

Dec 04, 2023

Protocol for Epidemiological Analysis and Data Visualization in Health

DOI

dx.doi.org/10.17504/protocols.io.j8nlko61dv5r/v1



Ronaldo Rodrigues de Oliveira Junior¹, Leandro do Prado Assunção², Lindomar José Pena³, Valéria Christina de Rezende Feres⁴

¹Laboratory of Molecular Biology and Applied Technologies for Laboratory Diagnosis, Faculty of Pharmacy, Federal University of Goiás (UFG), Goiânia, GO, Brazil;

²Laboratory of Molecular Patology, Institute of Biological Science, Federal University of Goiás (UFG), Goiânia, GO, Brazil;

³Virology and Experimental Therapy Laboratory, Aggeu Magalhães Institute, Oswaldo Cruz Foundation (Fiocruz), Pernambuco, PE, Brazil;

⁴Laboratory of Molecular Biology and Applied Technologies for Laboratory Diagnosis, Faculty of Pharmacy, Federal University of Goiás (UFG), Goiânia, GO, Brazil.



Ronaldo Rodrigues de Oliveira Junior

UFG

Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN  ACCESS



DOI: <https://dx.doi.org/10.17504/protocols.io.j8nlko61dv5r/v1>

Protocol Citation: Ronaldo Rodrigues de Oliveira Junior, Leandro do Prado Assunção, Lindomar José Pena, Valéria Christina de Rezende Feres 2023. Protocol for Epidemiological Analysis and Data Visualization in Health. **protocols.io**

<https://dx.doi.org/10.17504/protocols.io.j8nlko61dv5r/v1>

License: This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Protocol status: Working

We use this protocol and it's working

Created: November 13, 2023

Last Modified: December 04, 2023

Protocol Integer ID: 90849

Keywords: Data bank, Epidemiology, Data science, Health Information Systems, R programming language, data visualization in health, large health database, data visualization, monitoring population health, epidemiological analysis, survival data, data, population health, epidemiological study, graph, mortality, conducting epidemiological study, database, creation of graph, data processing, database of the sistema, figures from other database, python, big data, visualization, health, frequencies of comorbidity, other database, statistical analysis, informação, original data

Funders Acknowledgements:

Coordination for the Improvement of Higher Education Personnel, Capes

Grant ID: 0001

Abstract

In recent years, information contained in large health databases (big data) has been used in various scientific fields. These databases have been generated from original data that has expanded to encompass not only mortality and survival data but also other dimensions such as healthcare and infection control, becoming a valuable source for conducting epidemiological studies, effectiveness analyses of treatments, and monitoring population health, among other applications. For this purpose, the programming languages R and Python are most commonly used for data processing, statistical analysis, and the creation of graphs and tables. In this context, a protocol was developed using the database of the Sistema de Informação de Agravos de Notificação (SINAN) through the R programming language to create graphs showing the frequencies of comorbidities, symptoms, and a map of the distribution of deaths in the state of Goiás, Brazil, from 2015 to 2022. This script can be used to process and generate figures from other databases.

Troubleshooting

Before start

The RStudio program must be installed on the computer. If not, the link below provides the tutorial for download.

<https://posit.co/download/rstudio-desktop/>



Installing the packages

- 1 Install the packages that will be used throughout the analyses using the `install.packages('package name')` function. Then, press Ctrl + Enter. The code is as follows:

