



**Cursos Integrados
em Vigilância em Saúde**

Curso

**Análise de dados para a vigilância
em saúde – Curso Básico**

Módulo 4 - Análises básicas de dados para vigilância em saúde - Parte I

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Reitor Irineu Manoel de Souza

Vice-Reitora Joana Célia dos Passos

Pró-Reitora de Pós-graduação Werner Kraus

Pró-Reitor de Pesquisa e Inovação Jacques Mick

Pró-Reitor de Extensão Olga Regina Zigelli Garcia

CENTRO DE CIÊNCIAS DA SAÚDE

Diretor Fabrício de Souza Neves

Vice-Diretora Ricardo de Souza Magini

DEPARTAMENTO DE SAÚDE PÚBLICA

Chefe do Departamento Rodrigo Otávio Moretti Pires

Subchefe do Departamento Sheila Rúbia Lindner

Coordenadora do Curso Alexandra Crispim Boing

INSTITUTO TODOS PELA SAÚDE (ITPS)

Diretor presidente Jorge Kalil (Professor titular da Faculdade de Medicina da Universidade de São Paulo; Diretor do Laboratório de Imunologia do Incor)

ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA (ABRASCO)

Presidente Rosana Teresa Onocko Campos

EQUIPE DE PRODUÇÃO

Denis de Oliveira Rodrigues

Kamila de Oliveira Belo

Marcelo Eduardo Borges

Oswaldo Gonçalves Cruz

Alexandra Crispim Boing

Antonio Fernando Boing

Módulo 4 - Análises básicas de dados para vigilância em saúde - Parte I

Curso _____

**Análise de dados para a vigilância
em saúde – Curso Básico**

Dados Internacionais de Catalogação-na-Publicação (CIP)

A532 Análises básicas de dados para vigilância em saúde – Parte I/ Denis de Oliveira Rodrigues, Kamila de Oliveira Belo, Marcelo Eduardo Borges, Oswaldo Gonçalves Cruz . – Santa Catarina ; São Paulo ; Rio de Janeiro : UFSC ; ITPS ; Abrasco; 2022. 46p. (Análise de dados para a vigilância em saúde – Curso Básico; Módulo 4).

Publicação Online

10.52582/curso-analise-dados-vigilancia-modulo4-p1

1. Vigilância em saúde 2. Análise de dados I. Título

Sumário

O uso de estatísticas básicas para vigilância em saúde	06
1. Análise exploratória de dados da Vigilância em Saúde	07
2. Cálculo de frequências.....	12
2.1 Frequência absoluta	13
2.2 Frequência relativa ou porcentagem	21
3. Tipo de variáveis.....	23
4. Cálculo de medidas-resumo (média, mediana e quartis)	29
5. Cálculo de medidas de dispersão (variância e desvio padrão)	34
6. Avaliando a distribuição dos dados	37

O uso de estatísticas básicas para vigilância em saúde

Para seguir com este módulo você deve saber como importar dados de diferentes fontes, limpar e transformar textos e datas, criar tabelas com filtros escolhidos, unir dados de diversas fontes e exportar os dados tratados para diferentes formatos. Todo esse conteúdo nós vimos nos três primeiros módulos do curso. Lembre-se que você pode sempre voltar a eles para relembrar os códigos e dicas.

Neste módulo iremos aprender algumas das principais métricas para análise de dados utilizando o software **R**. Estas métricas são usadas dentro do fluxo de dados que você aprendeu no módulo anterior para analisar e investigar os diversos sistemas de informação, construindo um conjunto de dados de interesse e resumindo suas principais características. Estas etapas são fundamentais para o planejamento, o monitoramento e a avaliação das ações de vigilância em saúde.

Ao final deste módulo, você será capaz de:

- 1.** realizar os principais cálculos para análise exploratória de dados;
- 2.** calcular frequência absoluta e relativa;
- 3.** calcular média, mediana e quartis;
- 4.** avaliar a distribuição e a dispersão de dados.

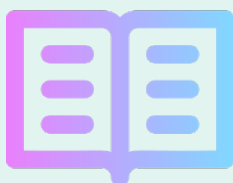
1. Análise exploratória de dados da Vigilância em Saúde

Para iniciar um relatório epidemiológico que narre a situação de um agravo, doença ou evento de saúde o profissional de vigilância deve realizar a análise *exploratória* de seus dados. Para manipular bancos de dados de forma eficiente, garantindo segurança à análise, o profissional deve possuir uma metodologia de compreensão e organização dos dados.

Para dar início a esta etapa, como primeiros passos para uma análise segura e reproduzível, o profissional de vigilância em saúde deve conhecer a estrutura de sua base de dados. Para isto deve sempre se fazer as seguintes perguntas:

- Quais informações estão presentes no banco de dados?
- Quais são as variáveis que o compõem?
- Quais os formatos dessas variáveis: textos, números, datas, categorias?
- O que cada linha significa?

Neste tópico iremos aprender como obter uma descrição básica dos dados escolhidos e responder a cada uma das perguntas listadas acima. Ao final seremos capazes de garantir a validade dos dados analisados e produzir nossa análise de situação de saúde com segurança.



Lembre-se de ter em mãos durante sua análise exploratória um **dicionário de dados**, documento que descreve os dados que serão utilizados: nomes ou variáveis, objetos, a estrutura dos dados ou suas fontes.

O dicionário de dados pode ser chamado também em alguns momentos de **glossário de dados**.

