

# Análises dos dados SIVEP-GRIPE, SINASC e SIM para painel Qualitados

Qualitados

13/maio/2025

## Bases, importações e devidos tratamentos.

A seguir, são carregados os pacotes do R (<https://www.r-project.org>) utilizados para filtragem e tratamento dos dados considerados no dashboard [https://observatorioobstetrico.shinyapps.io/oobr\\_qualitados2/](https://observatorioobstetrico.shinyapps.io/oobr_qualitados2/). Os dados do painel foram atualizados no dia 12/maio/2025.

```
#carregar pacotes
loadlibrary <- function(x) {
  if (!require(x, character.only = TRUE)) {
    install.packages(x, dependencies = T)
    if (!require(x, character.only = TRUE))
      stop("Package not found")
  }
}

packages <-
  c(
    "readr",
    "readxl",
    "janitor",
    "dplyr",
    "forcats",
    "stringr",
    "lubridate",
    "summarytools",
    "magrittr",
    "questionr",
    "knitr",
    "data.table",
    "writexl",
    "modelsummary",
    'coro',
    'getPass', 'httr'
  )
lapply(packages, loadlibrary)
```

# SIVEP-GRIPE

A base de dados SIVEP-Gripe (Sistema de Informação da Vigilância Epidemiológica da Gripe) contém os registros de casos e óbitos de SRAG (Síndrome Respiratória Aguda Grave). A notificação é compulsória para síndrome gripal, caracterizada por pelo menos dois dos seguintes sinais e sintomas: febre, mesmo que referida, calafrios, dor de garganta, dor de cabeça, tosse, coriza, distúrbios olfatórios ou de paladar, além de dispneia/desconforto respiratório, pressão persistente no peito, Saturação de O<sub>2</sub> menor que 95% no ar ambiente ou cor azulada dos lábios ou rosto. Indivíduos assintomáticos com confirmação laboratorial por biologia molecular ou exame imunológico para infecção por COVID-19 também são relatados.

Para notificações no Sivep-Gripe, devem ser considerados os casos hospitalizados em hospitais públicos e privados, bem como todas as mortes decorrentes de infecções respiratórias agudas graves, independentemente da hospitalização.

A vigilância da SRAG no Brasil é realizada pelo Ministério da Saúde (MS), por meio da Secretaria de Vigilância em Saúde (SVS), desde a pandemia de Influenza A (H1N1) em 2009. Para obter mais informações, acesse: <https://coronavirus.saude.gov.br/definicao-de-caso-e-notificacao>.

## Extração

Os dados de 2009 a 2019 são extraídos com auxílio da API da Plataforma de Ciência de Dados Aplicada à Saúde (PCDaS) e, em seguida, são tratados com base no fluxo ETL (Extração, Transformação e Carga), os dados para 2020 a 2024 são extraído do próprio Open Datasus. Durante a extração da API, os dados são filtrados utilizando consultas SQL, conforme demonstrado, em que a variável CS\_GESTANT assume os seguintes valores: 1-1º Trimestre; 2-2º Trimestre; 3-3º Trimestre; 4-Idade Gestacional Ignorada; 5-Não; 6-Não se aplica; 9-Ignorado.

```
# Dados 2009 - 2019 -----

# Função para converter os resultados das consultas para data.frame
convertRequestToDF <- function(request, column_names = c()){
  if("RequestError" %in% names(content(request)))
    stop(content(request)$RequestError)
  variables = unlist(content(request)$columns)
  variables = variables[names(variables) == "name"]
  if (!length(column_names)){
    column_names <- unname(variables)
  }
  values = content(request,)$rows
  df <- as.data.frame(do.call(rbind,lapply(values,function(r) {
    row <- r
    row[sapply(row, is.null)] <- NA
    rbind(unlist(row))
  } )))
  names(df) <- column_names
  return(df)
}

query_with_cursor <- generator(function(sql_query, token, nrows){
  tryCatch({
    json_api <- paste0('{"token": {"token": "',token,'"}, "sql": {"sql":
      {"query": "',sql_query,'" , "fetch_size": "',nrows,'" }}}')
    response <- POST(url = "https://bigdata-api.fiocruz.br/sql_query/",
```

```

        body = json_api, encode = "json")
df <- convertRequestToDF(response)
col_names <- colnames(df)
yield(df)
while(TRUE){
  json_api <- paste0('{"token": {"token": "',token,'"}, "sql": {"sql":
                    {"cursor": "',content(response)$cursor,'"} } }')
  response <- POST(url = "https://bigdata-api.fiocruz.br/sql_query/",
                  body = json_api, encode = "json")
  if(length(content(response)$rows)>0){
    yield(convertRequestToDF(response,col_names))
  }
  else return(NULL)
}
}, error=function(cond) message(paste0(cond,"\n",content(response))) )
})

convertColTypeToNum <- function(df, colname){
  df[,colname] <- as.numeric(as.character(df[,colname]))
  return(df)
}

anos <- c(2009:2019)
df_total_max3 <- data.frame()
for(i in anos){
  query <- paste0('SELECT (*)',
                  ' FROM \\"datasus-srags\\" WHERE (',
                  '(CAST(RIGHT(DT_SIN_PRI, 4) AS int) = ',i,') AND ',
                  '(CS_GESTANT = 1 OR CS_GESTANT = 1.0 OR ',
                  'CS_GESTANT = 2 OR CS_GESTANT = 2.0 OR ',
                  'CS_GESTANT = 3 OR CS_GESTANT = 3.0 OR ',
                  'CS_GESTANT = 4 OR CS_GESTANT = 4.0 OR ',
                  'PUERPERA = 1 OR PUERPERA = 1.0))')

  df_total <- data.frame()
  loop(for (df in query_with_cursor(query, token, nrows=10000)) {
    print(paste0('Número de registros recuperados a cada iteração: ', nrow(df)))
    df_total <- rbind(df_total,df)
  })
  df_total_max3 <- rbind(df_total,df_total_max3)
}

write_rds(df_total_max3,file = 'data1/Sivep_2009-2019.rds')

# Dados 2020-2023-----
#carregar pacotes
loadlibrary <- function(x) {
  if (!require(x, character.only = TRUE)) {
    install.packages(x, dependencies = T)
  }
}

```

```

    if (!require(x, character.only = TRUE))
      stop("Package not found")
  }
}

packages <-
  c(
    "readr",
    "readxl",
    "janitor",
    "dplyr",
    "forcats",
    "stringr",
    "lubridate",
    "summarytools",
    "magrittr",
    "questionr",
    "knitr",
    "data.table",
    "writexl",
    "modelsummary",
    'coro',
    'getPass', 'httr'
  )
lapply(packages, loadlibrary)
ckanr::ckanr_setup("https://opendatasus.saude.gov.br")

arqs <- ckanr::package_search("srag 2020")$results %>%
  purrr::map("resources") %>%
  purrr::map(purrr::keep, ~.x$mimetype == "text/csv") %>%
  purrr::map_chr(purrr::pluck, 1, "url")

arqs2 <- ckanr::package_search("srag 2021")$results %>%
  purrr::map("resources") %>%
  purrr::map(purrr::keep, ~.x$mimetype == "text/csv") %>%
  purrr::map_chr(purrr::pluck, 2, "url")

arqs3 <- ckanr::package_search("srag 2021")$results %>%
  purrr::map("resources") %>%
  purrr::map(purrr::keep, ~.x$mimetype == "text/csv") %>%
  purrr::map_chr(purrr::pluck, 3, "url")

dados_a <- fread(arqs[1], sep = ";")

dados_b <- fread(arqs[2], sep = ";")

dados_c <- fread(arqs2[1], sep = ";")

dados_d <- fread(arqs3[1], sep = ";")
dados_a$FATOR_RISC <- dados_a$FATOR_RISC %>% as.character()
dados_b$FATOR_RISC <- dados_b$FATOR_RISC %>% as.character()
dados_c$FATOR_RISC <- dados_c$FATOR_RISC %>% as.character()
dados_d$FATOR_RISC <- dados_d$FATOR_RISC %>% as.character()

```

```

dados_total <- full_join(dados_a, dados_b) %>%
  full_join(dados_c) %>%
  full_join(dados_d)
dados_total <- dados_total %>%
  filter(
    (CS_GESTANT == 1 | CS_GESTANT == 1.0 | CS_GESTANT == '1' |
     CS_GESTANT == '1.0' |
     CS_GESTANT == 2 | CS_GESTANT == 2.0 | CS_GESTANT == '2' |
     CS_GESTANT == '2.0' |
     CS_GESTANT == 3 | CS_GESTANT == 3.0 | CS_GESTANT == '3' |
     CS_GESTANT == '3.0' |
     CS_GESTANT == 4 | CS_GESTANT == 4.0 | CS_GESTANT == '4' |
     CS_GESTANT == '4.0' |
     PUERPERA == 1 | PUERPERA == 1.0 | PUERPERA == '1' |
     PUERPERA == '1.0')
  )
write_rds(dados_total, file = 'data1/Sivep_2020-2023.rds')

```

Há atualmente 65903 observações na base de dados e são as variáveis:

```
names(df)
```

```

## [1] "DT_NOTIFIC" "SEM_NOT" "DT_SIN_PRI" "SEM_PRI"
## [5] "SG_UF_NOT" "ID_REGIONA" "CO_REGIONA" "ID_MUNICIP"
## [9] "CO_MUN_NOT" "ID_UNIDADE" "CO_UNI_NOT" "CS_SEXO"
## [13] "DT_NASC" "NU_IDADE_N" "TP_IDADE" "COD_IDADE"
## [17] "CS_GESTANT" "CS_RACA" "CS_ESCOL_N" "ID_PAIS"
## [21] "CO_PAIS" "SG_UF" "ID_RG_RESI" "CO_RG_RESI"
## [25] "ID_MN_RESI" "CO_MUN_RES" "CS_ZONA" "SURTO_SG"
## [29] "NOSOCOMIAL" "AVE_SUINO" "FEBRE" "TOSSE"
## [33] "GARGANTA" "DISPNEIA" "DESC_RESP" "SATURACAO"
## [37] "DIARREIA" "VOMITO" "OUTRO_SIN" "OUTRO_DES"
## [41] "PUERPERA" "FATOR_RISC" "CARDIOPATI" "HEMATOLOGI"
## [45] "SIND_DOWN" "HEPATICA" "ASMA" "DIABETES"
## [49] "NEUROLOGIC" "PNEUMOPATI" "IMUNODEPRE" "RENAL"
## [53] "OBESIDADE" "OBES_IMC" "OUT_MORBI" "MORB_DESC"
## [57] "VACINA" "DT_UT_DOSE" "MAE_VAC" "DT_VAC_MAE"
## [61] "M_AMAMENTA" "DT_DOSEUNI" "DT_1_DOSE" "DT_2_DOSE"
## [65] "ANTIVIRAL" "TP_ANTIVIR" "OUT_ANTIV" "DT_ANTIVIR"
## [69] "HOSPITAL" "DT_INTERNA" "SG_UF_INTE" "ID_RG_INTE"
## [73] "CO_RG_INTE" "ID_MN_INTE" "CO_MU_INTE" "UTI"
## [77] "DT_ENTUTI" "DT_SAIDUTI" "SUPOORT_VEN" "RAIOX_RES"
## [81] "RAIOX_OUT" "DT_RAIOX" "AMOSTRA" "DT_COLETA"
## [85] "TP_AMOSTRA" "OUT_AMOST" "PCR_RESUL" "DT_PCR"
## [89] "POS_PCRFLU" "TP_FLU_PCR" "PCR_FLUASU" "FLUASU_OUT"
## [93] "PCR_FLUBLI" "FLUBLI_OUT" "POS_PCROUT" "PCR_VSR"
## [97] "PCR_PARA1" "PCR_PARA2" "PCR_PARA3" "PCR_PARA4"
## [101] "PCR_ADENO" "PCR_METAP" "PCR_BOCA" "PCR_RINO"
## [105] "PCR_OUTRO" "DS_PCR_OUT" "CLASSI_FIN" "CLASSI_OUT"
## [109] "CRITERIO" "EVOLUCAO" "DT_EVOLUCA" "DT_ENCERRA"
## [113] "DT_DIGITA" "HISTO_VGM" "PAIS_VGM" "CO_PS_VGM"
## [117] "LO_PS_VGM" "DT_VGM" "DT_RT_VGM" "PCR_SARS2"

```

## [121]	"PAC_COCBO"	"PAC_DSCBO"	"OUT_ANIM"	"DOR_ABD"
## [125]	"FADIGA"	"PERD_OLFT"	"PERD_PALA"	"TOMO_RES"
## [129]	"TOMO_OUT"	"DT_TOMO"	"TP_TES_AN"	"DT_RES_AN"
## [133]	"RES_AN"	"POS_AN_FLU"	"TP_FLU_AN"	"POS_AN_OUT"
## [137]	"AN_SARS2"	"AN_VSR"	"AN_PARA1"	"AN_PARA2"
## [141]	"AN_PARA3"	"AN_ADENO"	"AN_OUTRO"	"DS_AN_OUT"
## [145]	"TP_AM_SOR"	"SOR_OUT"	"DT_CO_SOR"	"TP_SOR"
## [149]	"OUT_SOR"	"DT_RES"	"RES_IGG"	"RES_IGM"
## [153]	"RES_IGA"	"ESTRANG"	"VACINA_COV"	"DOSE_1_COV"
## [157]	"DOSE_2_COV"	"DOSE_REF"	"FAB_COV_1"	"FAB_COV_2"
## [161]	"FAB_COVREF"	"LOTE_REF"	"LAB_PR_COV"	"LOTE_1_COV"
## [165]	"LOTE_2_COV"	"FNT_IN_COV"	"DOSE_2REF"	"FAB_COVRF2"
## [169]	"LOTE_REF2"	"TRAT_COV"	"TIPO_TRAT"	"OUT_TRAT"
## [173]	"DT_TRT_COV"	"CO_DETEC"	"VG_OMS"	"VG_OMSOUT"
## [177]	"VG_LIN"	"VG_MET"	"VG_METOUT"	"VG_DTRES"
## [181]	"VG_ENC"	"VG_REINF"	"REINF"	"FAB_ADIC"
## [185]	"LOT_RE_BI"	"FAB_RE_BI"	"DOSE_ADIC"	"DOS_RE_BI"
## [189]	"LOTE_ADIC"	"TABAG"	"CASO_SRAG"	"ARTRALGIA"
## [193]	"AVE_10_DIA"	"CALAFRIO"	"CONJUNTIV"	"CORIZA"
## [197]	"CO_LAB_IF"	"CO_LAB_PCR"	"CO_UF_INTE"	"CULT_AMOST"
## [201]	"CULT_OUT"	"CULT_RES"	"DOENCA_TRA"	"DS_IF_OUT"
## [205]	"DS_OAGEETI"	"DS_OUTMET"	"DS_OUTSUB"	"DT_CULTURA"
## [209]	"DT_HEMAGLU"	"DT_IF"	"DT_IFI"	"DT_OBITO"
## [213]	"DT_OUTMET"	"DT_PCR_1"	"HEMA_ETIOL"	"HEMA_RES"
## [217]	"HEMOGLOBI"	"HEM_TIPO_H"	"HEM_TIPO_N"	"ID_OCUPA_N"
## [221]	"IFI"	"IF_ADENO"	"IF_OUTRO"	"IF_PARA1"
## [225]	"IF_PARA2"	"IF_PARA3"	"IF_RESUL"	"IF_VSR"
## [229]	"LAB_IF"	"LAB_PCR"	"METABOLICA"	"MIALGIA"
## [233]	"MONITORA"	"NU_ANO"	"OUT_METODO"	"PCR"
## [237]	"PCR_AMOSTR"	"PCR_ETIOL"	"PCR_OUT"	"PCR_RES"
## [241]	"PCR_TIPO_H"	"PCR_TIPO_N"	"POS_IF_FLU"	"POS_IF_OUT"
## [245]	"REQUI_GAL"	"RES_ADNO"	"RES_FLUA"	"RES_FLUASU"
## [249]	"RES_FLUB"	"RES_OUTRO"	"RES_PARA1"	"RES_PARA2"
## [253]	"RES_PARA3"	"RES_VSR"	"SRAG2009FINAL"	"SRAG2010FINAL"
## [257]	"SRAG2011FINAL"	"SRAG2012FINAL"	"SRAG2013FINAL"	"SRAG2014FINAL"
## [261]	"SRAG2015FINAL"	"SRAG2017FINAL"	"SRAG2018FINAL"	"ST_TIPOFI"
## [265]	"TABAGISMO"	"TIPO_PCR"	"TPAUTOCTO"	"TP_FLU_IF"
## [269]	"watermark"			

## Tratamento

A base de dados do SIVEP-GRIPE utilizada no Painel Qualitados passa por um processo de reorganização, no qual os valores das observações que se enquadram em alguma das regras de indicadores de má qualidade dos dados (Incompletude, Implausibilidade ou Inconsistência) são substituídos. Os indicadores podem ser visualizados na aba de dicionário, na tabela de regras, para cada uma das respectivas bases de dados dentro do Painel. Por exemplo, os dados “NA” (Not Available) são substituídos por “Em Branco”. Tanto os dicionários de variáveis quanto o conjunto de regras estão disponíveis no GitHub do Painel, no seguinte endereço: <https://github.com/observatorioobstetrico/Qualitados>.

```
SIVEP_dic <- read_excel("data1/dicionarios.xlsx", sheet = "SIVEP")
df <- readRDS("data1/Sivep_2009-2022.rds")
df1 <- readRDS("data1/Sivep_2020-2023.rds")
variaveis_dic <- SIVEP_dic$`Codigo SIVEP`
```

```

df <- df[!( ( as.Date(df$DT_SIN_PRI, format = "%d/%m/%Y") %>%
lubridate::year() ) %in% c(2020,2021,2022)),]
df1 <- df1 %>%
  mutate_all(as.character)
df<-bind_rows(df1, df)
#BANCO AUXILIAR PARA CORRECAO DOS MUNICIPIOS
aux_muni2 <- abjData::muni %>%
  dplyr::select(uf_id,
                muni_id,
                muni_nm_clean,
                uf_sigla) %>%
  mutate_at("muni_id", as.character) %>%
  mutate(cod_mun = stringr::str_sub(muni_id, 1, 6))

#CRIANDO CLASSIFICACAO DE GESTANTE E PUERP E CORRIGINDO OS MUNICIPIOS
df_gest <- df %>%
  #CORRECAO MUNICIPIOS
  left_join(aux_muni2, by = c("ID_MUNICIP" = "cod_mun")) %>%
  mutate(SG_UF_NOT = ifelse(is.na(muni_nm_clean),
                            SG_UF_NOT, uf_sigla),
         ID_MUNICIP = ifelse(is.na(muni_nm_clean),
                              ID_MUNICIP, muni_nm_clean)) %>%
  mutate(
    #DATA DO PRIMEIRO SINTOMA
    dt_sint = as.Date(DT_SIN_PRI, format = "%d/%m/%Y"),
    #DATA DO NASCIMENTO
    dt_nasc = as.Date(DT_NASC, format = "%d/%m/%Y"),
    #ANO, BASEADO NA DATA DO PRIMEIRO SINTOMA
    ANO = lubridate::year(dt_sint),
    #MUNICIPIO
    MUNICIPIO = paste(ID_MUNICIP, "-", SG_UF_NOT)
  ) %>% select(-muni_nm_clean, -uf_sigla)
# CORRECAO DO ERRO QUE A FALTA DE PADRONIZACAO DOS DADOS OCASIONOU
df_gest <- df_gest %>% mutate_if(~ !is.character(.), as.character)
df_gest <- data.frame(lapply(df_gest,
  function(x) ifelse(x == "1.0", '1',
    ifelse(x == '2.0', '2',
      ifelse(x == '3.0', '3',
        ifelse(x == '4.0', '4',
          ifelse(x == '5.0', '5',
            ifelse(x == '6.0', '6',
              ifelse(x == '7.0', '7',
                ifelse(x == '8.0', '8',
                  ifelse(x == '9.0', '9', x)))))))))))

df_gest %>% nrow()#CONFERINDO SE VOLTOU TUDO

sivep2 <- df_gest

# INCOMPLETUDE -----
regras_incom <- fromJSON('data1/incompletude_sivep.json')

#VARIAVEIS DO DICIONARIO + VARIAVEIS PARA FILTRAGEM