Command

new command name

```
install.packages("magrittr") # Set of operators to facilitate writing
more readable code, especially when using the pipe operator (%>%).
install.packages("writexl") # Write Excel spreadsheets.
install.packages("foreign") # Provides functions to import/export
data to/from other formats, such as SPSS, SAS, and Stata.
install.packages("readxl") # Facilitates reading Excel data.
install.packages("readr") # Offers efficient functions for reading
"tidy" rectangular data in R.
install.packages("janitor") # Provides functions for data cleaning,
such as removing duplicate column names.
install.packages("summarytools") # Helps create statistical summaries
and overviews of datasets.
install.packages("stringr") # Offers functions for string
manipulation.
install.packages("skimr") # Generates descriptive statistics for each
variable in a dataset.
install.packages("lubridate") # Facilitates manipulation of dates and
times.
install.packages("descr") # Offers functions for descriptive
statistics.
install.packages("stringi") # Provides functions for string
manipulation with Unicode support.
install.packages("tidyverse") # A collection of packages (including
dplyr and ggplot2) for manipulating and visualizing data in a "tidy"
way.
install.packages("dplyr") # Offers functions for efficient data
manipulation.
install.packages("DescTools") # Provides various statistical and data
description tools.
install.packages("epitools") # Tools for epidemiology and public
health statistics.
install.packages("ggplot2") # Package for creating graphics and
visualizations.
install.packages("knitr") # Facilitates integrating R with dynamic
documents in formats like HTML, PDF, and Word.
install.packages("stats") # Contains basic statistical functions.
install.packages("plotly") # Allows creating interactive
visualizations.
install.packages("reclin") # Provides functions for analysis and
visualization of consumer complaint data.
```

```
install.packages("digest") # Generates data hashes to check integrity.
install.packages("DT") # Facilitates creating interactive tables in R.
install.packages("tidyr") # Facilitates data manipulation into "tidy"
formats.
install.packages("openxlsx") # Provides functions for reading and
writing Excel files.
install.packages("gt") # Offers a way to create beautiful and
formatted tables.
install.packages("sf") # Provides classes and methods for spatial
data.
install.packages("mapsf") # Facilitates creating maps with ggplot2
for spatial data.
install.packages("tmap") # Facilitates creating thematic maps.
install.packages("ggspatial") # Adds spatial functionality to ggplot2.
install.packages("geobr") # Provides geographical data for Brazil.
install.packages("ggmap") # Allows integrating Google Maps with
ggplot2.
install.packages("maps") # Provides maps for visualizations.
install.packages("networkD3") # Facilitates creating interactive
network visualizations.
```

Loading the packages

2 Load the packages.

Note: Every time you open the software, you should load the packages as indicated below.



Command

new command name

```
library(magrittr)
library(writexl)
library(foreign)
library(readxl)
library(readr)
library(janitor)
library(summarytools)
library(stringr)
library(skimr)
library(lubridate)
library(descr)
library(stringi)
library(tidyverse)
library(dplyr)
library(DescTools)
library(epitools)
library(ggplot2)
library(knitr)
library(stats)
library(plotly)
library(reclin)
library(digest)
library(DT)
library(tidyr)
library(openxlsx)
library(gt)
library(sf)
library(mapsf)
library(tmap)
library(ggspatial)
library(geobr)
library(ggmap)
library(maps)
library(networkD3)
```

Importing the database

- 3 Import the database into the R environment using the following function:

Command

new command name

```
sinan_geral <- read.dbf(file = 'Banco_Sinan.dbf', as.is = TRUE)
```

- i. The 'read.dbf' function is used to read data from a DBF (DataBase File) file.
- ii. The argument 'file = 'Banco_Sinan.dbf'' indicates the name of the DBF file to be imported.
- iii. The argument 'as.is = TRUE' is used to keep strings (character sets) as they are in the database.

Selecting the variables for the study

- 4 Filter the variables of interest using the 'select' function. Name the database as 'sinan' only.

Command

new command name

```
sinan <- select(sinan_geral, NM_PACIENT, ID_MN_RESI, DT_NASC,
NU_NOTIFIC, ID_AGRAVO, NU_IDADE_N, CLASSI_FIN, EVOLUCAO, CRITERIO,
CS_SEXO, CS_RACA, DT_NOTIFIC, DT_SIN_PRI, DT_CHIK_S1,
DT_CHIK_S2, RES_CHIKS1, RES_CHIKS2, DT_PCR, RESUL_PCR_, CLINC_CHIK,
DT_INTERNA, HOSPITALIZ, DT_OBITO, DT_ENCERRA, DIABETES, HEMATOLOG,
HEPATOPAT, RENAL, HIPERTENSA, ACIDO_PEPT, AUTO_IMUNE, CS_ESCOL_N,
FEBRE, MIALGIA, CEFALeia, EXANTEMA, VOMITO, NAUSEA, DOR_COSTAS,
CONJUNTIVIT , ARTRITE, ARTRALGIA, LEUCOPENIA, DOR_RETRO, ID_MUNICIP,
DS_OBS)
```

Cleaning and standardizing the variables

- 5 Adjust the database so that all records contained in it are written in the same way, ensuring that analyses are not compromised by typing differences.