Crie o hábito de consultar o **dicionário de dados** dos sistemas de informações que irá analisar! Caso o sistema não possua, você pode construir seu próprio dicionário de dados o que ampliará sua compreensão da estrutura e memorização das variáveis do banco.

Vamos praticar! Para nossa análise exploratória do Estado de Rosas utilizaremos a base de dados {NINDINET.dbf}, armazenada no diretório “bases” do curso e que já utilizamos anteriormente. Carregue os pacotes do `tidyverse` e `foreign`, e em seguida importe o banco de dados para o ambiente do RStudio. Lembre-se que os pacotes no R são bibliotecas que organizam e padronizam a distribuição de funções do R.

Acompanhe o *script* abaixo e não se preocupe com os pacotes novos que utilizaremos, eles serão explicados de forma detalhada no momento de aplicação neste módulo:

```
# carregando os pacotes necessários para análise
if(!require(tidyverse)) install.packages("tidyverse");library(tidyverse)
if(!require(foreign)) install.packages("foreign");library(foreign)
if(!require(DescTools)) install.packages("DescTools");library(DescTools)
if(!require(summarytools)) install.packages("summarytools");library(summarytools)
if(!require(gtsummary)) install.packages("gtsummary");library(gtsummary)
```

```
# criando objeto do tipo dataframe (tabela) {`dados`} com a base de dados {`NINDINET.dbf`}
dados <- read.dbf(file = 'Dados/NINDINET.dbf')
```

Agora, vamos visualizar o total de linhas e colunas da base de dados, sua estrutura, o tipo de variável de cada coluna e seus respectivos primeiros valores da tabela `dados`. Para isso utilizaremos a função `glimpse()`:

```
# utilizando a função glimpse()
glimpse(dados)
```



