

Mestrado Profissional em Avaliação e
Monitoramento de Políticas Públicas

Métodos Quantitativos I

Aula 1: Apresentação do curso e conceitos básicos

Professores: Daniel Grimaldi e Arthur Bragança

3º Trimestre - 2025

Apresentação do curso

Os Instrutores

Daniel Grimaldi: Economista graduado pela UFRJ, com mestrado na USP e Ph.D. pela George Mason. Trabalha com monitoramento e avaliação de política pública desde 2009, com experiências no Ipea, BNDES, BID e agora na Secretaria de Avaliação de Políticas Públicas e Assuntos Econômicos (SMA/MPO).

Arthur Bragança: Economista sênior na Prática Global de Meio Ambiente, Recursos Naturais e Economia Azul do Banco Mundial. É bacharel em Economia pela Universidade Federal de Minas Gerais, doutor em Economia na PUC-Rio e acadêmico visitante na Universidade de Harvard. Antes de ingressar no Banco Mundial, trabalhou como chefe de avaliação de políticas na Iniciativa de Política Climática (CPI).

Escopo do curso

Esse é um curso introdutório de **econometria**.

*The Econometric Society is an international society for the advancement of economic theory in its relation to **statistics** and **mathematics**. (...) Its main object shall be to promote studies that aim at a unification of the theoretical-quantitative and the empirical-quantitative approach to **economic problems** and that are penetrated by constructive and rigorous thinking similar to that which has come to **dominate in the natural sciences**. (Frisch 1933)*

Escopo do curso

Esse é um curso **introdutório de econometria aplicada**.

- Objetivo é que todos terminem o curso com capacidade de usar ferramental quantitativo para estudar problemas socioeconômicos.
 - Compreender, contratar e implementar análises econométricas.
- Não vamos nos aprofundar na teoria...
 - Corolário 1: não vamos cobrar provas formais de teoremas, estimadores etc

Escopo do curso

Esse é um curso **introdutório de econometria aplicada**.

- ❖ Objetivo é que todos terminem o curso com capacidade de usar ferramental quantitativo para estudar problemas socioeconômicos.
 - ❖ Compreender, contratar e implementar análises econométricas.
- ❖ ... mas vamos cobrar consolidação dos conceitos por meio de **aplicação direta do ferramental quantitativo...**
 - ❖ Corolário 2: vocês precisarão aprender e usar linguagem de programação ao longo do curso.

Visão Geral

- ❖ Todas as aulas terão uma parte conceitual e uma parte dedicada a programação aplicada
- ❖ A avaliação será feita por meio de participação em sala (20%) e 8 listas de exercícios (10% cada).
- ❖ Todo o material do curso será postado numa página do *Github*

Conceitos básicos

Espaço amostral e evento

- ❖ Um **espaço amostral** (Ω) é o conjunto de todos os resultados possíveis para um experimento aleatório.
- ❖ Um **evento** é qualquer conjunto de resultados definidos dentro do espaço amostral.
 - ❖ $A = \{A_1, A_2, \dots, A_n\}; A_i \in \Omega \forall i$
- ❖ Se o resultado A_i foi observado e $A_i \in A$, então dizemos que o evento A ocorreu.
 - ❖ Um evento B está contido em $A \Leftrightarrow B_i \in A \forall i$
 - ❖ A e B serão **eventos disjuntos** $\Leftrightarrow A \cap B = \emptyset$
 - ❖ O **complementar de** (A^c) é formado por todos os resultados que fazem parte do conjunto amostral, mas não estão contidos em A , de tal forma que $A \cap A^c = \emptyset$ e $A \cup A^c = \Omega$

Probabilidade

- ❖ Se Ω é enumerável, então $P(A) = \frac{\text{Qtd. de elementos de } A}{\text{Qtd. de elementos em } \Omega}$
- ❖ Se Ω não for enumerável, então $P(A) = \frac{\text{Comprimento de } A}{\text{Comprimento de } \Omega}$
- ❖ Uma função $\varphi(A, \Omega)$ é uma probabilidade \Leftrightarrow satisfaz os Axiomas de Kolmogorov:
 - ❖ (i) $P(\Omega) = 1$;
 - ❖ (ii) $\forall A \in \Omega, P(A) \geq 0$;
 - ❖ (iii) Para toda sequência A_1, A_2, \dots, A_n de eventos disjuntos, temos que $P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$