Command

new command name

```
# Inserting the `colnames()` function to visualize the variable names
colnames(sinan)

# Using the `clean_names()` function to edit the variable names
## This function makes all variable names lowercase, without accents,
and without spaces between words.

sinan <- clean_names(sinan)

# Viewing the variables after the transformation
colnames(sinan)
```

Selecting the records

6

Select records that meet all of the following criteria:

- i. Epidemiological or laboratory diagnostic criterion (1 or 2)
- ii. The final classification was chikungunya (13)
- iii. Evolved to death (2)

Note: Information within parentheses defines how the variables were filled.

Command

new command name

Filtering all records that meet the above-mentioned criteria:

```
sinan <- sinan %>%  
  filter(criterio %in% c(1, 2)) %>%  
  filter(classi_fin %in% c(13)) %>%  
  filter(evolucao %in% c(2))
```

Renaming the variables

- 7 Rename the variables using the 'rename' function from the dplyr package.

Command

new command name

Renaming variables using the 'rename' function from the dplyr package

```
sinan <- sinan %>%
  rename("numero notificacao" = nu_notific,
        agravo = id_agravo,
        "classificacao final" = classi_fin,
        sexo = cs_sexo,
        raca = cs_raca,
        "data notificacao" = dt_notific,
        "data primeiros sintomas" = dt_sin_pri,
        "data internacao" = dt_interna,
        hospitalizacao = hospitaliz,
        "data obito" = dt_obito,
        "data encerramento" = dt_encerra,
        "doenca hematologica" = hematolog,
        hepatopatias = hepatopat,
        "doenca renal cronica" = renal,
        hipertensao = hipertensa,
        "doenca acido peptica" = acido_pept,
        "doenca autoimune" = auto_imune,
        escolaridade = cs_escol_n,
        "dor nas costas" = dor_costas,
        conjuntivite = conjuntvit,
        "dor retrorbital" = dor_retro,
        idade = idade_anos,
        "ano do obito" = ano_obito,
        "ano da notificacao" = ano_notificacao,
        nome = nm_pacient,
        "codigo municipio residencia" = id_mn_resi,
        "data nascimento" = dt_nasc,
        "data coleta 1 soro" = dt_chik_s1,
        "data coleta 2 soro" = dt_chik_s2,
        "resultado coleta 1 soro" = res_chiks1,
        "resultado coleta 2 soro" = res_chiks2,
        "data PCR" = dt_pcr,
        "resultado PCR" = resul_pcr,
        "apresentacao clinica" = clinc_chik,
        "codigo municipio notificacao" = id_municip,
        observacao = ds_obs.
```

```
... ..
"nome municipio residencia" = nome_municipio_residencia,
"nome municipio notificacao" = nome_municipio_notificacao,
"dias entre inicio sintomas e obito" = dif_obt_e_inic_sint)
```

Handling the 'age' variable

- 8 The filling of the variable 'age' is different from the others. This process follows the following rule applied to the first digit:

1. Hour
2. Day
3. Month
4. Year

For example: 4056 → 56 years old; 3009 → 9 months old; 2024 → 24 days old

Command

Adding a new column for the 'age' variable and categorizing it into age groups

Adding a new column for the 'age' variable and categorizing it into age groups

Treating the 'idade' variable

Add a new formatted age column

```
sinan <- sinan %>%  
  mutate(idade_anos = ifelse(str_sub(nu_idade_n, 1, 1) == "4",  
    as.character(as.numeric(str_sub(nu_idade_n, 2, 4))),  
    ifelse(str_sub(nu_idade_n, 1, 1) == "3",  
    as.character(as.numeric(str_sub(nu_idade_n, 2, 4))/12),  
    ifelse(str_sub(nu_idade_n, 1, 1)  
    == "2", as.character(as.numeric(str_sub(nu_idade_n, 2, 4))/365),  
    "0"))))
```

Categorizing into age groups

Convert the 'idade_anos' column to numeric

```
sinan$idade_anos <- as.numeric(sinan$idade_anos)
```

Use the cut() function to stratify the IDADE_ANOS variable

```
sinan$`faixa etaria` <- cut(sinan$idade_anos,  
  breaks = c(0, 9, 19, 29, 39, 49, 59, 69,  
  79, 89, 99),  
  labels = c("0-9 anos", "10-19 anos", "20-  
  29 anos", "30-39 anos", "40-49 anos", "50-59 anos", "60-69 anos", "70-  
  79 anos", "80-89 anos", "90-100 anos"),  
  right = FALSE)
```

Creating new variables

9 Create some variables as per the code below:

Command

new command name

1. Creating the 'ano_obito' variable to contain only the year of death

```
sinan$ano_obito <- substr(sinan$dt_obito , 1, 4)
```