```

```

df_gest2 <-
  df_gest[,c(variaveis_dic,'ANO','MUNICIPIO','SG_UF_NOT','CLASSI_FIN')]

#VARIÁVEIS EM QUE O VALOR 9 É O VALOR IGNORADO:
variaveis_ign <- c('CS_SEXO','CS_RACA','CS_ESCOL_N','CS_ZONA','NOSOCOMIAL',
  'AVE_SUINO','FEBRE','TOSSE','GARGANTA','DISPNEIA',
  'DESC_RESP','SATURACAO','DIARREIA','VOMITO','OUTRO_SIN',
  'FATOR_RISC','CARDIOPATI','HEMATOLOGI','SIND_DOWN',
  'HEPATICA','ASMA','DIABETES','NEUROLOGIC','PNEUMOPATI',
  'IMUNODEPRE','RENAL','OBESIDADE','OUT_MORBI',
  'MAE_VAC','M_AMAMENTA','ANTIVIRAL','HOSPITAL','UTI',
  'SUPOUT_VEN','AMOSTRA','POS_PCRFLU','POS_PCRROUT',
  'EVOLUCAO','DOR_ABD','FADIGA','PERD_OLFT','PERD_PALA',
  'POS_AN_FLU','POS_AN_OUT','CS_GESTANT',
  'TOMO_RES','VACINA_COV','VACINA','PUERPERA',
  'CLASSI_FIN','RAIOX_RES' )

setdiff(variaveis_dic,variaveis_ign)
#SUBSTITUIR VALORES NA POR EM BRANCO
sivep <- replace(df_gest2,is.na(df_gest2) ,"Em Branco")

#SUBSTITUIR VALORES 9 POR IGNORADO
sivep[, variaveis_ign] <- lapply(sivep[, variaveis_ign],
  function(x) ifelse((x == '9'|x == '9.0'),
    "Ignorado", x))

# Calcular as porcentagens de valores 'Ignorados' e
# 'Em branco' por coluna so para ver se funcionou
colMeans(sivep == "Ignorado", na.rm = TRUE) * 100
colMeans(sivep == "Em Branco", na.rm = TRUE) * 100

# IMPLAUSIBILIDADE -----

regras_implau <- fromJSON('data1/implausibilidade_gestantes.json')
regras_implau2 <- fromJSON('data1/implausibilidade_puerperas.json')

# Criando vetores de variáveis improváveis e impossíveis
improvavel <- grep("_IMPROVAVEL", names(regras_implau), value = TRUE)
impossivel <- grep("_IMPOSSIVEL", names(regras_implau), value = TRUE)
impossivel2 <- grep("_IMPOSSIVEL", names(regras_implau2), value = TRUE)
impossivel <- c(impossivel2,impossivel) %>% unique()

# Criando um data.frame com as variáveis improváveis
df_improvavel <- data.frame(
  variavel = gsub(improvavel,pattern = '_IMPROVAVEL',replacement = ''))

# Criando um data.frame com as variáveis impossíveis
df_impossivel <- data.frame(
  variavel = gsub(impossivel,pattern = '_IMPOSSIVEL',replacement = ''))

# Trocando regras em string por booleanos
df_impossivel <- df_impossivel %>%
  mutate(condicao = case_when(
    grepl("CS_SEXO", variavel) ~

```



```

"CS_SEXO != 'F'",
grepl("NU_IDADE_N", variavel) ~
"as.integer(NU_IDADE_N) < 0 | as.integer(NU_IDADE_N) > 90",
grepl("CS_GESTANT", variavel) ~
"CS_GESTANT %in% c('1','2','3','4') & PUERPERA == '1' ",
grepl("DT_INTERNA", variavel) ~
"lubridate::year(as.Date(DT_INTERNA,format = '%d/%m/%Y')) < 2019
& !(is.na(lubridate::year(as.Date(DT_INTERNA,format = '%d/%m/%Y'))))",
grepl("DT_COLETA", variavel) ~
"lubridate::year(as.Date(DT_COLETA,format = '%d/%m/%Y')) < 2019 &
!(is.na(lubridate::year(as.Date(DT_COLETA,format = '%d/%m/%Y'))))",
grepl("TP_IDADE", variavel) ~
"TP_IDADE != '1' & TP_IDADE != '2' & TP_IDADE != '3'",
grepl("TP_ANTIVIR", variavel) ~
"TP_ANTIVIR != '1' & TP_ANTIVIR != '2' & TP_ANTIVIR != '3'",
grepl("SURTO_SG", variavel) ~
"SURTO_SG != '1' & SURTO_SG != '2' & SURTO_SG != 'Ignorado'",
grepl("NOSOCOMIAL", variavel) ~
"NOSOCOMIAL != '1' & NOSOCOMIAL != '2' & NOSOCOMIAL != 'Ignorado'",
grepl("AVE_SUINO", variavel) ~
"AVE_SUINO != '1' & AVE_SUINO != '2' & AVE_SUINO != 'Ignorado'",
grepl("FEBRE", variavel) ~
"FEBRE != '1' & FEBRE != '2' & FEBRE != 'Ignorado'",
grepl("TOSSE", variavel) ~
"TOSSE != '1' & TOSSE != '2' & TOSSE != 'Ignorado'",
grepl("GARGANTA", variavel) ~
"GARGANTA != '1' & GARGANTA != '2' & GARGANTA != 'Ignorado'",
grepl("DISPNEIA", variavel) ~
"DISPNEIA != '1' & DISPNEIA != '2' & DISPNEIA != 'Ignorado'",
grepl("DESC_RESP", variavel) ~
"DESC_RESP != '1' & DESC_RESP != '2' & DESC_RESP != 'Ignorado'",
grepl("SATURACAO", variavel) ~
"SATURACAO != '1' & SATURACAO != '2' & SATURACAO != 'Ignorado'",
grepl("DIARREIA", variavel) ~
"DIARREIA != '1' & DIARREIA != '2' & DIARREIA != 'Ignorado'",
grepl("VOMITO", variavel) ~
"VOMITO != '1' & VOMITO != '2' & VOMITO != 'Ignorado'",
grepl("OUTRO_SIN", variavel) ~
"OUTRO_SIN != '1' & OUTRO_SIN != '2' & OUTRO_SIN != 'Ignorado'",
grepl("FATOR_RISC", variavel) ~
"FATOR_RISC != '1' & FATOR_RISC != '2' & FATOR_RISC != 'Ignorado'",
grepl("CARDIOPATI", variavel) ~
"CARDIOPATI != '1' & CARDIOPATI != '2' & CARDIOPATI != 'Ignorado'",
grepl("HEMATOLOGI", variavel) ~
"HEMATOLOGI != '1' & HEMATOLOGI != '2' & HEMATOLOGI != 'Ignorado'",
grepl("SIND_DOWN", variavel) ~
"SIND_DOWN != '1' & SIND_DOWN != '2' & SIND_DOWN != 'Ignorado'",
grepl("HEPATICA", variavel) ~
"HEPATICA != '1' & HEPATICA != '2' & HEPATICA != 'Ignorado'",
grepl("ASMA", variavel) ~
"ASMA != '1' & ASMA != '2' & ASMA != 'Ignorado'",
grepl("DIABETES", variavel) ~
"DIABETES != '1' & DIABETES != '2' & DIABETES != 'Ignorado'",

```

```

grepl("NEUROLOGIC", variavel) ~
  "NEUROLOGIC != '1' & NEUROLOGIC != '2' & NEUROLOGIC != 'Ignorado'",
grepl("PNEUMOPATI", variavel) ~
  "PNEUMOPATI != '1' & PNEUMOPATI != '2' & PNEUMOPATI != 'Ignorado'",
grepl("IMUNODEPRE", variavel) ~
  "IMUNODEPRE != '1' & IMUNODEPRE != '2' & IMUNODEPRE != 'Ignorado'",
grepl("RENAL", variavel) ~
  "RENAL != '1' & RENAL != '2' & RENAL != 'Ignorado'",
grepl("OBESIDADE", variavel) ~
  "OBESIDADE != '1' & OBESIDADE != '2' & OBESIDADE != 'Ignorado'",
grepl("OUT_MORBI", variavel) ~
  "OUT_MORBI != '1' & OUT_MORBI != '2' & OUT_MORBI != 'Ignorado'",
grepl("VACINA", variavel) ~
  "VACINA != '1' & VACINA != '2' & VACINA != 'Ignorado'",
grepl("MAE_VAC", variavel) ~
  "MAE_VAC != '1' & MAE_VAC != '2' & MAE_VAC != 'Ignorado'",
grepl("M_AMAMENTA", variavel) ~
  "M_AMAMENTA != '1' & M_AMAMENTA != '2' & M_AMAMENTA != 'Ignorado'",
grepl("ANTIVIRAL", variavel) ~
  "ANTIVIRAL != '1' & ANTIVIRAL != '2' & ANTIVIRAL != 'Ignorado'",
grepl("HOSPITAL", variavel) ~
  "HOSPITAL != '1' & HOSPITAL != '2' & HOSPITAL != 'Ignorado'",
grepl("UTI", variavel) ~
  "UTI != '1' & UTI != '2' & UTI != 'Ignorado'",
grepl("AMOSTRA", variavel) ~
  "AMOSTRA != '1' & AMOSTRA != '2' & AMOSTRA != 'Ignorado'",
grepl("POS_PCRFLU", variavel) ~
  "POS_PCRFLU != '1' & POS_PCRFLU != '2' & POS_PCRFLU != 'Ignorado'",
grepl("POS_PCROUT", variavel) ~
  "POS_PCROUT != '1' & POS_PCROUT != '2' & POS_PCROUT != 'Ignorado'",
grepl("HISTO_VGM", variavel) ~
  "HISTO_VGM != '1' & HISTO_VGM != '2' & HISTO_VGM != 'Ignorado'",
grepl("DOR_ABD", variavel) ~
  "DOR_ABD != '1' & DOR_ABD != '2' & DOR_ABD != 'Ignorado'",
grepl("FADIGA", variavel) ~
  "FADIGA != '1' & FADIGA != '2' & FADIGA != 'Ignorado'",
grepl("PERD_OLFT", variavel) ~
  "PERD_OLFT != '1' & PERD_OLFT != '2' & PERD_OLFT != 'Ignorado'",
grepl("PERD_PALA", variavel) ~
  "PERD_PALA != '1' & PERD_PALA != '2' & PERD_PALA != 'Ignorado'",
grepl("POS_AN_FLU", variavel) ~
  "POS_AN_FLU != '1' & POS_AN_FLU != '2' & POS_AN_FLU != 'Ignorado'",
grepl("POS_AN_OUT", variavel) ~
  "POS_AN_OUT != '1' & POS_AN_OUT != '2' & POS_AN_OUT != 'Ignorado'",
grepl("TP_AM_SOR", variavel) ~
  "TP_AM_SOR != '1' & TP_AM_SOR != '2' & TP_AM_SOR != 'Ignorado'",
grepl("PUERPERA", variavel) ~
  "(PUERPERA %in% c('1')) & (CS_GESTANT %in% c('1','2','3','4'))"
))
df_improvavel <- df_improvavel %>%
mutate(condicao = case_when(
  grepl("NU_IDADE_N", variavel) ~
    "(as.integer(NU_IDADE_N) < 10 & as.integer(NU_IDADE_N) >= 0) |

```

```

(as.integer(NU_IDADE_N) > 55 & as.integer(NU_IDADE_N) <= 90)))

#Substituindo os valores do banco sivep por improvavel e impossivel
attach(sivep)
for(i in 1:nrow(df_impossivel)){
  var <- df_impossivel$variavel[i]
  cond <- df_impossivel$condicao[i]
  sivep[eval(parse
    (text = paste0(cond," & (",
      var," != 'Em Branco'"))),var]<- 'Impossivel'
}
for(i in 1:nrow(df_improvavel)){
  var <- df_improvavel$variavel[i]
  cond <- df_improvavel$condicao[i]
  sivep[eval(parse(text = paste0("(",cond,") & (",
    var," != 'Em branco' & ",
    var," != 'Ignorado' "))),var] <- 'Improvavel'
}
detach(sivep)
sivep_ic_ip <- sivep

# INCONSISTENCIA -----

regras_incon <- fromJSON('data1/SIVEP_Inconsistencias_Regras.json')

# Criando um data.frame com as variáveis improváveis
df_inconsistencia <- data.frame(
  variavel = names(regras_incon) %>% gsub(pattern = '_e_', replacement = ' e '))

# Trocando regras em string por booleanos
df_inconsistencia <- df_inconsistencia %>%
  mutate(condicao = case_when(
    grepl("CS_SEXO e CS_GESTANT", variavel) ~
      "(df_gest_aux$CS_SEXO %in% c('M', 'I')) &
      (df_gest_aux$CS_GESTANT %in% c('1','2','3','4'))",
    grepl("FATOR_RISC e COMORBIDADES", variavel) ~
      "((df_gest_aux$FATOR_RISC == '2' | df_gest_aux$FATOR_RISC == '9')
      & (df_gest_aux$CARDIOPATI == '1' | df_gest_aux$HEMATOLOGI == '1' |
      df_gest_aux$SIND_DOWN == '1' | df_gest_aux$HEPATICA == '1' |
      df_gest_aux$ASMA == '1' | df_gest_aux$DIABETES == '1' |
      df_gest_aux$NEUROLOGIC == '1' | df_gest_aux$PNEUMOPATI == '1' |
      df_gest_aux$IMUNODEPRE == '1' | df_gest_aux$RENAL == '1' |
      df_gest_aux$OBESIDADE == '1' | df_gest_aux$OBES_IMC == '1' |
      df_gest_aux$OUT_MORBI == '1')) | ((df_gest_aux$FATOR_RISC == '1') &
      (df_gest_aux$CARDIOPATI != '1' & df_gest_aux$HEMATOLOGI != '1' &
      df_gest_aux$SIND_DOWN != '1' & df_gest_aux$HEPATICA != '1' &
      df_gest_aux$ASMA != '1' & df_gest_aux$DIABETES != '1' &
      df_gest_aux$NEUROLOGIC != '1' & df_gest_aux$PNEUMOPATI != '1' &
      df_gest_aux$IMUNODEPRE != '1' & df_gest_aux$RENAL != '1' &
      df_gest_aux$OBESIDADE != '1' & df_gest_aux$OBES_IMC != '1' &
      df_gest_aux$OUT_MORBI != '1'))",
    grepl("VACINA e DT_UT_DOSE", variavel) ~
      "df_gest_aux$VACINA %in% c('2', '9') &

```

```

(df_gest_aux$DT_UT_DOSE != 'Em Branco')",
grepl("MAE_VAC e DT_VAC_MAE", variavel) ~
  "df_gest_aux$MAE_VAC %in% c('2', '9') &
(df_gest_aux$DT_VAC_MAE != 'Em Branco')",
grepl("DT_DOSEUNI e NU_IDADE_N", variavel) ~
  "(df_gest_aux$DT_DOSEUNI != 'Em Branco') &

(as.integer(df_gest_aux$NU_IDADE_N) <= '6' |
as.integer(df_gest_aux$NU_IDADE_N) >= '8')",
grepl("ANTIVIRAL e TP_ANTIVIR", variavel) ~ "df_gest_aux$ANTIVIRAL %in%
c('2', '9') & df_gest_aux$TP_ANTIVIR %in% c('1', '2', '3')",
grepl("HOSPITAL e DT_INTERNA", variavel) ~ "df_gest_aux$HOSPITAL %in%
c('2', '9') & (df_gest_aux$DT_INTERNA != 'Em Branco')",
grepl("UTI e DT_ENTUTI", variavel) ~ "(df_gest_aux$UTI == '2' |
df_gest_aux$UTI == '9') & (df_gest_aux$DT_ENTUTI != 'Em Branco') |
(df_gest_aux$HOSPITAL == '2' | df_gest_aux$HOSPITAL == '9') &
df_gest_aux$UTI == '1'",
grepl("RAIOX_RES e DT_RAIOX", variavel) ~ "(df_gest_aux$RAIOX_RES == '6' |
df_gest_aux$RAIOX_RES == '9') & (df_gest_aux$DT_RAIOX != 'Em Branco')",
grepl("AMOSTRA e DT_COLETA", variavel) ~ "(df_gest_aux$AMOSTRA == '6' |
df_gest_aux$AMOSTRA == '9') & (df_gest_aux$DT_COLETA != 'Em Branco')",
grepl("HISTO_VGM e Campos_VGMs", variavel) ~
  "(df_gest_aux$HISTO_VGM == '2' | df_gest_aux$HISTO_VGM == '9') &
(df_gest_aux$LO_PS_VGM != 'Em Branco') &
(df_gest_aux$DT_VGM != 'Em Branco') &
(df_gest_aux$DT_RT_VGM != 'Em Branco')",
grepl("TOMO_RES e DT_TOMO", variavel) ~
  "(df_gest_aux$TOMO_RES == '6' | df_gest_aux$TOMO_RES == '9') &
(df_gest_aux$DT_TOMO != 'Em Branco')",
grepl("TP_TES_AN e DT_RES_AN", variavel) ~ "((df_gest_aux$RES_AN == '4') &
(df_gest_aux$TP_TES_AN %in% c('1', '2')) | ((df_gest_aux$RES_AN == '4') &
(df_gest_aux$DT_RES_AN != 'Em Branco'))",
grepl("VACINA_COV e DOSES", variavel) ~
  "(df_gest_aux$VACINA_COV %in% c('2', '9')) &
((df_gest_aux$DOSE_1_COV != 'Em Branco') |
(df_gest_aux$DOSE_2_COV != 'Em Branco'))",
grepl("CLASSI_FIN_SRAG_INFLUENZA", variavel) ~
  "df_gest_aux$CLASSI_FIN == '1' & df_gest_aux$POS_PCRFLU %in%
c('2', '9') & df_gest_aux$POS_AN_FLU %in% c('2', '9')",
grepl("CLASSI_FIN_SRAG_OUTROS_VIRUS", variavel) ~
  "df_gest_aux$CLASSI_FIN == '1' &
df_gest_aux$PCR_OUTRO %in% c('2', '9') &
df_gest_aux$AN_OUTRO %in% c('2', '9')"
))
df_inconsistencia <- head(df_inconsistencia, -2)

# Criando colunas de inconsistencia no df_gest
df_gest_aux <- df_gest

#SUBSTITUIR VALORES NA POR EM BRANCO
df_gest_aux <- data.frame(lapply(df_gest_aux,
                                function(x) ifelse(is.na(x), "Em Branco", x)))
for(i in 1:nrow(df_inconsistencia)){