Proriedades da Probabilidade

- ❖ $P1. P(A) = 1 - P(A^C)$
- ❖ $P2.$ Sendo A e B dois eventos quaisquer, vale que $P(B) = P(B \cap A) + P(B \cap A^C)$
- ❖ $P3.$ Se $A \subset B$, então $P(A) \leq P(B)$
- ❖ $P4. P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- ❖ $P5.$ Para quaisquer eventos A_1, A_2, \dots , vale que: $P(\bigcup_{i=1}^{\infty} A_i) \leq \sum_{i=1}^{\infty} P(A_i)$

- Sendo $P(B) > 0$, a **probabilidade condicional** de A dado que ocorreu B ($P(A|B)$) é dada por $\frac{P(A \cap B)}{P(B)}$. Caso ($P(B) = 0 \Rightarrow P(A|B) = P(A)$).
- Eventos A e B são **independentes**
 $\Leftrightarrow P(A \cap B) = P(A)P(B)$
- Intuição: quando eventos são independentes, a ocorrência de um não informa nada sobre a ocorrência do outro.
 - $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$

O Básico de R

Por quê usar o R?

❖ **Gratuito**

- ❖ Tempo e orçamento direcionados para o trabalho

❖ **Comunidade ativa**

- ❖ Avanços metodológicos chegam primeiro no R;
- ❖ Farto material (gratuito) para treinamento;
- ❖ Diversos fóruns para troca de experiências.

❖ **É uma linguagem de programação, não um software**

- ❖ Amplitude maior de tarefas (data munging, data scrapping, recursos gráficos, automação...)
- ❖ Permite integração com outras linguagens (essa apresentação foi feita com **R + Latex**).

If statistics programs/languages were cars...



Instalando o R

- ❖ Primeiro, você precisa instalar o *R*
- ❖ Mas ter também o *RStudio* faz toda a diferença!
 - ❖ Interface gráfica mais agradável e recursos ‘point-and-click’;
 - ❖ Permite fazer a gestão, instalação e atualização de pacotes (**Sim, você vai precisar instalar/atualizar pacotes todo o tempo!**);
 - ❖ Permite integrar facilmente, via *RMarkdown*, programação em R com *LaTeX*, *SQL*, Python, Julia, C, C++ etc.

Essa apresentação foi gerada com um arquivo RMarkdown

The screenshot shows the RStudio IDE with the following components:

- Source Editor:** Contains R code for data structures and matrices.


```

102- require(tidyverse)
103- ...
104-
105- ## Data structures
106-
107- - vetor (unidimensional e homogêneo);
108- - matriz (bidimensional e homogêneo);
109- - Array (n-dimensional e homogêneo);
110- - Lista (unidimensional e heterogêneo);
111- - data frame (bidimensional e heterogêneo)
112-
113- ## Vetores
114-
115- ##Code chunk 2: Manipulando vetores##
116- \vsml1
117- "[[", echo=TRUE, eval=TRUE, tidy=TRUE, tidy.opts=list(arrow=TRUE, indent=2)]
118- # criando
119- vetor_1 <- c(1, 2, 3, 4, 5)
120- # editando
121- vetor_2 <- c(vetor_1, 6:10)
122- vetor_2
123- # operações com vetores
124- c(vetor_2[2])[2]
125-
126- [1] 1 2 3 4 5 6 7 8 9 10
127- [1] 4
128-
129- ## Matrices
130-
131- \vspace(0.6cm)
132- ##Code chunk 3: Manipulando matrizes##
133- \vs115
134- # Chunk 3
      
```
- Console:** Shows the output of the code.


```

[1] 1 2 3 4 5
[1] 1 2 3 4 5
> lista_1[[1]]
[1] [1] [2] [3] [4] [5]
[1] 1 3 5 7 9
[2] 2 4 6 8 100
> lista_1[[2]]
[1] 1 2 3 4 5
> # criando
> lista_1 <- list(matriz_1, vetor_1)
> lista_1[[1]]
[1] [1] [2] [3] [4] [5]
[1] 1 3 5 7 9
      
```
- Environment:** Shows the current environment with variables:

Object	Class	Value
lista_1	List of 2	
matriz_1	num [1:2, 1:5]	1 2 3 4 5 6 7 8 9 100
vetor_1	num [1:5]	1 2 3 4 5
vetor_2	num [1:10]	1 2 3 4 5 6 7 8 9 10
- R Documentation:** Shows the documentation for 'Matrix Transpose'.