2. Creating the 'ano_notificacao' variable to contain only the year of notification

```
sinan$ano_notificacao <- substr(sinan$dt_notific, 1, 4)
```

#3. Creating the 'nome do município de residência'

```
sinan$nome_municipio_residencia <- sinan$id_mn_resi
```

4. Creating the 'nome_municipio_notificacao' variable

```
sinan$nome_municipio_notificacao <- sinan$id_municip
```

5. Creating the 'diferenca_dia_sintomas_obito' variable

```
sinan <- sinan %>%  
  mutate(dif_obt_e_inic_sint = as.numeric(as.Date(dt_obito,  
"%Y/%m/%d") - as.Date(dt_sin_pri, "%Y/%m/%d")))
```

6. Creating the 'Evolução categorizada' variable

```
sinan$`Evolução categorizada` <- cut(sinan$dif_obt_e_inic_sint,  
                                   breaks = c(-Inf, 10, Inf),  
                                   labels = c("<= 10 dias", "> 10  
dias"),  
                                   right = FALSE)
```

6.1. Converting the 'Evolução categorizada' variable to numeric

```
sinan$`Evolução categorizada` <- as.numeric(sinan$`Evolução  
categorizada`)
```



Renaming the variable names

- 10 Rename and standardize the writing of variables.

Command

new command name

```
# Rename variables using the rename function from the Dplyr package
sinan <- sinan %>%
  rename("numero notificacao" = nu_notific,
        agravo = id_agravo,
        "classificacao final" = classi_fin,
        sexo = cs_sexo,
        raca = cs_raca,
        "data notificacao" = dt_notific,
        "data primeiros sintomas" = dt_sin_pri,
        "data internacao" = dt_interna,
        hospitalizacao = hospitaliz,
        "data obito" = dt_obito,
        "data encerramento" = dt_encerra,
        "doenca hematologica" = hematolog,
        hepatopatias = hepatopat,
        "doenca renal cronica" = renal,
        hipertensao = hipertensa,
        "doenca acido peptica" = acido_pept,
        "doenca autoimune" = auto_imune,
        escolaridade = cs_escol_n,
        "dor nas costas" = dor_costas,
        conjuntivite = conjuntvit,
        "dor retrorbital" = dor_retro,
        idade = idade_anos,
        "ano do obito" = ano_obito,
        "ano da notificacao" = ano_notificacao,
        nome = nm_pacient,
        "codigo municipio residencia" = id_mn_resi,
        "data nascimento" = dt_nasc,
        "data coleta 1 soro" = dt_chik_s1,
        "data coleta 2 soro" = dt_chik_s2,
        "resultado coleta 1 soro" = res_chiks1,
        "resultado coleta 2 soro" = res_chiks2,
        "data PCR" = dt_pcr,
        "resultado PCR" = resul_pcr,
        "apresentacao clinica" = clinc_chik,
        "codigo municipio notificacao" = id_municip,
        observacao = ds_obs,
        "nome municipio residencia" = nome_municipio_residencia,
        "nome municipio notificacao" = nome_municipio_notificacao.
```



```
"dias entre inicio sintomas e obito" = dif_obt_e_inic_sint)
```

Changing the label of the variable

- 11 Change the labels of the variables and convert them to 'factor'.

