Pesquisa reproduzível

Métodos Quantitativos Aplicados à Ciência Política

Frederico Bertholini

05.out.2020

1 Nunca esqueça

2 O Universo tidyverse

3 tydyr

4 Joins

5 Limpeza

6 Pesquisa reproduzível

7 Versionando projetos

8 ioslides no Rmarkdown

9 Gerenciamento de arquivos

Nunca esqueça

Pacotes e diretório de trabalho


```
lista.de.pacotes = c("tidyverse","haven","lubridate","jani
                     "stringr", "magrittr") # escreva a li

novos.pacotes <- lista.de.pacotes[!(lista.de.pacotes %in%
                                    installed.packages()
if(length(novos.pacotes) > 0) {install.packages(novos.paco
lapply(lista.de.pacotes, require, character.only=T)
#rm(lista.de.pacotes,novos.pacotes)
rm(list = ls())
gc()



# Definiindo o diretorio de trabalho como do arquivo local
setwd(dirname(rstudioapi::getActiveDocumentContext()$path)

setwd("/Volumes/Macintosh HD/MQCP_IPOL_2020/Slides/aula 04
```

O Universo tidyverse

Manifesto tidyverse

- 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/

Conjunto de dados

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

```
decisoes <- read_rds("dados/decisoes.rds") %>%
  janitor::clean_names() # com dois pontos eu não preciso
glimpse(decisoes)
```

```
## Rows: 11,731
## Columns: 9
## $ id_decisao     <chr> "11094999", "11093733", "1109367
## $ n_processo     <chr> "0057003-20.2017.8.26.0000", "00
## $ classe_assunto <chr> "Habeas Corpus / Homicídio Simpl
## $ municipio      <chr> "Cosmópolis", "São Paulo", "Ribe
## $ camara         <chr> "3ª Câmara de Direito Criminal",
## $ data_decisao   <chr> "19/12/2017", "19/12/2017", "19/
## $ data_registro  <chr> "19/12/2017", "19/12/2017", "19/
## $ juiz           <chr> "Luiz Antonio Cardoso", "Luiz An
## $ txt_decisao    <chr> NA, NA, NA, "Execução Penal -  C
```

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

tydyr

Alterando o formato de dados

Até agora, estudamos os principais ferramentas de transformação de dados do `dplyr`. Agora vamos aumentar nossa caixa-de-ferramentas com `tidyr`

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

```
processos <- read_rds("dados/processos_nested.rds")
```

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()`, -> substituídas por `pivot_longer` e `pivot_wider`
  - `nest()` e `unnest()`,
  - `separate()` e `unite()`

Onde estamos

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

gather()

- gather() empilha o banco de dados
- pivot_longer empilha de um jeito ainda mais fácil

```
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 Drog
##  3 11026431   municipio      Itapira
```

pivot_longer

Base `relig_income` do `tidyr` 3 variáveis:

- religion, nas linhas
- income, nas colunas e
- count, nas células

```
tidyr::relig_income
```

```
## # A tibble: 18 x 11
##    religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-
##    <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <
##  1 Agnostic      27        34        60        81
##  2 Atheist       12        27        37        52
##  3 Buddhist      27        21        30        34
##  4 Catholic     418       617       732       670
##  5 Don't k~      15        14        15        11
##  6 Evangel~     575       869      1064       982
##  7 Hindu          1         9         7         9
##  8 Histori~     228       244       236       238
```

```
tidyr::relig_income %>%
  pivot_longer(!religion,
               names_to = "income", # diz a varável onde e
               values_to = "count" # diz a variável onde e
               )
```

```
## # A tibble: 180 x 3
##    religion income             count
##    <chr>    <chr>              <dbl>
##  1 Agnostic <$10k                 27
##  2 Agnostic $10-20k               34
##  3 Agnostic $20-30k               60
##  4 Agnostic $30-40k               81
##  5 Agnostic $40-50k               76
##  6 Agnostic $50-75k              137
##  7 Agnostic $75-100k             122
##  8 Agnostic $100-150k            109
##  9 Agnostic >150k                 84
## 10 Agnostic Don't know/refused    96
```

spread()