Matrix Transpose

Description
Given a matrix or [data.frame](#) `x`, `t` returns the transpose of `x`.

Usage
`t(x)`

Arguments
`x` a matrix or data frame, typically.

Details
This is a generic function for which methods can be written. The description here applies to the default and "data.frame" methods.
A data frame is first coerced to a matrix: see [as.matrix](#). When `x` is a vector, it is treated as a column, i.e., the result is a 1-row matrix.

Value
A matrix, with `dim` and `dimnames` constructed appropriately from those of `x`, and other attributes except names copied across.

Note

Instalando pacotes

Code Chunk 1: Instalação de pacotes no R

```
# 'Simples'  
install.packages("tidyverse")  
require(tidyverse)  
  
# 'Sofisticado'  
if (!require(tidyverse)) {  
  install.packages("tidyverse")  
  require(tidyverse)  
}
```


Estruturas de dados em R

- ❏ Vetor (unidimensional e homogêneo);
- ❏ Matrix (bidimensional e homogêneo);
- ❏ Array (n-dimensional e homogêneo);
- ❏ Lista (unidimensional e heterogêneo);
- ❏ Data frame (bidimensional e heterogêneo)

Vetores

Code Chunk 2: Manipulando vetores

```
# criando  
vetor_1 <- c(1, 2, 3, 4, 5)  
# editando  
vetor_2 <- c(vetor_1, 6:10)  
vetor_2
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
# operações com vetores  
c(vetor_2 * 2)[2]
```

```
## [1] 4
```

Matrizes

Code Chunk 3: Manipulando matrizes

```
# criando
```

```
matrix_1 <- matrix(vetor_2, nrow = 2, ncol = 5)  
matrix_1[2, 5] <- 100  
matrix_1
```

```
##      [,1] [,2] [,3] [,4] [,5]  
## [1,]    1    3    5    7    9  
## [2,]    2    4    6    8   100
```

```
# operações com matrizes
```

```
matrix_1 * 2
```

```
##      [,1] [,2] [,3] [,4] [,5]  
## [1,]    2    6   10   14   18  
## [2,]    4    8   12   16  200
```

```
matrix_1 %*% t(matrix_1)
```

```
##      [,1] [,2]  
## [1,]  165 1000  
## [2,] 1000 10120
```

Listas

Code Chunk 4: Manipulando listas

```
# criando
```

```
lista_1 <- list(matrix_1, vetor_1)
```

```
lista_1[[1]]
```

```
##      [,1] [,2] [,3] [,4] [,5]
```

```
## [1,]    1    3    5    7    9
```

```
## [2,]    2    4    6    8   100
```

```
lista_1[[2]]
```

```
## [1] 1 2 3 4 5
```

```
lista_1[[1]][2, 5]
```

```
## [1] 100
```

```
lista_1[[2]][2]
```

```
## [1] 2
```

Criando bases no R

Code Chunk 5: Criando bases de dados no R

```
sample_n=10000
data <- data.frame("id"=paste("cpf", 1:sample_n, sep="_"),
                   "tipo_local"=sample(c("urbano", "rural"),
                                       size=sample_n,
                                       prob=c(0.84, 1-0.84),
                                       replace=TRUE))

head(data, 5)
```

```
##      id tipo_local
## 1 cpf_1      rural
## 2 cpf_2      urbano
## 3 cpf_3      urbano
## 4 cpf_4      urbano
## 5 cpf_5      rural
```