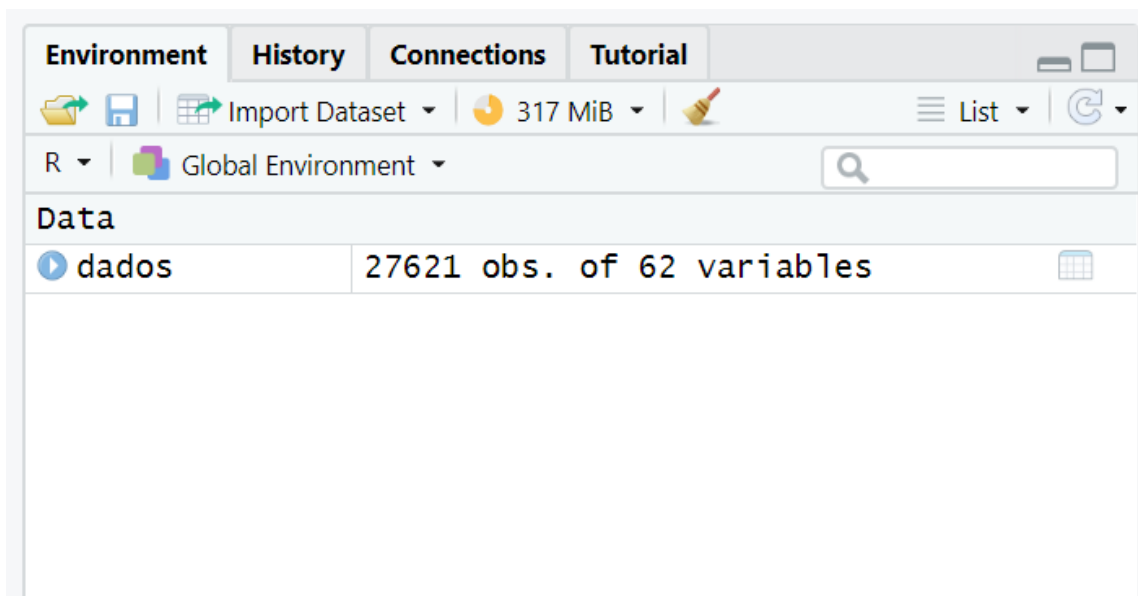

```
#> Rows: 27,621
#> Columns: 62
#> $ NU_NOTIFIC <fct> 7671320, 0855803, 8454645, 3282723, 9799526, 7275624, 82477...
#> $ TP_NOT <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,...
#> $ ID_AGRAVO <fct> A509, W64, X58, A90, B19, A90, A90, Y09, A90, W64, A90, A90...
#> $ CS_SUSPEIT <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ IN_AIDS <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CS_MENING <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ DT_NOTIFIC <date> 2012-04-11, 2010-09-17, 2010-10-19, 2008-04-14, 2011-06-20...
#> $ SEM_NOT <fct> 201215, 201037, 201042, 200816, 201125, 200807, 200750, 201...
#> $ NU_ANO <int> 2012, 2010, 2010, 2008, 2011, 2008, 2007, 2011, 2008, 2011,...
#> $ SG_UF_NOT <int> 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61,...
#> $ ID_MUNICIP <int> 610213, 610213, 610213, 610213, 610213, 610213, 610213, 610...
#> $ ID_REGIONA <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ ID_UNIDADE <int> 256100, 180142, 559191, 180142, 480722, 570404, 319816, 289...
#> $ DT_SIN_PRI <date> 2012-04-05, 2010-09-09, 2010-10-19, 2008-04-11, 2011-04-02...
#> $ SEM_PRI <fct> 201214, 201036, 201042, 200815, 201113, 200806, 200749, 201...
#> $ DT_NASC <date> 2012-04-04, 1988-04-23, 1971-03-25, 1928-05-29, 2002-09-18...
#> $ NU_IDADE_N <int> 2001, 4022, 4039, 4079, 4008, 4054, 4032, 4014, 4037, 4011,...
#> $ CS_SEXO <fct> M, M, M, F, M, F, F, F, F, F, M, M, M, M, F, F, F, F, F, F,...
#> $ CS_GESTANT <int> 6, 6, 6, 9, 6, 9, 9, 9, 6, 5, 6, 6, 6, 6, 9, 9, 9, 9, 6, 5,...
#> $ CS_RACA <int> 4, 1, NA, 4, 4, 9, NA, 1, 9, 4, NA, 9, NA, NA, 9, NA, 1, 9,...
#> $ CS_ESCOL_N <fct> 10, NA, NA, 02, 01, 09, NA, 09, 09, 01, 10, 09, 10, NA, 09, 09,...
#> $ SG_UF <int> 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33,...
#> $ ID_MN_RESI <int> 610213, 610213, 610250, 610213, 610250, 610213, 610213, 610...
#> $ ID_RG_RESI <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ ID_DISTRICT <fct> 05, 05, 05, 01, 01, 04, 05, 04, 01, 04, 01, 05, 04, 05, 03,...
#> $ ID_BAIRRO <fct> 020, 019, 020, 001, 001, 014, 020, 012, 003, 016, 003, 020,...
#> $ ID_LOGRADO <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ ID_GEO1 <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ ID_GEO2 <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CS_ZONA <int> 1, 1, NA, 1, 1, 1, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1, ...
#> $ ID_PAIS <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
#> $ NDUPLIC_N <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ IN_VINCULA <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ DT_INVEST <date> NA, NA, 2010-10-19, 2008-04-14, 2011-06-20, NA, NA, NA, 20...
#> $ ID_OCUPA_N <fct> NA, NA, NA, NA, 999991, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CLASSI_FIN <int> NA, NA, NA, 5, 1, 8, 8, 1, 1, NA, 8, 1, 8, 8, 1, NA, 1, 8, ...
#> $ CRITERIO <int> NA, NA, NA, 1, NA, NA, NA, NA, 2, NA, NA, 1, NA, NA, 2, NA,...
#> $ TPAUTOCTO <int> NA, NA, NA, NA, NA, NA, NA, NA, 1, NA, NA, 1, NA, NA, NA, N...
#> $ COUFINF <int> NA, NA, NA, NA, NA, NA, NA, NA, 61, NA, NA, 61, NA, NA, NA, NA,...
#> $ COPAISINF <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,...
#> $ COMUNINF <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CODISINF <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CO_BAINFC <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0...
#> $ NOBAIINF <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ DOENCA_TRA <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ EVOLUCAO <int> 1, NA, NA, NA, NA, NA, NA, NA, NA, 1, NA, NA, 1, NA, NA, NA, NA...
#> $ DT_OBITO <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
#> $ DT_ENCERRA <date> NA, 2010-10-16, 2010-10-19, 2008-06-19, 2011-06-20, 2008-0...
#> $ DT_DIGITA <date> 2012-11-09, 2010-11-17, 2011-03-14, 2008-04-24, 2011-09-14...
#> $ DT_TRANSUS <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
#> $ DT_TRANSDM <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, 2008-07-10, NA, NA, NA, NA...
#> $ DT_TRANSSE <date> 2012-11-13, 2010-11-23, 2011-04-12, 2010-11-16, 2011-09-19...
#> $ DT_TRANSRM <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
#> $ DT_TRANSRS <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
#> $ DT_TRANSSE <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
#> $ NU_LOTE_V <fct> 2012049, 2010047, 2011015, 2010044, 2011038, 2010043, 20100...
#> $ NU_LOTE_H <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CS_FLXRET <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
#> $ FLXRECEBI <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,...
#> $ MIGRADO_W <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CO_USUCAD <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
#> $ CO_USUALT <lg1> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
```

Observe que ao executarmos o comando `glimpse(dados)`, a função lhe retornará o nome de todas as variáveis da base. Estas variáveis estão identificadas a partir de estruturas com o seguinte padrão:

- `int`: variáveis compostas por números inteiros;
- `dbl`: variáveis numéricas compostas por números reais;
- `date`: variáveis no formato de data (aaaa/mm/dd);
- `fct`: variáveis categóricas (codificadas como “fatores” ou “*factors*”), e o número de categorias nesta variável (*levels*);
- `chr`: variáveis no formato de strings de texto (*character*).