- spread() espalha uma variável nas colunas e preenche com outra variável
- Função inversa de gather
- Bem mais fácil com pivot_wider

```
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_r
##    <chr>      <chr>  <chr>          <chr>         <chr>
##  1 11026431   5ª Câ~ Apelação / Tr~ 30/11/2017   01/12/
##  2 11026432   5ª Câ~ Apelação / Fu~ 30/11/2017   01/12/
```

pivot_wider


```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  pivot_longer(!id_decisao,
               names_to = "tipo", # diz a varável onde ent
               values_to = "info" # diz a variável onde en
               ) %>%
  # 1. coluna a ser espalhada
  # 2. valores da coluna
  pivot_wider(names_from = "tipo",
              values_from = "info")
```

```
## # A tibble: 11,666 x 7
##    id_decisao n_processo classe_assunto municipio camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
```

- 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


```
## `summarise()` regrouping output by 'juiz' (override wit

## # A tibble: 65 x 5
## # Groups:   juiz [65]
##    juiz                                droga n_droga total
##    <chr>                               <dbl>   <dbl> <dbl>
##  1 Ivana David                            57     101   158
##  2 Diniz Fernando                         66     132   198
##  3 Sérgio Ribas                            1       2     3
##  4 Cesar Augusto Andrade de Castro        25      52    77
##  5 Paulo Rossi                            20      43    63
##  6 Sérgio Mazina Martins                  54     117   171
##  7 Andrade Sampaio                        35      79   114
##  8 Moreira da Silva                       29      67    96
##  9 Machado de Andrade                      3       7    10
## 10 Silmar Fernandes                       44     104   148
## # ... with 55 more rows
```

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

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


```
## `summarise()` regrouping output by 'juiz' (override wit

## # 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)
```

```
## Rows: 37,579
## Columns: 5
## $ n_processo <chr> "0000003-71.2016.8.26.0073", "000000
## $ id         <int> 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2,
## $ name       <chr> "JOSE MARIA JUSTINO NETO", "Defensor
## $ part       <chr> "Apelante", "Apelante", "Apelado", "
## $ role       <chr> "Apelante", "Apelante", "Apelado", "
```

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
```

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 camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
##  5 11093374   0052938-7~ Mandado de Se~ São Paulo 8ª Câ
##  6 11093320   9000723-7~ Agravo de Exe~ Presiden~ 8ª Câ
##  7 11091506   0003276-8~ Apelação / Tr~ Bertioga  8ª Câ
##  8 11093326   9000298-1~ Agravo de Exe~ Taubaté   8ª Câ
##  9 11092475   0004653-3~ Apelação / Tr~ Aparecida 8ª Câ
## 10 11093773   2221930-6~ Habeas Corpus~ Jandira   3ª Câ
```

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 muni
##    <chr>           <int> <chr>      <chr>           <chr
##  1 0000276-8~          2 11051087   Apelação / Tr~ Itap
##  2 0000276-8~          2 11093633   Embargos de D~ Itap
##  3 0000358-1~          2 11108278   Embargos de D~ São
##  4 0000358-1~          2 11028129   Apelação / Ro~ São
##  5 0002236-1~          2 11041351   Apelação / Co~ Nhan
##  6 0002236-1~          2 11041352   Apelação / Co~ Nhan
##  7 0004453-2~          2 11041132   Apelação / Tr~ São
##  8 0004453-2~          2 11093635   Embargos de D~ São
##  9 0004636-5~          3 11032094   Apelação / Tr~ Olím
## 10 0004636-5~          3 11032093   Apelação / Tr~ Olím
```

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:


```
a <- 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           pa
##    <chr>      <chr>                     <list>          <l
##  1 11109089   0003779-93.2015.8.26.~ <tibble [14 x ~ <t
##  2 11109088   3001293-25.2013.8.26.~ <tibble [13 x ~ <t
##  3 11108246   0063566-45.2015.8.26.~ <tibble [14 x ~ <t
##  4 11108245   0003528-84.2015.8.26.~ <tibble [14 x ~ <t
##  5 11109087   0008470-76.2015.8.26.~ <tibble [14 x ~ <t
##  6 11109086   0013767-62.2012.8.26.~ <tibble [14 x ~ <t
##  7 11109085   3019561-54.2013.8.26.~ <tibble [14 x ~ <t
##  8 11108348   0003072-91.2017.8.26.~ <tibble [11 x ~ <t
##  9 11108725   0009578-41.2017.8.26.~ <tibble [12 x ~ <t
## 10 11108347   3001116-52.2013.8.26.~ <tibble [12 x ~ <t
## # ... with 159 more rows
```

Exemplo de right join:


```
b <- 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           pa
##    <chr>      <chr>                   <list>          <l
##  1 11109089   0003779-93.2015.8.26.~ <tibble [14 x ~ <t
##  2 11109088   3001293-25.2013.8.26.~ <tibble [13 x ~ <t
##  3 11108246   0063566-45.2015.8.26.~ <tibble [14 x ~ <t
##  4 11108245   0003528-84.2015.8.26.~ <tibble [14 x ~ <t
##  5 11109087   0008470-76.2015.8.26.~ <tibble [14 x ~ <t
##  6 11109086   0013767-62.2012.8.26.~ <tibble [14 x ~ <t
##  7 11109085   3019561-54.2013.8.26.~ <tibble [14 x ~ <t
##  8 11108348   0003072-91.2017.8.26.~ <tibble [11 x ~ <t
##  9 11108725   0009578-41.2017.8.26.~ <tibble [12 x ~ <t
## 10 11108347   3001116-52.2013.8.26.~ <tibble [12 x ~ <t
## # ... with 11,628 more rows
```

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 camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
##  5 11093374   0052938-7~ Mandado de Se~ São Paulo 8ª Câ
##  6 11093320   9000723-7~ Agravo de Exe~ Presiden~ 8ª Câ
##  7 11091506   0003276-8~ Apelação / Tr~ Bertioga  8ª Câ
##  8 11093326   9000298-1~ Agravo de Exe~ Taubaté   8ª Câ
##  9 11092475   0004653-3~ Apelação / Tr~ Aparecida 8ª Câ
## 10 11093773   2221930-6~ Habeas Corpus~ Jandira   3ª Câ
## # ... with 2,750 more rows, and 3 more variables: data_
## #   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 muni
##    <chr>           <int> <chr>      <chr>           <chr
##  1 0000276-8~          2 11051087   Apelação / Tr~ Itap
##  2 0000276-8~          2 11093633   Embargos de D~ Itap
##  3 0000358-1~          2 11108278   Embargos de D~ São
##  4 0000358-1~          2 11028129   Apelação / Ro~ São
##  5 0002236-1~          2 11041351   Apelação / Co~ Nhan
##  6 0002236-1~          2 11041352   Apelação / Co~ Nhan
##  7 0004453-2~          2 11041132   Apelação / Tr~ São
##  8 0004453-2~          2 11093635   Embargos de D~ São
##  9 0004636-5~          3 11032094   Apelação / Tr~ Olím
## 10 0004636-5~          3 11032093   Apelação / Tr~ Olím
## # ... with 104 more rows, and 3 more variables: data_re
## #   juiz <chr>, txt_decisao <chr>
```

- Janitor exemplos
- Missing e imputação
- mice
- Outliers (critérios, limpeza e gráficos)
- `stringi` e `stringr`–> expressões regulares

Pesquisa reproduzível

Por quê?