```

```

    df_gest_aux[[df_inconsistencia$variavel[i]]] <- 'Nao'
  }
df_gest_aux %>% colnames() #VENDO SE DEU CERTO

# Verificando a condição de inconsistência para cada variável

for(i in 1:(nrow(df_inconsistencia))){
  var <- df_inconsistencia$variavel[i]
  cond <- df_inconsistencia$condicao[i]
  df_gest_aux[eval(parse(text = paste0(cond))) ,var] <- 'Inconsistencia'
}
n <- nrow(df_inconsistencia)
maxi <- ncol(df_gest_aux)

# CONCATENANDO E MUDANDO NOME DAS COLUNAS -----

sivep <- cbind(sivep_ic_ip,df_gest_aux[, (maxi - n + 1):maxi])

#RENOMEANDO AS COLUNAS COM BASE NO DICIONARIO

nomes_colunas <- colnames(sivep)

# Substituindo os nomes originais pelos novos
for(i in seq_along(SIVEP_dic$`Codigo SIVEP`)) {
  nomes_colunas <- gsub(SIVEP_dic$`Codigo SIVEP`[i],
                        SIVEP_dic$`Codigo Qualitados`[i],
                        nomes_colunas)
}

# Atribuindo os novos nomes de colunas ao dataframe
colnames(sivep) <- nomes_colunas

# CRIAR REGRAS DO SIVEP -----
#inconsistencia
regras_incon <- regras_incon |> as.data.frame() |> t() |> as.data.frame()
regras_incon <- cbind(regras_incon |> row.names(),regras_incon)
regras_incon |> row.names() <- NULL
regras_incon |> colnames() <- c('Variavel','Regra')
regras_incon$Variavel <- regras_incon$Variavel |> gsub(pattern = '_e_',
                                                         replacement = ' e ')

regras_incon$Indicador <- 'Inconsistência'
regras_incon <- regras_incon[-c(17,18),]

#implausibilidade
regras_implau <- regras_implau |> as.data.frame() |> t() |> as.data.frame()
regras_implau <- cbind(regras_implau |> row.names(),regras_implau)
regras_implau |> row.names() <- NULL
regras_implau |> colnames() <- c('Variavel','Regra')
regras_implau$Variavel <- regras_implau$Variavel |>
  gsub(pattern = '_IMPOSSIVEL', replacement = '')
regras_implau$Regra <- regras_implau$Regra |>
  gsub(pattern = 'de gestantes ', replacement = '')
regras_implau$Regra <- regras_implau$Regra |>

```

```

    gsub(pattern = 'Gestantes ', replacement = 'Gestantes e puérperas ')
regras_implau$Regra[4] <- 'Gestantes e puérperas ao mesmo tempo'
regras_implau$Indicador <- 'Implausibilidade'

#incompletude
regras_incom <- regras_incom |> as.data.frame() |> t() |> as.data.frame()
regras_incom <- cbind(regras_incom |> row.names(),regras_incom)
regras_incom |> row.names() <- NULL
regras_incom |> colnames() <- c('Variavel','Regra')
regras_incom$Indicador <- 'Incompletude'

#CORRECAO PARA CODIGO DO QUALIDADES
regras_sivep <- rbind(regras_incom,regras_implau,regras_incom)
for(i in seq_along(SIVEP_dic$`Codigo SIVEP`)) {
  for(j in 1:ncol(regras_sivep)){
    regras_sivep[,j] <- gsub(SIVEP_dic$`Codigo SIVEP`[i],
                           SIVEP_dic$`Codigo Qualidades`[i],
                           regras_sivep[,j])
  }
}

regras_sivep$Variavel <- regras_sivep$Variavel %>%
  gsub(pattern = '_IMPROVAVEL',replacement = '')

#DESCRICAO DOS INDICADORES
desc_incom <- 'análise das informações que estão faltando na base de dados,
seja porque não foram preenchidas ("dados em branco") ou porque a
resposta era desconhecida ("dados ignorados").'
desc_implau <- "análise das informações que são improváveis e/ou dificilmente
possam ser consideradas aceitáveis dadas as características de sua natureza."
desc_incon <- "informações que parecem ilógicas e/ou incompatíveis a
partir da análise da combinação dos dados informados em dois
ou mais campos do formulário."

var_sivep_incon <-
  regras_sivep[regras_sivep$Indicador=='Inconsistência','Variavel']
#VARIÁVEIS AUXILIARES PARA INCONSISTENCIA
Var_incon_relacao <- list(
  c('SEXO','IDADE_GEST'),
  c('FATOR_RISCO','CARDIOPATI', 'HEMATOLOGI', 'SIND_DOWN', 'HEPÁTICA',
    'ASMA', 'DIABETES',
    'NEUROLÓGICA', 'PNEUMOPATIA', 'IMUNODEPRESSAO', 'RENAL_CRON', 'OBESIDADE',
    'OBES_IMC', 'OUT_FATOR_RISCO'),
  c('VACINA','DT_VACINA_GRIPE'),
  c('MAE_VACINA', 'DT_VACINA_MAE' ),
  c('DT_DOSE_UNICA','IDADE'),
  c('ANTIVIRAL','TIPO_ANTIVIRAL'),
  c('INTERNACAO','DT_INTERNACAO'),
  c('UTI', 'DT_UTI', 'INTERNACAO'),
  c('RESULT_RAIOX', 'DT_RAIOX' ),
  c('AMOSTRA_DIAG', 'DT_COLETA_AMO' ),
  c('HIST_VIAGEM','LO_PS_VGM', 'DT_VGM', 'DT_RT_VGM'),
  c('RESULT_TOMOGR', 'DT_TOMOGRAFIA' ),

```

```

c('RES_AN' , 'TIPO_ANTIGENICO' , 'DT_RES_ANTIGENICO' ),
c('VACINA_COVID' , 'DOSE_1_COV' , 'DOSE_2_COV' ),
c('CLASSI_FIN' , 'PCR_INFLU' , 'ANTIGENICO_INFLU' ),
c('CLASSI_FIN' , 'PCR_OUTRO' , 'AN_OUTRO')
)
names(Var_incon_relacao) <-
  regras_sivep[regras_sivep$Indicador == 'Inconsistência','Variavel']
Var_incon_relacao <-
  Var_incon_relacao[var_sivep_incon] %>% unlist() %>% unname()
Var_incon_relacao <-
  Var_incon_relacao[Var_incon_relacao %in% colnames(sivep)]

#VARIÁVEIS PARA FILTRO
var_sivep_implau <-
  regras_sivep$Variavel[regras_sivep$Indicador == 'Implausibilidade'] %>%
  unique()
var_sivep_incom <-
  regras_sivep$Variavel[regras_sivep$Indicador == 'Incompletude'] %>%
  unique()
dados_oobr_qualitados_SIVEP_2009_2023 <- sivep
#DADOS
usethis::use_data(dados_oobr_qualitados_SIVEP_2009_2023,overwrite = T)
#VARIÁVEIS PARA FILTRO
usethis::use_data(Var_incon_relacao,overwrite = T)
usethis::use_data(var_sivep_incom,overwrite = T)
usethis::use_data(var_sivep_implau,overwrite = T)
usethis::use_data(var_sivep_incon,overwrite = T)
#DESCRICAO
usethis::use_data(desc_incom, overwrite = T)
usethis::use_data(desc_implau, overwrite = T)
usethis::use_data(desc_incon, overwrite = T)
#DICIONARIO
usethis::use_data(SIVEP_dic,overwrite = T)
usethis::use_data(regras_sivep,overwrite = T)

```

## Análise dos dados de caracterização

### Classificação caso de SRAG

A variável que indica a classificação é a CLASSI\_FIN, que possui as seguintes categorias: 1 - SRAG por influenza 2 - SRAG por outro vírus respiratório 3 - SRAG por outro agente etiológico 4 - SRAG não especificado 5 - SRAG por COVID-19

```

#tabela de frequência para a classificação
questionr::freq(
  dados_oobr_qualitados_SIVEP_2009_2023$CLASSI_FIN,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
  kable(caption = "Tabela de frequências para classificação do caso ",
        digits = 2)

```

Table 1: Tabela de frequências para classificação do caso

	n	%
1	6677	10.5
2	1383	2.2
3	3833	6.0
4	23290	36.5
5	26141	41.0
Em Branco	2451	3.8
Ignorado	43	0.1
Total	63818	100.0

É perceptível que a maior concentração de dados está nas categorias de COVID-19 e SRAG não especificado. Todos os totais de dados considerados ‘Em Branco’, ‘Implausíveis’ e outras categorias similares serão apresentados posteriormente no documento.

### Indicativo de Gestante ou Puérpera

Neste ponto, é realizada uma alteração nos dados para visualizar o trimestre gestacional e se a pessoa é puérpera ou não.

```
#tabela de frequência para a classificação
questionr::freq( dados_oobr_qualitados_SIVEP_2009_2023 |> mutate(
  classi_gesta_puerp = case_when(
    IDADE_GEST == '1' ~ "1tri",
    IDADE_GEST == '2' ~ "2tri",
    IDADE_GEST == '3' ~ "3tri",
    IDADE_GEST == '4' ~ "IG_ig",
    ( IDADE_GEST == '5' & PUERPERA == '1' ) ~ "puerp",
    (IDADE_GEST == '9' & PUERPERA == '1') ~ "puerp",
    TRUE ~ "não"
  )) |> select(classi_gesta_puerp),
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
kable(caption =
  "Tabela de frequências para classificação do trimestre gestacional ",
  digits = 2)
```

Table 2: Tabela de frequências para classificação do trimestre gestacional

	n	%
1tri	6817	10.7
2tri	14937	23.4
3tri	25801	40.4



	n	%
IG_ig	2240	3.5
não	3903	6.1
puerp	10120	15.9
Total	63818	100.0

É importante observar que as implausibilidades e incompletudes são classificadas como “NÃO” no contexto mencionado.

### Período Gestacional

A variável IDADE\_GEST representa o período gestacional e assume os seguintes valores: 1 - 1º Trimestre; 2 - 2º Trimestre; 3 - 3º Trimestre; 4 - Idade Gestacional Ignorada; 5 - Não; 6 - Não se aplica; Ignorado.

```
#tabela de frequência para gestação
questionr::freq(
  dados_oobr_qualidadados_SIVEP_2009_2023$IDADE_GEST,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
kable(caption = "Tabela de frequências para variável
sobre gestação", digits = 2)
```

Table 3: Tabela de frequências para variável sobre gestação

	n	%
0	1	0.0
1	6817	10.7
2	14937	23.4
3	25801	40.4
4	2240	3.5
5	10120	15.9
6	1193	1.9
Ignorado	680	1.1
Impossível	2029	3.2
Total	63818	100.0

Neste caso, os dados em que a variável IDADE\_GEST assume os valores 1, 2, 3 ou 4, e a variável PUERPERA assume o valor 1 - É puérpera, são classificados como “Impossíveis”.

### Sexo

O Paineiro se limita aos dados em que o indivíduo observado foi classificado como gestante ou puérpera. Portanto, qualquer dado que indique “M” - Homem é considerado impossível. No total, existem 11 observações com essa classificação.

```
#tabela de frequência para sexo
questionr::freq(
  dados_oobr_qualitados_SIVEP_2009_2023$SEXO,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
  kable(caption = "Tabela de frequências para sexo", digits = 2)
```

Table 4: Tabela de frequências para sexo

	n	%
F	63332	99.2
Impossivel	486	0.8
Total	63818	100.0

## Idade

A variável IDADE representa a idade do indivíduo como um valor numérico. Nesse contexto, os dados cujos valores sejam maiores que 55 ou menores que 10 são classificados como implausíveis, sendo considerados como impossíveis ou improváveis.

```
#tabela de frequência para gestação
questionr::freq(
  dados_oobr_qualitados_SIVEP_2009_2023$IDADE,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
  kable(caption = "Tabela de frequências para variável
idade", digits = 2)
```

Table 5: Tabela de frequências para variável idade

	n	%
10	19	0.0
11	21	0.0
12	32	0.1
13	76	0.1
14	243	0.4
15	504	0.8
16	761	1.2
17	1030	1.6
18	1230	1.9
19	1568	2.5
20	1715	2.7
21	1932	3.0
22	2036	3.2

	n	%
23	2201	3.4
24	2192	3.4
25	2270	3.6
26	2238	3.5
27	2266	3.6
28	2238	3.5
29	2138	3.4
30	2152	3.4
31	2163	3.4
32	1990	3.1
33	1914	3.0
34	1869	2.9
35	1849	2.9
36	1629	2.6
37	1485	2.3
38	1321	2.1
39	1180	1.8
40	904	1.4
41	612	1.0
42	472	0.7
43	312	0.5
44	247	0.4
45	147	0.2
46	83	0.1
47	73	0.1
48	63	0.1
49	53	0.1
50	42	0.1
51	41	0.1
52	44	0.1
53	54	0.1
54	65	0.1
55	56	0.1
Impossivel	14592	22.9
Improvavel	1696	2.7
Total	63818	100.0

## Raça

A variável RACA representa a raça do indivíduo e possui as seguintes categorias: 1 - Branca; 2 - Preta; 3 - Amarela; 4 - Parda; 5 - Indígena; Ignorado

```
#tabela de frequência para gestação
questionr::freq(
  dados_oobr_qualitados_SIVEP_2009_2023$RACA,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
  kable(caption = "Tabela de frequências para variável
```

```
Raça/Cor", digits = 2)
```

Table 6: Tabela de frequências para variável Raça/Cor

	n	%
1	25508	40.0
2	3836	6.0
3	485	0.8
4	25925	40.6
5	333	0.5
Em Branco	956	1.5
Ignorado	6775	10.6
Total	63818	100.0

Os dados da população apresentam uma predominância majoritária nas categorias de raça “Branca” e “Parda”.

### UF de Notificação

A variável SG\_UF\_NOT representa a Unidade Federativa (UF) do estado de notificação do caso de SRAG. Ela assume diferentes valores correspondentes aos estados do Brasil.

```
#tabela de frequência para gestação
questionr::freq(
  dados_oobr_qualitados_SIVEP_2009_2023$SG_UF_NOT,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
  kable(caption = "Tabela de frequências para variável
    UF de notificação", digits = 2)
```

Table 7: Tabela de frequências para variável UF de notificação

	n	%
AC	265	0.4
AL	767	1.2
AM	1561	2.4
AP	301	0.5
BA	1816	2.8
CE	3200	5.0
DF	1804	2.8
ES	490	0.8
GO	1770	2.8
MA	779	1.2
MG	5447	8.5
MS	1504	2.4
MT	1316	2.1

	n	%
PA	2140	3.4
PB	1907	3.0
PE	1660	2.6
PI	890	1.4
PR	7135	11.2
RJ	4602	7.2
RN	655	1.0
RO	604	0.9
RR	108	0.2
RS	2876	4.5
SC	2781	4.4
SE	403	0.6
SP	16609	26.0
TO	428	0.7
Total	63818	100.0

## Escolaridade

A variável ESCOLARIDADE representa o nível de escolaridade do paciente e possui as seguintes categorias: 0 - Sem escolaridade/Analfabeto; 1 - Fundamental 1º ciclo, 1ª a 5ª série; 2 - Fundamental 2º ciclo, 6ª a 9ª série; 3 - Médio, 1º ao 3º ano; 4 - Superior; 5 - Não se aplica; Ignorado. Para os níveis de escolaridade fundamental e médio, deve-se considerar a última série ou ano concluído.

```
#tabela de frequência para gestação
questionr::freq(
  dados_oobr_qualidadados_SIVEP_2009_2023$ESCOLARIDADE,
  cum = FALSE,
  total = TRUE,
  na.last = FALSE,
  valid = FALSE
) %>%
kable(caption = "Tabela de frequências para variável
Escolaridade", digits = 2)
```

Table 8: Tabela de frequências para variável Escolaridade

	n	%
0	300	0.5
0.0	52	0.1
1	3776	5.9
10.0	161	0.3
2	7062	11.1
3	12969	20.3
4	3756	5.9
5	849	1.3
6	1316	2.1
7	192	0.3
8	315	0.5
Em Branco	16273	25.5
Ignorado	16797	26.3

	n	%
Total	63818	100.0

Entre as observações, o índice de ensino médio concluído é o de maior frequência. Isso significa que a categoria correspondente ao nível de escolaridade “Médio, 1º ao 3º ano” é a mais comum.

## Análise dos Indicadores

Segue abaixo a frequência para cada variável e cada indicador apresentado no painel. As regras de decisão podem ser observadas na aba de tratamento ou no próprio dicionário do painel.

