Compartilhando o código em R

Vitor Marinho

10/4/23

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

```
1 + 1
```

```
[1] 2
```

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

2 Índice de Gini

Compartilhando Código em R

Este tutorial aborda o cálculo e visualização do Índice de Gini, uma medida estatística amplamente utilizada para avaliar a desigualdade na distribuição de renda em populações ou regiões. O Índice de Gini varia de 0 (igualdade perfeita) a 1 (desigualdade máxima) e é essencial para economistas, pesquisadores e formuladores de políticas públicas compreenderem a desigualdade socioeconômica.

Contato: **transformacao.digital@fjp.mg.gov.br**

O Índice de Gini é uma medida estatística amplamente utilizada para avaliar a desigualdade na distribuição de renda em uma população ou país. Ele fornece uma medida numérica que varia de 0 a 1, onde 0 representa igualdade perfeita (ou seja, todas as pessoas possuem a mesma renda) e 1 representa desigualdade máxima (ou seja, uma única pessoa detém toda a renda, enquanto as demais não possuem nenhuma) (Hoffmann 2006).

Ao calcular o Índice de Gini, é possível obter uma compreensão clara do quão desigual é a distribuição de renda em uma sociedade. Ele é frequentemente usado por economistas, pesquisadores e formuladores de políticas públicas para medir e comparar a desigualdade em diferentes regiões e ao longo do tempo. Com base nessa medida, é possível identificar áreas de alta desigualdade e direcionar esforços para promover políticas sociais e econômicas mais igualitárias.

Aqui estão alguns exemplos de problemas de pesquisa que podem ser analisados usando esse índice:

1. Desigualdade de renda entre diferentes grupos demográficos (por exemplo, gênero, raça, etnia).
2. Variação da desigualdade de renda em diferentes regiões geográficas.
3. Impacto das políticas governamentais na desigualdade de renda.
4. Relação entre desigualdade de renda e pobreza.
5. Efeitos da globalização na distribuição de renda.
6. Influência da educação e qualificação profissional na desigualdade de renda.
7. Efeitos da progressividade ou regressividade do sistema tributário na desigualdade de renda.

8. Comparação da desigualdade de renda ao longo do tempo.

Esses são apenas alguns exemplos, e existem inúmeras questões de pesquisa que podem ser abordadas usando o Índice de Gini como medida de desigualdade de renda.

Cada problema pode exigir abordagens metodológicas diferentes e fontes de dados específicas, mas o Índice de Gini oferece uma base sólida para explorar e compreender questões relacionadas à desigualdade socioeconômica.

2.1 Cálculo do Índice de Gini

O Índice de Gini é calculado da seguinte maneira:

$$G = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|$$

Onde: - (G) é o Índice de Gini. - (n) é o número de observações na amostra. - (x_i) são os valores ordenados da variável de renda. - (x_j) são os valores ordenados da mesma variável de renda. - (| |) é o valor absoluto. - ( ) é a média da variável de renda.

3 Como Calcular o Índice de Gini no R

Neste tutorial, você aprenderá como calcular o Índice de Gini no R usando dados de PIB municipal e população. Antes de começar, certifique-se de ter os pacotes necessários instalados. Caso contrário, você pode instalá-los usando a função `install.packages()`.

```
# Lista de pacotes necessários
pacotes <- c("sidrar", "ineq", "tidyverse",'ggspatial',
             "ggplot2", "plotly", 'geobr', 'gt')

# Verifica se os pacotes estão instalados e instala se necessário
install.packages(setdiff(x = pacotes,
                         y = rownames(installed.packages())))

# Carrega os pacotes
lapply(X = pacotes,
       FUN = library,
       character.only = TRUE)
```

3.0.1 Importando Dados

Primeiro, vamos importar os dados de PIB municipal e população. O pacote SIDRAR nos permite puxar essas bases diretamente para o diretório do R via API.

```
# PIB municipal
pib <- get_sidra(api = '/t/5938/n6/all/v/37/p/last%201/d/v37%200' )
```

```
All others arguments are desconsidered when 'api' is informed
```

```
# População
populacao <- get_sidra(api = '/t/6579/n6/all/v/all/p/2020')
```

```
All others arguments are desconsidered when 'api' is informed
```

Agora, vamos limpar esses dados para reter apenas as informações relevantes.

```
pib <- pib %>% janitor::clean_names() %>%
  select(municipio_codigo,
  municipio,
  ano,
  renda = valor
)

populacao <- populacao %>% janitor::clean_names() %>% select(
  municipio_codigo,
  municipio,
  ano,
  habitantes = valor
)
```

3.0.2 Preparando os Dados

Em seguida, vamos juntar as bases de dados e criar a variável de PIB per capita.

```
# Preparando a base de dados
dados <- left_join(pib, populacao) %>%
  separate(municipio, into = c("municipio", "uf"), sep = " - ") %>%
  filter(uf=='MG') %>%
  mutate(pib_per_cp = renda/habitantes)
```

```
Joining with `by = join_by(municipio_codigo, municipio, ano)`
```

Aqui, estamos filtrando os dados para considerar apenas municípios em Minas Gerais (UF == 'MG').

Lendo nossa base de dados

```
dados %>%
  head(10) %>%
  gt()
```

municipio_codigo	municipio	uf	ano	renda	habitantes	pib_per_cp
3100104	Abadia dos Dourados	MG	2020	153873	7006	21.96303
3100203	Abaeté	MG	2020	555814	23250	23.90598

3100302	Abre Campo	MG	2020	226522	13444	16.84930
3100401	Acaiaca	MG	2020	58057	3994	14.53605
3100500	Açucena	MG	2020	138557	9368	14.79046
3100609	Água Boa	MG	2020	172998	13523	12.79287
3100708	Água Comprida	MG	2020	159718	1992	80.17972
3100807	Aguanil	MG	2020	71565	4522	15.82596
3100906	Águas Formosas	MG	2020	254012	19247	13.19749
3101003	Águas Vermelhas	MG	2020	214552	13599	15.77704

3.0.3 Calcular o Índice de Gini no R

Vamos calcular o Índice de Gini para cada município.

```
# Crie uma nova coluna para a acumulação de renda per capita
dados <- dados %>%
  mutate(acum_renda_per_capita = cumsum(pib_per_cp))

# Crie uma nova coluna para a acumulação de pessoas
dados <- dados %>%
  mutate(acum_pessoas = cumsum(habitantes))

# Calcule o Índice de Gini para cada município
gini_municipios <- dados %>%
  group_by(municipio) %>%
  summarize(gini = 1 - 2 * sum(acum_renda_per_capita / (acum_pessoas + 1)),
            code_muni = municipio_codigo)
# Imprima o Índice de Gini para cada município
gini_municipios %>%
  head(10) %>%
  gt()
```

municipio	gini	code_muni
Abadia dos Dourados	0.9937311	3100104
Abaeté	0.9969680	3100203
Abre Campo	0.9971297	3100302
Acaiaca	0.9967605	3100401
Aguanil	0.9947901	3100807
Aimorés	0.9962572	3101102
Aiuruoca	0.9961477	3101201
Alagoa	0.9959787	3101300