Exportando bases no R

Code Chunk 6: Exportando bases

```
# gerando diretorio
if(dir.exists("./01 - Bases")){
  unlink("./01 - Bases", recursive = TRUE)
}
dir.create("./01 - Bases", showWarnings = FALSE)

# em formato nativo do R
save(data, file="./01 - Bases/teste_data.Rdata")
# em arquivo de texto (txt, csv etc)
write.table(data, file = "./01 - Bases/teste_data.txt", sep = " ",
            row.names = FALSE)
write.table(data, file = "./01 - Bases/teste_data.csv", sep = ";",
            row.names = FALSE)
write_csv2(data, file="./01 - Bases/teste_data.csv", append = TRUE)
# write_delim é outra opção

# em arquivo formato excel (requer package openxlsx)
write.xlsx(data, file="./01 - Bases/teste_data.xlsx")

# em arquivo STATA (requer package: haven)
write_dta(data, path = "./01 - Bases/teste_data.dta")
```

Importando dados

Code Chunk 7: Importando dados no R

```
# R native file
load(file="./01 - Bases/teste_data.Rdata")

# arquivo de texto (txt, csv etc)
data <- read_delim(file="./01 - Bases/teste_data.txt", delim = " ")
data <- read_csv2(file="./01 - Bases/teste_data.csv")

# arquivo em formato excel
data <- read.xlsx("./01 - Bases/teste_data.xlsx", sheet = 1)
str(data)

## 'data.frame':    10000 obs. of  2 variables:
## $ id           : chr  "cpf_1" "cpf_2" "cpf_3" "cpf_4" ...
## $ tipo_local: chr  "rural" "urbano" "urbano" "urbano" ...
```

Code Chunk 8: Criando variáveis na base

```
data %<>%  
  mutate(urbano = case_when(tipo_local=="urbano" ~ 1,  
                             TRUE ~ 0))  
head(data)
```

```
##      id tipo_local urbano  
## 1 cpf_1      rural        0  
## 2 cpf_2      urbano        1  
## 3 cpf_3      urbano        1  
## 4 cpf_4      urbano        1  
## 5 cpf_5      rural        0  
## 6 cpf_6      rural        0
```

Code Chunk 9: bases de dados relacionadas

```
data2 <- data.frame("id"=paste("cpf", 1:sample_n, sep="_"),
                    "classe_renda"=sample(c("1sm", "1sm+"),
                                           size=sample_n,
                                           prob=c(0.65, 1-0.65),
                                           replace=TRUE))

data %<>%
  left_join(data2, by=c("id"))

# existem outras opções de join: (inner_join, right_join, full_join)
head(data)
```

```
##      id tipo_local urbano classe_renda
## 1 cpf_1      rural      0         1sm+
## 2 cpf_2      urbano      1          1sm
## 3 cpf_3      urbano      1         1sm+
## 4 cpf_4      urbano      1          1sm
## 5 cpf_5      rural      0          1sm
## 6 cpf_6      rural      0         1sm+
```

group_by

Code Chunk 10: agregando a base de dados

```
tab.probs <- data %>%
  group_by(tipo_local, classe_renda) %>%
  summarise(qtd = n_distinct(id)) %>%
  group_by(tipo_local) %>%
  mutate(qtd_local = sum(qtd),
         prob_renda_cond_local = qtd / qtd_local,
         ) %>%
  arrange(tipo_local)
head(tab.probs)
```

```
## # A tibble: 4 x 5
## # Groups:   tipo_local [2]
##   tipo_local classe_renda    qtd qtd_local prob_renda_cond_local
##   <chr>      <chr>      <int>    <int>          <dbl>
## 1 rural      1sm          1124     1666          0.675
## 2 rural      1sm+           542     1666          0.325
## 3 urbano    1sm         5451     8334          0.654
## 4 urbano    1sm+         2883     8334          0.346
```

Estrutura para looping

Code Chunk 11: Usando for para implementar looping

```
# Criando status de renda
data$baixa_renda <- NA
for (i in 1:nrow(data)){
  data$baixa_renda[i] <- ifelse(data$classe_renda[i]=="1sm", 1, 0)
}
head(data[,c("id", "classe_renda", "baixa_renda")])
```

```
##      id classe_renda baixa_renda
## 1 cpf_1      1sm+         0
## 2 cpf_2      1sm         1
## 3 cpf_3      1sm+         0
## 4 cpf_4      1sm         1
## 5 cpf_5      1sm         1
## 6 cpf_6      1sm+         0
```

