-36~ Agravo de Execuç~ Campinas  2ª
##  5 11107383   2218460-27~ Habeas Corpus / ~ Sorocaba  2ª
##  6 11107331   0006928-63~ Agravo de Execuç~ Sorocaba  2ª
##  7 11107651   0000297-54~ Apelação / Tráfi~ Junqueir~ 2ª
##  8 11107485   2225548-19~ Habeas Corpus / ~ Nazaré P~ 2ª
##  9 11107335   0006934-70~ Agravo de Execuç~ Sorocaba  2ª
## 10 11107340   0006682-67~ Agravo de Execuç~ Sorocaba  2ª
## # ... with 304 more rows, and 3 more variables: data_reg
## #   juiz <chr>, txt_decisao <chr>
```

mutate

`mutate`

- Aceita várias novas colunas iterativamente.
- Novas variáveis devem ter o mesmo `length` que o `nrow` do bd original ou `1`.

mutate em ação

```
decisoes %>%
  select(n_processo, data_decisao, data_registro) %>%
  mutate(tempo = dmy(data_registro) - dmy(data_decisao))
```

```
## # A tibble: 11,731 x 4
##    n_processo                 data_decisao data_registro
##    <chr>                      <chr>        <chr>
##  1 0057003-20.2017.8.26.0000 19/12/2017   19/12/2017
##  2 0052762-03.2017.8.26.0000 19/12/2017   19/12/2017
##  3 0055169-79.2017.8.26.0000 19/12/2017   19/12/2017
##  4 9000580-82.2017.8.26.0032 14/12/2017   19/12/2017
##  5 0052938-79.2017.8.26.0000 14/12/2017   19/12/2017
##  6 9000723-79.2017.8.26.0482 14/12/2017   19/12/2017
##  7 0003276-86.2015.8.26.0075 14/12/2017   19/12/2017
##  8 9000298-11.2017.8.26.0625 14/12/2017   19/12/2017
##  9 0004653-39.2015.8.26.0028 14/12/2017   19/12/2017
## 10 2221930-66.2017.8.26.0000 19/12/2017   19/12/2017
## # ... with 11,721 more rows
```

Exercício

- Crie uma coluna binária `drogas` que vale `TRUE` se no texto da decisão algo é falado de drogas e `FALSE` caso contrário. – Dica: `str_detect`

Obs.: Considere tanto a palavra 'droga' como seus sinônimos, ou algum exemplo de droga e retire os casos em que `txt_decisao` é vazio

Resolução

```
decisoes %>%
  filter(!is.na(txt_decisao)) %>%
  mutate(txt_decisao = tolower(txt_decisao),
         droga = str_detect(txt_decisao,
    "droga|entorpecente|psicotr[óo]pico|maconha|haxixe|coca
  dplyr::select(n_processo,droga)
```

```
## # A tibble: 6,933 x 2
##    n_processo                  droga
##    <chr>                       <lgl>
##  1 9000580-82.2017.8.26.0032 FALSE
##  2 0052938-79.2017.8.26.0000 FALSE
##  3 9000723-79.2017.8.26.0482 FALSE
##  4 0003276-86.2015.8.26.0075 TRUE
##  5 9000298-11.2017.8.26.0625 TRUE
##  6 0004653-39.2015.8.26.0028 TRUE
##  7 9000788-34.2017.8.26.0269 FALSE
##  8 9000673-53.2017.8.26.0482 FALSE
```

summarise

`summarise`

- Retorna um vetor de tamanho 1 a partir de uma operação com as variáveis (aplicação de uma função).
- Geralmente é utilizado em conjunto com `group_by()`.
- Algumas funções importantes: `n()`, `n_distinct()`.

Em ação

```
decisoes %>%
  select(n_processo, municipio, data_decisao) %>%
  #         pega ano da decisão
  mutate(ano_julgamento = year(dmy(data_decisao)),
         # pega o ano do processo 0057003-20.2017.8.26.000
         ano_proc = str_sub(n_processo, 12, 15),
         # transforma o ano em inteiro
         ano_proc = as.numeric(ano_proc),
         # calcula o tempo em anos
         tempo_anos = ano_julgamento - ano_proc) %>%
  group_by(municipio) %>%
  summarise(n = n(),
            media_anos = mean(tempo_anos),
            min_anos = min(tempo_anos),
            max_anos = max(tempo_anos))
```

Resultado

```
## # A tibble: 315 x 5
##    municipio                n media_anos min_anos max_ano
##    <chr>                <int>      <dbl>    <dbl>    <dbl
##  1 Adamantina              17      0.765        0
##  2 Aguaí                   19      1.16         0
##  3 Águas de Lindóia         5      1.4          0
##  4 Agudos                   8      3.25         0
##  5 Altinópolis              7      0.857        0
##  6 Americana               56      1.41         0
##  7 Américo Brasiliense      9      1.56         0
##  8 Amparo                   9      2.11         0
##  9 Andradina               41      0.707        0
## 10 Angatuba                 4      0.5          0
## # ... with 305 more rows
```

usando count()

A função count(), simplifica um group_by %>% summarise %>% ungroup:

```
decisoes %>%
  count(juiz, sort = TRUE) %>%
  mutate(prop = n / sum(n),
         prop = scales::percent(prop))
```