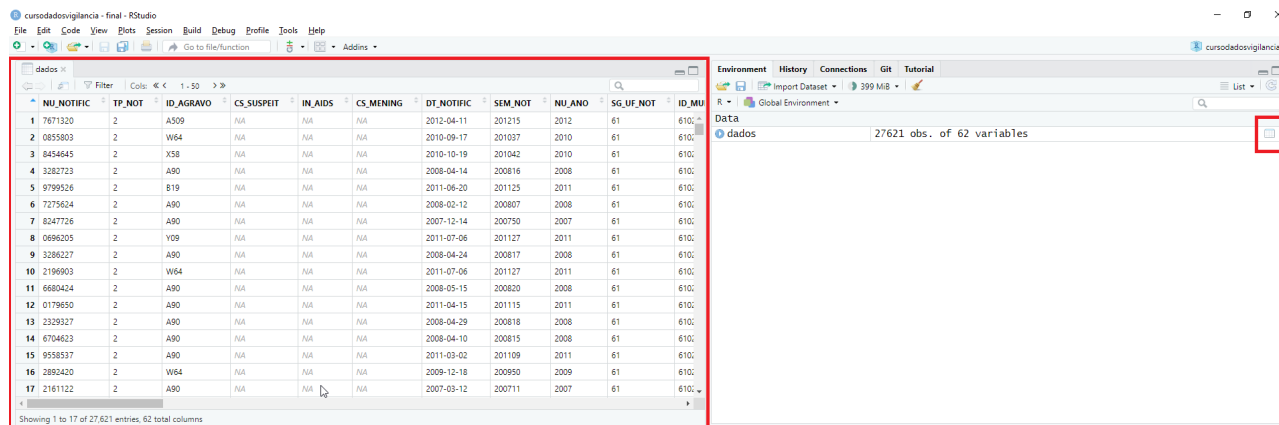
Você poderá também visualizar os dados de outra maneira, ou seja, de forma mais resumida. Para isto deve selecionar ou clicar no nome da sua tabela no canto direito superior do **RStudio** dentro da aba **Environment**. Veja esta área do R na Figura 1 abaixo. Clique na tabela que acabamos de criar chamada **dados**.

Figura 1: Tela do painel **Environment** com a tabela `{dados}` carregada no **RStudio**.



Clicando na tabela `{dados}`, iremos abrir uma nova aba com as informações presentes em cada coluna (Figura 2). Utilizando as barras de rolagens da sua tela para baixo você poderá acessar as diferentes linhas e colunas de toda tabela `{dados}` salva no R.

Figura 2: Tela da IDE RStudio com a tabela {dados} carregada.



	NU_NOTIFIC	TP_NOT	ID_AGRAVO	CS_SUSPEIT	IN_AIDS	CS_MENING	DT_NOTIFIC	SEM_NOT	NU_ANO	SG_UF_NOT	ID_MU
1	7671320	2	A509	N/A	N/A	N/A	2012-04-11	201215	2012	61	6101
2	0855803	2	W64	N/A	N/A	N/A	2010-09-17	201037	2010	61	6101
3	8454645	2	X58	N/A	N/A	N/A	2010-10-19	201042	2010	61	6101
4	3282723	2	A90	N/A	N/A	N/A	2008-04-14	200816	2008	61	6101
5	9799526	2	B19	N/A	N/A	N/A	2011-06-20	201125	2011	61	6101
6	7275624	2	A90	N/A	N/A	N/A	2008-02-12	200807	2008	61	6101
7	8247726	2	A90	N/A	N/A	N/A	2007-12-14	200750	2007	61	6101
8	0696205	2	Y09	N/A	N/A	N/A	2011-07-06	201127	2011	61	6101
9	3286227	2	A90	N/A	N/A	N/A	2008-04-24	200817	2008	61	6101
10	2196903	2	W64	N/A	N/A	N/A	2011-07-06	201127	2011	61	6101
11	6680424	2	A90	N/A	N/A	N/A	2008-05-15	200820	2008	61	6101
12	0179650	2	A90	N/A	N/A	N/A	2011-04-15	201115	2011	61	6101
13	2329327	2	A90	N/A	N/A	N/A	2008-04-29	200818	2008	61	6101
14	6704623	2	A90	N/A	N/A	N/A	2008-04-10	200815	2008	61	6101
15	9558537	2	A90	N/A	N/A	N/A	2011-03-02	201109	2011	61	6101
16	2892420	2	W64	N/A	N/A	N/A	2009-12-18	200950	2009	61	6101
17	2161122	2	A90	N/A	N/A	N/A	2007-03-12	200711	2007	61	6101

Pronto, agora que conhecemos as variáveis e a estrutura da tabela {dados} poderemos escolher quais as variáveis serão necessárias para construção de uma análise dados de vigilância em saúde.



Lembre-se que ao tentar visualizar bases de dados grandes, a tela irá mostrar apenas um número máximo de colunas de cada vez. Você pode “mudar a página” para visualizar mais colunas com os botões > para ver a próxima “página”, e >> para ir até a última página. Estes botões estão localizados na porção superior da visualização de dados.

Acompanhe nas sessões abaixo as medidas mais utilizadas em epidemiologia, como cálculo de frequência, proporção e porcentagem, valores mínimos e máximos, média, mediana, quartis, variância, desvio padrão e medidas de dispersão dos dados.

2. Cálculo de frequências

Considere que o profissional de vigilância necessita produzir um relatório analisando as notificações de casos suspeitos ou confirmados de dengue e hepatites virais. Para esta análise você escolheu utilizar o Sistema de Informação em Saúde Sinan Net.

Para construir o relatório, você precisará conhecer o número de casos e para isso irá criar uma tabela de frequência, ou seja, contar quantas vezes um caso foi notificado no banco de dados {**NINDINET.dbf**}, disponível no menu lateral “Arquivos”, do módulo.

2.1 Frequência absoluta

Agora vamos calcular a frequência absoluta dos valores de colunas específicas na tabela `dados`. Lembre-se que a base `{NINDINET.dbf}` foi utilizada para gerar o objeto `{dados}`.