Criando functions

Code Chunk 12: Criando functions

```
# Criando function
gen_tag_renda <- function(classe_renda_i){
  ifelse(classe_renda_i=="1sm", 1, 0)
}
data %<>%
  mutate(baixa_renda2 = sapply(classe_renda, gen_tag_renda))

head(data[,c("id", "classe_renda", "baixa_renda", "baixa_renda2")])
```

```
##      id classe_renda baixa_renda baixa_renda2
## 1 cpf_1      1sm+         0         0
## 2 cpf_2      1sm         1         1
## 3 cpf_3      1sm+         0         0
## 4 cpf_4      1sm         1         1
## 5 cpf_5      1sm         1         1
## 6 cpf_6      1sm+         0         0
```

Hands on!

Desenho de programa

- ❖ Vamos implementar um programa de combate à pobreza. A população-alvo são indivíduos em famílias com baixa renda - renda *per capita* inferior a 1sm (Evento A).
 - ❖ Não temos informação sobre a renda formal das famílias...
 - ❖ ... mas podemos estimar suas probabilidades.
- ❖ O programa consistirá em uma transferência incondicional de 1sm para qualquer indivíduo que aderir ao programa.
- ❖ Existem duas opções de implementação: (i) presencial ou (ii) digital (via Govbr).

Propensão a adesão

O que sabemos?

- ❖ Propensão à adesão (Evento B) depende da renda.
- ❖ Entre os indivíduos de baixa renda, 90% estão propensos a aderir;
 - ❖ $P(B|A) = 0.9$
- ❖ Entre os indivíduos com alta renda, 40% estão propensos a aderir
 - ❖ $P(B|A^C) = 0.4$.

Capacidade de adesão presencial

O que sabemos?

- ❖ Capacidade de adesão no modelo presencial (Evento C_p) depende apenas da localidade.
- ❖ Entre indivíduos em área urbana (Evento D), 95% conseguirão se inscrever;
 - ❖ $P(C_p|D) = 0.95$
- ❖ Entre indivíduos em área rural, apenas 30% conseguirão se inscrever por falta de pontos de inscrição
 - ❖ $P(C_p|D^C) = 0.3$

Capacidade de adesão virtual

O que sabemos?

- ❖ Capacidade de adesão virtual (Evento C_v) depende apenas da renda.
- ❖ Entre indivíduos de alta renda, 90% consegue se inscrever;
 - ❖ $P(C_v|A^C) = 0.95$
- ❖ Entre indivíduos de baixa renda, 55% consegue se inscrever
 - ❖ $P(C_v|A) = 0.55$

Falha de focalização

Qual é a falha de focalização (Φ) esperada para cada opção de implementação?

✚ $\Phi = \frac{\text{Qtd. indivíduos fora do público-alvo}}{\text{Qtd. de beneficiários}}$

✚ Adesão Presencial: $\Phi_p = P(A^C | B \cap C_p)$

✚ Adesão Virtual: $\Phi_p = P(A^C | B \cap C_v)$

Adesão presencial

$$\Phi_p = \frac{P(A^C \cap B \cap C_p)}{P(B \cap C_p)} \quad (1)$$

$$= \frac{P(A^C \cap B) P(C_p)}{P(B) P(C_p)} \quad (2)$$

$$= \frac{P(B|A^C) P(A^C)}{P(B)} \quad (3)$$

$$= \frac{0.4 P(A^C)}{P(B)} \quad (4)$$

Notas: ¹ Equação (1) vale pela definição de probabilidade condicional. ² Equação (2) é possível porque o Evento C_p é independente de A e B. ³ A passagem de (2) para (3) ocorre também pela definição de probabilidade condicional.

Calculando $P(B)$

$$P(B) = P(B \cap A) + P(B \cap A^C) \quad (5)$$

$$P(B) = P(B|A) P(A) + P(B|A^C) P(A^C) \quad (6)$$

$$P(B) = 0.9 P(A) + 0.4 P(A^C) \quad (7)$$

$$\Rightarrow \Phi_p = \frac{0.4 P(A^C)}{0.9 P(A) + 0.4 P(A^C)}$$

Notas: ¹ (5) vale por conta de P1. ² A passagem de (5) para (6) é possível pela definição de probabilidade condicional.