```
## # A tibble: 100 x 3
##    juiz                          n prop
##    <chr>                     <int> <chr>
##  1 Gilberto Ferreira da Cruz   237 2.02%
##  2 Francisco Orlando           226 1.93%
##  3 Diniz Fernando              198 1.69%
##  4 Walter da Silva             183 1.56%
##  5 De Paula Santos             182 1.55%
##  6 Machado de Andrade          182 1.55%
##  7 Newton Neves                180 1.53%
```

+ fácil ainda

mas sem formato %

```
decisoes %>%
  count(juiz, sort = TRUE) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 100 x 3
##    juiz                          n   prop
##    <chr>                     <int>  <dbl>
##  1 Gilberto Ferreira da Cruz   237 0.0202
##  2 Francisco Orlando           226 0.0193
##  3 Diniz Fernando              198 0.0169
##  4 Walter da Silva             183 0.0156
##  5 De Paula Santos             182 0.0155
##  6 Machado de Andrade          182 0.0155
##  7 Newton Neves                180 0.0153
##  8 Leme Garcia                 179 0.0153
##  9 Grassi Neto                 177 0.0151
```

arrange

arrange

- Simplesmente ordena de acordo com as opções.
- Utilizar `desc()` para ordem decrescente ou o sinal de menos (-).

Exercício

- Quem são os cinco relatores mais prolixos?

– Dica: use `str_length()` – Lembre-se da função `head()`

Resolução

```
decisoes %>%
  filter(!is.na(txt_decisao)) %>%
  mutate(tamanho = str_length(txt_decisao)) %>%
  group_by(juiz) %>%
  summarise(n = n(),
            tamanho_mediana = median(tamanho)) %>%
  filter(n >= 10) %>%
  arrange(desc(tamanho_mediana)) %>%
  head()
```

```
## # A tibble: 6 x 3
##   juiz                                n tamanho_mediana
##   <chr>                           <int>           <dbl>
## 1 Airton Vieira                     154           3146.
## 2 Ely Amioka                         81           1847
## 3 Grassi Neto                       141           1675
## 4 Alcides Malossi Junior             95           1541
## 5 Cesar Augusto Andrade de Castro    77           1341
```

Vamos versionar nossos projetos a partir de agora

- Versionamento -> https://www.curso-r.com/blog/2017-07-17-rstudio-e-github/
- Instruções adicionais de instalação http://r-bio.github.io/git-installation/

Exercitando o que sabemos até aqui

- Carregue o arquivo `decisoes.rds` em um objeto chamado `decisoes`.
- Crie um objeto contendo o tempo médio entre decisão e registro por juiz, apenas para processos relacionados a drogas nos municípios de Campinas ou Limeira.

– Obs.: a nova "singularidade" da base de dados será o `juiz`. Na base original, a singularidade era o `processo`

- Salve o objeto resultante em um arquivo chamado `juizes_drogas_CL.rds`.

Resolução

- Carregando

```
#setwd()
decisoes <- read_rds("CADS2018/Exercícios/dados/decisoes.rd
```

Resolução

- tempo médio entre decisão e registro, por juiz, para processos relacionados a drogas nos municípios de Campinas ou Limeira

```
juizes_drogas_CL <- decisoes %>%
  # selecionando as colunas utilizadas (só pra usar o sele
  select(juiz,municipio,txt_decisao,data_registro,data_dec
  # criando variável "droga" a partir do texto da decisão
  mutate(txt_decisao = tolower(txt_decisao),
         droga = str_detect(txt_decisao,
    "droga|entorpecente|psicotr[óo]pico|maconha|haxixe|coca
  # variável tempo,
         tempo = dmy(data_registro) - dmy(data_decisao)) %>
  filter(droga ==TRUE,municipio %in% c("Campinas","Limeira"
  group_by(juiz) %>%
  summarise(tempo_medio = mean(tempo,na.rm=T))
```

Resolução

- Salvando o objeto `juizes_drogas_CL.rds`

```
write_rds(juizes_drogas_CL,"juizes_drogas_CL.rds")
```

Exercitando o versionamento

- Faça commit e push do script e do arquivo `.rds`

tydyr

A partir dessa aula, sempre versione

- Baixe os dados da pasta exercícios (ou faça pull do seu GitHub)
- Configure o GitHub na sua máquina

– Versionamento -> https://www.curso-r.com/blog/2017-07-17-rstudio-e-github/

– Instruções adicionais de instalação http://r-bio.github.io/git-installation/

- Rode todos os pacotes (usando o macetinho) –> pode baixar o script do exercício 6, que já tem tudo.
- Repositório no GitHub https://github.com/fredbsr/aulas_ENAP/tree/master/CADS2018

Alterando o formato de dados

Até agora, estudamos os principais ferramentas de transformação de dados do `dplyr`. Agora vamos aumentar nosso toolkit com `tidyr`

- Vamos utilizar uma nova base de dados, que completa a de decisões.