Para esta avaliação, utilizaremos as funções do pacote `dplyr`. Os três passos a seguir serão necessários para a construção da tabela de frequência:

1. Primeiro passo, selecionaremos (filtre) os registros de pacientes por agravos de interesse. Os códigos registrados na tabela do `{NINDINET.dbf}` para filtrar o agravo dengue e hepatite viral não especificada são respectivamente `A90` e `B19`, de acordo com o CID-10. Estas informações estão contidas na coluna `ID_AGRAVO`. Você irá escrever assim: `filter(ID_AGRAVO %in% c("A90", "B19")) |>`;
2. Segundo passo, será necessário organizar os casos encontrados, agrupando de acordo com cada categoria, ou seja, os dois agravos, utilizando a função `group_by()`. Assim: `group_by(ID_AGRAVO) |>`;
3. Terceiro passo, utilize a função `summarise()` para que o `R` retorne as informações de interesse para cada valor por agravo. Essa função necessita de dois argumentos para ser executada:
 - o nome da nova coluna contendo as informações que queremos calcular;
 - o comando indicando qual informação de cada grupo queremos obter.

Logo, neste exemplo, vamos criar uma coluna com o nome `total`, embora qualquer outro nome possa ser escolhido, e vamos contar o número de elementos em cada categoria. Para isso, vamos utilizar a função `n()`. Assim, nosso comando para esse passo será: `summarise(total = n())`.

Veja como fica o *script* com todos os comandos descritos acima:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")  
# com a função filter()
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# agrupando os agravos (coluna ID_AGRAVO) com a função group_by()
```

```
group_by(ID_AGRAVO) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n())
```

```
#> # A tibble: 2 × 2  
#>   ID_AGRAVO total  
#>   <fct>      <int>  
#> 1 A90        12781  
#> 2 B19         955
```

Observe o *output* do código executado, você deverá encontrar os seguintes valores: 12.781 registros de pacientes notificados com dengue (**A90**), e 955 pacientes notificados de hepatites virais (**B19**). Caso você encontre algo diferente, retorne ao código e verifique se todos os comandos estão iguais ao exemplo acima, inclusive vírgulas e parênteses.

Agora, vamos estudar e verificar se existiu diferença entre sexo para notificação dos casos de dengue e hepatites virais. Para isto, calcularemos a frequência absoluta das notificações por sexo no {NINDINET.dbf}. Lembre-se que usaremos a mesma lógica de construção dos códigos utilizados acima. Observe e siga o *script* abaixo:

```
dados |>
```

```
# agrupando com a função group_by() os indivíduos por sexo (coluna CS_SEX0)
```

```
group_by(CS_SEX0) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n())
```

```
#> # A tibble: 3 × 2  
#>   CS_SEX0 total  
#>   <fct>    <int>  
#> 1 F      13567  
#> 2 I         56  
#> 3 M     13998
```

Perceba que é possível visualizar três valores para a variável sexo (CS_SEX0): **F** para feminino com 13.567 registros; **M** para masculino com 13.998 registros e; **I** para ignorado com apenas 56 registros. No dia a dia na vigilância é comum encontrarmos registros de pacientes em que a informação sobre o sexo está ausente. Agora, podemos ter a seguinte dúvida:

Será que estes registros ignorados ou nulos são mais frequentes no agravo dengue (A90) do que quando analisamos o agravo hepatite(B19)?

Como profissional de vigilância em saúde, para responder à pergunta acima, necessitaremos aprofundar nossa análise, sendo necessário cruzar informações sobre duas variáveis epidemiológicas: identificação do agravo e o sexo do paciente. Dessa forma, deve-se incluir em seu relatório a distribuição das notificações dos agravos por sexo. Para isto, permaneceremos utilizando a função de agrupamento `group_by()`, para que calcule a frequência para cada grupo de categorias. Siga o *script* abaixo:

```
dados |>
```

```
#filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")  
# com a função filter()
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO) e indivíduos  
# por sexo (coluna CS_SEX0)
```

```
group_by(ID_AGRAVO, CS_SEX0) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n())
```

```
#> # A tibble: 5 × 3  
#> # Groups:   ID_AGRAVO [2]  
#>   ID_AGRAVO CS_SEX0 total  
#>   <fct>     <fct>   <int>  
#> 1 A90      F        6827  
#> 2 A90      I         11  
#> 3 A90      M        5943  
#> 4 B19      F         418  
#> 5 B19      M        537
```

Perceba que no *output* do *script* executado, conforme exemplo acima, as linhas foram ordenadas em ordem alfabética: primeiro para a coluna `ID_AGRAVO` (A90 e B19), e em seguida para a coluna `CS_SEX0` (F, I e M).

Podemos ordenar as linhas da tabela de acordo com o que desejarmos. Então, vamos lá ordenar. Para praticar, você irá ordenar as linhas de acordo com o número de registros (coluna `total`). Para este ordenamento utilizamos as funções `arrange()`, que ordena a coluna, e `desc()`, para indicar que a ordem será do maior valor para o menor. Veja:

```
dados |>
```

```
#filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO) e indivíduos
```

```
# por sexo (coluna CS_SEX0)
```

```
group_by(ID_AGRAVO, CS_SEX0) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n()) |>
```

```
# ordenando a frequência de casos (coluna total) em ordem decrescente com a  
# função arrange()
```