Albertina	0.9955812	3101409
Alfenas	0.9970799	3101607

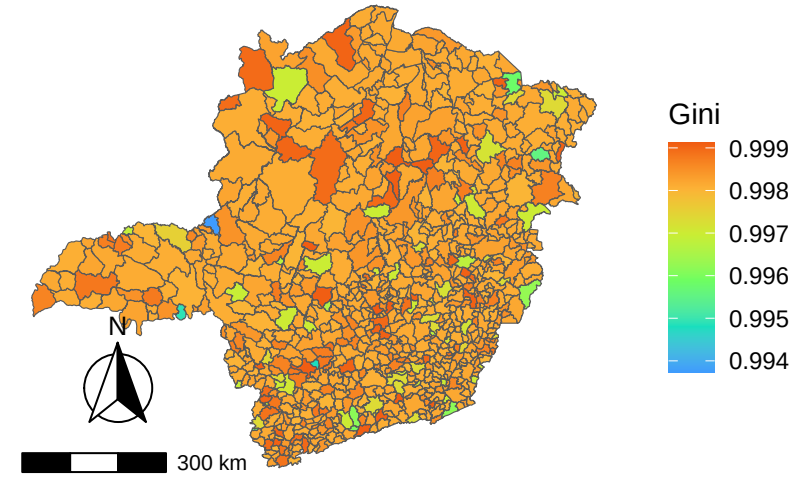
3.0.4 Visualizando os Resultados

Por fim, podemos visualizar a distribuição espacial do Índice de Gini.

```
mun <- read_municipality(code_muni = "MG", year = 2010)

# Converter coluna
gini_municipios$code_muni <- as.double(gini_municipios$code_muni)

# Juntar coordenadas para criar o mapa
gini_municipios<- left_join(gini_municipios, mun , by = 'code_muni')
```

Agora, vamos plotar os resultados em um mapa interativo.

```
# Criar o gráfico ggplot com o Índice Gini como hovertext
ggplot_obj <- gini_municipios %>%
  ggplot() +
  geom_sf(data = gini_municipios$geom, aes(fill = gini_municipios$gini, text = paste("Muni
  scale_fill_viridis_c(option = 15, begin = 0.2, end = 0.8,
                       name = 'Gini') +
  theme(panel.grid = element_line(colour = "transparent"),
        panel.background = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank()) +
  labs(title = "Índice de Gini 2020 dos Municípios de MG",
       subtitle = 'Calculado com base no PIB per capita',
       caption = 'Fonte: Elaboração própria', size = 8)+
  annotation_scale(location = "bl", width_hint = 0.3)+
  annotation_north_arrow(location = "bl", which_north = "true",
        pad_x = unit(0.3, "in"), pad_y = unit(0.3, "in"),
        style = north_arrow_fancy_orienteering) +
  theme(panel.grid = element_line(colour = "transparent"),panel.background =  element_blan

# Converter o gráfico ggplot para plotly com o hovertext
interactive_plot <- ggplotly(ggplot_obj, tooltip = "text")

# Exibir o gráfico interativo
ggplot_obj
```



Fonte: Elaboração própria

4 Preparação de Dados: Pacote DataMaid

Tutorial Transformação Digital nº 2

Neste tutorial, você aprenderá como utilizar o pacote DataMaid no ambiente R para a preparação de dados, uma etapa crucial na análise de dados. O DataMaid é uma ferramenta poderosa que auxilia na verificação e limpeza de dados, fornecendo um documento para análise da estrutura dos dados. Ele é capaz de identificar diversos tipos de erros e inconsistências nos dados, como classes incorretas, duplicatas, inconsistências de capitalização, valores improváveis, espaços em branco, indicadores de falta não reconhecidos e muito mais.

Contato: transformacao.digital@fjp.mg.gov.br

Nesta oficina, aprenderemos como usar o pacote DataMaid no R para uma etapa prévia a crítica de dados: a preparação dos dados.

Tutorial disponível: https://github.com/vitor-marinho-fjp/Critica_Datamaid

O que é o DataMaid?

Um assistente de limpeza de dados capaz de fornecer um documento para ser lido e avaliado por uma pessoa. Uma ferramenta para auxiliar na lógica/verificação de erros tanto em colunas quanto em linhas. (Petersen and Ekstrøm 2019)

4.0.1 Exemplos de erros em verificações de dados para a limpeza de dados:

- Classe incorreta
- Duplicatas
- Consistência de capitalização (**B**elo **H**orizonte vs **B**elo **h**orizonte)
- Valor improvável (peso = 1000, idade = 201)
- Espaços em branco
- Indicadores de falta não reconhecidos
- Quantidade de faltantes (NA)

- Observações/categorias únicas com contagem baixa
- Dados imprecisos (data de falecimento antes da data de nascimento)

4.0.2 Inserção do dataMaid no Fluxo de trabalho em Ciência de Dados

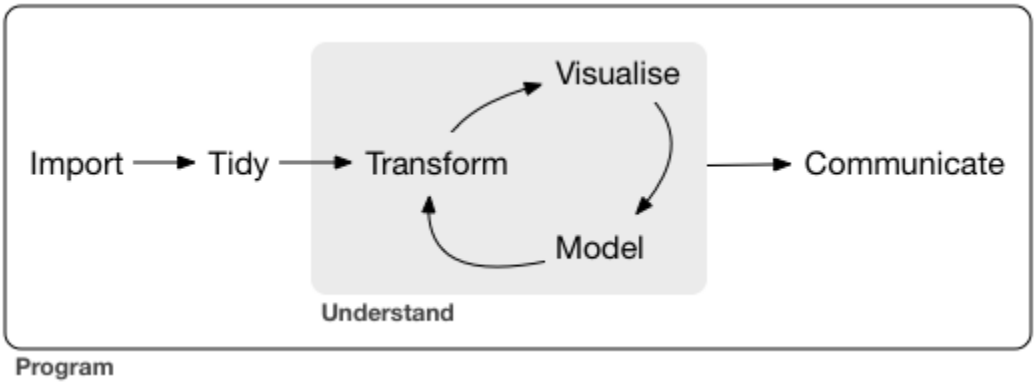


Figure 4.1: Fonte: Wickham & Grolemund (2017).

Foco nos dois primeiros passos: Import → Tidy

5 Import/Tidy

5.1 Pacotes utilizados

```
### Instalação e Carregamento do Pacote

# Lista de pacotes necessários
pacotes <- c('dataMaid', 'tidyverse', 'readxl', 'gt')

# Verifica se os pacotes estão instalados e instala se necessário
install.packages(setdiff(x = pacotes,
                         y = rownames(installed.packages())))

# Carrega os pacotes
lapply(X = pacotes,
       FUN = library,
       character.only = TRUE)
```

Documentação:

5.2 Carregando os Dados de Exemplo Indicadores de Saúde

Disponível em: base_dados

```
dados_datamaid <- read_excel("dados/dados_datamaid.xlsx")

dados_datamaid%>%
  head(5) %>%
  gt()
```

CHAVE	IBGE6	IBGE7	ANO	S_TXBRUTAMORT	S_TXBRUTAMORTPAD	S_TXMOISQ
2000310010	310010	3100104	2000	7.45	6.47	
2000310020	310020	3100203	2000	3.80	3.44	

2000310030	310030	3100302	2000	5.84	5.04
2000310040	310040	3100401	2000	6.43	5.16
2000310050	310050	3100500	2000	6.70	5.95

5.3 Análise Inicial dos Dados

A função `makeDataReport` produz um relatório de visão geral dos dados em que *resume* o conteúdo do conjunto de dados e *sinaliza possíveis problemas*. Esses potenciais erros são identificados executando um conjunto de *verificações de validação* específicas da classe, de modo que diferentes verificações sejam realizadas em diferentes tipos de variáveis. As etapas de verificação podem ser personalizadas de acordo com a entrada do usuário e/ou tipo de dados da variável inserida

Para cada variável, um conjunto de funções de pré-verificação (controladas pelo argumento preChecks) é executado primeiro e depois uma bateria de funções é aplicada dependendo da classe da variável.