```
processos <- read_rds("CADS2018/Exercícios/dados/processos_
```

Fomato tidy

- Hadley Wickham http://r4ds.had.co.nz/tidy-data.html

Funções do pacote

- Enquanto o `dplyr` faz recortes na base (com `filter()`e `select()`) e adições simples (`mutate()`, `summarise()`), o `tidyr` mexe no **formato** da tabela (`gather()`, `spread()`) e faz modificações menos triviais.
- As funções do `tidyr` geralmente vêm em pares com seus inversos:
  - `gather()` e `spread()`,
  - `nest()` e `unnest()`,
  - `separate()` e `unite()`

Onde estamos

http://r4ds.had.co.nz/wrangle-intro.html

gather()

- gather() empilha o banco de dados

```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  # 1. nome da coluna que vai guardar os nomes de colunas
  # 2. nome da coluna que vai guardar os valores das colun
  # 3. seleção das colunas a serem empilhadas
  gather(key="variavel", value="valor", -id_decisao) %>%
  arrange(id_decisao)
```

```
## # A tibble: 69,996 x 3
##    id_decisao variavel       valor
##    <chr>      <chr>          <chr>
##  1 11026431   n_processo     0000009-51.2015.8.26.0546
##  2 11026431   classe_assunto Apelação / Tráfico de Droga
##  3 11026431   municipio      Itapira
##  4 11026431   camara         5ª Câmara de Direito Crimi
```

spread()

- spread() espalha uma variável nas colunas e preenche com outra variável
- É essencialmente a função inversa de gather

```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  gather(key, value, -id_decisao) %>%
  # 1. coluna a ser espalhada
  # 2. valores da coluna
  spread(key, value)
```

```
## # A tibble: 11,666 x 7
##    id_decisao camara  classe_assunto  data_decisao data_
##    <chr>      <chr>   <chr>           <chr>        <chr>
##  1 11026431   5ª Câm~ Apelação / Trá~ 30/11/2017   01/12
##  2 11026432   5ª Câm~ Apelação / Fur~ 30/11/2017   01/12
##  3 11026433   5ª Câm~ Apelação / Rou~ 30/11/2017   01/12
```

Exercício

- Qual juiz julga a maior proporção de processos que tratam de drogas

– Dica: construa um `data.frame` contendo as colunas juiz, n_processos_drogas, n_processos_n_drogas e total_processos, remodelando os dados para haver um juiz por linha e utilizando `spread()`

Resolução

```
## # A tibble: 65 x 5
## # Groups:   juiz [65]
##    juiz                    droga n_droga total proporcao
##    <chr>                   <dbl>   <dbl> <dbl>     <dbl>
##  1 Airton Vieira              23     131   154     0.149
##  2 Alcides Malossi Junior     23      72    95     0.242
##  3 Alexandre Almeida          41     122   163     0.252
##  4 Amaro Thomé                36      96   132     0.273
##  5 Andrade Sampaio            35      79   114     0.307
##  6 Angélica de Almeida         2       6     8     0.25
##  7 Antonio Tadeu Ottoni        0       1     1     0
##  8 Bandeira Lins               0       2     2     0
##  9 Camargo Aranha Filho       32     109   141     0.227
## 10 Camilo Léllis              32     133   165     0.194
## # ... with 55 more rows
```

Exercício

- Qual quantidade mensal de decisões por juiz?
- Dica: use `data_decisao` `dmy()` e `month()`

Resolução

```
decisoes %>%
  filter(!is.na(txt_decisao)) %>%
  mutate(txt_decisao = tolower(txt_decisao),
         droga = str_detect(txt_decisao,
    "droga|entorpecente|psicotr[óo]pico|maconha|haxixe|coca
    droga=case_when(
      droga==TRUE ~ "droga",
      droga==FALSE ~ "n_droga"
    )) %>%
  group_by(juiz,droga) %>%
  summarise(n=n()) %>%
  spread(droga,n,fill = 0) %>%
  mutate(total=droga+n_droga,
         proporcao=droga/total)
```

Resultado

```
## # A tibble: 65 x 5
## # Groups:   juiz [65]
##    juiz                    droga n_droga total proporcao
##    <chr>                   <dbl>   <dbl> <dbl>     <dbl>
##  1 Airton Vieira              23     131   154     0.149
##  2 Alcides Malossi Junior     23      72    95     0.242
##  3 Alexandre Almeida          41     122   163     0.252
##  4 Amaro Thomé                36      96   132     0.273
##  5 Andrade Sampaio            35      79   114     0.307
##  6 Angélica de Almeida         2       6     8     0.25
##  7 Antonio Tadeu Ottoni        0       1     1     0
##  8 Bandeira Lins               0       2     2     0
##  9 Camargo Aranha Filho       32     109   141     0.227
## 10 Camilo Léllis              32     133   165     0.194
## # ... with 55 more rows
```

Exemplo para o ggplot

Unindo e separando colunas

- `unite` junta duas ou mais colunas usando algum separador (_, por exemplo).
- `separate` faz o inverso de `unite`, e uma coluna em várias usando um separador.

Exemplo de separação de colunas

- Olhe os valores da variável classe_assunto

Exemplo de separação de colunas

- Vamos separar a coluna classe_assunto em duas colunas
- coluna classe e coluna assunto
- Existe separador? -> sim, /
- Usei count apenas em assunto

Em ação

```
decisoes %>%
  select(n_processo, classe_assunto) %>%
  separate(classe_assunto, c('classe', 'assunto'), sep = '
           extra = 'merge', fill = 'right') %>%
  count(assunto, sort = TRUE)

## count é um jeito resumido de usar group_by() %>% summar
```

Em ação

```
## # A tibble: 152 x 2
##    assunto                                    n
##    <chr>                                  <int>
##  1 Tráfico de Drogas e Condutas Afins      2441
##  2 Pena Privativa de Liberdade             1106
##  3 Roubo Majorado                          1093
##  4 Furto Qualificado                        838
##  5 Roubo                                    780
##  6 Progressão de Regime                     607
##  7 Furto                                    450
##  8 Receptação                               353
##  9 Homicídio Qualificado                    329
## 10 Crimes de Trânsito                       322
## # ... with 142 more rows
```

List columns: `nest()` e `unnest()`

`nest()` e `unnest()` são operações inversas e servem para tratar dados complexos, como o que temos em `processos`

```
d_partes <- processos %>%
  select(n_processo, partes) %>%
  unnest(partes)
```

As list columns são uma forma condensada de guardar dados que estariam em múltiplas tabelas. Por exemplo, uma alternativa à colocar as `partes` numa list column seria guardar a tabela `d_partes` separadamente.