```
arrange(desc(total))
```

```
#> # A tibble: 5 × 3  
#> # Groups:   ID_AGRAVO [2]  
#>   ID_AGRAVO CS_SEX0 total  
#>   <fct>     <fct>   <int>  
#> 1 A90      F        6827  
#> 2 A90      M        5943  
#> 3 B19      M         537  
#> 4 B19      F         418  
#> 5 A90      I          11
```

No retorno acima, os campos com sexo **I** de ignorado estão todos concentrados no agravo **A90**. Isso demonstra que a variável **CS_SEX0** no agravo **B19** tem o preenchimento melhor.

Algo que pode ser valioso em algumas análises com R é pedir que ele retorne os valores que estão em branco nas colunas. Assim, você conseguirá saber todos os possíveis valores da sua coluna. Para este ordenamento, o pacote `dplyr` possui uma função auxiliar que agrupa os dados por uma categoria, e calcula a frequência de cada um simultaneamente. Esta função é `count()`. Agora vamos praticar:

1. Primeiro, utilize a função `filter()` para filtrar a tabela apenas os agravos de interesse;
2. Segundo, inclua a função `mutate()` e, dentro dela, a função `drop.levels()`. Isso fará que o R remova categorias (os chamados levels) que não atendem ao filtro;
3. Por fim, inclua a função `count()`, que necessitará receber o argumento `drop = FALSE` para contabilizar as categorias com frequência igual a zero (0).

Veja abaixo como ficaria a nova distribuição no *script* seguindo o passo-a-passo com os comandos descritos:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
```

```
# categorias (levels) em branco após o filtro usando a função drop.levels()
```

```
mutate(ID_AGRAVO = drop.levels(ID_AGRAVO)) |>
```

```
# contando o total de casos por agravo (coluna ID_AGRAVO) e sexo (coluna CS_SEX0)
```

```
# combinações de categorias com contagem 0 permanecem na tabela pelo uso do
```

```
# argumento .drop = FALSE
```

```
count(ID_AGRAVO, CS_SEX0, .drop = FALSE)
```

```
#>   ID_AGRAVO CS_SEXO    n
#> 1      A90      F 6827
#> 2      A90      I   11
#> 3      A90      M 5943
#> 4      B19      F   418
#> 5      B19      I     0
#> 6      B19      M   537
```

Observe que a nova tabela tem uma nova linha com a combinação `CS_SEXO = I`, referente ao agravo (`ID_AGRAVO = B19`), sendo o total de `I` igual a zero (0). Você deve ter percebido que utilizamos uma linha com a função `droplevels()`. Mas por que ela é necessária?

Mesmo utilizando a função `filter()` para filtrar apenas os agravos de interesse da tabela `{dados}`, a coluna `ID_AGRAVO` é uma variável do tipo factor, ou seja, a coluna retém a informação de todas as categorias (`levels`) existentes antes da filtragem. Para corrigir este detalhe e garantir que só serão retornados os agravos `A90` e `B19`, precisamos então utilizar a função `drop.levels()` dentro da função `mutate()`. Assim o comando utilizado foi: `mutate(ID_AGRAVO = droplevels(ID_AGRAVO))`.



Em alguns momentos da rotina da análise da vigilância em saúde poderá ser necessário ordenar categorias de uma forma diferente da ordem alfabética. Para isto, devemos transformar a coluna que queremos ordenar em um **fator ordenado**.

Lembre-se que no R *fatores* são um conjunto de categorias, e os valores que essas variáveis assumem são os níveis (*levels*), ou simplesmente as categorias que compõem o fator. Os fatores são necessários sempre que precisamos representar variáveis categóricas (como as de sexo: F, I, e M, ou raça/cor: Branco, Preto, Amarelo, Pardo e Indígena). Assim podem ser ordenados como desejamos.

Observe abaixo como seria construir um *script* para visualizar os dados de casos notificados por sexo na seguinte ordem: Feminino (F), Masculino (M), e Ignorado (I). Para esta ordenação utilizaremos a função **factor** e para especificar a ordem desejada utilizaremos o argumento **levels**, e por fim adicionaremos o argumento **ordered = TRUE**, para dizer que queremos ordená-los conforme os *levels*. Veja as linhas de script com os códigos necessários:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# utilizando a função mutate() para transformar a coluna CS_SEX0  
# em uma variável do tipo "factor", com três categorias ordenadas
```

```
mutate(CS_SEX0 = factor(CS_SEX0, levels = c("F", "M", "I"),  
ordered = TRUE)) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO) e  
# indivíduos por sexo (coluna CS_SEX0)
```

```
group_by(ID_AGRAVO, CS_SEX0) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n())
```