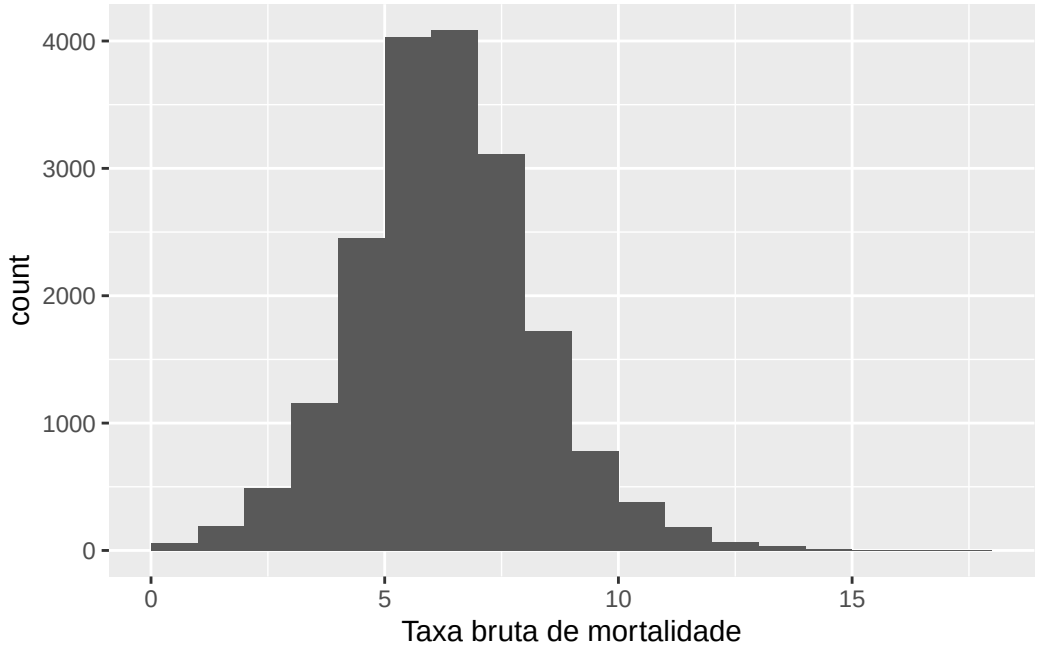
6 Extra: Personalizando o Report

Além da análise inicial, você pode criar regras personalizadas para a análise inicial dos dados.

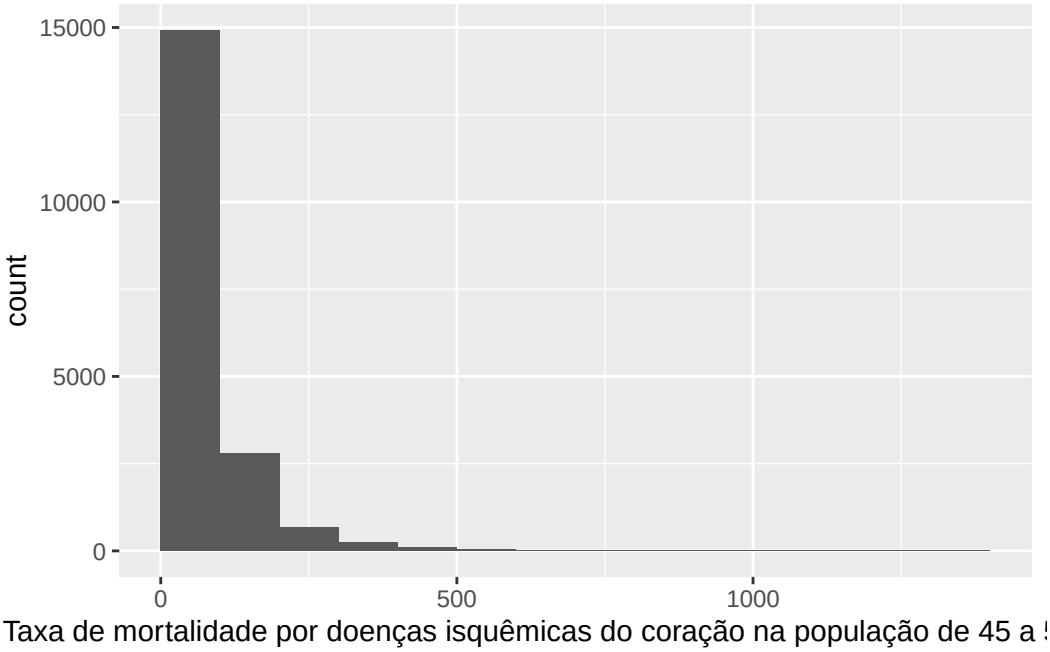
Por exemplo, podemos verificar a variável ***S_TXBRUTAMORT***

Trace a distribuição de uma variável.

```
standardVisual(dados_datamaid$S_TXBRUTAMORT, "Taxa bruta de mortalidade")
```




```
standardVisual(dados_datamaid$S_TXMOISQCOR45A59, "Taxa de mortalidade por doenças isquêmic
```



6.1 Trabalhando com funções personalizadas

6.1.1 Funções de sumarização

Pode-se criar funções que seja do interesse a verificação. Por exemplo, um função para ocorrências de valores iguais a zero: `countZeros()`

Criando a função:

```
countZeros <- function(v, ...) {
  val <- length(which(v == 0))
  summaryResult(list(feature = "No. zeros", result = val, value = val))
}
```

```
countZeros(dados_datamaid$S_TXMOHOMI15A29)
```

```
No. zeros: 12322
```

uma outra função útil seria a contagem dos valores atípicos (outliers) em uma variável numérica `v` usando a abordagem do intervalo interquartil (IQR) com um fator de limiar (threshold):

```r
#Função para Identificar Outliers:
countOutliers <- function(v, threshold = 1.5) {
  q1 <- quantile(v, 0.25)
  q3 <- quantile(v, 0.75)
  iqr <- q3 - q1
  lower_bound <- q1 - threshold * iqr
  upper_bound <- q3 + threshold * iqr
  val <- length(which(v < lower_bound | v > upper_bound))
  summaryResult(list(feature = "No. outliers", result = val, value = val))
}
```

Além dessas, outras possibilidades são possíveis, por exemplo, criar uma função que calcule estatísticas (média, mediana, etc.) para uma variável numérica, segmentadas por grupos de uma variável categórica. Isso pode ser útil para entender as diferenças entre grupos:

```r
calculateStatsByGroup <- function(data, numeric_var, categorical_var) {
  stats_by_group <- data %>%
    group_by({{categorical_var}}) %>%
    summarise(
      Mean = mean({{numeric_var}}, na.rm = TRUE),
      Median = median({{numeric_var}}, na.rm = TRUE),
      SD = sd({{numeric_var}}, na.rm = TRUE)
    )
  summaryResult(list(
    feature = paste("Summary Statistics by", as_label(categorical_var)),
    result = stats_by_group,
    value = NULL
  ))
}
```

6.1.2 Adicionando novas funções ao `report`

A inclusão das funções criadas anteriormente são realizadas da seguinte maneira:

```r
makeDataReport(dados_datamaid, summaries = setSummaries(
  character = defaultCharacterSummaries(add = c("countZeros", "countOutliers")),
  factor = defaultFactorSummaries(add = c("countZeros", "countOutliers")),
  labelled = defaultLabelledSummaries(add = c("countZeros", "countOutliers")),
  numeric = defaultNumericSummaries(add = c("countZeros", "countOutliers")),
  integer = defaultIntegerSummaries(add = c("countZeros", "countOutliers")),
```

```
    logical = defaultLogicalSummaries(add = c("countZeros", "countOutliers"))
  ), replace = TRUE, output = "html")
```

```
Error in quantile.default(v, 0.25) :
  missing values and NaN's not allowed if 'na.rm' is FALSE
```

Funções do DataMaid

name	descrição
identifyCaseIssues	Identificar problemas
identifyLoners	Identificar variáveis com < 6 obs.
identifyMissing	Identificar valores ausentes mal codificados
identifyNums	Identificar variáveis numéricas ou inteiras classificadas incorretamente
identifyOutliers	Identificar outliers
identifyOutliersTBStyle	Identify outliers (Turkish Boxplot style)
identifyWhitespace	Identifique espaços em branco prefixados e sufixados
isCPR	Identify Danish CPR numbers
isEmpty	Verifique se a variável contém apenas um único valor
isKey	Verifique se a variável é uma chave
isSingular	Verifique se a variável contém apenas um único valor
isSupported	Verifique se a classe da variável é suportada pelo dataMaid.