```
glimpse(d_partes)
```

```
## Observations: 37,579
## Variables: 5
## $ n_processo <chr> "0000003-71.2016.8.26.0073", "0000003
## $ id         <int> 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1
## $ name       <chr> "JOSE MARIA JUSTINO NETO", "Defensori
## $ part       <chr> "Apelante", "Apelante", "Apelado", "A
## $ role       <chr> "Apelante", "Apelante", "Apelado", "A
```

Duplicatas

Para retirar duplicatas, utilizar `distinct`. Ele considera apenas a primeira linha em que encontra um padrão para as combinações de variáveis escolhidas e descarta as demais.

```
decisoes %>%
  distinct(municipio)
```

```
## # A tibble: 315 x 1
##    municipio
##    <chr>
##  1 Cosmópolis
##  2 São Paulo
##  3 Ribeirão Preto
##  4 Araçatuba
##  5 Presidente Prudente
##  6 Bertioga
##  7 Taubaté
##  8 Aparecida
```

Por coluna

Para manter as demais colunas, use `.keep_all=`:

```
decisoes %>%
  distinct(municipio, camara,
           .keep_all = TRUE)
```

```
## # A tibble: 2,760 x 9
##    id_decisao n_processo  classe_assunto    municipio ca
##    <chr>      <chr>       <chr>             <chr>     <c
##  1 11094999   0057003-20~ Habeas Corpus / ~ Cosmópol~ 3ª
##  2 11093733   0052762-03~ Habeas Corpus / ~ São Paulo 3ª
##  3 11093677   0055169-79~ Habeas Corpus / ~ Ribeirão~ 3ª
##  4 11093270   9000580-82~ Agravo de Execuç~ Araçatuba 8ª
##  5 11093374   0052938-79~ Mandado de Segur~ São Paulo 8ª
##  6 11093320   9000723-79~ Agravo de Execuç~ Presiden~ 8ª
##  7 11091506   0003276-86~ Apelação / Tráfi~ Bertioga  8ª
##  8 11093326   9000298-11~ Agravo de Execuç~ Taubaté   8ª
##  9 11092475   0004653-39~ Apelação / Tráfi~ Aparecida 8ª
```

janitor::get_dupes()

Use `janitor::get_dupes()` para averiguar os casos em que há repetição de combinações de colunas.

```
decisoes %>%
  get_dupes(n_processo)
```

```
## # A tibble: 114 x 10
##    n_processo   dupe_count id_decisao classe_assunto
##    <chr>             <int> <chr>      <chr>
##  1 0000276-86.~          2 11051087   Apelação / Tráfic
##  2 0000276-86.~          2 11093633   Embargos de Decla
##  3 0000358-10.~          2 11108278   Embargos de Decla
##  4 0000358-10.~          2 11028129   Apelação / Roubo
##  5 0002236-18.~          2 11041351   Apelação / Contra
##  6 0002236-18.~          2 11041352   Apelação / Contra
##  7 0004453-20.~          2 11041132   Apelação / Tráfic
##  8 0004453-20.~          2 11093635   Embargos de Decla
##  9 0004636-51.~          3 11032094   Apelação / Tráfic
```

Joins

Dados relacionais

- Hadley Wickham http://r4ds.had.co.nz/relational-data.html

Principais funções

Para juntar tabelas, usar `inner_join`, `left_join`, `anti_join`, etc.

Visualizando

Exemplo de inner join:

```
decisoes %>%
  filter(data_registro == "18/01/2018", !is.na(id_decisao))
  select(id_decisao, n_processo) %>%
  inner_join(processos, "n_processo")
```

```
## # A tibble: 169 x 5
##    id_decisao n_processo                    infos
##    <chr>      <chr>                         <list>
##  1 11109089   0003779-93.2015.8.26.0597 <tibble [14 x 2]
##  2 11109088   3001293-25.2013.8.26.0510 <tibble [13 x 2]
##  3 11108246   0063566-45.2015.8.26.0050 <tibble [14 x 2]
##  4 11108245   0003528-84.2015.8.26.0400 <tibble [14 x 2]
##  5 11109087   0008470-76.2015.8.26.0072 <tibble [14 x 2]
##  6 11109086   0013767-62.2012.8.26.0624 <tibble [14 x 2]
##  7 11109085   3019561-54.2013.8.26.0405 <tibble [14 x 2]
##  8 11108348   0003072-91.2017.8.26.0521 <tibble [11 x 2]
##  9 11108725   0009578-41.2017.8.26.0050 <tibble [12 x 2]
## 10 11108347   3001116-52.2013.8.26.0028 <tibble [12 x 2]
## # ... with 159 more rows
```

Exemplo de right join:

```
decisoes %>%
  filter(data_registro == "18/01/2018", !is.na(id_decisao))
  select(id_decisao, n_processo) %>%
  right_join(processos, "n_processo")
```