Command

new command name

```
sinan$agravo = factor(sinan$agravo,
                      label=c("chikungunya"),
                      levels = c("A92.0"))

sinan$`classificacao final` = factor(sinan$`classificacao final`,
                                     label=c("chikungunya"),
                                     levels = c("13"))

sinan$criterio = factor(sinan$criterio,
                        label=c("laboratorial", "clinico"),
                        levels = c("1", "2"))

sinan$sexo = factor(sinan$sexo,
                    label=c("feminino", "masculino"),
                    levels = c("F", "M"))

sinan$evolucao = factor(sinan$evolucao,
                        label=c("óbito pelo agravo"),
                        levels = c("2"))

sinan$raca = factor(sinan$raca,
                    label=c("branca", "preta", "amarela", "parda",
                             "indígena", "ignorado"),
                    levels = c("1", "2", "3", "4", "5", "9"))

sinan$diabetes = factor(sinan$diabetes,
                        label=c("sim", "nao"),
                        levels = c("1", "2"))

sinan$hepatopatias = factor(sinan$hepatopatias,
                             label=c("sim", "nao"),
                             levels = c("1", "2"))
```

```
sinan$`doenca hematologica` = factor(sinan$`doenca hematologica`,
                                     label=c("sim", "nao"),
                                     levels = c("1", "2"))

sinan$`doenca renal cronica` = factor(sinan$`doenca renal cronica`,
                                     label=c("sim", "nao"),
                                     levels = c("1", "2"))

sinan$hipertensao = factor(sinan$hipertensao,
                           label=c("sim", "nao"),
                           levels = c("1", "2"))

sinan$`doenca acido peptica` = factor(sinan$`doenca acido peptica`,
                                     label=c("sim", "nao"),
                                     levels = c("1", "2"))

sinan$`doenca autoimune` = factor(sinan$`doenca autoimune`,
                                  label=c("sim", "nao"),
                                  levels = c("1", "2"))

sinan$escolaridade = factor(sinan$escolaridade,
                             label=c("Iletrado", "1-3 anos", "4 anos",
                                     "5-8 anos", "9 anos", "10-12 anos", "13 anos", "13-17 anos", "18",
                                     "Ignorado", "Não se aplica"),
                             levels = c("0", "1", "2", "3", "4", "5",
                                     "6", "7", "8", "9", "10"))

sinan$febre= factor(sinan$febre,
                    label=c("sim", "nao"),
                    levels = c("1", "2"))

sinan$mialgia= factor(sinan$mialgia,
                      label=c("sim", "nao"),
                      levels = c("1", "2"))

sinan$cefaleia= factor(sinan$cefaleia,
                       label=c("sim", "nao"),
                       levels = c("1", "2"))

sinan$exantema= factor(sinan$exantema,
                       label=c("sim", "nao"),
```



```
levels = c("1", "2"))
```

Removing variables that will no longer be used

- 12 Remove the variables that were used only to create others.

For example: the variable 'nu_idade,' which represents the age of patients but is encoded, will no longer be necessary since it has been decoded to create another variable for this purpose.

Command

new command name

Select only the columns of interest for the next steps and organize them better

```
sinan <- select(sinan,
  "numero notificacao",
  "nome",
  "data nascimento",
  "escolaridade",
  "idade",
  "faixa etaria",
  "sexo",
  "raca",
  "nome municipio residencia",
  "codigo municipio residencia",
  "codigo municipio notificacao",
  "nome municipio notificacao",
  "data notificacao",
  "ano da notificacao",
  "diabetes",
  "hipertensao",
  "doenca renal cronica",
  "hepatopatias",
  "doenca hematologica",
  "doenca acido peptica",
  "doenca autoimune",
  "data primeiros sintomas",
  "febre",
  "mialgia",
  "cefaleia",
  "exantema",
  "vomito",
  "nausea",
  "dor nas costas",
  "conjuntivite",
  "artrite",
  "artralgia",
  "leucopenia",
  "dor retrorbital",
  "data coleta 1 soro".
```



```
        "resultado coleta 1 soro",
        "data coleta 2 soro",
        "resultado coleta 2 soro",
        "data PCR",
        "resultado PCR",
        "agravo",
        "classificacao final",
        "criterio",
        "apresentacao clinica",
        "hospitalizacao",
        "data internacao",
        "evolucao",
        "data obito",
        "dias entre inicio sintomas e obito",
        "ano do obito",
        "data encerramento",
        "observacao",
        "Evolução categorizada")
```