```
#> # A tibble: 5 × 3  
#> # Groups:   ID_AGRAVO [2]  
#>   ID_AGRAVO CS_SEX0 total  
#>   <fct>     <ord>   <int>  
#> 1 A90      F        6827  
#> 2 A90      M        5943  
#> 3 A90      I         11  
#> 4 B19      F        418  
#> 5 B19      M        537
```

Pronto, ordenamos nosso cálculo de frequência absoluta da coluna **CS_SEX0** tanto para o agravo **A90** quanto para o **B19**, e conseguimos ordenar o sexo também em ordem alfabética. Assim: F, M e I.

2.2 Frequência relativa ou porcentagem

Agora que sabemos como calcular a frequência absoluta dos valores de colunas específicas na tabela `{dados}`, um segundo passo a ser feito é calcular também a proporção de cada um desses agravos em nosso conjunto de dados.

Para saber esta proporção, vamos realizar um cálculo de divisão: dividindo a frequência absoluta de cada categoria (agravo) pela frequência absoluta total dos valores (soma de todas os notificados por dengue e hepatites não especificadas).

Abaixo, observe o script que possui os códigos de comando e perceba que adicionamos uma linha com a função `mutate()` para que seja possível dividir cada valor da coluna `total` pela soma geral da coluna (`sum(total)`). Veja abaixo os comandos e os resultados:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)
```

```
group_by(ID_AGRAVO) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n()) |>
```

```
# calculando a proporção de casos ao dividir o total pela soma de valores  
# total pelo uso das funções mutate() e sum()
```

```
mutate(proporcao = total / sum(total))
```

```
#> # A tibble: 2 × 3  
#>   ID_AGRAVO total proporcao  
#>   <fct>      <int>      <dbl>  
#> 1 A90        12781      0.930  
#> 2 B19         955      0.0695
```

Ao executarmos esta operação, obtemos uma tabela que contém a coluna “proporcao” com valores decimais entre 0 e 1 onde A90 possui 0,93 dos registros e B19 possui apenas 0,0695. Para facilitar a nossa visualização, convertemos esse valor para um valor do tipo porcentagem (0% a 100%), da seguinte forma:

1. Primeiro, realizamos os mesmos cálculos do exemplo anterior;
2. Segundo, multiplicamos o valor resultante da coluna **proporcao** por 100;
3. Terceiro, arredondamos o resultado em duas casas decimais utilizando a função **round()** e o seu argumento **digits = 2**.

Observe o *script* abaixo:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")  
# com a função filter()
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)
```

```
group_by(ID_AGRAVO) |>
```

```
# calculando a frequência de casos com as funções summarise() e n()
```

```
summarise(total = n()) |>
```

```
# calculando a proporção de casos ao dividir o total pela soma de valores total  
# pelo uso das funções mutate() e sum()
```

```
mutate(proporcao = total / sum(total)) |>
```

```
# criando uma nova coluna (porcentagem) com o uso da função mutate()  
# utilizando a função round() para arredondar em duas casas decimais a  
# porcentagem, após multiplicar a coluna "proporcao" por 100
```

```
mutate(porcentagem = round(proporcao * 100, digits = 2))
```

```
#> # A tibble: 2 × 4  
#>   ID_AGRAVO total proporcao porcentagem  
#>   <fct>      <int>      <dbl>      <dbl>  
#> 1 A90        12781      0.930      93.0  
#> 2 B19         955      0.0695     6.95
```

Pronto, agora temos uma tabela para ser adicionada à análise de situação da saúde de Rosas, com a frequência relativa dos dados entre os agravos A90 e B19, ou seja, a dengue (A90) possui 93% (proporção = 0,93) dos registros e as hepatites virais (B19) 6,95% (proporção = 0,0695).

Tente reproduzir esta etapa com o {NINDINET.dbf} de seu município ou estado e visualize as frequências absolutas e relativas avaliando quais os agravos de maior magnitude.

3. Tipo de variáveis

Muitas vezes, precisamos saber de forma rápida qual o maior valor ou o menor valor de uma coluna do banco de dados. Agora, vamos praticar e encontrar esses valores na tabela {dados} criada a partir da Ficha de Notificação Individual {NINDINET.dbf}.

Considere que seja necessário fazer um levantamento das datas mais recentes e mais antigas de notificação dos casos de dengue e das hepatites virais que estamos analisando.

Para isso, você irá utilizar as funções `max()` para encontrarmos os valores máximos de data de notificação (data recente), e `min()` para encontrar os valores mínimos (data antiga). Atente-se: os valores com as datas de notificações estão na coluna de nome `DT_NOTIFIC`, conforme você estudou no dicionário de dados.

Acompanhe abaixo a nossa avaliação, utilizando a mesma estrutura dos exemplos anteriores:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral (código "B19")  
# com a função filter()
```

```
filter(ID_AGRAVO %in% c("A90", "B19")) |>
```

```
# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as  
# categorias (levels) em branco após o filtro usando a função droplevels()
```

```
mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>
```

```
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)
```

```
group_by(ID_AGRAVO) |>
```

```
# calculando novos dados a partir das colunas agrupadas com o uso da função  
# summarise(), calculando a primeira e última data de notificação com as  
# funções min() e max() na coluna DT_NOTIFIC, respectivamente.
```

```
summarise(primeira_data = min(DT_NOTIFIC),  
          ultima_data = max(DT_NOTIFIC))
```

```
#> # A tibble: 2 × 3  
#>   ID_AGRAVO primeira_data ultima_data  
#>   <fct>      <date>      <date>  
#> 1 A90      2007-01-03      2012-12-23  
#> 2 B19      2007-01-04      2012-12-27
```

Observe que no agravo dengue (A90) a data mais antiga é a 03/01/2007 e a data mais recente é 23/12/2012 e para as hepatites virais (B19) a data mais antiga é 04/01/2007 e a mais atual é 27/12/2012. Perceba que, no R as datas são mostradas com o ano primeiro, depois o mês e, por último, o dia.



A composição do *script* é similar aos exercícios anteriores. Perceba que as funções utilizadas são bastante flexíveis e podem ser sempre adaptadas em diferentes contextos, funcionam tanto para valores numéricos quanto datas.