6.2 Recursos Adicionais

Documentação DataMaid - https://github.com/ekstroem/DataMaid

Documentação DataMaid : https://cran.r-project.org/web/packages/dataMaid/index.html

```
vignette("extending_dataMaid")
```

7 Citação

Marinho,V.; Gonçalves, C. **Preparação de Dados: Pacote DataMaid**. Tutorial Transformação Digital. Fundação João Pinheiro, n. 2, 2023. Disponível em: https://rpubs.com/fjp/datamaid.

8 Workshop: Crítica e Imputação de Dados no R: Validate

Tutorial Transformação Digital nº 3

Neste tutorial, exploramos o poderoso pacote validate do R, que é uma ferramenta essencial para validar dados e garantir a qualidade dos resultados em projetos. O pacote validate oferece uma variedade de funções que nos permitem verificar a validade dos dados criando regras de crítica.

Contato: transformacao.digital@fjp.mg.gov.br

O pacote validate do R é uma ferramenta poderosa que pode ser usada para validar dados e garantir a qualidade dos resultados de um projeto. Ele oferece uma variedade de funções que podem ser usadas para verificar a validade de dados, incluindo:

- **Validação de tipos de dados:** O pacote validate pode ser usado para verificar se os dados estão no formato correto. Por exemplo, você pode usar a função `is.numeric()` para verificar se uma variável é um número.
- **Validação de valores:** O pacote validate pode ser usado para verificar se os valores estão dentro de um intervalo aceitável. Por exemplo, você pode usar a função `between()` para verificar se um valor está entre dois valores especificados.
- **Validação de regras:** O pacote validate pode ser usado para verificar se os dados atendem a regras específicas. Por exemplo, você pode usar a função `validate()` para verificar se um valor é maior que outro valor.

Confrontar os dados com as regras e armazenar os resultados

Regras de crítica são expressões que são avaliadas ao ser confrontadas com dados, e resultam em um objeto do tipo 'logical' (TRUE ou FALSE)

Verificações de tipo da variável: is.numeric, is.character,. . .

Comparações: <, <=, ==, identical, !=, %in%, >=, >

Operadores lógicos: |, &, if, !, all, any

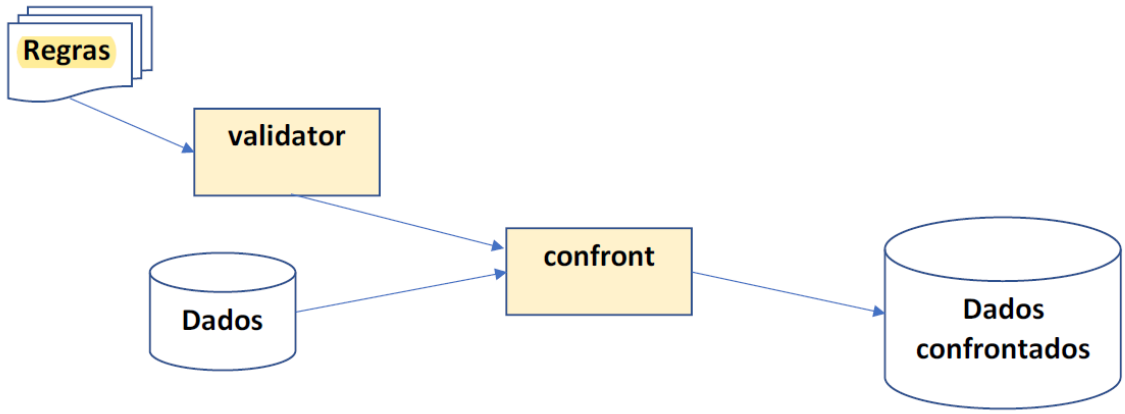


Figure 8.1: Van der Loo; de Jonge (2019).

9 Padronização nas regras de crítica

A padronização dos nomes permite que seja possível a identificação da classificação da regra e a presença do atributo de flexibilidade. Define-se a seguinte proposta de nomenclatura para as regras de crítica, conforme Silva (2020):

1 carácter indicando finalidade + 1 carácter indicando flexibilidade + Sequência numérica

Classificação por finalidade

- Tipo (T): refere-se a classe da variável, são realizadas verificações no sentido de identificar se os dados são numéricos, caracteres, lógicos, entre outros.
- Validade ou Intervalo (V): refere-se aos intervalos estabelecidos matematicamente para um dado ou indicador. Verificações de valores positivos, intervalos entre 0 e 1, são exemplos de verificação de validades.
- Fluxo (F): quando se estabelece que uma variável só existirá dado que outra existe, pode-se estabelecer uma regra de fluxo. Pode ser usada para os casos de questionários com as verificações dos "pulos".
- Consistência (C): refere-se aos casos em que se verificam as relações matemáticas com outras variáveis, por exemplo, parcelas de um total não podem ser maiores do que o próprio total.
- Distribuição (D): regras de distribuição estabelecem parâmetros esperados para as estatísticas descritivas da variável como média, moda, mediana, máximo, mínimo e as medidas de dispersão.

Classificação por flexibilidade

- Flexível (F): construída com parâmetros esperados, mas caso algum caso falhe a regra, precisa ser investigado o motivo, entendendo se a regra deveria ser outra ou se é o caso de se tratar de um valor atípico que futuramente merecerá uma imputação, ou ainda, simplesmente é um valor atípico explicado por alguma situação.
- Inflexível (I): necessariamente precisa ser seguida, uma regra inflexível é rígida e não existe exceção a sua condição estipulada.

Exemplos

TI01: regra de tipo e inflexível

VI08: regra de validade e inflexível

CF02: regra de consistência e flexível

Por definição, regras da classe tipo, validade e fluxo são inflexíveis. As regras da classe consistência podem ser flexíveis e inflexíveis e, por fim, a regra do tipo distribuição é sempre flexível.

10 Leitura dos dados

Disponível em: base_dados

```
### Instalação e Carregamento do Pacote

# Lista de pacotes necessários
pacotes <- c('validate', 'tidyverse', 'gt','readxl')

# Verifica se os pacotes estão instalados e instala se necessário
install.packages(setdiff(x = pacotes,
                         y = rownames(installed.packages())))

# Carrega os pacotes
lapply(X = pacotes,
       FUN = library,
       character.only = TRUE)
```

```
# Carregando dados

dados <- read_excel("dados/dados_validate.xlsx")


dados %>%
  head(5) %>%
  gt()
```

IBGE7	S_TXBRUTAMORT	S_TXBRUTAMORT_t_1	S_TXBRUTAMORTPAD	S_TXMOISQCC
3100104	6.631924	7.494074	5.266178	
3100203	6.901495	6.694450	5.551739	
3100302	6.972738	6.554374	5.409386	
3100401	7.218662	7.363035	5.462740	
3100500	6.133833	5.827141	4.717667	

A função `validator` serve para definir um conjunto de regras de críticas que podem ser

aplicadas a diferentes tipos de conjuntos de dados.

Verificar valores nulos:

- Verifique se não há valores nulos em colunas críticas, como "IBGE7," e outras colunas-chave.

Validar faixas de valores:

- Verifique se os valores em colunas numéricas estão dentro de faixas aceitáveis, por exemplo, garantir que as taxas estejam entre 0 e 100.

Verificar valores ausentes em grupos de colunas relacionadas:

- Verifique se as colunas relacionadas, como as taxas de mortalidade, têm valores ausentes em conjunto.