Changing the format of variables representing dates

13 Standardize the date format. The code is as follows:

Command

new command name

```
# Format the date of birth variable
```

```
sinan <- sinan %>%  
  mutate(`data nascimento` = as.Date(`data nascimento`)) %>%  
  mutate(`data nascimento` = format(`data nascimento`, "%d-%m-%Y"))
```

```
# Format the notification date variable
```

```
sinan <- sinan %>%  
  mutate(`data notificacao` = as.Date(`data notificacao`)) %>%  
  mutate(`data notificacao` = format(`data notificacao`, "%d-%m-%Y"))
```

```
# Format the date of first symptoms variable
```

```
sinan <- sinan %>%  
  mutate(`data primeiros sintomas` = as.Date(`data primeiros  
sintomas`)) %>%  
  mutate(`data primeiros sintomas` = format(`data primeiros  
sintomas`, "%d-%m-%Y"))
```

```
# Format the variable date of first serum collection
```

```
sinan <- sinan %>%  
  mutate(`data coleta 1 soro` = as.Date(`data coleta 1 soro`)) %>%  
  mutate(`data coleta 1 soro` = format(`data coleta 1 soro`, "%d-%m-  
%Y"))
```

```
# Format the variable collection date second serum collection
```

```
sinan <- sinan %>%  
  mutate(`data coleta 2 soro` = as.Date(`data coleta 2 soro`)) %>%  
  mutate(`data coleta 2 soro` = format(`data coleta 2 soro`, "%d-%m-  
%Y"))
```

```
# Format the PCR date variable
```

```
sinan <- sinan %>%  
  mutate(`data PCR` = as.Date(`data PCR`)) %>%  
  mutate(`data PCR` = format(`data PCR`, "%d-%m-%Y"))
```

```
# Format the hospitalization date variable
```



```
sinan <- sinan %>%
  mutate(`data internacao` = as.Date(`data internacao`)) %>%
  mutate(`data internacao` = format(`data internacao`, "%d-%m-%Y"))

# Format the date of death variable
sinan <- sinan %>%
  mutate(`data obito` = as.Date(`data obito`)) %>%
  mutate(`data obito` = format(`data obito`, "%d-%m-%Y"))

# Format the case closing date variable
sinan <- sinan %>%
  mutate(`data encerramento` = as.Date(`data encerramento`)) %>%
  mutate(`data encerramento` = format(`data encerramento`, "%d-%m-%Y"))
```

Exporting the processed database

14 Export the database (optional)

Command

new command name

```
# Export spreadsheet for preliminary analysis
```

```
# Write a dataframe to a CSV file
```

```
write.csv(sinan, "sinan_tratado_2023.09.05.csv", row.names = FALSE)
```

```
# Write a dataframe to a text file using readr
```

```
write_delim(sinan, "sinan_tratado_2023.09.05.txt", delim = "\t")
```

```
# Write the dataframe to an XLSX file
```

```
write_xlsx(sinan, "sinan_tratado_2023.09.14.xlsx")
```

Creating the relative frequency bar chart of comorbidities

15 The use of charts is one of the most commonly employed visual methods for data visualization. Therefore, bar charts will be created to demonstrate the frequencies of comorbidities in relation to deaths.

15.1 i. Conversion of variables to numeric

Command**new command name**

```
# i. Convert categorical variables to numeric values (0 for "não" and  
1 for "sim")
```

```
sinan <- sinan %>%  
  mutate(  
    diabetes = ifelse(diabetes == "sim", 1, 0),  
    hipertensao = ifelse(hipertensao == "sim", 1, 0),  
    `doenca renal cronica` = ifelse(`doenca renal cronica` == "sim",  
1, 0),  
    hepatopatias = ifelse(hepatopatias == "sim", 1, 0),  
    `doenca hematologica` = ifelse(`doenca hematologica` == "sim", 1,  
0),  
    `doenca acido peptica` = ifelse(`doenca acido peptica` == "sim",  
1, 0),  
    `doenca autoimune` = ifelse(`doenca autoimune` == "sim", 1, 0)  
  )
```

15.2 ii. Calculate the percentage of patients for each comorbidity

Command**new command name**

```
# ii. Calculate the percentage of patients for each comorbidity

comorbidades_porcentagem <- sinan %>%
  summarise(
    diabetes = mean(diabetes) * 100,
    hipertensao = mean(hipertensao) * 100,
    `doenca renal cronica` = mean(`doenca renal cronica`) * 100,
    hepatopatias = mean(hepatopatias) * 100,
    `doenca hematologica` = mean(`doenca hematologica`) * 100,
    `doenca acido peptica` = mean(`doenca acido peptica`) * 100,
    `doenca autoimune` = mean(`doenca autoimune`) * 100
  )
```

15.3 iii. Transform the dataset from wide to long format**Command****new command name**

```
#iii. Transform dataset from wide format to long (tidy)
comorbidades_porcentagem_longo <- comorbidades_porcentagem %>%
  pivot_longer(cols = everything(), names_to = "Comorbidade",
    values_to = "Porcentagem")
```