### Incompletude

As regras aqui utilizadas são apresentadas no dicionário do painel, mostramos abaixo a frequência relativa a cada variável do painel apresentando alguma das incompletudes. Lembrando que o sivep possui um total de 35792 observações. Essas frequências representam a porcentagem de ocorrência de cada valor em relação ao total de observações para cada variável.

```
tabela_resultados <- data.frame(Variavel = character(),
                                Ignorado = numeric(),
                                `Em Branco` = numeric(),
                                `Porcentagem Incompletude` = character(),
                                row.names = NULL, stringsAsFactors = FALSE)

# Iteração sobre as colunas do dataframe original
for (col in colnames(dados_oobr_qualitados_SIVEP_2009_2023)) {
  # Contagem dos casos "Ignorado" e "Em Branco"
  contagem_ignorado <- sum(dados_oobr_qualitados_SIVEP_2009_2023[[col]] == "Ignorado", na.rm = TRUE)
  contagem_em_branco <- sum(dados_oobr_qualitados_SIVEP_2009_2023[[col]] == "Em Branco", na.rm = TRUE)
  porc <- (contagem_ignorado + contagem_em_branco)/length(dados_oobr_qualitados_SIVEP_2009_2023[[col]])
  # Adição dos resultados à tabela

  tabela_resultados <- rbind(tabela_resultados,
                              data.frame(Variavel = col,
                                          Ignorado = contagem_ignorado,
                                          `Em Branco` = contagem_em_branco,
                                          `Porcentagem Incompletude` = paste0(
                                            round(porc*100,2), '%'))
  }
tabela_resultados |> kable()
```

Variavel	Ignorado	Em.Branco	Porcentagem.Incompletude
SEXO	0	0	0%
IDADE	0	0	0%
TIPO_IDADE	0	14472	22.68%
RACA	6775	956	12.11%
ESCOLARIDADE	16797	16273	51.82%
ZONA	408	18456	29.56%
SURTO_SG	1561	48684	78.73%
SRAG_POS	3092	20715	37.3%

Variavel	Ignorado	Em.Branco	Porcentagem.Incompletude
AVES_SUINOS	6605	20902	43.1%
FEBRE	479	7338	12.25%
TOSSE	381	5252	8.83%
GARGANTA	1077	11002	18.93%
DISPENEIA	458	7717	12.81%
DESC_RESPIRATORIO	617	17125	27.8%
SATURAÇÃO	1009	17941	29.69%
DIARREIA	878	19330	31.67%
VOMITO	711	26574	42.75%
OUTRO_SINT	1322	12584	21.79%
FATOR_RISCO	0	15837	24.82%
CARDIOPATI	895	30508	49.21%
HEMATOLOGI	356	44980	71.04%
SIND_DOWN	566	38656	61.46%
HEPÁTICA	632	38704	61.64%
ASMA	319	44496	70.22%
DIABETES	307	44146	69.66%
NEUROLÓGICA	610	38621	61.47%
PNEUMOPATIA	913	31291	50.46%
IMUNODEPRESSAO	933	31339	50.57%
RENAL_CRON	914	31500	50.79%
OBESIDADE	673	38244	60.98%
OUT_FATOR_RISCO	767	28085	45.21%
MAE_VACINA	16	63774	99.96%
MAE_AMAMENTA	12	63782	99.96%
ANTIVIRAL	4912	14033	29.69%
TIPO_ANTIVIRAL	0	58722	92.01%
INTERNACAO	336	1051	2.17%
DT_INTERNACAO	0	2039	3.2%
UTI	610	13836	22.64%
SUPORT_VENT	1326	13693	23.53%
AMOSTRA_DIAG	143	9426	14.99%
DT_COLETA_AMO	0	8550	13.4%
RT-PCR_INFLU	1341	52485	84.34%
RT-PCR_OUTRO	34	48053	75.35%
EVOLUCAO	1860	5275	11.18%
HIST_VIAGEM	0	14473	22.68%
DOR_ABD	704	33399	53.44%
FADIGA	684	32662	52.25%
PERDA_OLFT	870	33035	53.13%
PERDA_PALADAR	886	33104	53.26%
ANTIGENICO_INFLU	534	58668	92.77%
ANTIGENICO_OUTRO	62	57396	90.03%
IDADE_GEST	680	0	1.07%
DT_VACINA_GRIPE	0	15157	23.75%
DT_VACINA_MAE	0	15845	24.83%
DT_DOSE_UNICA	0	15845	24.83%
DT_UTI	0	13050	20.45%
RESULT_RAIOX	3867	22287	40.98%
DT_RAIOX	0	6719	10.53%
RESULT_TOMOGR	1829	37203	61.16%
DT_TOMOGRFIA	0	15845	24.83%

Variavel	Ignorado	Em.Branco	Porcentagem.Incompletude
TIPO_ANTIGENICO	0	45991	72.07%
DT_RESULT_ANTTIGENICO	0	15844	24.83%
VACINA_COVID	3843	35410	61.51%
VACINA	16988	12279	45.86%
TIPO_AMOSTRA	531	59872	94.65%
PUERPERA	309	34412	54.41%
DOSE1_COVID	0	30732	48.16%
DOSE2_COVID	0	30732	48.16%
LOCAL_VIAGEM	0	27464	43.03%
DT_VIAGEM	0	27464	43.03%
DT_RETORNO_VIAGEM	0	27464	43.03%
RESULT_ANT	262	21887	34.71%
ANO	0	0	0%
MUNICIPIO	0	0	0%
SG_UF_NOT	0	0	0%
CLASSI_FIN	43	2451	3.91%
SEXO e IDADE_GEST	0	0	0%
FATOR_RISCO e COMORBIDADES	0	0	0%
VACINA e DT_VACINA_GRIPE	0	0	0%
MAE_VACINA e DT_VACINA_MAE	0	0	0%
DT_DOSE_UNICA e IDADE	0	0	0%
ANTIVIRAL e TIPO_ANTIVIRAL	0	0	0%
INTERNACAO e DT_INTERNACAO	0	0	0%
UTI e DT_UTI	0	0	0%
RESULT_RAIOX e DT_RAIOX	0	0	0%
AMOSTRA_DIAG e DT_COLETA_AMO	0	0	0%
HIST_VIAGEM e Campos_VGMs	0	0	0%
RESULT_TOMOGR e DT_TOMOGRAFIA	0	0	0%
TIPO_ANTIGENICO e	0	0	0%
DT_RESULT_ANTTIGENICO			
VACINA_COVID e DOSES	0	0	0%
CLASSI_FIN_SRAG_INFLUENZA	0	0	0%
CLASSI_FIN_SRAG_OUTROS_VIRUS	0	0	0%
TIPO_ANTIGENICO e	0	0	0%
DT_RES_ANTIGENICO			

## Implausibilidade

```
tabela_resultados <- data.frame(Variavel = character(),
                                Implausivel = numeric(),
                                `Impossivel` = numeric(),
                                `Porcentagem Implausibilidade` = character(),
                                row.names = NULL, stringsAsFactors = FALSE)

# Iteração sobre as colunas do dataframe original
for (col in colnames(dados_oobr_qualitados_SIVEP_2009_2023)) {
  # Contagem dos casos "Ignorado" e "Em Branco"
  contagem_Implausivel <- sum(dados_oobr_qualitados_SIVEP_2009_2023[[col]] == "Improvavel", na.rm = TRUE)
  contagem_Impossivel <- sum(dados_oobr_qualitados_SIVEP_2009_2023[[col]] == "Impossivel", na.rm = TRUE)
  porc <- (contagem_Implausivel + contagem_Impossivel)/length(dados_oobr_qualitados_SIVEP_2009_2023[[col]])
  # Adição dos resultados à tabela
}
```



```
tabela_resultados <- rbind(tabela_resultados,
                           data.frame(Variavel = col,
                                       Implausivel = contagem_Implausivel,
                                       Impossivel = contagem_Impossivel,
                                       `Porcentagem Implausibilidade` = paste0(
                                         round(porc*100,2), '%')))}

tabela_resultados |> kable()
```

Variavel	Implausivel	Impossivel	Porcentagem.Implausibilidade
SEXO	0	486	0.76%
IDADE	1696	14592	25.52%
TIPO_IDADE	0	0	0%
RACA	0	0	0%
ESCOLARIDADE	0	0	0%
ZONA	0	0	0%
SURTO_SG	0	0	0%
SRAG_POS	0	0	0%
AVES_SUINOS	0	51	0.08%
FEBRE	0	0	0%
TOSSE	0	0	0%
GARGANTA	0	0	0%
DISPNEIA	0	0	0%
DESC_RESPIRATORIO	0	0	0%
SATURAÇÃO	0	0	0%
DIARREIA	0	0	0%
VOMITO	0	0	0%
OUTRO_SINT	0	0	0%
FATOR_RISCO	0	14895	23.34%
CARDIOPATI	0	0	0%
HEMATOLOGI	0	0	0%
SIND_DOWN	0	0	0%
HEPÁTICA	0	0	0%
ASMA	0	0	0%
DIABETES	0	0	0%
NEUROLÓGICA	0	0	0%
PNEUMOPATIA	0	0	0%
IMUNODEPRESSAO	0	1033	1.62%
RENAL_CRON	0	0	0%
OBESIDADE	0	0	0%
OUT_FATOR_RISCO	0	0	0%
MAE_VACINA	0	0	0%
MAE_AMAMENTA	0	0	0%
ANTIVIRAL	0	33	0.05%
TIPO_ANTIVIRAL	0	0	0%
INTERNACAO	0	0	0%
DT_INTERNACAO	0	12453	19.51%
UTI	0	0	0%
SUPPORT_VENT	0	0	0%
AMOSTRA_DIAG	0	187	0.29%
DT_COLETA_AMO	0	6002	9.4%
RT-PCR_INFLU	0	0	0%

Variavel	Implausivel	Impossivel	Porcentagem.Implausibilidade
RT-PCR_OUTRO	0	0	0%
EVOLUCAO	0	0	0%
HIST_VIAGEM	0	38802	60.8%
DOR_ABD	0	0	0%
FADIGA	0	0	0%
PERDA_OLFT	0	0	0%
PERDA_PALADAR	0	1	0%
ANTIGENICO_INFLU	0	0	0%
ANTIGENICO_OUTRO	0	0	0%
IDADE_GEST	0	2029	3.18%
DT_VACINA_GRIPE	0	0	0%
DT_VACINA_MAE	0	0	0%
DT_DOSE_UNICA	0	0	0%
DT_UTI	0	0	0%
RESULT_RAIOX	0	0	0%
DT_RAIOX	0	0	0%
RESULT_TOMOGR	0	0	0%
DT_TOMOGRAFIA	0	0	0%
TIPO_ANTIGENICO	0	0	0%
DT_RESULT_ANTTIGENICO	0	0	0%
VACINA_COVID	0	0	0%
VACINA	0	0	0%
TIPO_AMOSTRA	0	0	0%
PUERPERA	0	2029	3.18%
DOSE1_COVID	0	0	0%
DOSE2_COVID	0	0	0%
LOCAL_VIAGEM	0	0	0%
DT_VIAGEM	0	0	0%
DT_RETORNO_VIAGEM	0	0	0%
RESULT_ANT	0	0	0%
ANO	0	0	0%
MUNICIPIO	0	0	0%
SG_UF_NOT	0	0	0%
CLASSI_FIN	0	0	0%
SEXO e IDADE_GEST	0	0	0%
FATOR_RISCO e COMORBIDADES	0	0	0%
VACINA e DT_VACINA_GRIPE	0	0	0%
MAE_VACINA e DT_VACINA_MAE	0	0	0%
DT_DOSE_UNICA e IDADE	0	0	0%
ANTIVIRAL e TIPO_ANTIVIRAL	0	0	0%
INTERNACAO e DT_INTERNACAO	0	0	0%
UTI e DT_UTI	0	0	0%
RESULT_RAIOX e DT_RAIOX	0	0	0%
AMOSTRA_DIAG e	0	0	0%
DT_COLETA_AMO			
HIST_VIAGEM e Campos_VGMs	0	0	0%
RESULT_TOMOGR e	0	0	0%
DT_TOMOGRAFIA			
TIPO_ANTIGENICO e	0	0	0%
DT_RESULT_ANTTIGENICO			
VACINA_COVID e DOSES	0	0	0%
CLASSI_FIN_SRAG_INFLUENZA	0	0	0%

Variavel	Implausivel	Impossivel	Porcentagem.Implausibilidade
CLASSI_FIN_SRAG_OUTROS_VIRUS	0	0	0%
TIPO_ANTIGENICO e	0	0	0%
DT_RES_ANTIGENICO			

## Inconsistência

As regras utilizadas para identificar as inconsistências no banco de dados podem ser visualizadas na aba de dicionário do painel. Nessa seção, é possível encontrar as informações detalhadas sobre as regras adotadas para determinar as inconsistências nos dados. Recomenda-se consultar essa aba para obter mais detalhes.

```
tabela_resultados <- data.frame(Variavel = character(),
                                `Inconsistência` = numeric(),
                                `Porcentagem Inconsistência` = character(),
                                row.names = NULL, stringsAsFactors = FALSE)
# Iteração sobre as colunas do dataframe original
for (col in colnames(dados_oobr_qualitados_SIVEP_2009_2023)) {
  # Contagem dos casos "Ignorado" e "Em Branco"
  contagem <- sum(dados_oobr_qualitados_SIVEP_2009_2023[[col]] == "Inconsistencia", na.rm = TRUE)
  porc <- (contagem)/length(dados_oobr_qualitados_SIVEP_2009_2023[[col]])
  # Adição dos resultados à tabela
  if (contagem > 0) {
    tabela_resultados <- rbind(tabela_resultados,
                                data.frame(Variavel = col,
                                              `Inconsistência` = contagem,
                                              `Porcentagem Implausibilidade` = paste0(
                                                round(porc*100,2), '%'))))
  }
}
tabela_resultados |> kable()
```

Variavel	Inconsistência	Porcentagem.Implausibilidade
SEXO e IDADE_GEST	13	0.02%
FATOR_RISCO e COMORBIDADES	5445	8.53%
VACINA e DT_VACINA_GRIPE	28790	45.11%
MAE_VACINA e DT_VACINA_MAE	13	0.02%
DT_DOSE_UNICA e IDADE	46642	73.09%
INTERNACAO e DT_INTERNACAO	1647	2.58%
UTI e DT_UTI	32206	50.47%
RESULT_RAIOX e DT_RAIOX	23140	36.26%
AMOSTRA_DIAG e DT_COLETA_AMO	7622	11.94%
HIST_VIAGEM e Campos_VGMs	11942	18.71%
RESULT_TOMOGR e DT_TOMOGRAFIA	18473	28.95%
TIPO_ANTIGENICO e	12081	18.93%
DT_RESULT_ANTTIGENICO		
VACINA_COVID e DOSES	15184	23.79%

## SINASC

O Sistema de Informações sobre Nascidos Vivos (SINASC) foi oficialmente implantado a partir de 1990, com o propósito de coletar dados sobre os nascimentos ocorridos em todo o território nacional, fornecendo

informações relevantes sobre a natalidade para todos os níveis do Sistema de Saúde.

O SINASC é gerenciado pelo Ministério da Saúde em parceria com as Secretarias Estaduais e Municipais de Saúde. Seu objetivo principal é subsidiar a formulação, implementação e avaliação de políticas públicas relacionadas à saúde materno-infantil.

## Extração

Para a base de dados do SINASC, assim como para a base de dados do SIVEP-GRIPE e do SIM (Sistema de Informações sobre Mortalidade), a extração dos dados foi realizada por meio da API disponibilizada pela PCDas (Plataforma de Ciência de Dados Aplicada à Saúde) abrangendo os anos de 1996 a 2021, e pelo Open datatus para os dados preliminares referentes a 2022.

Durante a extração dos dados por meio da API, os mesmos são devidamente filtrados, tratados e subdivididos em três bases distintas. Essa subdivisão ocorre devido ao tamanho excessivo do arquivo completo, buscando otimizar o processamento e análise dos dados. Cada uma das três bases corresponde a um dos indicadores trabalhados no painel.

As bases finais resultantes contêm informações sobre o número de casos dos indicadores e o total de observações, agrupados por município-UF, ano e variável em questão. Essa organização permite uma visualização e análise mais eficiente dos dados, facilitando a compreensão dos padrões e tendências relacionados aos indicadores monitorados no painel.

```
# INCOMPLETEDE #####
```

```
import glob
import pandas as pd
from collections import Counter
import datetime as dt

import warnings
warnings.filterwarnings("ignore")

regras_ignorados = {}
regras_ignorados['LOCNASC'] = [9]
regras_ignorados['ESTCIVMAE'] = [9]
regras_ignorados['ESMAE'] = [9]
regras_ignorados['GESTACAO'] = [9]
regras_ignorados['GRAVIDEZ'] = [9]
regras_ignorados['PARTO'] = [9]
regras_ignorados['CONSULTAS'] = [9]
regras_ignorados['CONSULTAS'] = [9]
regras_ignorados['SEXO'] = [0, 9, 'I']
regras_ignorados['RACACOR'] = [9]
regras_ignorados['IDANOMAL'] = [8,9]
regras_ignorados['ESMAE2010'] = [9]
regras_ignorados['TPMETESTIM'] = [8,9]
regras_ignorados['TPMETESTIM'] = [99]
regras_ignorados['TPAPRESENT'] = [9]
regras_ignorados['STRABPART'] = [9]
regras_ignorados['STCESPARTO'] = [9]
regras_ignorados['TPNASCASSI'] = [9]
regras_ignorados['TPFUNCRESP'] = [0]
regras_ignorados['ESMAEAGR1'] = [9]
regras_ignorados['TPROBSON'] = [11,12]
```

```

regras_ignorados['IDADEMAE'] = [99]
regras_ignorados['PESO'] = [9999]

for f in glob.glob('SINASC_dataset/*.csv'):

    df = pd.read_csv(f)
    print(len(df))
    ano = f.split('/')[1].split('_')[3].split('.')[0]
    codmun = df['CODMUNNASC']
    estado = f.split('/')[1].split('_')[2]

    df_ignorados = df.copy()
    df_totais = df.isna()
    df_nulos = df_totais.copy()

    df_totais['ANO'] = ano
    df_totais['CODMUNNASC'] = codmun

    df_totais = df_totais.groupby(['ANO', 'CODMUNNASC']) \
        .count() \
        .reset_index() \
        .melt(id_vars=['ANO', 'CODMUNNASC'])

    df_totais.columns = ['ANO', 'CODMUNNASC', 'VARIABEL', 'TOTAIS']

    df_nulos['CODMUNNASC'] = codmun
    df_nulos['ANO'] = ano

    df_nulos = df_nulos.groupby(['ANO', 'CODMUNNASC']).sum() \
        .reset_index() \
        .melt(id_vars=['ANO', 'CODMUNNASC'])
    df_nulos.columns = ['ANO', 'CODMUNNASC', 'VARIABEL', 'NULOS']

    for c in df_ignorados.columns:
        if c in regras_ignorados:
            df_ignorados[c] = df_ignorados[c].isin(regras_ignorados[c])
        else:
            if c not in ['ANO', 'CODMUNNASC']:
                df_ignorados.drop(columns=[c], inplace=True)

    df_ignorados['CODMUNNASC'] = codmun
    df_ignorados['ANO'] = ano

    df_ignorados = df_ignorados.groupby(['ANO', 'CODMUNNASC']) \
        .sum() \
        .reset_index().melt(id_vars=['ANO', 'CODMUNNASC'])
    df_ignorados.columns = ['ANO', 'CODMUNNASC', 'VARIABEL', 'IGNORADOS']

    df_ignorados = df_ignorados.fillna(0)

    x = df_totais.merge(df_nulos, how='left', on=['ANO', 'CODMUNNASC', 'VARIABEL'])