Estimando $P(A)$ e $P(A^c)$

Code Chunk 13: Download de dados do IBGE

```
api_call = "/t/3278/n1/all/v/allxp/p/all/c386/allxt/c1/allxt/c86/2776,2777,2778"
tab3278 <- get_sidra(api = api_call)
tab3278 <- tab3278[,
  c(5, 12, 14)]
names(tab3278) <- c("qtd",
  "classe_renda", "tipo_local")
head(tab3278)
```

##	qtd	classe_renda	tipo_local
## 2	3603733	9681	1
## 3	1436581	9681	1
## 4	139021	9681	1
## 5	7837682	9681	1
## 6	48947	9681	1
## 7	1944916	9681	2

Estimando $P(A)$ e $P(A^c)$

Code Chunk 14: Estimando probabilidade de adesao

```
tabA <- tab3278 %>%  
  filter(!is.na(qtd)) %>%  
  mutate(classe_renda = case_when(classe_renda<=9683 ~ "A",  
                                  TRUE ~ "Ac")) %>%  
  group_by(classe_renda) %>%  
  summarise(qtd_classe_renda = sum(qtd)) %>%  
  mutate(prob = qtd_classe_renda / sum(qtd_classe_renda))  
head(tabA)
```

```
## # A tibble: 2 x 3  
##   classe_renda qtd_classe_renda prob  
##   <chr>          <dbl> <dbl>  
## 1 A             104280061 0.648  
## 2 Ac             56727459 0.352
```

Falha de focalização: presencial

$$P(A) = 0.648 \quad (8)$$

$$P(A^C) = 0.352 \quad (9)$$

$$\Rightarrow \Phi_p = \frac{0.4 * 0.352}{0.9 * 0.648 + 0.4 * 0.352}$$

$$\Phi_p \approx 19.5\%$$

Adesão virtual

$$\Phi_v = \frac{P(A^C \cap B \cap C_v)}{P(B \cap C_v)} \quad (10)$$

$$= \frac{P(A^C \cap B \cap C_v)}{P(A^C \cap B \cap C_v) \cup P(A \cap B \cap C_v)} \quad (11)$$

$$= \frac{P(A^C) P(B|A^C) P(C_v|A^C \cap B)}{P(A^C) P(B|A^C) P(C_v|A^C \cap B) \cup P(A) P(B|A) P(C_v|A \cap B)} \quad (12)$$

$$= \frac{P(A^C) P(B|A^C) P(C_v|A^C)}{P(A^C) P(B|A^C) P(C_v|A^C) \cup P(A) P(B|A) P(C_v|A)} \quad (13)$$

$$= \frac{P(A^C) P(B|A^C) P(C_v|A^C)}{P(A^C) P(B|A^C) P(C_v|A^C) + P(A) P(B|A) P(C_v|A)} \quad (14)$$

Notas: ¹ Equação (10) vale pela definição de probabilidade condicional. ² Equação (11) é possível por P2. ³ A passagem de (11) para (12) ocorre também pela definição de probabilidade condicional. ⁴ A passagem de (12) para (13) ocorre porque C_v só depende de B por meio de A. ⁵ Equação (14) vale por P4.

Falha de focalização: virtual

Todos os termos da Equação 13 são já conhecidos. Basta substituir:

$$P(A) = 0.648 \quad (15)$$

$$P(A^C) = 0.352 \quad (16)$$

$$P(B|A^C) = 0.4 \quad (17)$$

$$P(B|A) = 0.9 \quad (18)$$

$$P(C_v|A^C) = 0.95 \quad (19)$$

$$P(C_v|A) = 0.55 \quad (20)$$

$$\Rightarrow \Phi_v = \frac{0.352 * 0.4 * 0.95}{(0.352 * 0.4 * 0.95) + (0.648 * 0.9 * 0.55)}$$

$$\Phi_v \approx 29.4\%$$

Validação por simulação

Vamos construir um Processo Gerador de Dados (PGD) para simular o processo de adesão ao nosso programa

- ❖ O PGD deve utilizar as mesmas probabilidades primárias
 - ❖ Para isso, vamos precisar estimar a proporção de indivíduos em áreas urbanas e rurais - $P(D)$
- ❖ Com isso, podemos conferir nossos resultados