15.4 iv. Create the chart

Command

new command name

i.v. Creating the chart

```
ggplot(data = comorbidades_porcentagem_longo, aes(x =  
reorder(Comorbidade, -Porcentagem), y = Porcentagem, fill =  
Porcentagem)) +  
  geom_bar(stat = "identity", width = 0.3) + # Barras preenchidas  
  geom_bar(stat = "identity", width = 0.3, color = "black", position  
= "identity", fill = NA, size = 0.8) + # Contorno das barras  
  labs(title = "Porcentagem de Pacientes em Relação às Comorbidades",  
        x = "Comorbidades",  
        y = "Porcentagem (%)",  
        caption = "Fonte: SINAN/SUVISA") +  
  scale_fill_gradient(low = "lightblue3", high = "darkblue") + #  
  Definir cores do degradê  
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +  
  theme(panel.background = element_blank(),  
        panel.grid.major = element_blank(),  
        panel.grid.minor = element_blank(),  
        panel.border = element_blank(), # Remover borda do painel  
        axis.line = element_line(color = "black", size = 0.3)) + #  
  Linhas dos eixos X e Y  
  scale_y_continuous(expand = c(0, 0), limits = c(0, 15)) + #  
  Definir limites do eixo Y  
  guides(fill = "none") # Remover a legenda de cores
```

Creating the relative frequency chart of symptoms in relation to deaths

16 Create the symptom chart in the same manner as mentioned earlier. The script is provided below:

16.1 i. Convert categorical data into numerical data

Command

new command name

```
# Convert categorical variables to numeric values (0 for "no" and 1 for "yes")
```

```
sinan <- sinan %>%  
  mutate(  
    febre = ifelse(febr == "sim", 1, 0),  
    mialgia = ifelse(mialgia == "sim", 1, 0),  
    cefaleia = ifelse(cefaleia == "sim", 1, 0),  
    exantema = ifelse(exantema == "sim", 1, 0),  
    vomito = ifelse(vomito == "sim", 1, 0),  
    nausea = ifelse(nausea == "sim", 1, 0),  
    `dor nas costas` = ifelse(`dor nas costas` == "sim", 1, 0),  
    conjuntivite = ifelse(conjuntivite == "sim", 1, 0),  
    artrite = ifelse(artrite == "sim", 1, 0),  
    artralgia = ifelse(artralgia == "sim", 1, 0),  
    leucopenia = ifelse(leucopenia == "sim", 1, 0),  
    `dor retrorbital` = ifelse(`dor retrorbital` == "sim", 1, 0))
```

16.2 ii. Calculating the percentage of patients for each symptom

Command**new command name**

ii. Calculating the percentage of patients with each symptom

```
sintomas_porcentagem_menor_10_dias <- menor_10_dias %>%  
  summarise(  
    febre = mean(febre) * 100,  
    mialgia = mean(mialgia) * 100,  
    cefaleia = mean(cefaleia) * 100,  
    exantema = mean(exantema) * 100,  
    vomito = mean(vomito) * 100,  
    nausea = mean(nausea) * 100,  
    `dor nas costas` = mean(`dor nas costas`) * 100,  
    conjuntivite = mean(conjuntivite) * 100,  
    artrite = mean(artrite) * 100,  
    artralgia = mean(artralgia) * 100,  
    leucopenia = mean(leucopenia) * 100,  
    `dor retrorbital` = mean(`dor retrorbital`) * 100  
  )
```

16.3 iii. Transform the data from wide to long format

Command**new command name**

```
# Transform the dataset from wide to long format (tidy)  
sintomas <- sintomas %>%  
  pivot_longer(cols = everything(), names_to = "Sintoma", values_to =  
    "Porcentagem")
```

16.4 iv. Create the chart

Command

new command name

```
# Create a stacked bar chart with gradient colors and outline
ggplot(data = sintomas, aes(x = reorder(Sintoma, -Porcentagem), y =
Porcentagem, fill = Porcentagem)) +
  geom_bar(stat = "identity", width = 0.3) + # Barras preenchidas
  geom_bar(stat = "identity", width = 0.3, color = "black", position
= "identity", fill = NA, size = 1) + # Contorno das barras
  labs(title = "Porcentagem dos Sintomas Presentes no óbitos por
CHIKV ",
        x = "Sintomas",
        y = "Porcentagem (%)",
        caption = "Fonte: SINAN/SUVISA") +
  scale_fill_gradient(low = "plum1", high = "darkred") + # Definir
cores do degradê
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(), # Remover borda do painel
        axis.line = element_line(color = "black", size = 0.3)) + #
Linhas dos eixos X e Y
  scale_y_continuous(expand = c(0, 0)) + # Remover expansão no eixo Y
  guides(fill = "none") # Remover a legenda de cores
```