Verificar valores booleanos:

- Verifique se as colunas com valores lógicos (TRUE/FALSE) estão preenchidas corretamente.

Validar que certas colunas não possuem valores nulos e são numéricas:

- Certifique-se de que determinadas colunas têm valores numéricos válidos e não nulos.

A função `confront` serve para aplicar o conjunto de regras de crítica a uma base de dados específica.

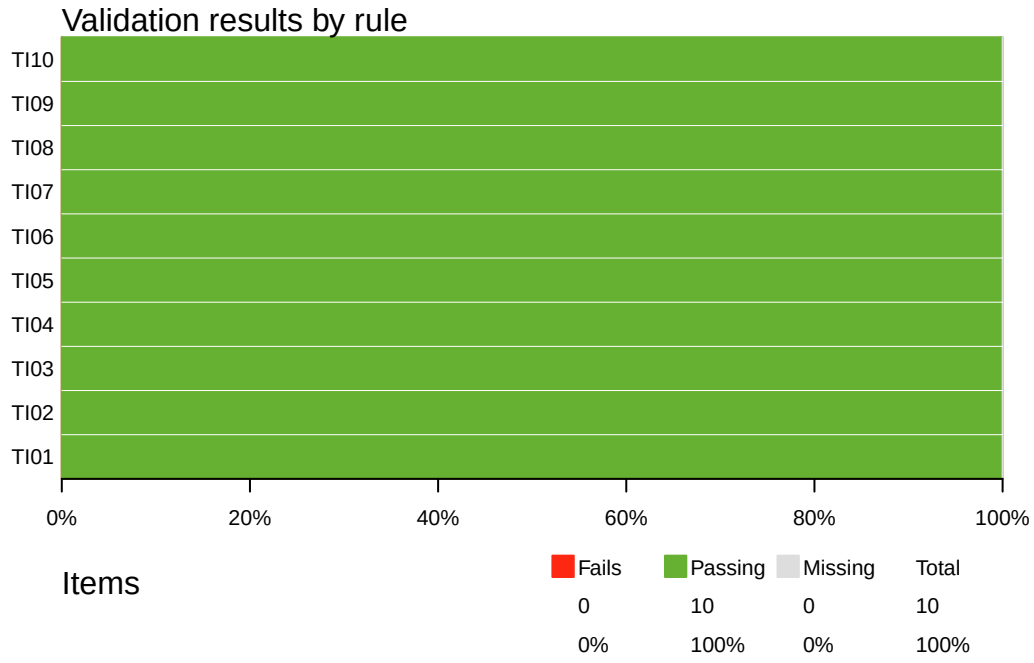
11 Regras de tipo (T)

```
# Crie um conjunto de regras de validade
regras_tipo <- validator(
  #Taxa bruta de mortalidade
  TI01 = is.numeric(S_TXBRUTAMORT),
  #Taxa de mortalidade por homicídio da população total
  TI02 = is.numeric(S_TXMOHOMI),
  #Taxa de mortalidade por homicídio da população de 15 a 29 anos
  TI03 = is.numeric(S_TXMOHOMI15A29),
  #Mortalidade proporcional da população idosa
  TI04 = is.numeric(S_OBITO60),
  #Proporção de nascidos vivos com baixo peso
  TI05 = is.numeric(S_NASCBAIXOPESO),
  #Casos confirmados notificados de sífilis congênita em menores de 1 ano
  TI06 = is.numeric(S_OBINFSIFILS),
  #Casos confirmados notificados de raiva humana
  TI07 = is.numeric(S_OBRAIVA),
  #Proporção de internações por doenças de veiculação hídrica
  TI08 = is.numeric(S_INTERDVHID),
  #Proporção de internações por doenças relacionadas ao saneamento ambiental inadequado
  TI09 = is.numeric(S_INTERDRSAI),
  #Existência de Conselho Municipal de Saúde
  TI10 = is.character(U_CONSSAU)
)

# Aplica regras de crítica aos dados
base_conf_tipo <- confront(dados, regras_tipo)

# Obtém resumo da aplicação das regras aos dados
summary(base_conf_tipo) %>% gt()
```

name	items	passes	fails	nNA	error	warning	expression
TI01	1	1	0	0	FALSE	FALSE	is.numeric(S_TXBRUTAMORT)
TI02	1	1	0	0	FALSE	FALSE	is.numeric(S_TXMOHOMI)

TI03	1	1	0	0	FALSE	FALSE	is.numeric(S_TXMOHOMI15A29)
TI04	1	1	0	0	FALSE	FALSE	is.numeric(S_OBITO60)
TI05	1	1	0	0	FALSE	FALSE	is.numeric(S_NASCBAIXOPESO)
TI06	1	1	0	0	FALSE	FALSE	is.numeric(S_OBINFSIFILS)
TI07	1	1	0	0	FALSE	FALSE	is.numeric(S_OBRAIVA)
TI08	1	1	0	0	FALSE	FALSE	is.numeric(S_INTERDVHID)
TI09	1	1	0	0	FALSE	FALSE	is.numeric(S_INTERDRSAI)
TI10	1	1	0	0	FALSE	FALSE	is.character(U_CONSSAU)

```
# gráfico
plot(base_conf_tipo)
```



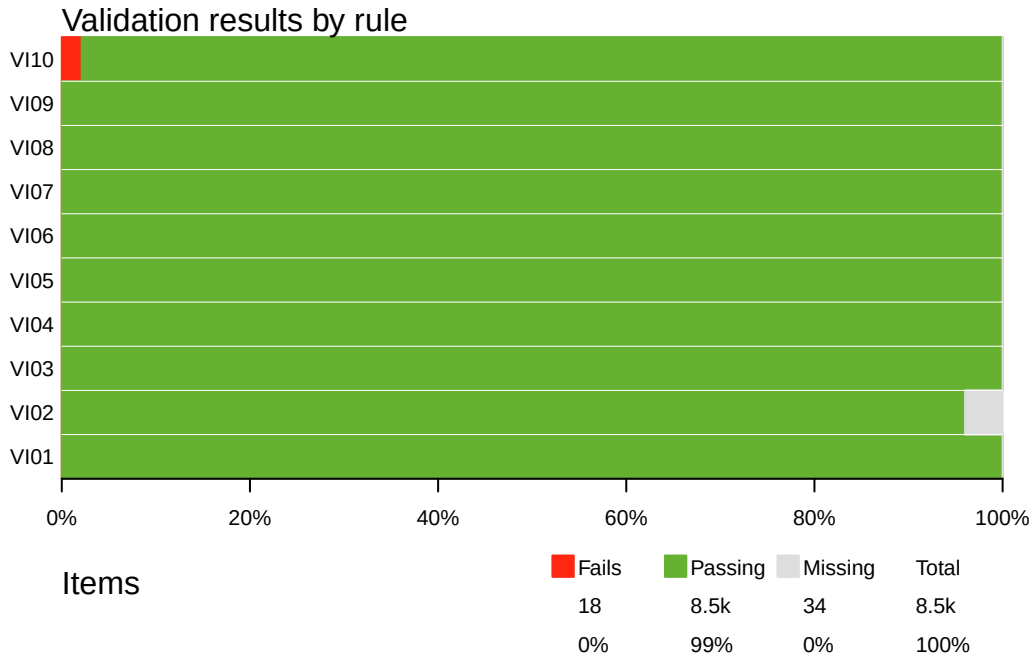
12 Regras de validade (V)

```
regras_validade <- validator(
  #Taxa bruta de mortalidade
  VI01 = (S_TXBRUTAMORT>= 0),
  #Taxa de mortalidade por homicídio da população total
  VI02 = (S_TXMOHOMI >= 0),
  #Taxa de mortalidade por homicídio da população de 15 a 29 anos
  VI03 = (S_TXMOHOMI15A29 >= 0),
  #Mortalidade proporcional da população idosa
  VI04 = (S_OBITO60 >= 0),
  #Proporção de nascidos vivos com baixo peso
  VI05 = (S_NASCBAIXOPESO >= 0 & S_NASCBAIXOPESO <= 100),
  #Casos confirmados notificados de sífilis congênita em menores de 1 ano
  VI06 = (S_OBINFSIFILS >= 0),
  #Casos confirmados notificados de raiva humana
  VI07 = (S_OBRAIVA >= 0),
  #Proporção de internações por doenças de veiculação hídrica
  VI08 = (S_INTERDVHID >= 0 & S_INTERDVHID <= 100),
  #Proporção de internações por doenças relacionadas ao saneamento ambiental inadequado
  VI09 = (S_INTERDRSAI >= 0 & S_INTERDRSAI <= 100),
  #Existência de Conselho Municipal de Saúde
  VI10 = (U_CONSSAU %in% c("Sim","Não"))
)

# Aplica regras de crítica aos dados
base_conf_validade <- confront(dados, regras_validade)

# Obtém resumo da aplicação das regras aos dados
summary(base_conf_validade) %>% gt()
```