```

```

x = x.merge(df_ignorados, how='left', on=['ANO', 'CODMUNNASC', 'VARIABEL'])
x = x.reset_index()

x = x[['ANO', 'CODMUNNASC', 'VARIABEL', 'NULOS', 'IGNORADOS', 'TOTAIS']]

x = x.fillna(0)

x.to_csv('SINASC_dataset/resultados/Incompletude_{ }_{ }.csv' \
        .format(estado, ano),
        index=None, compression='gzip')

incompletude = pd.DataFrame()

for f in glob.glob('SINASC_dataset/resultados/Incompletude_*.csv'):
    df = pd.read_csv(f, compression='gzip')
    incompletude = pd.concat([incompletude, df], axis=0)

incompletude.fillna(0, inplace=True)
incompletude = incompletude[~incompletude.VARIABEL.isin(['contador', 'NOVO'])]
incompletude.to_csv('SINASC_Incompletude_v2.csv', index=None, compression='gzip')

# gera regras

import json

regras = {}
for r in regras_ignorados:
    regras["IGNORADOS_" + r] = "Se o campo " + r + \
        " estiver preenchido com " + str(regras_ignorados[r])

with open('SINASC_Incompletude_Regras.json', 'w') as fp:
    json.dump(regras, fp, indent=4)

#IMPLAUSIBILIDADE

colunas_implausibilidade = ['ANO', 'ESTADO', 'CODMUNNASC', 'LOCNASC',
                             'IDADEMAE', 'ESTCIVMAE', 'ESMAE', 'QTDFILVIVO',
                             'QTDFILMORT', 'GESTACAO', 'GRAVIDEZ', 'PARTO',
                             'CONSULTAS', 'DTNASC', 'HORANASC', 'SEXO',
                             'APGAR1', 'APGAR5', 'RACACOR', 'PESO', 'IDANOMAL',
                             'DTCADASTRO', 'CODANOMAL', 'ESMAE2010', 'DTNASCMAE',
                             'QTDGESTANT', 'QTDPARTNOR', 'QTDPARTCES', 'IDADEPAI',
                             'DTULTMENST', 'SEMAGESTAC', 'TPMETESTIM', 'TPAPRESENT',
                             'STTRABPART', 'STCESPARTO', 'TPNASCASSI', 'TPFUNCRESP',
                             'TPDOCRESP', 'TPROBSON', 'SERIESMAE', 'CONSPRENAT',
                             'MESPRENAT', 'ESMAEAGR1', 'PARIDADE']

# aplica as regras para variaveis com opcoes
regras_gerais = { 'LOCNASC': [1,2,3,4,5,9],
                  'ESTCIVMAE': [1,2,3,4,5,9],
                  'ESMAE': [1,2,3,4,5,9],
                  'GESTACAO': [1,2,3,4,5,6,9],
                  'GRAVIDEZ': [1,2,3,9],

```

```

        'PARTO': [1,2,9],
        'CONSULTAS': [1,2,3,4,9],
        'SEXO': [1,2,9,0, 'M', 'F', 'I'],
        'RACACOR': [1,2,3,4,5],
        'IDANOMAL': [1,2,9],
        'ESMAE2010': [1,2,3,4,5,9],
        'TPMETESTIM': [1,2,9],
        'TPAPRESENT': [1,2,3,9],
        'STTRABPART': [1,2,3,9],
        'STCESPARTO': [1,2,3,9],
        'TPNASCASSI': [1,2,3,4,9],
        'TPFUNCRESP': [1,2,3,4,5,9],
        'TPDOCRESPI': [1,2,3,4,5],
        'TPROBSON': list(range(1,13)), # 1 a 12
        'SERIESMAE': list(range(1,9)), # 1 a 8
        'MESPRENAT': list(range(1,11)) + [99], # 1 a 10 e 99
        'ESMAEAGR1': list(range(1,13)), # 1 a 12,
    }

# implausibilidade #####
for f in glob.glob('SINASC_dataset/*.csv'):

    df = pd.read_csv(f)
    ano = f.split('/')[1].split('_')[3].split('.')[0]
    estado = f.split('/')[1].split('_')[2]
    codmun = df['CODMUNNASC']

    aux_cols = []
    for c in colunas_implausibilidade:
        if c in df.columns:
            aux_cols.append(c)

    aux = df[aux_cols]

    aux['ANO'] = ano
    aux['CODMUNNASC'] = codmun

    print(ano, estado)

    for col in regras_gerais.keys():
        if col in aux_cols:
            aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
                (~aux[col].isin(regras_gerais[col]))

# REGRAS ESPECÍFICAS

col = 'IDADEMAE'
if col in aux_cols:
    aux[col] = pd.to_numeric(aux[col], errors='coerce')

    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
        ((aux[col] < 10) | (aux[col] > 55))

```

```

for col in ['QTDFILVIVO', 'QTDFILMORT']:
    if col not in aux_cols:
        continue
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
        ((aux[col] < 0) | (aux[col] > 70))

col = 'PESO'
if col in aux_cols:
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] < 0) | \
        (aux[col] > 11000))

for col in ['QTDGESTANT', 'QTDPARTNOR', 'QTDPARTCES']:
    if col not in aux_cols:
        continue
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] < 0) | \
        (aux[col] > 27))

col = 'IDADEPAI'
if col in aux_cols:
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] < 10) | \
        (aux[col] > 99))

col = 'SEMAGESTAC'
if col in aux_cols:
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & (aux[col] < 20)

col = 'CONSPRENAT'
if col in aux_cols:
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & (aux[col] < 0)

for col in ['DTNASC', 'DTCADASTRO']:
    if col in aux_cols:
        aux[col] = pd.to_numeric(df[col].astype(str).str[-4:],
                                errors='coerce')
        aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
            (aux[col] > dt.date.today().year)

col = 'DTNASCMAE'
if col in aux_cols:
    aux[col] = pd.to_numeric(df[col].astype(str).str[-4:],
                            errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] > 2012) | \
        (aux[col] < 1967))

col = 'HORANASC'
if col in aux_cols:
    if df[col].dtype == "object":
        df[col] = df[col].str.replace(";", "")

```



```

        df[col] = pd.to_numeric(df[col], errors='coerce')
        hora = pd.to_numeric(df[col], errors='coerce') // 100
        minuto = pd.to_numeric(df[col], errors='coerce') % 100
        aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & (df[col] > 59) & \
            (hora > 23) & (minuto > 59) # 00:59 vira 59 só

for col in ['APGAR1', 'APGAR5']:
    if col in aux_cols:
        aux[col] = pd.to_numeric(aux[col], errors='coerce')
        aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
            ((aux[col] < 0) | (aux[col] > 10))

col = 'PARIDADE'
if col in aux_cols:
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
        ((aux[col] < 0) | (aux[col] > 27))

aux_cols = []

for c in aux.columns:
    if 'IMPLAUSIVEL' in c:
        aux_cols.append(c)

aux_cols = ['ANO', 'ESTADO', 'CODMUNNASC'] + aux_cols

df_implausiveis = aux[aux_cols]

df_implausiveis.fillna(0, inplace=True)

df_implausiveis = df_implausiveis.groupby(['ANO', 'CODMUNNASC']) \
    .sum() \
    .reset_index() \
    .melt(id_vars=['ANO', 'CODMUNNASC'])
df_implausiveis.columns = ['ANO', 'CODMUNNASC', 'VARIABEL', 'IMPLAUSIVEIS']

df['ANO'] = ano
df['CODMUNNASC'] = codmun

df_totais = df[['ANO', 'CODMUNNASC']]
df_totais['TOTAIS'] = 1

df_totais = df_totais.groupby(['ANO', 'CODMUNNASC'])['TOTAIS'] \
    .sum().reset_index()
df_totais.columns = ['ANO', 'CODMUNNASC', 'TOTAIS']

df_totais.set_index(['ANO', 'CODMUNNASC'], inplace=True)
df_implausiveis.set_index(['ANO', 'CODMUNNASC'], inplace=True)

x = df_totais.join(df_implausiveis, how='left')
x = x.reset_index()

x = x[['ANO', 'CODMUNNASC', 'VARIABEL', 'IMPLAUSIVEIS', 'TOTAIS']]

```

```

x = x.fillna(0)

x.to_csv('SINASC_dataset/resultados/Implausiabilidade_{}_{}.csv' \
        .format(estado, ano),
        index=None, compression='gzip')
implausibilidade = pd.DataFrame()

for f in glob.glob('SINASC_dataset/resultados/Implausiabilidade_*.csv'):
    df = pd.read_csv(f, compression='gzip')
    implausibilidade = pd.concat([implausibilidade, df], axis=0)

implausibilidade.fillna(0, inplace=True)
implausibilidade.to_csv('SINASC_Implausiabilidade_v2.csv', index=None,
                        compression='gzip')

# gera regras

import json

regras = {}
regras["IDADEMAE"] = "Se campo IDADEMAE for menor que 10 ou maior que 55"
regras["QTDFILVIVO"] = "Se campo QTDFILVIVO for menor que 0 ou maior que 70"
regras["QTDFILMORT"] = "Se campo QTDFILMORT for menor que 0 ou maior que 70"
regras["PESO"] = "Se campo PESO for menor que 0 ou maior que 11000"
regras["QTDGESTANT"] = "Se campo QTDGESTANT for menor que 0 ou maior que 27"
regras["QTDPARTNOR"] = "Se campo QTDPARTNOR for menor que 0 ou maior que 27"
regras["QTDPARTCES"] = "Se campo QTDPARTCES for menor que 0 ou maior que 27"
regras["IDADEPAI"] = "Se campo IDADEPAI for menor que 10 ou maior que 99"
regras["SEMAGESTAC"] = "Se campo SEMAGESTAC for menor que 20"
regras["CONSPRENAT"] = "Se campo SEMAGESTAC for menor que 0"
regras["DTNASC"] = "Se campo DTNASC for maior que a data da última atualização \
                    dos dados"

regras["DTCADASTRO"] = "Se campo DTCADASTRO for menor que a data \
                        da última atualização dos dados"

regras["DTNASCMAE"] = "Se campo DTNASCMAE for menor que 1967 ou maior que 2012"
regras["HORANASC"] = "Se campo HORANASC não for uma hora válida"
regras["APGAR1"] = "Se campo APGAR1 for menor que 0 ou maior que 10"
regras["APGAR5"] = "Se campo APGAR5 for menor que 0 ou maior que 10"
regras["PARIDADE"] = "Se campo APGAR5 for menor que 0 ou maior que 27"

for k in regras_gerais.keys():
    if k not in regras.keys():
        regras[k] = "Se o campo " + k + "não for preenchido com " + \
                    str(regras_gerais[k])

with open('SINASC_Implausiabilidade_Regras.json', 'w') as fp:
    json.dump(regras, fp, indent=4)

#INCONSISTENCIA #####

regras = {}

```

```

regras['LOCNASC_e_PARTO'] = "Se campo LOCNASC for 2,3,4,5 e o \
                             campo PARTO estiver preenchido com 2"
regras['PARTO_e_STCESPARTO'] = "Se o campo STCESPARTO estiver preenchido \
                                como 1 e o campo PARTO estiver como 2 ou 9"
regras['TPROBSON_e_composicao'] = "Se o campo TPROBSON estiver preenchido \
                                  entre 1 e 10 e qualquer um dos \
                                  campos QTDGESTANT,QTDPARTNOR, \
                                  QTDPARTCES,SEMAGESTAC,TPAPRESENT, \
                                  STTRABPART estiverem em branco"

# inconsistencia
for f in glob.glob('SINASC_dataset/*.csv'):

    df = pd.read_csv(f)

    ano = f.split('/')[1].split('_')[3].split('.')[0]
    estado = f.split('/')[1].split('_')[2]
    codmun = df['CODMUNNASC']

    df['PESO'] = df['PESO'].apply(pd.to_numeric, errors='coerce')

    df['parto_prematuro'] = df['GESTACAO'] <= 4

    aux_cols = []

    base = df

    # LOCNASC e PARTO
    base['LOCNASC_e_PARTO_INCONSISTENTES'] = (base['LOCNASC'] \
                                                .isin([2,3,4,5])) \
                                                & (base['PARTO'] == 2)

    # PARTO e STCESPARTO
    if 'STCESPARTO' in base.columns:
        base['PARTO_e_STCESPARTO_INCONSISTENTES'] = (base['STCESPARTO'] == 1) & \
                                                    (base['PARTO'].isin([2,9]))

    # TPROBSON e composicao
    if 'TPROBSON' in base.columns:
        base['TPROBSON_e_composicao_INCONSISTENTES'] = (base['TPROBSON'] \
                                                         .isin([1,2,3,4,5,6,7,8,9,10])) & \
                                                         ((~base[['QTDGESTANT','QTDPARTNOR', \
                                                         'QTDPARTCES','SEMAGESTAC', \
                                                         'TPAPRESENT','STTRABPART']] \
                                                         .isna()).sum(axis = 1) > 0)

    # PARTO_PREMATURO e PESO
    base['PARTO_PREMATURO_e_PESO_INCONSISTENTES'] = (base['parto_prematuro'] == 1) \
                                                    & (base['PESO'] > 2500)

    aux_cols = []
    for c in base.columns:
        if 'INCONSISTENTES' in c:

```

```

        aux_cols.append(c)

aux = base[aux_cols]

aux['ANO'] = ano
aux['CODMUNNASC'] = codmun

df_inconsistentes = aux

df_inconsistentes.fillna(0, inplace=True)

df_inconsistentes = df_inconsistentes.groupby(['ANO', 'CODMUNNASC']) \
    .sum() \
    .reset_index() \
    .melt(id_vars=['ANO', 'CODMUNNASC'])
df_inconsistentes.columns=['ANO', 'CODMUNNASC', 'VARIABEL', 'INCONSISTENTES']

df['ANO'] = ano
df['CODMUNNASC'] = codmun

df_totais = df[['ANO', 'CODMUNNASC']]
df_totais['TOTAIS'] = 1

df_totais = df_totais.groupby(['ANO', 'CODMUNNASC'])['TOTAIS'] \
    .sum().reset_index()
df_totais.columns = ['ANO', 'CODMUNNASC', 'TOTAIS']

df_totais.set_index(['ANO', 'CODMUNNASC'], inplace=True)
df_inconsistentes.set_index(['ANO', 'CODMUNNASC'], inplace=True)

x = df_totais.join([df_inconsistentes], how='left')
x = x.reset_index()

x = x[['ANO', 'CODMUNNASC', 'VARIABEL', 'INCONSISTENTES', 'TOTAIS']]

x = x.fillna(0)

x.to_csv('SINASC_dataset/resultados/Inconsistencia_{}_{}.csv' \
        .format(estado, ano),
        index=None, compression='gzip')

inconsistencias = pd.DataFrame()

for f in glob.glob('SINASC_dataset/resultados/Inconsistencia_*.csv'):
    df = pd.read_csv(f, compression='gzip')
    inconsistencias = pd.concat([inconsistencias, df], axis=0)

inconsistencias.fillna(0, inplace=True)
inconsistencias.to_csv('SINASC_Inconsistencia_v2.csv',
                      index=None, compression='gzip')

# gera regras

```

```
import json

with open('SINASC_Inconsistencias_Regras.json', 'w') as fp:
    json.dump(regras, fp, indent=4)
```

## Tratamento

Após a extração dos dados via API, é realizado o tratamento dos dados no software R para adequá-los ao modelo de dados utilizado no painel. Esse tratamento envolve várias etapas, incluindo a substituição dos códigos dos municípios pelos seus respectivos nomes e a associação das variáveis aos códigos correspondentes utilizado no Qualidados.

Os conjuntos de dados são trabalhados separadamente para cada indicador, garantindo que as informações sejam devidamente organizadas e associadas as abas correspondentes do painel. Dessa forma, cada indicador terá seu próprio conjunto de dados tratados, contendo as informações necessárias para a análise e exibição.

```
#pacotes
library(rjson)
library(readr)
library(dplyr)
library(readxl)
SINASC_dic <- read_excel("data1/dicionarios.xlsx", sheet = "SINASC")
usethis::use_data(SINASC_dic, overwrite = T)
##### INCOMPLETUDE #####

regras_sinasc_incom <-
  c(fromJSON(file = 'data1/SINASC_Incompletude_Regras.json'))
Sinasc_incom <-
  read_csv("data1/SINASC_Incompletude_v2.csv", show_col_types = FALSE )

#FILTRAR APENAS PARA VARIÁVEIS PRESENTES NO DICIONARIO
vars <- SINASC_dic$`Codigo SINASC` %>% unique()
Sinasc_incom$VARIABLE %>% unique() %>% setdiff(vars)
Sinasc_incom <- Sinasc_incom[Sinasc_incom$VARIABLE %in% vars,]

#ACRESCENTAR A COLUNA DE MUNICIPIOS E MUNICIPIOS

aux_muni2 <- abjData::muni %>%
  dplyr::select(uf_id,
               muni_id,
               muni_nm_clean,
               uf_sigla) %>%
  mutate_at("muni_id", as.character) %>%
  mutate(cod_mun = stringr::str_sub(muni_id, 1, 7))

aux_muni2 <- rbind(aux_muni2, aux_muni2|>
  mutate(cod_mun = stringr::str_sub(muni_id, 1, 6)))

Sinasc_incom$CODMUNNASC <- as.character(format(Sinasc_incom$CODMUNNASC ,
                                               scientific = FALSE))
Sinasc_incom$CODMUNNASC <- gsub(' ', '', Sinasc_incom$CODMUNNASC)
```

```

Sinasc_incom <- Sinasc_incom %>%
  rename(cod_mun = CODMUNNASC ) %>%
  left_join(aux_muni2 ,by='cod_mun') %>%
  select(-muni_id,-uf_id) %>%
  mutate(uf_id = stringr::str_sub(cod_mun,1,2))

Sinasc_incom[is.na(Sinasc_incom$uf_sigla)==T,'uf_sigla']<-
  Sinasc_incom[is.na(Sinasc_incom$uf_sigla)==T,]>
  left_join(unique(aux_muni2[,c('uf_id','uf_sigla')]),by = 'uf_id') >
  dplyr::select(uf_sigla.y)

Sinasc_incom[is.na(Sinasc_incom$muni_nm_clean)==T,'muni_nm_clean'] <-
  'Não informado'

Sinasc_incom$CODMUNNASC <- Sinasc_incom$muni_nm_clean
Sinasc_incom$ESTADO <- Sinasc_incom$uf_sigla
Sinasc_incom[,c('cod_mun','uf_id','uf_sigla','muni_nm_clean')] <- NULL
Sinasc_incom[is.na(Sinasc_incom$ESTADO),"ESTADO"] <- 'Não Informado'

regras_sinasc_incom <-regras_sinasc_incom |> as.data.frame() |> t() |>
  as.data.frame()
regras_sinasc_incom <- cbind(regras_sinasc_incom|> row.names(),
                             regras_sinasc_incom)
regras_sinasc_incom |> row.names() <- NULL
regras_sinasc_incom |> colnames() <- c('Variável','Regra')
regras_sinasc_incom$Variável <- regras_sinasc_incom$Variável |>
  gsub(pattern = 'IGNORADOS_', replacement = '')
regras_sinasc_incom$Regra <- regras_sinasc_incom$Regra |>
  gsub(pattern = 'estiver', replacement = ' estiver')

var_aux <- Sinasc_incom$VARIABEL |> unique()
Sinasc_incom <- merge(Sinasc_incom,
                     SINASC_dic[,c("Codigo Qualidados", "Codigo SINASC") ],
                     by.x="VARIABEL", by.y="Codigo SINASC", all=TRUE)
Sinasc_incom <- Sinasc_incom[Sinasc_incom$VARIABEL %in% var_aux,]
Sinasc_incom$VARIABEL <- Sinasc_incom`Codigo Qualidados`
Sinasc_incom`Codigo Qualidados` <- NULL
vars_incom_sinasc <- unique(Sinasc_incom$VARIABEL)

##### IMPLAUSIBILIDADE #####

regras_sinasc_implau <-
  c(fromJSON(file = 'data1/SINASC_Implausibilidade_Regras.json'))
Sinasc_implau <- read_csv('data1/SINASC_Implausibilidade_v2.csv',
                          show_col_types = FALSE)
Sinasc_implau$VARIABEL <- Sinasc_implau$VARIABEL |>
  gsub(pattern = "_IMPLAUSIVEL", replacement = '')
Sinasc_implau$VARIABEL %>% unique()

#FILTRAR APENAS PARA VARIÁVEIS PRESENTES NO DICIONARIO

Sinasc_implau$VARIABEL %>% unique() %>% setdiff(vars)