Creating the map of municipalities in Goiás

17 "Now we will create the map. The region used as an example is the state of Goiás, Brazil, showing the frequency of deaths by municipalities. The code is provided below:"

17.1 i. Import the database from IBGE for the state of Goiás

Command

new command name

```
# i. Import the IBGE database for Goiás
```

```
read_municipality() # baixa os dados de todos os municipios  
brasileiros
```

```
read_municipality(code_muni = 'GO')
```

```
go <- read_municipality(code_muni = 'GO')
```

17.2 ii. Processing the IBGE database

Command

new command name

```
# ii. Cleaning characters and standardizing variables

# Inserting the `colnames()` function to check the variables
colnames(go)

# Using the `clean_names()` function to edit variable names
##This function makes the variable names all standardized, written in
lowercase, without accents and without spaces between words.
go <- clean_names(go)

##Delete accents and special characters
RemoveAcentos = function(textoComAcentos) {
  if(!is.character(textoComAcentos)){
    on.exit()
  }
  letrasComAcentos =
"áéíóúÁÉÍÓÚýÝàèìòùÀÈÌÒÙâêîôûÂÊÎÔÛãõÃÕñÑäëïöüÄËÏÖÜçÇ´`^~""*, "
  letrasSemAcentos =
"aeiouAEIOUyYaeiouAEIOUaeiouAEIOUaoAOonNaeiouAEIOUycC      "
  textoSemAcentos = chartr(
    old = letrasComAcentos,
    new = letrasSemAcentos,
    x = textoComAcentos
  )
  return(textoSemAcentos)
}

go$name_muni = as.character(go$name_muni)
go = go %>% mutate_if(is.character, RemoveAcentos)

#leave comments in lowercase letters (dplyr package)

# Use the tolower() function to convert municipal names to lowercase

go <- go %>%
  mutate(municipio = tolower(name_muni))
```

17.3 iii. Perform a join of the IBGE database with the death frequency table

Command

new command name

```
#iii. Perform a left join between objects
mapa <- merge(go, dados_frequencia, by.x = "municipio", by.y =
"Municipio", all.x = TRUE)

# Fill in missing values with 0 in the "Frequencia" variable
mapa$Frequencia[is.na(mapa$Frequencia)] <- 0

#Rename the variable 'Frequencia' to 'Occurrence of deaths due to
CHIKV'
mapa = rename (mapa, "Ocorrência de óbitos por CHIKV" = Frequencia)
```

17.4 iv. Generating the map

Command

new command name

```
# iv. Creating the map
```

```
ggplot() + geom_sf(data = mapa) # the + sign tells R to add another layer generated by the other function (which receives the spatial data)
```

```
#the "data" argument of the function is used to specify the data to be plotted on the graph
```

```
## Adding color to the map using the fill argument
```

```
ggplot() + geom_sf(data = mapa, aes(fill=`Ocorrência de óbitos por CHIKV`))+
```

```
#This line of code creates the map of Goiás based on IBGE data. Additionally, the argument 'aes(fill=code_state)' colored the map according to the state code number.
```

```
## Defining the graphic scale of the map with the annotation_scale() function
```

```
annotation_scale(location = "br",  
                  height = unit(.1, "cm")) +
```

```
# Setting the North-pointing arrow view
```

```
annotation_north_arrow(location = "tr",  
                        height = unit(1, "cm"),  
                        style = north_arrow_fancy_orienteering)+
```

```
# Adding the theme
```

```
theme_bw() +
```

```
# Adding a caption to the data source
```

```
labs(caption = "Fonte: Sinan/SUVISA") +
```

```
# Set the color scale
```

```
scale_fill_gradient(trans = "reverse", low = "red", high =  
"mintcream") +
```

```
# Removing color legend
```

```
guides(fill = FALSE)
```



Protocol references

1. Chiavegatto Filho, A. D. P. **Uso de big data em saúde no Brasil: perspectivas para um futuro próximo.** *Epidemiologia e Serviços de Saúde*, v. 24, n. 2, p. 325–332, jun. 2015. Acesso em: 05 de novembro de 2023
2. Coronel, C.; Morris, S. **Database Systems Design, Implementation, & Management.** 2017. Disponível em: www.cengage.com/highered. Acesso em: 05 nov de 2023.
3. DATASUS - Departamento de Informática do Sistema Único de Saúde. **Sistema de Informação do SUS (DATASUS).** Disponível em: <https://datasus.saude.gov.br/aceso-a-informacao/>. Acesso em: 01 dez. 2023.