name	items	passes	fails	nNA	error	warning	expression
VI01	853	853	0	0	FALSE	FALSE	(S_TXBRUTAMORT - 0 >= -1e-08)
VI02	853	819	0	34	FALSE	FALSE	(S_TXMOHOMI - 0 >= -1e-08)
VI03	853	853	0	0	FALSE	FALSE	(S_TXMOHOMI15A29 - 0 >= -1e-08)

VI04	853	853	0	0	FALSE	FALSE	(S_OBITO60 - 0 >= -1e-08)
VI05	853	853	0	0	FALSE	FALSE	(S_NASCBAIXOPESO - 0 >= -1e-08 & S_NASC
VI06	853	853	0	0	FALSE	FALSE	(S_OBINFSIFILS - 0 >= -1e-08)
VI07	853	853	0	0	FALSE	FALSE	(S_OBRAIVA - 0 >= -1e-08)
VI08	853	853	0	0	FALSE	FALSE	(S_INTERDVHID - 0 >= -1e-08 & S_INTERDVH
VI09	853	853	0	0	FALSE	FALSE	(S_INTERDRSAI - 0 >= -1e-08 & S_INTERDRS
VI10	853	835	18	0	FALSE	FALSE	(U_CONSSAU %vin% c(”Sim”, ”Não”))

```
# gráfico
plot(base_conf_validade)
```

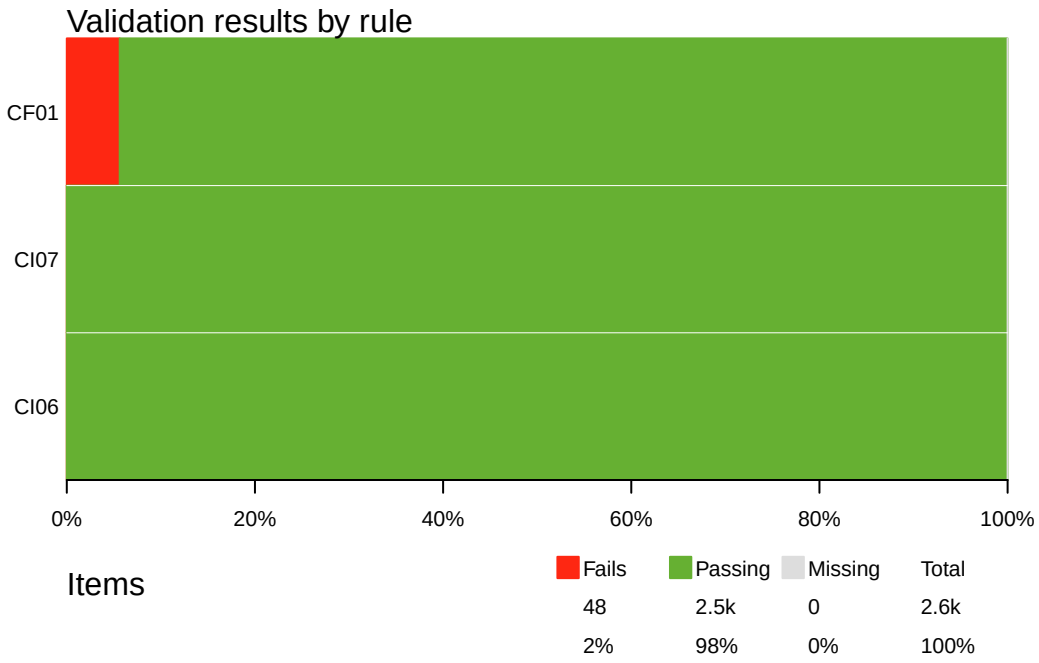


13 Regras de consistência (C)

```
regras_consistencia <- validator(
  #Taxa bruta de mortalidade
  CF01 = (S_TXBRUTAMORT/S_TXBRUTAMORT_t_1) <= 1.10,
  #Casos confirmados notificados de sífilis congênita em menores de 1 ano
  CI06 = (S_OBINFSIFILS <= D_POPP0),
  #Casos confirmados notificados de raiva humana
  CI07 = (S_OBRAIVA <= D_POPT)
)

# Aplica regras de crítica aos dados
base_conf_consistencia <- confront(dados, regras_consistencia)

# Obtém resumo da aplicação das regras aos dados
summary(base_conf_consistencia) %>% gt()
```

name	items	passes	fails	nNA	error	warning	expression
CF01	853	805	48	0	FALSE	FALSE	(S_TXBRUTAMORT/S_TXBRUTAMORT_t_1)
CI06	853	853	0	0	FALSE	FALSE	(S_OBINFSIFILS - D_POPP0 <= 1e-08)
CI07	853	853	0	0	FALSE	FALSE	(S_OBRAIVA - D_POPT <= 1e-08)

```
# gráfico
plot(base_conf_consistencia)
```

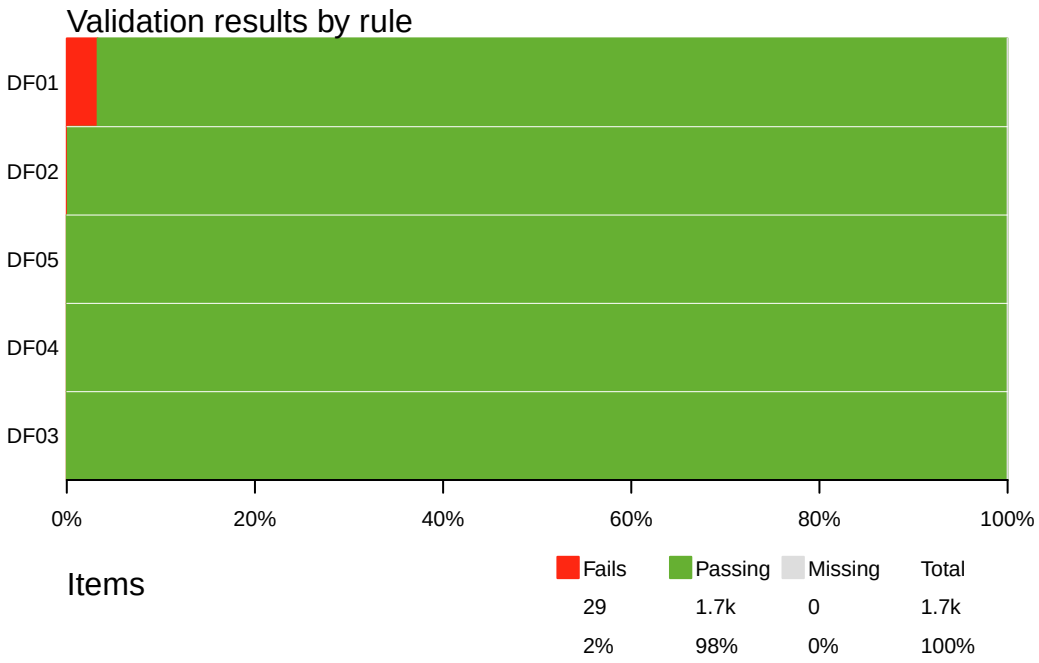

Validation results by rule
CF01
CI07
CI06
0%
20%
40%
60%
80%
100%
Items
Fails
Passing
Missing
Total
48
2.5k
0
2.6k
2%
98%
0%
100%