```

```

Sinasc_implau <- Sinasc_implau[Sinasc_implau$VARIABEL %in% vars,]

#ACRESCENTAR A COLUNA DE MUNICIPIOS E MUNICIPIOS

Sinasc_implau$CODMUNNASC <- as.character(format(Sinasc_implau$CODMUNNASC ,
                                                scientific = FALSE))
Sinasc_implau$CODMUNNASC <- gsub(' ', '', Sinasc_implau$CODMUNNASC)

Sinasc_implau <- Sinasc_implau %>%
  rename(cod_mun = CODMUNNASC ) %>%
  left_join(aux_muni2 ,by='cod_mun') %>% select(-muni_id,-uf_id)

Sinasc_implau <- Sinasc_implau |>
  mutate(uf_id = stringr::str_sub(cod_mun,1,2))

Sinasc_implau[is.na(Sinasc_implau$uf_sigla)==T,'uf_sigla']<-
  Sinasc_implau[is.na(Sinasc_implau$uf_sigla)==T,]|>
  left_join(unique(aux_muni2[,c('uf_id','uf_sigla')]),by = 'uf_id') |>
  dplyr::select(uf_sigla.y)

Sinasc_implau[is.na(Sinasc_implau$muni_nm_clean)==T,'muni_nm_clean'] <-
  'Não informado'

Sinasc_implau$CODMUNNASC <- Sinasc_implau$muni_nm_clean
Sinasc_implau$ESTADO <- Sinasc_implau$uf_sigla
Sinasc_implau[is.na(Sinasc_implau$ESTADO),'ESTADO'] <- 'Não informado'

Sinasc_implau[,c('cod_mun','uf_id','uf_sigla','muni_nm_clean')] <- NULL

regras_sinasc_implau <-regras_sinasc_implau |> as.data.frame() |>
  t() |> as.data.frame()
regras_sinasc_implau <- cbind(regras_sinasc_implau|> row.names(),
                             regras_sinasc_implau)
regras_sinasc_implau |> row.names() <- NULL
regras_sinasc_implau |> colnames() <- c('Variável','Regra')
regras_sinasc_implau$Regra <- regras_sinasc_implau$Regra |>
  gsub(pattern = 'não',replacement = ' não')

var_aux <- Sinasc_implau$VARIABEL |> unique()
Sinasc_implau <- merge(Sinasc_implau,
                      SINASC_dic[,c("Codigo Qualidades", "Codigo SINASC") ]
                      , by.x="VARIABEL", by.y="Codigo SINASC", all=TRUE)
Sinasc_implau <- Sinasc_implau[Sinasc_implau$VARIABEL %in% var_aux,]
Sinasc_implau$VARIABEL <- Sinasc_implau$`Codigo Qualidades`
Sinasc_implau$`Codigo Qualidades` <- NULL
vars_implau_sinasc <- unique(Sinasc_implau$VARIABEL)

##### INCONSISTÊNCIA #####

Sinasc_incon<- read_csv("data1/SINASC_Inconsistencia_v2.csv")
regras_sinasc_incon <-
  c(fromJSON(file = 'data1/SINASC_Inconsistencias_Regras.json'))

```

```

var_incon_sinasc <-Sinasc_incon$VARIABEL |>
  stringr::str_sub(1,nchar(Sinasc_incon$VARIABEL)-15) |>
  unique() |>
  gsub(pattern = '_', replacement = ' ')
nomes_incon <- Sinasc_incon$VARIABEL |> unique()
var_incon_sinasc |> names() <- nomes_incon
# ACRESCENTAR MUNICIPIO
Sinasc_incon$CODMUNNASC <- as.character(format(Sinasc_incon$CODMUNNASC ,
                                              scientific = FALSE))
Sinasc_incon$CODMUNNASC <- gsub(' ','',Sinasc_incon$CODMUNNASC)
Sinasc_incon <- Sinasc_incon %>%
  rename(cod_mun = CODMUNNASC ) %>%
  left_join(aux_muni2 ,by='cod_mun')

Sinasc_incon[,c('muni_id','uf_id')] <- NULL

Sinasc_incon <- Sinasc_incon |>
  mutate(uf_id = stringr::str_sub(cod_mun,1,2))

Sinasc_incon[is.na(Sinasc_incon$uf_sigla)==T,'uf_sigla']<-
  Sinasc_incon[is.na(Sinasc_incon$uf_sigla)==T,]|>
  left_join(unique(aux_muni2[,c('uf_id','uf_sigla')]),by = 'uf_id') |>
  dplyr::select(uf_sigla.y)

Sinasc_incon[is.na(Sinasc_incon$muni_nm_clean)==T,'muni_nm_clean'] <-
  'Não informado'

Sinasc_incon$CODMUNNASC <- Sinasc_incon$muni_nm_clean
Sinasc_incon$ESTADO <- Sinasc_incon$uf_sigla
Sinasc_incon[,c('cod_mun','uf_id','uf_sigla','muni_nm_clean')] <- NULL
Sinasc_incon[is.na(Sinasc_incon$ESTADO) == T,'ESTADO'] <- 'Não informado'

regras_sinasc_incon <-regras_sinasc_incon |> as.data.frame() |> t() |>
  as.data.frame()
regras_sinasc_incon <- cbind(regras_sinasc_incon|> row.names(),
                             regras_sinasc_incon)
regras_sinasc_incon |> row.names() <- NULL
regras_sinasc_incon |> colnames() <- c('Variável','Regra')
regras_sinasc_incon$Variável <- regras_sinasc_incon$Variável |>
  gsub(pattern = '_',replacement = ' ')

dados_oobr_qualitados_SINASC_Implausibilidade <- Sinasc_implau
dados_oobr_qualitados_SINASC_Incompletude <- Sinasc_incom
dados_oobr_qualitados_SINASC_Inconsistencia <- Sinasc_incon
##### EXPORTACAO #####
usethis::use_data(dados_oobr_qualitados_SINASC_Implausibilidade, overwrite = TRUE)
usethis::use_data(vars_implau_sinasc, overwrite = TRUE)
usethis::use_data(dados_oobr_qualitados_SINASC_Incompletude, overwrite = TRUE)
usethis::use_data(vars_incom_sinasc, overwrite = TRUE)
usethis::use_data(dados_oobr_qualitados_SINASC_Inconsistencia, overwrite = TRUE)
usethis::use_data(var_incon_sinasc, overwrite = TRUE)

```



```
##### REGRAS #####
regras_sinasc_implau$Indicador <- 'Implausibilidade'
regras_sinasc_incon$Indicador <- 'Inconsistência'
regras_sinasc_incom$Indicador <- 'Incompletude'

for(i in seq_along(SINASC_dic$`Codigo SINASC`)) {
  for(j in 1:ncol(regras_sinasc_implau)){
    regras_sinasc_implau[,j] <- gsub(SINASC_dic$`Codigo SINASC`[i],
                                     SINASC_dic$`Codigo Qualificados`[i],
                                     regras_sinasc_implau[,j])
  }
}
regras_sinasc_implau <-
  regras_sinasc_implau[regras_sinasc_implau$Variável %in% vars_implau_sinasc,]
for(i in seq_along(SINASC_dic$`Codigo SINASC`)) {
  for(j in 1:ncol(regras_sinasc_incom)){
    regras_sinasc_incom[,j] <- gsub(SINASC_dic$`Codigo SINASC`[i],
                                     SINASC_dic$`Codigo Qualificados`[i],
                                     regras_sinasc_incom[,j])
  }
}

regras_sinasc_incom <-
  regras_sinasc_incom[regras_sinasc_incom$Variável %in% vars_incom_sinasc,]
regras_sinasc <-
  rbind(regras_sinasc_implau, regras_sinasc_incom, regras_sinasc_incon)

usethis::use_data(regras_sinasc, overwrite = TRUE)
```

## Análise

Devido à disponibilidade de variáveis no banco de dados, é possível apresentar apenas o número máximo de observações por níveis de Incompletude, Implausibilidade e Inconsistência, assim como para cada uma das variáveis.

Agora serão exibidas as tabelas de frequência relativa para cada um dos indicadores. É importante notar que, em alguns casos, os dados totais para determinadas variáveis podem diferir. Isso ocorre devido à ausência de certas variáveis em determinados anos, os quais são mencionados no dicionário do painel.

## Incompletude

```
dados_oobr_qualificados_SINASC_Incompletude_1996_2022 |>
  group_by(VARIAVEL) |>
  summarise(Nulos = sum(NULOS),
            Ignorados = sum(IGNORADOS),
            `Porcentagem Incompletude` = paste0(round((
              sum(NULOS + IGNORADOS)/sum(TOTAIS))*100,2), '%'),
            Total = sum(TOTAIS)) |>
```

```
select(VARIAVEL, Nulos, Ignorados, `Porcentagem Incompletude`, Total) |>
kable()
```

VARIAVEL	Nulos	Ignorados	Porcentagem Incompletude	Total
ANOMALIA_COG	2597961	1168494	5.79%	64998302
APGAR1	5365701	0	6.74%	79637102
APGAR5	5631493	0	7.07%	79637102
CESAREA_ANTES_PART	6859363	972151	21.09%	37126352
CODMUNNATU	4884674	0	13.16%	37126352
COD_ANOMALIA_COG	79145708	0	99.38%	79637102
CONSULTAS_PRE_NAT	4735978	0	13.84%	34222325
CONSUL_PRE_NATAL	1417765	1758504	3.99%	79637102
DIF_OBITO_RECEB	10423	0	0.03%	37126352
DOC_RESP	177503	0	0.69%	25541508
DO_EPIDEMIOLOGICA	7256	0	0.02%	37126352
DO_NOVA	1699	0	0%	37126352
DT_ATUALIZACAO_REG	336419	0	0.69%	48779017
DT_CADASTRO_DN	196761	0	0.4%	48779017
DT_DECLARACAO	438742	0	1.72%	25541508
DT_MENSTRUACAO	18844412	0	50.76%	37126352
DT_NASCIMENTO	0	0	0%	79637102
DT_NASCIMENTO_MAE	4424001	0	11.92%	37126352
DT_RECEBIMENTO__LOT	20122952	0	78.79%	25541508
ESCOLARIDADE	1895791	1953277	4.83%	79637102
ESCOLARIDADE_2010	566609	182012	2.39%	31351324
ESCOL_2010_AGR	417418	163129	2.04%	28445535
ESTABELECIMENTO	2529288	0	3.93%	64293318
ESTADO_CIVIL	11254302	1017088	15.41%	79637102
FUNCAO_RESP	1117151	69	4.37%	25541508
GEST_PRE_NATAL	5230954	0	14.09%	37126352
GRUPO_ROBSON	3819770	1585771	15.8%	34222325
HORA_NASCIMENTO	349131	0	0.71%	49087382
IDADE_DA_MAE	179562	134581	0.39%	79637102
IDADE_PAI	23019296	0	62%	37126352
LOCAL_NASCIMENTO	16968	647029	0.83%	79637102
METODO_UTILIZADO	8696893	0	23.43%	37126352
NASCI_ASSISTIDO	239887	24560	0.93%	28445535
NATURALMAE	4957118	0	13.35%	37126352
NUM_LOTE	31795	0	0.09%	37126352
NUM_PARTOS	0	0	0%	25541508
OCUPACAO_CBO	21594698	0	28.45%	75900261
PAIS_RESID	8925877	0	24.04%	37126352
PARTO_INDUZIDO	4877718	884517	15.52%	37126352
PESO_NASC	292544	87836	0.48%	79637102
QTD_FILHOS_M	13652709	0	17.14%	79637102
QTD_FILHOS_V	7542808	0	9.47%	79637102
QTD_GESTACOES	5585726	0	15.05%	37126352
QTD_PARTO_CESAREA	6384592	0	17.2%	37126352
QTD_PARTO_NORMAL	6084441	0	16.39%	37126352
RACA	11542603	19313	14.52%	79637102
RACA_MAE	5432206	0	14.63%	37126352
RESIDENCIA_MUNI	0	0	0%	79637102

VARIAVEL	Nulos	Ignorados	Porcentagem Incompletude	Total
SEMANAS_GEST	4950081	0	13.33%	37126352
SEMA_GESTACAO	1668594	465866	2.68%	79637102
SERIE_ESC_MAE	16983947	0	45.75%	37126352
SEXO	0	65154	0.08%	79637102
TIPO_GRAVIDEZ	765601	76772	1.06%	79637102
TIPO_PARTO	126061	125611	0.32%	79637102
TIPO_RN	4526617	376629	13.21%	37126352
UF_NATURA_MAE	489545	0	1.72%	28445535
VERSAO_SISTEMA	35263	0	0.09%	37126352

## Implausibilidade

```
dados_oobr_qualitados_SINASC_Implausibilidade_1996_2022 |>
  group_by(VARIAVEL) |>
  summarise(Implausiveis = sum(IMPLAUSIVEIS),
            `Porcentagem Implausibilidade` = paste0(
              round((sum(IMPLAUSIVEIS)/sum(TOTAIS))*100,2), '%'),
            Total = sum(TOTAIS)) |>
  select(VARIAVEL, Implausiveis, `Porcentagem Implausibilidade`, Total) |>
  kable()
```

VARIAVEL	Implausiveis	Porcentagem Implausibilidade	Total
ANOMALIA_COG	658	0%	64998302
APGAR1	85131	0.11%	79637102
APGAR5	78567	0.1%	79637102
CESAREA_ANTES_PART	0	0%	37126352
CONSULTAS_PRE_NAT	0	0%	34222325
CONSUL_PRE_NATAL	3683656	4.63%	79637102
DOC_RESP	2514205	9.84%	25541508
DT_CADASTRO_DN	119	0%	48779017
DT_NASCIMENTO	0	0%	79637102
DT_NASCIMENTO_MAE	5187648	13.97%	37126352
ESCOLARIDADE	7484753	9.4%	79637102
ESCOLARIDADE_2010	157444	0.5%	31351324
ESCOL_2010_AGR	133699	0.47%	28445535
ESTADO_CIVIL	0	0%	79637102
FUNCAO_RESP	69	0%	25541508
GEST_PRE_NATAL	0	0%	37126352
GRUPO_ROBSON	0	0%	34222325
HORA_NASCIMENTO	28	0%	49087382
IDADE_DA_MAE	136326	0.17%	79637102
IDADE_PAI	86	0%	37126352
LOCAL_NASCIMENTO	0	0%	79637102
METODO_UTILIZADO	15117474	40.72%	37126352
NASCI_ASSISTIDO	0	0%	28445535
NUM_PARTOS	0	0%	25541508
PARTO_INDUZIDO	0	0%	37126352
PESO_NASC	0	0%	79637102
QTD_FILHOS_M	866369	1.09%	79637102

VARIAVEL	Implausíveis	Porcentagem Implausibilidade	Total
QTD_FILHOS_V	467291	0.59%	79637102
QTD_GESTACOES	2535	0.01%	37126352
QTD_PARTO_CESAREA	3668	0.01%	37126352
QTD_PARTO_NORMAL	3823	0.01%	37126352
RACA	19314	0.02%	79637102
SEMANAS_GEST	3947	0.01%	37126352
SEMA_GESTACAO	456183	0.57%	79637102
SERIE_ESC_MAE	552	0%	37126352
SEXO	0	0%	79637102
TIPO_GRAVIDEZ	119	0%	79637102
TIPO_PARTO	0	0%	79637102
TIPO_RN	0	0%	37126352

## Inconsistência

```
df <- dados_oobr_qualitados_SINASC_Inconsistencia_1996_2022 |> group_by(VARIAVEL) |>
  summarise(`Inconsistências` = sum(INCONSISTENTES),
            `Porcentagem Inconsistências` = paste0(
              round((sum(INCONSISTENTES)/sum(TOTAIS))*100,2),
              '%'),
            Total = sum(TOTAIS)) |>
  select(VARIAVEL, `Inconsistências`,
         `Porcentagem Inconsistências`, Total)
df$VARIAVEL <- df$VARIAVEL |>
  gsub(pattern = '_e_', replacement = ' e ') |>
  gsub(pattern = '_INCONSISTENTES', replacement = '')
kable(df)
```

VARIAVEL	Inconsistências	Porcentagem Inconsistências	Total
LOCNASC e PARTO	289049	0.36%	79637102
PARTO_PREMATURO e PESO	3009060	3.78%	79637102
PARTO e STCESPARTO	9014258	24.28%	37126352
TPROBSON e composicao	28806879	84.18%	34222325

## SIM

O Sistema de Informação Sobre Mortalidade (SIM), desenvolvido pelo Ministério da Saúde em 1975, é resultado da integração de mais de quarenta modelos de instrumentos utilizados ao longo dos anos para coletar dados sobre mortalidade no país. O SIM possui uma variedade de variáveis que, a partir das causas de morte atestadas pelos médicos, permitem a construção de indicadores e a realização de análises epidemiológicas que contribuem para a eficiência da gestão em saúde.

O processo de informatização do SIM teve início em 1979. Doze anos depois, com a implementação do Sistema Único de Saúde (SUS) e a descentralização das responsabilidades, a coleta de dados foi transferida para os estados e municípios, por meio de suas respectivas Secretarias de Saúde. O objetivo do SIM é reunir dados quantitativos e qualitativos sobre óbitos ocorridos no Brasil, sendo considerado uma ferramenta essencial para a gestão da saúde, fornecendo subsídios para a tomada de decisões em diversas áreas da assistência à saúde. No âmbito federal, o SIM está sob a responsabilidade da Secretaria de Vigilância em Saúde.

## Extração

A extração dos dados foi realizada por meio da API da PCDas para o período de 1996 a 2022, utilizando a linguagem Python como suporte, e os dados de 2023 preliminares pelo open datasus. Durante o processo de extração, os dados são tratados para garantir a qualidade e consistência das informações. Assim como foi feito para a base de dados do SINASC, os dados são subdivididos para cada um dos indicadores trabalhados, visando reduzir o tamanho das bases.