```
## # A tibble: 11,638 x 5
##    id_decisao n_processo                    infos
##    <chr>      <chr>                         <list>
##  1 <NA>       0000003-71.2016.8.26.0073 <tibble [11 x 2]
##  2 <NA>       0000004-09.2017.8.26.0142 <tibble [12 x 2]
##  3 <NA>       0000004-34.2016.8.26.0630 <tibble [12 x 2]
##  4 <NA>       0000004-59.2015.8.26.0633 <tibble [14 x 2]
##  5 <NA>       0000004-62.2014.8.26.0611 <tibble [14 x 2]
##  6 <NA>       0000006-04.2017.8.26.0651 <tibble [12 x 2]
##  7 <NA>       0000006-06.2015.8.26.0576 <tibble [12 x 2]
##  8 <NA>       0000006-63.2017.8.26.0599 <tibble [12 x 2]
##  9 <NA>       0000006-74.2010.8.26.0125 <tibble [14 x 2]
## 10 <NA>       0000006-82.2016.8.26.0604 <tibble [11 x 2]
## # ... with 11,628 more rows
```

Exercício

- Crie um objeto contendo informações sobre os tamanhos das bancadas dos partidos (arquivo `bancadas.rds`), suas respectivas coligações eleitorais para 2018 (arquivo `coligacoes.xlsx`) e o grau de concordância com a agenda do Gov Temer (arquivo `governismo_temer.xlsx`).

Resolução

Exercício

Com base no objeto criado:

- Crie uma coluna unindo partido e candidato, sem excluir as originais
- Bônus: use `group_by` e `summarise` para identificar

– qual candidato tem a coligação com menor média de concordância e

– qual candidato tem a coligação com maior soma da proporção total de assentos.

Resolução

Função `expand.grid()` pode ser util

```
expand.grid(ano=c(2010:2012),mes=c(1:2))
```

```
##    ano mes
## 1 2010   1
## 2 2011   1
## 3 2012   1
## 4 2010   2
## 5 2011   2
## 6 2012   2
```

Limpeza

Duplicatas

Para retirar duplicatas, utilizar `distinct`. Ele considera apenas a primeira linha em que encontra um padrão para as combinações de variáveis escolhidas e descarta as demais.

```
decisoes %>%
  distinct(municipio)
```

```
## # A tibble: 315 x 1
##    municipio
##    <chr>
##  1 Cosmópolis
##  2 São Paulo
##  3 Ribeirão Preto
##  4 Araçatuba
##  5 Presidente Prudente
##  6 Bertioga
##  7 Taubaté
##  8 Aparecida
##  9 Jandira
## 10 Flórida Paulista
## # ... with 305 more rows
```

Por coluna

Para manter as demais colunas, use `.keep_all=`:

```
decisoes %>%
  distinct(municipio, camara,
           .keep_all = TRUE)
```

```
## # A tibble: 2,760 x 9
##    id_decisao n_processo  classe_assunto    municipio ca
##    <chr>      <chr>       <chr>             <chr>     <c
##  1 11094999   0057003-20~ Habeas Corpus / ~ Cosmópol~ 3ª
##  2 11093733   0052762-03~ Habeas Corpus / ~ São Paulo 3ª
##  3 11093677   0055169-79~ Habeas Corpus / ~ Ribeirão~ 3ª
##  4 11093270   9000580-82~ Agravo de Execuç~ Araçatuba 8ª
##  5 11093374   0052938-79~ Mandado de Segur~ São Paulo 8ª
##  6 11093320   9000723-79~ Agravo de Execuç~ Presiden~ 8ª
##  7 11091506   0003276-86~ Apelação / Tráfi~ Bertioga  8ª
##  8 11093326   9000298-11~ Agravo de Execuç~ Taubaté   8ª
##  9 11092475   0004653-39~ Apelação / Tráfi~ Aparecida 8ª
## 10 11093773   2221930-66~ Habeas Corpus / ~ Jandira   3ª
## # ... with 2,750 more rows, and 3 more variables: data_r
## #   juiz <chr>, txt_decisao <chr>
```

janitor::get_dupes()

Use `janitor::get_dupes()` para averiguar os casos em que há repetição de combinações de colunas.

```
decisoes %>%
  get_dupes(n_processo)
```

```
## # A tibble: 114 x 10
##    n_processo   dupe_count id_decisao classe_assunto
##    <chr>             <int> <chr>      <chr>
##  1 0000276-86.~          2 11051087   Apelação / Tráfic
##  2 0000276-86.~          2 11093633   Embargos de Decla
##  3 0000358-10.~          2 11108278   Embargos de Decla
##  4 0000358-10.~          2 11028129   Apelação / Roubo
##  5 0002236-18.~          2 11041351   Apelação / Contra
##  6 0002236-18.~          2 11041352   Apelação / Contra
##  7 0004453-20.~          2 11041132   Apelação / Tráfic
##  8 0004453-20.~          2 11093635   Embargos de Decla
##  9 0004636-51.~          3 11032094   Apelação / Tráfic
## 10 0004636-51.~          3 11032093   Apelação / Tráfic
## # ... with 104 more rows, and 4 more variables: data_dec
## #   data_registro <chr>, juiz <chr>, txt_decisao <chr>
```

- Janitor exemplos
  http://sfirke.github.io/janitor/articles/janitor.html
- Missing e imputação https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/
- Outliers
- `stringi` e `stringr`

srvyr

Referências

- Survey http://r-survey.r-forge.r-project.org/survey/
- `srvyr` https://cran.r-project.org/web/packages/srvyr/vignettes/srvyr-vs-survey.html
- Dados amostrais complexos ou aletórias simples (apenas peso amostral)

Desenho amostral

- `as_survey()`

– ids

– strata

– fpc

– nest

– weights

Pronto, agora trabalhe como se estivesse em um *tibble* `tbl_df` normal, que será um `tbl_svy`

Este não é um curso de amostragem

- Para maior aprofundamento, leia:

– http://r-survey.r-forge.r-project.org/svybook/

– https://faculty.psau.edu.sa/filedownload/doc-12-pdf-532657fe8ef0e20eada1f34972a4b0dc-original.pdf

– http://r-survey.r-forge.r-project.org/survey/survey-census.pdf

Exemplo motivador

- Dados de desempenho escolar por escola
  http://r-survey.r-forge.r-project.org/survey/html/api.html