14 Regras de distribuição (D)

```
regras_distribuicao <- validator(
    #Casos confirmados notificados de sífilis congênita em menores de 1 ano
  DF01 = (S_OBINFSIFILS  < 5),
  #Casos confirmados notificados de raiva humana
  DF02 = (S_OBRAIVA < 5),
  #Proporção de internações por doenças de veiculação hídrica
  DF03 = (mean(S_INTERDVHID) <= 4),
  #Proporção de internações por doenças relacionadas ao saneamento ambiental inadequado
  DF04 = (mean(S_INTERDRSAI) <= 4),
  #Existência de Conselho Municipal de Saúde
  DF05 = (mean(U_CONSSAU=="Sim") >= 0.95)
)
# Aplica regras de crítica aos dados
base_conf_distribuicao <- confront(dados, regras_distribuicao)

# Obtém resumo da aplicação das regras aos dados
summary(base_conf_distribuicao) %>% gt()
```

name	items	passes	fails	nNA	error	warning	expression
DF01	853	825	28	0	FALSE	FALSE	(S_OBINFSIFILS < 5)
DF02	853	852	1	0	FALSE	FALSE	(S_OBRAIVA < 5)
DF03	1	1	0	0	FALSE	FALSE	(mean(S_INTERDVHID) <= 4)
DF04	1	1	0	0	FALSE	FALSE	(mean(S_INTERDRSAI) <= 4)
DF05	1	1	0	0	FALSE	FALSE	(mean(U_CONSSAU == ”Sim”) >= 0.95)

```
# gráfico
plot(base_conf_distribuicao)
```


Validation results by rule
DF01
DF02
DF05
DF04
DF03
0%
20%
40%
60%
80%
100%
Items
Fails
Passing
Missing
Total
29
1.7k
0
1.7k
2%
98%
0%
100%

15 Citação

Gonçalves, C; Marinho,V.. **Crítica e Imputação de Dados no R: Validate**. Tutorial Transformação Digital. Fundação João Pinheiro, n. 2, 2023. Disponível em: https://rpubs.com/fjp/validate.

16 Referências

Silva, P.L.d.N. (2020). Crítica e Imputação de Dados. Notas de aula - Escola Nacional de Ciências Estatísticas.

van der Loo, M. P. J., & de Jonge, E. (2021). Data Validation Infrastructure for R. Journal of Statistical Software, 97(10), 1–31. https://doi.org/10.18637/jss.v097.i10

17 Análise de componentes principais

Compartilhando o código em R

Este tutorial aborda a análise de componentes principais (ACP) em R. Exploraremos os conceitos fundamentais da ACP e demonstraremos como implementá-la em R. O tutorial inclui exemplos práticos e código fonte para ajudar os leitores a compreender e aplicar a ACP em seus próprios projetos de análise de dados.

Contanto:transformacao.digital@fjp.mg.gov.br

A análise de componentes principais (ACP) é uma técnica estatística amplamente utilizada para redução de dimensionalidade e extração de informações relevantes de conjuntos de dados multivariados. O objetivo principal da ACP é transformar um conjunto de variáveis correlacionadas em um novo conjunto de variáveis não correlacionadas, chamadas de componentes principais. Cada componente principal é uma combinação linear das variáveis originais e é ordenado de acordo com a quantidade de variação que ele captura. Isso permite a identificação dos padrões mais significativos dos dados, facilitando a interpretação e visualização. (Manly 1994)

Estes são alguns exemplos de aplicações:

Estudos de qualidade de vida: Através da ACP, é possível identificar dimensões principais que influenciam a qualidade de vida em uma determinada região, permitindo que políticas públicas sejam direcionadas de forma mais eficaz.

Análise de desigualdade social: A ACP pode ser aplicada em indicadores sociais e econômicos para compreender os principais fatores que contribuem para a desigualdade entre grupos populacionais.

Análise de dados de saúde: A ACP é útil para identificar padrões em grandes conjuntos de dados de saúde, como fatores de risco e grupos de doenças.

Estudos de mercado: A ACP é aplicada em pesquisas de mercado para identificar segmentos de clientes com características semelhantes, ajudando as empresas a direcionar suas estratégias de marketing.

Análise de indicadores econômicos: Através da ACP, é possível reduzir um grande número de indicadores econômicos em poucos componentes principais, facilitando a compreensão das principais tendências e correlações na economia.

Previsão econômica: A ACP pode ser usada para reduzir a dimensionalidade de séries temporais econômicas e melhorar a precisão das previsões de indicadores macroeconômicos.

18 Como calcular ACP no R

Antes de iniciar o processo de ACP, é necessário carregar as bibliotecas necessárias.

```
# Lista de pacotes necessários
pacotes <- c("tidyverse", "readxl", "FactoMineR", "factoextra", "gt")

# Verifica se os pacotes estão instalados e instala se necessário
install.packages(setdiff(x = pacotes,
                         y = rownames(installed.packages())))

# Carrega os pacotes
lapply(X = pacotes,
       FUN = library,
       character.only = TRUE)
```

18.0.0.1 Importação dos dados:

Os dados utilizados para a análise de componentes principais são apenas um exemplo, a base comtém informações sobre o PIB, Número de óbitos, Despesas com saúde e número de beneficiários do bolsa família. Para calcular ACP os dados devem conter as variáveis numéricas.

Disponível em: base_dados

```
# Importar dados
df_acp <-read_excel("dados/dados_acp.xlsx")
```

18.0.0.2 Visualização da matriz de dados

Podemos usar o comando `gt()` para visualizar uma tabela estatistica do conjunto de dados. Isso ajuda a garantir que os dados foram carregados corretamente antes de prosseguir com a análise.

```r
df_acp %>%
  head(5) %>%
  gt()
```

ano	sigla_uf	id_municipio	numero_obitos	desp_saude	populacao_atendida_agua	populacao_ate
2019	MG	3100104	48	5088031	3997	
2019	MG	3100203	169	15933256	19540	
2019	MG	3100302	86	8937634	9356	
2019	MG	3100401	36	3325094	3982	
2019	MG	3100500	76	4597114	3263	

18.0.0.3 Limpeza e seleção de variáveis

Antes de proceder com ACP vamos selecionar algumas variaveis

```r
dados<- df_acp %>% select(pib, numero_obitos, desp_saude, pessoas_pbf)
```

18.0.1 Função de Redução de dimensionalidade

ACP é realizada usando o pacote `FactoMineR`. Nesse exemplo, o cálculo é feito com comando o `PCA`, onde "dados" é o dataframe que contém as variáveis a serem analisadas:

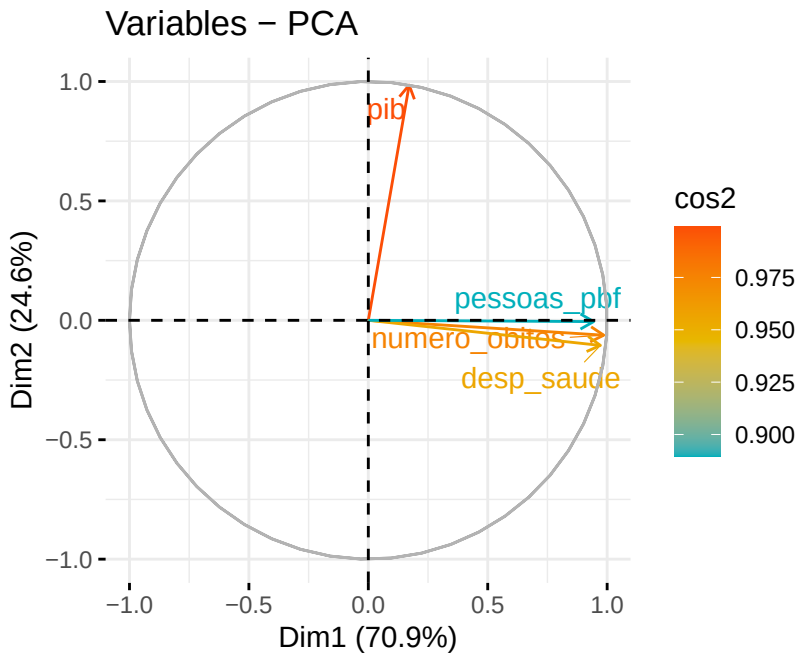
```r
acp <- PCA(dados, graph=F)
```

18.1 Visualização e exploração dos resultados:

Para visualizar a qualidade das variáveis no mapa de fatores, usa-se a função `fviz_pca_var()`

O gráfico que exibe a qualidade ($\cos^2$) de cada variável em relação aos componentes principais é uma representação visual da contribuição de cada variável para a formação desses componentes. O "$\cos^2$" é a proporção da variância da variável original que é explicada pelo componente principal específico.

```r
fviz_pca_var(acp, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping
             )
```



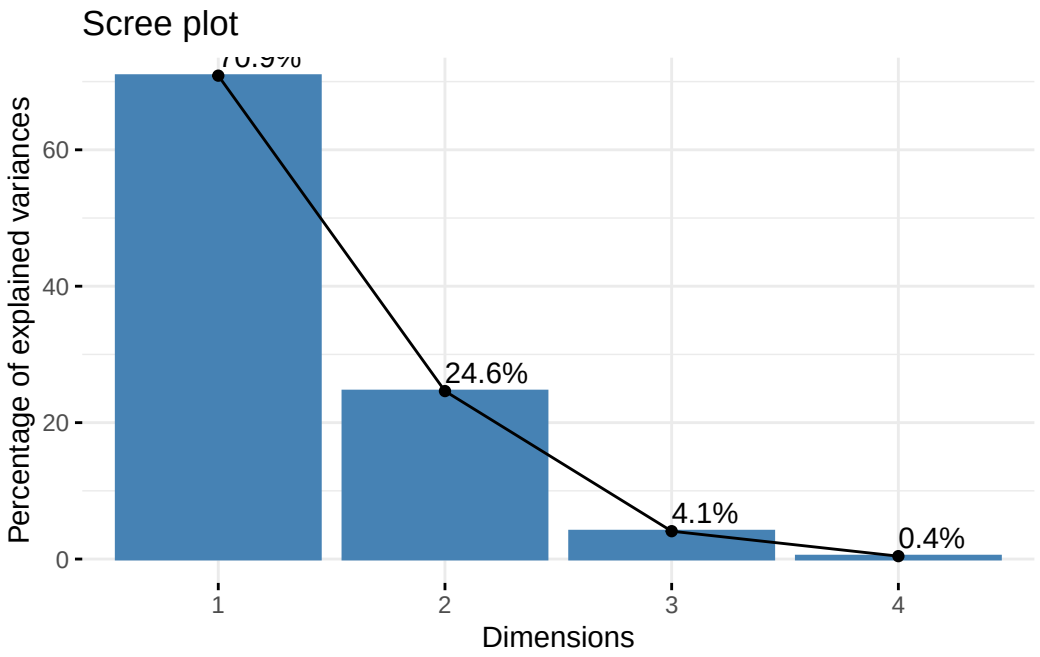
Quando realizamos a Análise de Componentes Principais (ACP), estamos buscando encontrar novas variáveis (os componentes principais) que sejam combinações lineares das variáveis originais, de modo que eles capturem a maior quantidade possível de variação dos dados. Cada componente principal é uma combinação ponderada das variáveis originais, e a quantidade de variação explicada por cada componente é medida pelos autovalores associados a eles.

O gráfico de qualidade das variáveis em relação aos componentes principais ajuda a identificar quais variáveis têm uma forte influência na definição de cada componente principal e quais têm uma influência mais fraca. Essa informação é útil para entender quais variáveis são mais importantes para explicar a estrutura dos dados e quais têm menos impacto.

18.1.1 Visualização da comunaliade explicada por cada componente:

A função `fviz_eig` é utilizada para exibir a inércia explicada por cada componente principal e o pacote `plotly` permite deixar o gráfico interativo. Ela mostra o quanto de variação dos dados é explicada por cada componente:

```
fviz_eig(acp, addlabels=TRUE, ylim = c(0,70))
```



18.1.2 Sumarização dos resultados:

A função `facto_summarize` é usada para resumir as informações sobre as variáveis e os componentes principais obtidos a partir da ACP. No exemplo, a sumarização é feita para os dois primeiros componentes principais (axes = 1:2):

```
facto_summarize(acp, "var", axes = 1:2) %>% gt()
```

name	Dim.1	Dim.2	coord	cos2	contrib
pib	0.1710860	0.984893502	0.9992856	0.9992856	26.16030
numero_obitos	0.9857684	-0.062278615	0.9756180	0.9756180	25.54070
desp_saude	0.9717458	-0.105023373	0.9553198	0.9553198	25.00931
pessoas_pbf	0.9431881	-0.005357553	0.8896325	0.8896325	23.28969

Analisando as coordenadas das variáveis nos dois componentes principais:

A variável despesas com saude tem uma forte relação positiva com o primeiro componente principal (Dim.1), mas uma relação negativa muito pequena com o segundo componente principal (Dim.2). Isso sugere que essa variável tem um papel dominante na definição do primeiro componente principal, enquanto sua influência no segundo componente é praticamente insignificante.

A variável populacao_atendida_agua também possui uma relação muito forte e positiva com o primeiro componente principal (Dim.1), mas uma relação negativa muito pequena com o segundo componente principal (Dim.2). Isso indica que essa variável também é um importante contribuinte para a definição do primeiro componente principal.

Assim como as duas variáveis anteriores, a "populacao_atentida_esgoto" tem uma relação positiva forte com o primeiro componente principal (Dim.1) e uma relação negativa pequena com o segundo componente principal (Dim.2). Isso sugere que essa variável é relevante para o primeiro componente principal.

A variável pessoas_beneficiarias_pbf tem uma relação positiva relativamente alta com o primeiro componente principal (Dim.1) e uma relação positiva menor com o segundo componente principal (Dim.2). Isso indica que essa variável contribui significativamente para ambos os componentes principais, mas tem uma importância maior no primeiro.

Por fim, o PIB tem uma relação positiva muito alta com o segundo componente principal (Dim.2) e uma relação positiva menor com o primeiro componente principal (Dim.1). Isso sugere que essa variável é essencialmente representada pelo segundo componente principal e tem uma importância menor no primeiro.

19 Recursos Adicionais

http://www.sthda.com/english/wiki/wiki.php?id_contents=7851

19.0.1 Referências

References

Hoffmann, Rodolfo. 2006. *Estatística Para Economistas*. 1ª edição. Cengage Learning.

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Manly, Bryan F. J. 1994. *Multivariate Statistical Methods: A Primer*. 2nd ed. Boca Raton: Chapman & Hall/CRC.

Petersen, Anne Helby, and Claus Thorn Ekstrøm. 2019. "**dataMaid**: Your Assistant for Documenting Supervised Data Quality Screening in *R*." *Journal of Statistical Software* 90 (6). https://doi.org/10.18637/jss.v090.i06.