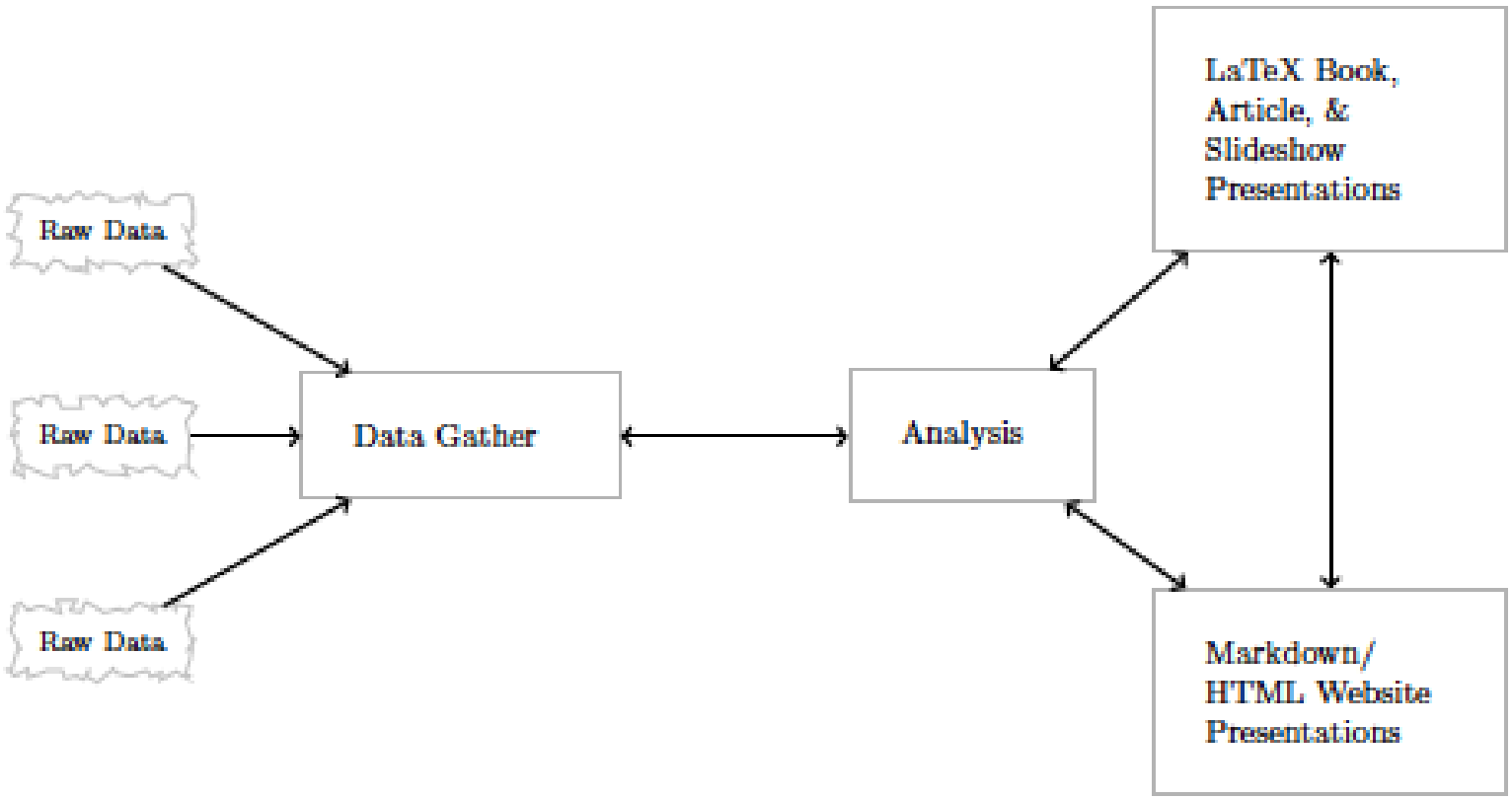
- Pra ciência
- Pra você

Ferramentas

- R e RStudio (ok)
- Github
- knitr e rmarkdown
- LaTeX

Fluxo de trabalho

1. Coleta
2. Análise
3. Comunicação


Raw Data
Raw Data
Raw Data
Data Gather
Analysis
LaTeX Book, Article, & Slideshow Presentations
Markdown/ HTML Website Presentations

Dicas

1. Documente tudo!
2. Tudo é um arquivo (de texto).
3. Todos os arquivos devem ser legíveis (por humanos).
4. Relacione explicitamente seus arquivos.
5. Tenha um plano para organizar, armazenar e disponibilizar seus arquivos.

Versionando projetos

Repositório: Criação de repositório do projeto no Github

.Rproj: Criação do Projeto no RStudio

Commit: Editando e “Commitando” as mudanças no código

Push: Subindo os commits para o Github

Pull: Baixando o estado atual do projeto

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 Markdown ==> 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

Gerenciamento de arquivos