Estimando $P(D)$

Code Chunk 15: Estimando $P(D)$

```
tabD <- tab3278 %>%  
  filter(!is.na(qtd)) %>%  
  mutate(tipo_local = case_when(tipo_local==1 ~ "urbano",  
                                TRUE ~ "rural")) %>%  
  group_by(tipo_local) %>%  
  summarise(qtd_local = sum(qtd)) %>%  
  mutate(probD = qtd_local / sum(qtd_local))  
head(tabD)
```

```
## # A tibble: 2 x 3  
##   tipo_local qtd_local probD  
##   <chr>      <dbl> <dbl>  
## 1 rural      24309163 0.151  
## 2 urbano    136698357 0.849
```

Code Chunk 16: Construindo PGD

```
sample_n=100000
data <- data.frame("id"=paste("cpf", 1:sample_n, sep="_"),
  "baixa_renda" = sample(0:1, size=sample_n,
    prob=c(1-0.648, 0.648),
    replace=TRUE),
  "tipo_local" = sample(c("urbano", "rural"),
    size=sample_n,
    prob=c(0.849, 1-0.849),
    replace=TRUE)) %>%
mutate(prop_adesao = case_when(baixa_renda==1 ~ 0.9,
  TRUE ~ 0.4),
  cap_p = case_when(tipo_local=="urbano" ~ 0.95,
  TRUE ~ 0.3),
  cap_v = case_when(baixa_renda==1 ~ 0.55,
  TRUE ~ 0.95),
  prob_adesao_p = prop_adesao * cap_p,
  prob_adesao_v = prop_adesao * cap_v)
```

Simulação

Code Chunk 17: Simulando a adesão em cada cenário

```
adesao_p <- function(prob_i){sample(0:1, size=1,  
                                   prob=c(1-prob_i, prob_i))}  
adesao_v <- function(prob_i){sample(0:1, size=1,  
                                   prob=c(1-prob_i, prob_i))}  
data %<>%  
  mutate(benef_p = sapply(prob_adesao_p, adesao_p),  
         benef_v = sapply(prob_adesao_v, adesao_v))  
head(data, 3)
```

```
##      id baixa_renda tipo_local prop_adesao cap_p cap_v prob_adesao_p  
## 1 cpf_1           1      urbano          0.9  0.95  0.55          0.855  
## 2 cpf_2           1      urbano          0.9  0.95  0.55          0.855  
## 3 cpf_3           0      rural           0.4  0.30  0.95          0.120  
##   prob_adesao_v benef_p benef_v  
## 1           0.495      1       0  
## 2           0.495      1       1  
## 3           0.380      0       0
```


Falha de focalização: simuladas

Code Chunk 18: Adesão presencial

```
sum(data$benef_p*(1-data$baixa_renda))/sum(data$benef_p)
```

```
## [1] 0.1941821
```

Code Chunk 19: Adesão virtual

```
sum(data$benef_v*(1-data$baixa_renda))/sum(data$benef_v)
```

```
## [1] 0.2959276
```

Encerramento

Preparação para a próxima aula

- ❏ Instalem o *R* e o *RStudio*;
- ❏ Tentem replicar os códigos da aula de hoje;
- ❏ Se quiserem, podem trazer o laptop para as aulas seguintes.

Dúvidas com R

- ❖ Google:
 - ❖ 'how to [o que você quiser] R'
 - ❖ 'how to [o que você quiser] R Cran' (se estiver procurando por um package específico)
 - ❖ Fóruns importantes para dúvidas de programação são [stackoverflow](#) e [R-Bloggers](#)
- ❖ Programadores de R adoram montar Cheat sheets, que são pequenos resumos de comandos. Elas são muito úteis (principalmente no início). Salvamos diversas delas no [GitHub do curso](#).

Referências estruturadas

- ❖ R Basics: *R programming for data science*, *R for data science* e *R Cookbook*
- ❖ Manipulação de dados: as referências de *dplyr* e *tidyr*
- ❖ Gráficos: as referências de *ggplot2* e *R Graph Gallery*
- ❖ Tabelas: as referências de *Kable* e *Stargazer*
- ❖ RMarkdown: *The Definitive guide*
- ❖ Econometrics: *Causal inference: the mixtape* e *The Effect*