```
data(api) #dados de desempenho escolar
```

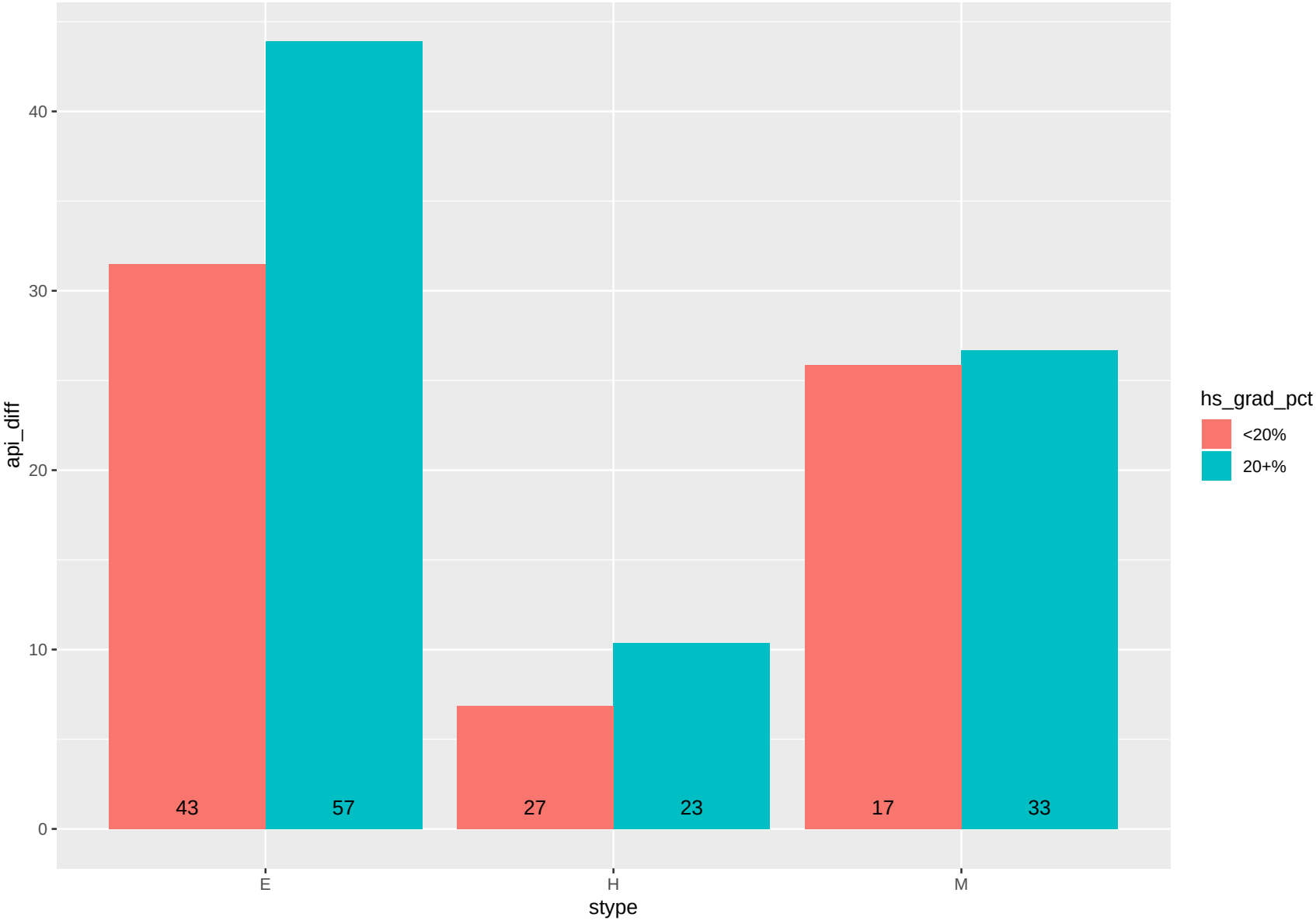
- `hsg` percentual de pais com graduação
- `stype` é o tipo de escola (Elementary/Middle/High School)

```
out <- apistrat %>%
  mutate(hs_grad_pct = cut(hsg, c(0, 20, 100), include.lowe
                           labels = c("<20%", "20+%"))) %>%
  group_by(stype, hs_grad_pct) %>%
  summarize(api_diff = weighted.mean(api00 - api99, pw),
            n = n())
```

Estimativas pontuais

Variável `api-diff`