As bases finais resultantes apresentam apenas o número de casos dos indicadores e o total de observações, agrupados por Município-UF, Ano e variável.

```
## INCOMPLETUDE
regras_ignorados = {}
regras_ignorados['TIPOBITO'] = ['NA']
regras_ignorados['SEXO'] = ['I','O']
regras_ignorados['RACACOR'] = ['NA']
regras_ignorados['ESTCIV'] = [9]
regras_ignorados['ESC'] = [9]
regras_ignorados['ESMAE'] = [9]
regras_ignorados['QTDFILVIVO'] = [99]
regras_ignorados['QTDFILMORT'] = [99]
regras_ignorados['GRAVIDEZ'] = [9]
regras_ignorados['GESTACAO'] = [9]
regras_ignorados['PARTO'] = [8,9]
regras_ignorados['OBITOPARTO'] = [9]
regras_ignorados['OBITOGRAV'] = [8,9]
regras_ignorados['OBITOPUERP'] = [99]
regras_ignorados['ASSISTMED'] = [9]
regras_ignorados['EXAME'] = [9]
regras_ignorados['CIRURGIA'] = [9]
regras_ignorados['NECROPSIA'] = [9]
regras_ignorados['CIRCOBITO'] = [0]
regras_ignorados['ACIDTRAB'] = [9]
regras_ignorados['FONTE'] = [9]
regras_ignorados['TPMORTEOCO'] = [9]
regras_ignorados['FONTEINV'] = [9]
regras_ignorados['ESMAEAGR1'] = [9]
regras_ignorados['ESCFALAGR1'] = [9]

# incompletude
cont = 0
for f in glob.glob('SIM_dataset/*.csv'):

    df = pd.read_csv(f)
    ano = df['DTOBITO'] % 10000
    codmun = df['CODMUNOCOR']

    df_ignorados = df.copy()
    df_totais = df.isna()

    df_totais = df.isna()
    df_nulos = df_totais.copy()

    df_totais[df_totais == True] = 1
    df_totais[df_totais == False] = 1
```

```

df_totais['ANO'] = ano
df_totais['CODMUNOCOR'] = codmun

df_totais = df_totais.groupby(['ANO', 'CODMUNOCOR']) \
    .count() \
    .reset_index().melt(id_vars=['ANO', 'CODMUNOCOR'])
df_totais.columns = ['ANO', 'CODMUNOCOR', 'VARIABEL', 'TOTAIS']

df_nulos['CODMUNNASC'] = codmun
df_nulos['ANO'] = ano

df_nulos = df_nulos.groupby(['ANO', 'CODMUNOCOR']) \
    .sum() \
    .reset_index().melt(id_vars=['ANO', 'CODMUNOCOR'])
df_nulos.columns = ['ANO', 'CODMUNOCOR', 'VARIABEL', 'NULOS']

for c in df_ignorados.columns:
    if c in regras_ignorados:
        df_ignorados[c] = df_ignorados[c].isin(regras_ignorados[c])
    else:
        if c not in ['ANO', 'CODMUNOCOR']:
            df_ignorados.drop(columns=[c], inplace=True)

df_ignorados['CODMUNOCOR'] = codmun
df_ignorados['ANO'] = ano

df_ignorados = df_ignorados.groupby(['ANO', 'CODMUNOCOR']) \
    .sum() \
    .reset_index() \
    .melt(id_vars=['ANO', 'CODMUNOCOR'])
df_ignorados.columns = ['ANO', 'CODMUNOCOR', 'VARIABEL', 'IGNORADOS']

df_ignorados = df_ignorados.fillna(0)

df_totais.set_index(['ANO', 'CODMUNOCOR', 'VARIABEL'], inplace=True)
df_nulos.set_index(['ANO', 'CODMUNOCOR', 'VARIABEL'], inplace=True)
df_ignorados.set_index(['ANO', 'CODMUNOCOR', 'VARIABEL'], inplace=True)

x = df_totais.join([df_nulos, df_ignorados], how='left')
x = x.reset_index()

x = x[['ANO', 'CODMUNOCOR', 'VARIABEL', 'NULOS', 'IGNORADOS', 'TOTAIS']]

x = x.fillna(0)

x.to_csv('SIM_dataset/resultados/Incompletude_p{}.csv'.format(cont),
        index=None, compression='gzip')

cont += 1

incompletude = pd.DataFrame()

for f in glob.glob('SIM_dataset/resultados/Incompletude_p*.csv'):

```

```

df = pd.read_csv(f, compression='gzip')
incompletude = pd.concat([incompletude, df], axis=0)

incompletude.fillna(0, inplace=True)
incompletude = incompletude[~incompletude.VARIAVEL.isin(['contador', 'NOVO'])]
incompletude.to_csv('SIM_Incompletude_v2.csv', index=None, compression='gzip')

# gera regras

regras = {}
for r in regras_ignorados:
    regras["IGNORADOS_" + r] = "Se o campo " + r + \
        "estiver preenchido com " + str(regras_ignorados[r])

with open('SIM_Incompletude_Regras.json', 'w') as fp:
    json.dump(regras, fp, indent=4)

### implausibilidade #####
# aplica as regras para variaveis com opcoes
regras_gerais = {'TIPOBITO': [1, 2],
    'SEXO': [1, 2, 9, 0, 'M', 'F', 'I'],
    'RACACOR': [1, 2, 3, 4, 5],
    'ESTCIV': [1, 2, 3, 4, 5, 9],
    'ESC': [1, 2, 3, 4, 5, 9],
    'LOCOCOR': [1, 2, 3, 4, 5, 9],
    'ESCMAC': [1, 2, 3, 4, 5, 9],
    'GRAVIDEZ': [1, 2, 3, 9],
    'GESTACAO': [1, 2, 3, 4, 5, 6, 9],
    'PARTO': [1, 2, 9],
    'OBITOPARTO': [1, 2, 3, 9],
    'OBITOGRAV': [1, 2, 9],
    'OBITOPUERP': [1, 2, 3, 9],
    'ASSISTMED': [1, 2, 3, 9],
    'EXAME': [1, 2, 3, 9],
    'CIRURGIA': [1, 2, 3, 9],
    'NECROPSIA': [1, 2, 3, 9],
    'CIRCOBITO': [1, 2, 3, 4, 9],
    'ACIDTRAB': [1, 2, 9],
    'FONTE': [1, 2, 3, 4, 9],
    'SERIESCMAC': list(range(1, 9)),
    'TPMORTEOCO': [1, 2, 3, 4, 5, 8, 9],
    'TPPOS': [1, 2],
    'ATESTANTE': list(range(1, 6)),
    'FONTEINV': [1, 2, 3, 4, 6, 7, 8, 9],
    'ESCMACAGR1': list(range(1, 13)),
    'ESCFALAGR1': list(range(1, 13)),
}

colunas_implausibilidade = regras_gerais.keys()

cont = 0
# implausibilidade
for f in glob.glob('SIM_dataset/*.csv'):

```

```

df = pd.read_csv(f)
ano = df['DTOBITO'] % 10000
codmun = df['CODMUNOCOR']

aux_cols = []
for c in columnas_implausibilidade:
    if c in df.columns:
        aux_cols.append(c)

aux = df[aux_cols]

aux['ANO'] = ano
aux['CODMUNOCOR'] = codmun

for col in regras_gerais.keys():
    if col in aux_cols:
        aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & \
            (~aux[col].isin(regras_gerais[col]))

# REGRAS ESPECÍFICAS

for col in ['IDADE', 'IDADEMAE']:
    if col not in aux_cols:
        continue
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] < 0) \
        | (aux[col] > 120))

for col in ['QTDFILVIVO', 'QTDFILMORT']:
    if col not in aux_cols:
        continue
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & (aux[col] != 99) & \
        ((aux[col] < 0) | (aux[col] > 70))

col = 'PESO'
if col in aux_cols:
    aux[col] = pd.to_numeric(aux[col], errors='coerce')
    aux[col + "_IMPLAUSIVEL"] = (~aux[col].isna()) & ((aux[col] < 0) | \
        (aux[col] > 11000))

aux_cols = []

for c in aux.columns:
    if 'IMPLAUSIVEL' in c:
        aux_cols.append(c)

aux_cols = ['ANO', 'CODMUNOCOR'] + aux_cols

df_implausiveis = aux[aux_cols]

df_implausiveis.fillna(0, inplace=True)

```



```

df_implausiveis = df_implausiveis.groupby(['ANO', 'CODMUNOCOR']) \
    .sum().reset_index() \
    .melt(id_vars=['ANO', 'CODMUNOCOR'])
df_implausiveis.columns = ['ANO', 'CODMUNOCOR', 'VARIABEL', 'IMPLAUSIVEIS']

df['ANO'] = ano
df['CODMUNOCOR'] = codmun

df_totais = df[['ANO', 'CODMUNOCOR']]
df_totais['TOTAIS'] = 1

df_totais = df_totais.groupby(['ANO', 'CODMUNOCOR'])['TOTAIS'] \
    .sum().reset_index()
df_totais.columns = ['ANO', 'CODMUNOCOR', 'TOTAIS']

df_totais.set_index(['ANO', 'CODMUNOCOR'], inplace=True)
df_implausiveis.set_index(['ANO', 'CODMUNOCOR'], inplace=True)

x = df_totais.join([df_implausiveis], how='left')
x = x.reset_index()

x = x[['ANO', 'CODMUNOCOR', 'VARIABEL', 'IMPLAUSIVEIS', 'TOTAIS']]

x = x.fillna(0)

x.to_csv('SIM_dataset/resultados/Implausibilidade_p{}.csv'.format(cont),
        index=None, compression='gzip')

cont += 1

implausibilidade = pd.DataFrame()

for f in glob.glob('SIM_dataset/resultados/Implausibilidade_p*.csv'):
    df = pd.read_csv(f, compression='gzip')
    implausibilidade = pd.concat([implausibilidade, df], axis=0)

implausibilidade.fillna(0, inplace=True)
implausibilidade = implausibilidade[~implausibilidade.VARIABEL \
    .isin(['contador', 'NOVO'])]
implausibilidade.to_csv('SIM_Implausibilidade_v2.csv',
    index=None, compression='gzip')

# gera regras

regras = {}
regras["IDADE"] = "Se campo IDADE for menor que 0 ou maior que 120"
regras["IDADEMAE"] = "Se campo IDADEMAE for menor que 0 ou maior que 120"
regras["QTDFILVIVO"] = "Se campo QTDFILVIVO for menor que 0 ou maior que 70"
regras["QTDFILMORT"] = "Se campo QTDFILMORT for menor que 0 ou maior que 70"
regras["PESO"] = "Se campo PESO for menor que 0 ou maior que 11000"

for k in regras_gerais.keys():

```

```

if k not in regras.keys():
    regras[k] = "Se o campo " + k + "não for preenchido com " + \
        str(regras_gerais[k])

with open('SIM_Implausibilidade_Regras.json', 'w') as fp:
    json.dump(regras, fp, indent=4)

## inconsistencia #####

regras = {}
cont = 0

def convert_date(col):

    col = col.fillna(0)
    data_string = col.apply(int).apply(str)
    y = data_string.str[-4:]
    m = data_string.str[-6:-4]
    d = data_string.str[:-6]
    return y.str.cat(m.str.cat(d.str.zfill(2)))

# inconsistencias
for f in glob.glob('SIM_dataset/*.csv'):

    df = pd.read_csv(f)
    ano = df['DTOBITO'] % 10000
    codmun = df['CODMUNOCOR']

    aux_cols = [
        'DTOBITO',
        'DTNASC',
        'SEXO',
        'OBITOPARTO',
        'OBITOGRAV',
        'OBITOPUERP',
        'LOCOCOR',
        'FONTE'
    ]

    aux = df[aux_cols]

    # DTOBITO menor que DTNASC
    regras['DTOBITO_e_DTNASC'] = "Se a data de óbito for menor \
        que a data de nascimento"
    aux['DTOBITO'] = convert_date(aux['DTOBITO'])
    aux['DTNASC'] = convert_date(aux['DTNASC'])

    aux['DTOBITO_e_DTNASC_INCONSISTENTES'] = (aux['DTOBITO'] < aux['DTNASC'])

    # se SEXO estiver como M ou I e os campos OBITOPARTO,
    # OBITOGRAV, OBITOPUERP estiverem preenchidos
    regras['SEXO_e_OBITO'] = "Se SEXO for diferente de 'M','I' e os campos \
        relativos a óbitos em mulheres estiverem preenchidos"

```

```

obito_preenchido = (~aux['OBITOPARTO'].isna()) | (~aux['OBITOGRAV'] \
    .isna()) | (~aux['OBITOPUERP'].isna())
aux['SEXO_e_OBITO_INCONSISTENTES'] = (aux['SEXO'].isin(['M','I'])) & \
    (obito_preenchido)

# Se OBITOPARTO preenchido como 3 e OBITOPUERP
## estiver como 3 ou OBITOGRAV estiver como 1;
# Se OBITOPARTO preenchido como 1 ou 2 e OBITOGRAV estiver como 2 ou
## OBITOPUERP estiver como 1 ou 2;
# Se OBITOPARTO preenchido como 9, OBITOGRAV estiver como 1 ou 2 ou
## OBITOPUERP estiver como 1,2 ou 3
regras['OBITO_PUERPERIO_GRAVIDEZ'] = "Se OBITOPARTO e OBITOPUERP estiver \
    como 3 ou OBITOGRAV estiver como 1;"
regras['OBITO_PUERPERIO_GRAVIDEZ'] += "Se OBITOPARTO estiver como 1 ou 2 e \
    OBITOGRAV estiver como 2, ou OBITOPUERP estiver como 1 ou 2;"
regras['OBITO_PUERPERIO_GRAVIDEZ'] += "Se OBITOPARTO estiver como 9 \
    e OBITOGRAV estiver como 1 ou 2, ou OBITOPUERP estiver como 1, 2 ou 3"

parte_1 = (aux['OBITOPARTO'] == 3) & ((aux['OBITOPARTO'] == 3) | \
    (aux['OBITOGRAV'] == 1))
parte_2 = (aux['OBITOPARTO'].isin([1,2])) & ((aux['OBITOGRAV'] == 2) | \
    (aux['OBITOPUERP'].isin([1,2])))
parte_3 = (aux['OBITOPARTO'] == 9) & ((aux['OBITOGRAV'].isin([1,2])) | \
    (aux['OBITOPUERP'].isin([1,2,3])))
aux['OBITO_PUERPERIO_GRAVIDEZ_INCONSISTENTES'] = (parte_1) | (parte_2) | \
    parte_3

# Preenchido como 1 e o item Morte durante o puerperio também for
# preenchido como 1,2 ou 9
regras['OBITOGRAV_e_OBITOPUERP'] = "Se OBITOGRAV estiver como 1 e \
    OBITOPUERP estiver como 1, 2 ou 9"

aux['OBITOGRAV_e_OBITOPUERP_INCONSISTENTES'] = (aux['OBITOGRAV'] == 1) & \
    (aux['OBITOPUERP'].isin([1,2,3]))

# Preenchido como 1 ou 2 e o item morte durante a gravidez
# estiver preenchido como 1 ou 9
regras['OBITOPUERP_e_OBITOGRAV'] = "Se OBITOGRAV estiver como 1 ou 2 e \
    OBITOGRAV estiver como 1 ou 9"

aux['OBITOPUERP_e_OBITOGRAV_INCONSISTENTES'] = (aux['OBITOGRAV'] \
    .isin([1,2])) & (aux['OBITOGRAV'].isin([1,9]))

# Se FONTE diferente de 2 e LOCOCOR for igual a 1
regras['FONTE_E_LOCOCOR'] = "Se FONTE estiver diferente de 2 e \
    LOCOCOR estiver como 1"

aux['FONTE_E_LOCOCOR_INCONSISTENTES'] = (aux['FONTE'] != 2) & \
    (aux['LOCOCOR'] == 1)

aux_cols = []
for c in aux.columns:
    if 'INCONSISTENTES' in c:

```

```

        aux_cols.append(c)

aux = aux[aux_cols]

aux['ANO'] = ano
aux['CODMUNOCOR'] = codmun

df_inconsistentes = aux

df_inconsistentes.fillna(0, inplace=True)

df_inconsistentes = df_inconsistentes.groupby(['ANO', 'CODMUNOCOR']) \
    .sum().reset_index().melt(id_vars=['ANO', 'CODMUNOCOR'])
df_inconsistentes.columns = ['ANO', 'CODMUNOCOR', 'VARIABEL', \
                             'INCONSISTENTES']

df['ANO'] = ano
df['CODMUNOCOR'] = codmun

df_totais = df[['ANO', 'CODMUNOCOR']]
df_totais['TOTAIS'] = 1

df_totais = df_totais.groupby(['ANO', 'CODMUNOCOR'])['TOTAIS'] \
    .sum().reset_index()
df_totais.columns = ['ANO', 'CODMUNOCOR', 'TOTAIS']

df_totais.set_index(['ANO', 'CODMUNOCOR'], inplace=True)
df_inconsistentes.set_index(['ANO', 'CODMUNOCOR'], inplace=True)

x = df_totais.join([df_inconsistentes], how='left')
x = x.reset_index()

x = x[['ANO', 'CODMUNOCOR', 'VARIABEL', 'INCONSISTENTES', 'TOTAIS']]

x = x.fillna(0)

x.to_csv('SIM_dataset/resultados/Inconsistencia_p{}.csv'.format(cont),
        index=None, compression='gzip')

cont += 1
inconsistencias = pd.DataFrame()

for f in glob.glob('SIM_dataset/resultados/Inconsistencia_p*.csv'):
    df = pd.read_csv(f, compression='gzip')
    inconsistencias = pd.concat([inconsistencias, df], axis=0)

inconsistencias.fillna(0, inplace=True)
inconsistencias = inconsistencias[~inconsistencias.VARIABEL \
    .isin(['contador', 'NOVO'])]
inconsistencias.to_csv('SIM_Inconsistencia_v2.csv', index=None,
                      compression='gzip')

```

```
# gera regras
```

```
with open('SIM_Inconsistencia_Regras.json', 'w') as fp:  
    json.dump(regras, fp, indent=4)
```