```
ggplot(data = out, aes(x = stype, y = api_diff, group = hs_
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(y = 0, label = n), position = position_dodg
```


40
30
20
10
0
api_diff
43
57
27
23
17
33
E
H
M
stype
hs_grad_pct
<20%
20+%

Estimativas intervalares

- `survey_total`
- `survey_mean`
- `survey_median()`
- `survey_ratio()`
- `survey_quantile()`
- Vamos ao live code

PNAD

- PNADcIBGE https://rpubs.com/BragaDouglas/335574
- ADSFree http://asdfree.com/pesquisa-nacional-por-amostra-de-domicilios-continua-pnadc.html

Exercício

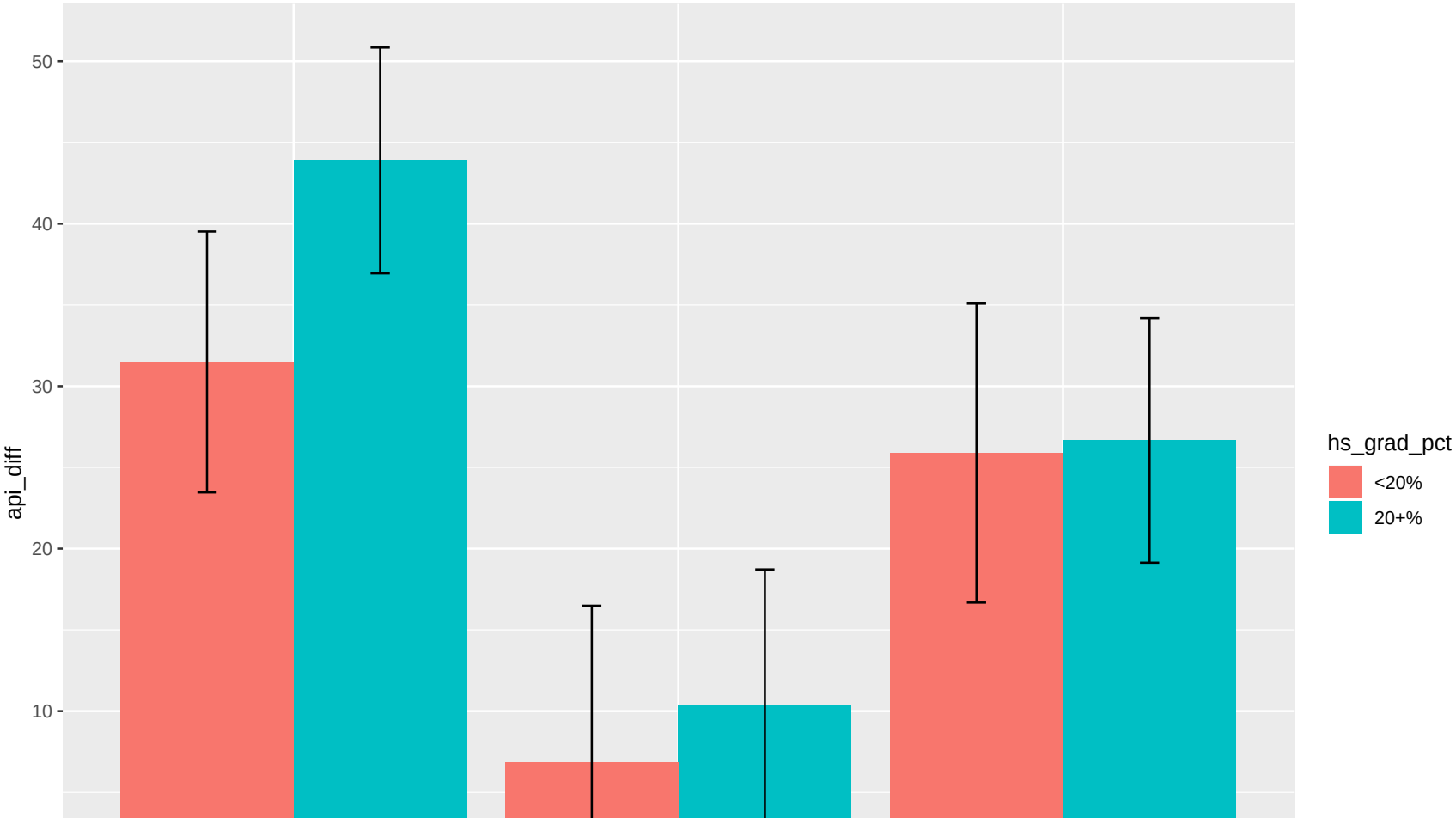
- Carregue os dados de exemplo do pacote survey `data(api)`
- Qual a proporção de escolas por tipo? (use a variável `stype`)
- Quantas escolas há por tipo? (use a variável `stype`)

Resolução

```
strat_design <- apistrat %>%
  as_survey(strata = stype, fpc = fpc, weight  = pw)
```

```
out <- strat_design %>%
  mutate(hs_grad_pct = cut(hsg, c(0, 20, 100), include.lowe
                           labels = c("<20%", "20+%"))) %>%
  group_by(stype, hs_grad_pct) %>%
  summarize(api_diff = survey_mean(api00 - api99, vartype =
            n=unweighted(n()))
```