## Tratamento

No caso do SIM, assim como foi feito para o SINASC, também é realizado um trabalho de adequação dos dados. Os códigos de identificação dos municípios são substituídos pelos respectivos nomes, a fim de tornar os dados mais compreensíveis e facilitar a análise. Além disso, os dados são reformulados e estruturados de maneira adequada para serem recebidos e processados pelo painel.

```
## code to prepare `SIM` dataset goes here  
library(rjson)  
library(readr)  
library(dplyr)  
library(readxl)  
SIM_dic <- read_excel("data1/dicionarios.xlsx", sheet = "SIM")  
##### INCOMPLETUDE #####  
  
regras_sim_incom <- c(fromJSON(file = 'data1/SIM_Incompletude_Regras.json'))  
SIM_Incom <- read_csv("data1/SIM_Incompletude_v2.csv", show_col_types = FALSE )  
  
#ACRESCENTAR A COLUNA DE MUNICIPIOS E MUNICIPIOS  
#####  
aux_muni2 <- abjData::muni %>%  
  dplyr::select(uf_id,  
               muni_id,  
               muni_nm_clean,  
               uf_sigla) %>%  
  mutate_at("muni_id", as.character) %>%  
  mutate(cod_mun = stringr::str_sub(muni_id, 1, 7))  
  
aux_muni2 <- rbind(aux_muni2,aux_muni2|>  
  mutate(cod_mun = stringr::str_sub(muni_id, 1, 6)))  
#####  
  
SIM_Incom$CODMUNOCOR <- as.character(format(SIM_Incom$CODMUNOCOR ,  
                                             scientific = FALSE))  
SIM_Incom$CODMUNOCOR <- gsub(' ', '',SIM_Incom$CODMUNOCOR)  
  
SIM_Incom <- SIM_Incom %>%  
  rename(cod_mun = CODMUNOCOR ) %>%  
  left_join(aux_muni2 ,by='cod_mun')  
  
SIM_Incom[,c('muni_id','uf_id')] <- NULL  
  
SIM_Incom <- SIM_Incom |>  
  mutate(uf_id = stringr::str_sub(cod_mun,1,2))
```

```

SIM_Incom[is.na(SIM_Incom$uf_sigla)==T,'uf_sigla']<-
  SIM_Incom[is.na(SIM_Incom$uf_sigla)==T,]>
  left_join(unique(aux_muni2[,c('uf_id','uf_sigla')]),by = 'uf_id') |>
  dplyr::select(uf_sigla.y)

SIM_Incom[is.na(SIM_Incom$muni_nm_clean)==T,'muni_nm_clean'] <- 'Não informado'

SIM_Incom$CODMUNNASC <- SIM_Incom$muni_nm_clean
SIM_Incom$ESTADO <- SIM_Incom$uf_sigla
SIM_Incom[,c('cod_mun','uf_id','uf_sigla','muni_nm_clean')] <- NULL
var_sim_tirar <- c('CODBAIOCOR',
                  'CODCART',
                  'CODMUNCART',
                  'CONTADOR',
                  'DTREGCART',
                  'EXPDIFDATA',
                  'NUMREGCART',
                  'UFINFORM',
                  'ALTCAUSA',
                  'DTCADINF',
                  'DTCADINV',
                  'DTCONCASO',
                  'DTCONINV',
                  'ESTABDESCR',
                  'FONTES',
                  'FONTESINF',
                  'MORTEPARTO',
                  'NUDIASINF',
                  'NUDIASOBCO',
                  'NUDIASOBIN',
                  'ORIGEM',
                  'TPNIVELINV',
                  'TPOBITOCOR',
                  'TPRESGINFO')

SIM_Incom <- SIM_Incom[!(SIM_Incom$VARIABEL %in% var_sim_tirar),]
var_aux <- SIM_Incom$VARIABEL |> unique()
SIM_Incom <- merge(SIM_Incom, SIM_dic[,c("Codigo Qualidades", "Codigo SIM") ],
                  by.x="VARIABEL", by.y="Codigo SIM", all=TRUE)
SIM_Incom <- SIM_Incom[SIM_Incom$VARIABEL %in% var_aux,]
SIM_Incom$VARIABEL <- SIM_Incom$`Codigo Qualidades`
SIM_Incom$`Codigo Qualidades` <- NULL
vars_incom_sim<- unique(SIM_Incom$VARIABEL)
##### REGRAS
df_aux <- regras_sim_incom |> as.data.frame() |> t() |> as.data.frame()
df_aux<- cbind(row.names(df_aux),df_aux)
df_aux |> row.names() <- NULL
df_aux$`row.names(df_aux)` <- df_aux$`row.names(df_aux)` |>
  gsub(pattern = 'IGNORADOS_', replacement = '')
colnames(df_aux) <- c('Variável','Regra')
regras_sim_incom <- df_aux

usethis::use_data(SIM_Incom, overwrite = TRUE)
usethis::use_data(vars_incom_sim, overwrite = TRUE)

```

```
##### IMPLAUSIBILIDADE #####

regras_sim_implau <-
  c(fromJSON(file = 'data1/SIM_Implausibilidade_Regras.json'))
SIM_Implau <-
  read_csv("data1/SIM_Implausibilidade_v2.csv",show_col_types = FALSE )
SIM_Implau$VARIABEL <- SIM_Implau$VARIABEL |>
  gsub(pattern = '_IMPLAUSIVEL',replacement = '')

SIM_Implau$CODMUNOCOR <- as.character(format(SIM_Implau$CODMUNOCOR ,
                                             scientific = FALSE))
SIM_Implau$CODMUNOCOR <- gsub(' ',',',SIM_Implau$CODMUNOCOR)

SIM_Implau <- SIM_Implau %>%
  rename(cod_mun = CODMUNOCOR ) %>%
  left_join(aux_muni2 ,by='cod_mun')

SIM_Implau[,c('muni_id','uf_id')] <- NULL

SIM_Implau <- SIM_Implau |>
  mutate(uf_id = stringr::str_sub(cod_mun,1,2))

SIM_Implau[is.na(SIM_Implau$uf_sigla)==T,'uf_sigla']<-
  SIM_Implau[is.na(SIM_Implau$uf_sigla)==T,]|>
  left_join(unique(aux_muni2[,c('uf_id','uf_sigla')]),by = 'uf_id') |>
  dplyr::select(uf_sigla.y)

SIM_Implau[is.na(SIM_Implau$muni_nm_clean)==T,'muni_nm_clean'] <-
  'Não informado'

SIM_Implau$CODMUNNASC <- SIM_Implau$muni_nm_clean
SIM_Implau$ESTADO <- SIM_Implau$uf_sigla
SIM_Implau[,c('cod_mun','uf_id','uf_sigla','muni_nm_clean')] <- NULL
SIM_Implau <- SIM_Implau[!(SIM_Implau$VARIABEL %in% var_sim_tirar),]
var_aux <- SIM_Implau$VARIABEL |> unique()
SIM_Implau <- merge(SIM_Implau,
                   SIM_dic[,c("Codigo Qualidados", "Codigo SIM") ],
                   by.x="VARIABEL", by.y="Codigo SIM", all=TRUE)
SIM_Implau <- SIM_Implau[SIM_Implau$VARIABEL %in% var_aux,]
SIM_Implau$VARIABEL <- SIM_Implau$`Codigo Qualidados`
SIM_Implau$`Codigo Qualidados` <- NULL
vars_implau_sim<- unique(SIM_Implau$VARIABEL)
##### REGRAS
df_aux <- regras_sim_implau |> as.data.frame() |> t() |> as.data.frame()
df_aux<- cbind(row.names(df_aux),df_aux)
df_aux |> row.names() <- NULL
colnames(df_aux) <- c('Variável','Regra')
regras_sim_implau <- df_aux
```

```

usethis::use_data(SIM_Implau, overwrite = TRUE)
usethis::use_data(vars_implau_sim, overwrite = TRUE)

##### Inconsistencia
regras_sim_incon <-
  c(fromJSON(file = 'data1/SIM_Inconsistencia_Regras.json'))
SIM_Incon <- read_csv("data1/SIM_Inconsistencia_v2.csv", show_col_types = FALSE )
SIM_Incon$VARIABEL <- SIM_Incon$VARIABEL |>
  gsub(pattern = '_INCONSISTENTES', replacement = '')
SIM_Incon$VARIABEL <- SIM_Incon$VARIABEL |>
  gsub(pattern = '_', replacement = ' ')

SIM_Incon$CODMUNOCOR <- as.character(format(SIM_Incon$CODMUNOCOR ,
                                             scientific = FALSE))
SIM_Incon$CODMUNOCOR <- gsub(' ', '', SIM_Incon$CODMUNOCOR)

SIM_Incon <- SIM_Incon %>%
  rename(cod_mun = CODMUNOCOR ) %>%
  left_join(aux_muni2 , by='cod_mun')

SIM_Incon[,c('muni_id', 'uf_id')] <- NULL

SIM_Incon <- SIM_Incon |>
  mutate(uf_id = stringr::str_sub(cod_mun, 1, 2))

SIM_Incon[is.na(SIM_Incon$uf_sigla)==T, 'uf_sigla'] <-
  SIM_Incon[is.na(SIM_Incon$uf_sigla)==T, ] |>
  left_join(unique(aux_muni2[,c('uf_id', 'uf_sigla')]), by = 'uf_id') |>
  dplyr::select(uf_sigla.y)

SIM_Incon[is.na(SIM_Incon$muni_nm_clean)==T, 'muni_nm_clean'] <- 'Não informado'

SIM_Incon$CODMUNNASC <- SIM_Incon$muni_nm_clean
SIM_Incon$ESTADO <- SIM_Incon$uf_sigla
SIM_Incon[,c('cod_mun', 'uf_id', 'uf_sigla', 'muni_nm_clean')] <- NULL
SIM_Incon <- SIM_Incon[!(SIM_Incon$VARIABEL %in% SIM_Incon),]
vars_incon_sim <- unique(SIM_Incon$VARIABEL)
##### REGRAS
df_aux <- regras_sim_incon |> as.data.frame() |> t() |> as.data.frame()
df_aux <- cbind(row.names(df_aux), df_aux)
df_aux |> row.names() <- NULL
df_aux$`row.names(df_aux)` <- df_aux$`row.names(df_aux)` |>
  gsub(pattern = '_', replacement = ' ')
colnames(df_aux) <- c('Variável', 'Regra')
regras_sim_incon <- df_aux

usethis::use_data(SIM_Incon, overwrite = TRUE)
usethis::use_data(vars_incon_sim, overwrite = TRUE)

##### REGRAS #####
regras_sim_implau$Indicador <- 'Implausibilidade'

```



```

regras_sim_incom$Indicador <- 'Incompletude'
regras_sim_incon$Indicador <- 'Inconsistência'

for(i in seq_along(SIM_dic$`Codigo SIM`)) {
  for(j in 1:ncol(regras_sim_implau)){
    regras_sim_implau[,j] <- gsub(SIM_dic$`Codigo SIM`[i],
                                SIM_dic$`Codigo Qualificados`[i],
                                regras_sim_implau[,j])
  }
}
regras_sim_implau <-
  regras_sim_implau[regras_sim_implau$Variável %in% vars_implau_sim,]
for(i in seq_along(SIM_dic$`Codigo SIM`)) {
  for(j in 1:ncol(regras_sim_incom)){
    regras_sim_incom[,j] <- gsub(SIM_dic$`Codigo SIM`[i],
                                SIM_dic$`Codigo Qualificados`[i],
                                regras_sim_incom[,j])
  }
}
regras_sim_incom <-
  regras_sim_incom[regras_sim_incom$Variável %in% vars_incom_sim,]
regras_sim <- rbind(regras_sim_implau, regras_sim_incom, regras_sim_incon)
usethis::use_data(regras_sim, overwrite = TRUE)
SIM_dic<-
  SIM_dic[SIM_dic$`Codigo Qualificados` %in% c(vars_implau_sim,
                                              vars_incom_sim, vars_incon_sim),
          ]
usethis::use_data(SIM_dic, overwrite = T)

```

## Análise

Devido à disponibilidade das variáveis no banco de dados, é possível apresentar apenas o número máximo de observações para cada nível de Incompletude, Implausibilidade e Inconsistência. Além disso, será fornecida a frequência dos indicadores para cada variável presente no conjunto de dados.

### Incompletude

```

dados_oobr_qualitados_SIM_Incompletude_1996_2022 |>
  group_by(VARIAVEL) |>
  summarise(Nulos = sum(NULOS),
            Ignorados = sum(IGNORADOS),
            `Porcentagem Incompletude` = paste0(round(
              (sum(NULOS + IGNORADOS)/sum(TOTAIS))*100,2), '%'),
            Total = sum(TOTAIS)) |>
  select(VARIAVEL, Nulos, Ignorados, `Porcentagem Incompletude`, Total) |>
  kable()

```

VARIAVEL	Nulos	Ignorados	Porcentagem Incompletude	Total
ACIDENTE_TRAB	74858	870	97.33%	77804
ASSIST_MEDICA	15019	2653	22.71%	77804
CAUSA_BASICA	219	0	0.37%	58496
CAUSA_CID_10	0	0	0%	77804
CAUSA_EXT_MAT	37261	0	99.2%	37563
CAUSA_SCB	28218	0	57.39%	49167
CID_ATESTADO	1	0	0%	28234
CID_LINHA_A	7143	0	9.18%	77804
CID_LINHA_B	20764	0	26.69%	77804
CID_LINHA_C	39584	0	50.88%	77804
CID_LINHA_D	60267	0	77.46%	77804
CID_LINHA_II	57520	0	73.93%	77804
CIRURGIA	42690	1935	57.36%	77804
CODIFICADO	0	0	0%	28234
DIF_OBITO_RECEB	1	0	0%	34157
DT_ATESTADO	504	0	0.86%	58496
DT_CADASTRO	721	0	1.23%	58496
DT_INVESTIG	26623	0	45.51%	58496
DT_NASC	276	0	0.35%	77804
DT_OBITO	0	0	0%	77804
DT_RECEBI_CENTRAL	4578	0	7.83%	58496
DT_RECEBI_ORIGINAL	2	0	0.01%	34700
ESCOLARIDADE	11452	9321	26.7%	77804
ESCOLARIDADE_2010	6707	0	17.86%	37563
ESCOL_2010_AGR	5367	2383	24.77%	31294
ESCOL_MAE	76549	124	98.55%	77804
ESCOL_MAE_2010	36505	0	97.18%	37563
ESCOL_MAE_2010_AGR	30533	57	97.75%	31294
ESTABELECIMENTO	11696	0	16.88%	69300
EST_CIVIL	3936	2595	8.39%	77804
EXAM_COMPLEM	42177	2304	57.17%	77804
FONTE_INF	70625	803	91.81%	77804
FONTE_INV	25822	41	44.21%	58496
HORA_OBITO	3286	0	5.62%	58496
IDADE	5	0	0.01%	77804
IDADE_MAE	76605	0	98.46%	77804
LOCAL_OBITO	60	0	0.08%	77804
MEDICO_ATEST	4465	0	6.44%	69300
MORTE_GRAV	1563	415	2.54%	77804
MORTE_PARTO	76584	38	98.48%	77804
MORTE_PUERP	3670	0	4.72%	77804
MUNICIPIO_NATU	1783	0	6.32%	28234
MUNICIPIO_RES	0	0	0%	77804
MUNICIPIO_SVO_IML	28828	0	76.75%	37563
NATURALIDADE	15952	0	20.5%	77804
NECROPSIA	9608	1956	14.86%	77804
NUMERO_LOTE	24	0	0.09%	28234
NUM_FILH_MORT	76317	394	98.6%	77804
NUM_FILH_VIVOS	76554	90	98.51%	77804
OBITO_INV	13012	0	22.24%	58496
OCUP_CBO2002	14067	0	18.08%	77804
OCUP_MAE	76657	0	98.53%	77804

VARIAVEL	Nulos	Ignorados	Porcentagem Incompletude	Total
PESO_NASC	76456	0	98.27%	77804
RACA	6285	0	8.08%	77804
SEM_GEST	33212	0	97.23%	34157
SEM_GESTACAO	76624	32	98.52%	77804
SERIE_FALECIDO	25531	0	74.75%	34157
SERIE_MAE	33705	0	98.68%	34157
SEXO	1	0	0%	77804
STATUS_DO_EPID	1	0	0%	37563
STATUS_DO_NOVA	0	0	0%	37563
STATUS_INSTAL	7	0	0.02%	28234
TIPO_GRAVIDEZ	76553	10	98.4%	77804
TIPO_MORTE_GRAV	522	1	1.53%	34157
TIPO_OBITO	0	0	0%	77804
TIPO_PARTO	76552	10	98.4%	77804
TP_ACIDENTE	69079	4	88.79%	77804
VERSAO_SELETOR	1154	0	4.09%	28234
VERSAO_SISTEM	0	0	0%	28234
NA	37648	0	79.12%	47581

## Implausibilidades

```
dados_oobr_qualitados_SIM_Implausibilidade_1996_2022 |>
  group_by(VARIAVEL) |>
  summarise(Implausiveis = sum(IMPLAUSIVEIS),
            `Porcentagem Implausibilidade` = paste0(
              round((sum(IMPLAUSIVEIS)/sum(TOTAIS))*100,2), '%'),
            Total = sum(TOTAIS)) |>
  select(VARIAVEL, Implausiveis, `Porcentagem Implausibilidade`, Total) |>
  kable()
```

VARIAVEL	Implausiveis	Porcentagem Implausibilidade	Total
ACIDENTE_TRAB	3	0%	77804
ASSIST_MEDICA	0	0%	77804
CIRURGIA	1	0%	77804
ESCOLARIDADE	2124	2.73%	77804
ESCOL_2010_AGR	706	2.26%	31294
ESCOL_MAE	34	0.04%	77804
ESCOL_MAE_2010_AGR	40	0.13%	31294
EST_CIVIL	0	0%	77804
EXAM_COMPLEM	1	0%	77804
FONTE_INF	1	0%	77804
FONTE_INV	440	0.75%	58496
LOCAL_OBITO	21	0.03%	77804
MEDICO_ATEST	1	0%	69300
MORTE_GRAV	0	0%	77804
MORTE_PARTO	0	0%	77804
MORTE_PUERP	1324	1.7%	77804
NECROPSIA	1	0%	77804
OBITO_INV	45484	77.76%	58496

VARIAVEL	Implausíveis	Porcentagem Implausibilidade	Total
RACA	0	0%	77804
SEM_GESTACAO	7	0.01%	77804
SERIE_MAE	0	0%	34157
SEXO	0	0%	77804
TIPO_GRAVIDEZ	1	0%	77804
TIPO_MORTE_GRAV	0	0%	34157
TIPO_OBITO	0	0%	77804
TIPO_PARTO	1	0%	77804
TP_ACIDENTE	4	0.01%	77804

## Inconsistência

```
df <- dados_oobr_qualitados_SIM_Inconsistencia_1996_2022 |> group_by(VARIAVEL) |>
  summarise(`Inconsistências` = sum(INCONSISTENTES),
            `Porcentagem Inconsistências` = paste0(round(
              (sum(INCONSISTENTES)/sum(TOTAIS))*100,2), '%'),
            Total = sum(TOTAIS)) |>
  select(VARIAVEL, `Inconsistências`, `Porcentagem Inconsistências`, Total)
df$VARIAVEL <- df$VARIAVEL |>
  gsub(pattern = '_e_', replacement = ' e ') |>
  gsub(pattern = '_INCONSISTENTES', replacement = '')
kable(df)
```

VARIAVEL	Inconsistências	Porcentagem Inconsistências	Total
DTOBITO e DTNASC	0	0%	77804
FONTE E LOCOCOR	62233	79.99%	77804
OBITO PUERPERIO GRAVIDEZ	1220	1.57%	77804
OBITOGRAV e OBITOPUERP	27769	35.69%	77804
OBITOPUERP e OBITOGRAV	34482	44.32%	77804
SEXO e OBITO	0	0%	77804