```
out %>%
  ggplot(aes(x = stype, y = api_diff, group = hs_grad_pct,
                          ymax = api_diff_upp, ymin = api_diff
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(position = position_dodge(width = 0.9), wid
  geom_text(aes(y = 0, label = n), position = position_dodg
```



Exercício

- Carregue os dados de exemplo do pacote survey `data(api)`, use o data.frame `apisrs` e expanda a amostra usando apenas variável fpc contendo a contagem finita de população `as_survey(fpc=fpc)`
- Usando a variável `stype` crie uma nova variável indicando se a escola é de nível fundamental (categorias **E** e **M** de `stype`) ou de nível médio (categoria *H* de `stype`). Dica: use `mutate`e `cas_when`.
- Qual a proporção de escolas por nível (Fundamental, Médio)? (use a variável nova que você criou e a função `survey_mean`),

– Qual o coeficiente de varição dessa proporção?

- Quantas escolas há por nível? (use a variável `stype` e a função `survey_total`)

– Qual é o limite inferior e o limite superior da quantidade de escolas por tipo

Resolução

```
data(api)

srs_design_srvyr <- apisrs %>%
  as_survey(fpc = fpc) %>%
  mutate(nivel=case_when(
    stype=="E"~"Fundamental",
    stype=="M"~"Fundamental",
    stype=="H"~"Médio"
  ))
```

```r
resolucao <- srs_design_srvyr %>%
  group_by(nivel) %>%
  summarize(proporcao = survey_mean(vartype = "cv"),
            n=survey_total(vartype = "ci"))
```

ggplot

Princípios

- O que você quer mostrar? https://i1.wp.com/www.tatvic.com/blog/wp-content/uploads/2016/12/Pic_2.png
- Elementos que podem **destacar** ou **confundir** o que você quer mostrar.

– Exemplo Codeplan

- vamos tentar alternar “teoria” com live code

Recursos

- R Cookbook http://www.cookbook-r.com/Graphs/
- STHDA http://www.sthda.com/english/wiki/be-awesome-in-ggplot2-a-practical-guide-to-be-highly-effective-r-softw
- R Graph Gallery https://www.r-graph-gallery.com/
- Extensões http://www.ggplot2-exts.org/

Elementos do ggplot

- Dados
- Geometrias
- Estéticas
- Escalas (estética)
- Escalas (eixos)
- Tema
- Facet

Dados `data =`

- Dado empilhado?
- Cada coluna será uma entrada!

Geometrias `geom_`

- `geom_tipo_de_geometria`
- Recursos +
- cheat sheet https://www.rstudio.com/wp-content/uploads/2016/03/ggplot2-cheatsheet-portuguese.pdf
- manual ggplot https://ggplot2.tidyverse.org/reference/

Estéticas aes()

- x (xmax e xmin)
- y
- color
- fill
- shape
- group
- size

Escalas (estética) `scale_`

- `scale_color_xx`
- `scale_fill_xx`
- `scale_shape_xx`

Escalas (eixos) `scale_x`

- Contínua `scale_x_continuous`
- Discreta `scale_x_discrete`
- Série de tempo `zoo` –> `scale_yearmon`

Tema

- Customização total da visualização
- Eixos
- Texto `element_text`
- linhas de grade

Facet

- facet_grid
- facet_wrap

Gráficos com interatividade

- ggiraph
- plotly (ggplotly)

Exercício

- Carregue os dados de exemplo do pacote survey `data(api)`, use o data.frame `apisrs`
- Crie o objeto tbl_svy `amostra_expandida` expandindo a amostra aleatória simples usando apenas a variável (coluna) "pw", contendo o peso amostral. Dica: execute `as_survey(weight=pw)`.
- Usando a variável `stype` crie uma nova variável indicando se a escola é de nível fundamental (categorias **E** e **M** de `stype`) ou de nível médio (categoria *H* de `stype`). Dica: use `mutate`e `case_when`.
- Faça um gráfico de barras comparando a variação média das notas de 1999 (`api99`) e 2000 (`api00`) por nível e utilize as estimativas intervalares. Dica: olhe o código da aula 07, utilize `geom_errorbar` para a estimativa intervalar.

Resolução

```
data(api)

amostra_expandida <- apisrs %>%
  as_survey(weight = pw) %>%
  mutate(nivel=case_when(
    stype=="E"~"Fundamental",
    stype=="M"~"Fundamental",
    stype=="H"~"Médio"
  ))
```

```r
out <- amostra_expandida %>%
  group_by(nivel) %>%
  summarise(api_diff = survey_mean(api00 - api99, vartype =
```

```r
out %>%
  ggplot(aes(x = nivel, y = api_diff, fill = nivel,color=ni
                          ymax = api_diff_upp, ymin = api_diff
  geom_bar(stat = "identity",alpha=0.6) +
  geom_errorbar(width = 0,size=3)
```

ioslides no Rmarkdown

Trabalhando com slides no RMarkdown

- Manual https://rmarkdown.rstudio.com/lesson-11.html
- Galeria https://rmarkdown.rstudio.com/gallery.html

File ==> New file ==> R Markdonw ==> Presentation

- HTML (ioslides)

Trabalhando no .rmd

- Opções e detalhes do ioslides https://rmarkdown.rstudio.com/ioslides_presentation_format#overview
- Mais referências https://bookdown.org/yihui/rmarkdown/ioslides-presentation.html
- Montando o arquivo `index.rmd`

gh-pages

- novo “branch”
- nome `gh-pages`
- arquivo `index.html` precisa estar na raiz
- a cada alteração de `index.rmd` e `index.html`, merge de master para gh-pages OU SIMPLESMENTE apague o branch e recrie o gh-pages
- Suestão: só crie o branch gh-pages quando concluir seu trabalho e fizer o
- Seu site estará no endereço ==> nome_de_usuario.github.io/nome_do_repositorio/
- ATENÇÃO: não esqueça da barra final no endereço