Agora que já sabemos qual a maior e a menor data de notificação por agravos, podemos aprofundar a análise e nos perguntar:

Qual seria a menor e maior idade dos indivíduos notificados (suspeitos e confirmados) para cada um destes agravos?

Vamos praticar para responder essa questão? Siga os passos abaixo:

1. Primeiro, faremos um passo auxiliar, onde calcularemos a idade em anos de cada caso conforme sua data de nascimento. Esta etapa aprendemos no Módulo 3 e vamos treinar aqui. Além disso, vamos utilizar algumas funções aninhadas, uma dentro da outra. Acompanhe o script abaixo com atenção:

```
# criando uma nova tabela (dataframe) chamada {`dados_idade`} que receberá as  
# transformações a seguir
```

```
dados_idade <- dados |>
```

```
# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,  
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre  
# data de primeiros sintomas e data de notificação e transformando em  
# número inteiro com a função as.integer() e seguido da divisão por 365.25,  
# e, no final, arredondamento para o menor número inteiro com uso da função floor()
```

```
mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25))
```

Perceba que para cálculo com datas, precisamos estar atentos às sequências de transformações e cálculos. Outro detalhe é que, ao explicarmos as funções do *script*, começamos de dentro pra fora, ou seja, da diferença entre datas, depois da divisão por 365.25, depois a transformação em número inteiro e só ao final falamos sobre o arredondamento com a função `floor()`. Isso é necessário, pois o R começa a realizar a operação nesta mesma sequência e, por isso, preferimos explicar dessa forma.

Vamos visualizar a coluna criada? Abaixo, estamos visualizando apenas algumas colunas (NU_NOTIFIC, DT_SIN_PRI, DT_NASC, IDADE_ANOS) da base criada anteriormente, utilizando a função `select()`:

```
# utilizando o head para visualizar as primeiras linhas da base {`dados_idade`}
# selecionando quatro colunas apenas: NU_NOTIFIC, DT_SIN_PRI, DT_NASC, IDADE_ANOS

head(dados_idade |> select(NU_NOTIFIC, DT_SIN_PRI, DT_NASC, IDADE_ANOS))
```

```
#>   NU_NOTIFIC DT_SIN_PRI   DT_NASC IDADE_ANOS
#> 1    7671320 2012-04-05 2012-04-04         0
#> 2    0855803 2010-09-09 1988-04-23        22
#> 3    8454645 2010-10-19 1971-03-25        39
#> 4    3282723 2008-04-11 1928-05-29        79
#> 5    9799526 2011-04-02 2002-09-18         8
#> 6    7275624 2008-02-06 1953-08-01        54
```

2. Agora, vamos praticar calculando a idade mínima e máxima de cada registro a partir do cálculo das idades revisado antes. Cuidado ao usar as funções `max()` e `min()`, pois se possuírmos alguns valores iguais a `NA` (nulo ou em branco), a função também irá retornar o valor `NA`. Para ignorarmos estes valores, devemos adicionar o argumento `na.rm = TRUE`. Veja o *script* abaixo e, novamente, **vá com atenção**:

```
# criando uma nova tabela (dataframe) chamada {`dados_idade_2`}

dados_idade_2 <- dados |>

# filtrando os agravos de dengue (código "A90") e hepatite viral
# (código "B19") com a função filter()

filter(ID_AGRAVO %in% c("A90", "B19")) |>

# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
# categorias (levels) em branco após o filtro usando a função droplevels()

mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>

# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)

group_by(ID_AGRAVO) |>

# calculando novos dados a partir das colunas agrupadas com o uso da função
# summarise(), calculando a menor e maior idade com as funções min() e max()
# na coluna IDADE_ANOS, respectivamente.

summarise(
  menor_idade = min(IDADE_ANOS, na.rm = TRUE),
  maior_idade = max(IDADE_ANOS, na.rm = TRUE)
)
```

Vamos agora visualizar a tabela recém-criada. Perceba que não precisamos usar a função `head()` aqui, pois no *script* acima, nós resumizamos a base `{dados}` e, por isso, a visualização dela fica mais agradável e sem a necessidade de utilizar outras funções. Basta digitar o nome da tabela e clicar no botão “Run”.

```
# visualizando a tabela {`dados_idade_2`}
```

```
dados_idade_2
```

```
#> # A tibble: 2 × 3  
#>   ID_AGRAVO menor_idade maior_idade  
#>   <fct>         <dbl>         <dbl>  
#> 1 A90             0             98  
#> 2 B19             0             89
```

Observe que o resultado foi que as menores idades para as notificações de dengue (A90) são 0 anos e a maior foi 98 anos, para B19 a menor idade também foi 0 anos e maior idade encontrada foi 89 anos, ignorando os valores nulos.



ATENÇÃO

Lembre-se durante o seu cálculo de idade de desconsiderar os valores ignorados e ou todos os valores `nulos/vazios` ou `NA` utilizando o argumento: `na.rm = TRUE` que você aprendeu, anteriormente.

4. Cálculo de medidas-resumo (média, mediana e quartis)

Além dos valores máximos ou mínimos, podemos calcular outras estatísticas de interesse epidemiológico para seguir com a análise dos dados do Estado de Rosas. Neste tópico, iremos conhecer medidas como médias, medianas e quartis, de forma semelhante ao que vimos anteriormente.

O cálculo destas medidas é uma adaptação das funções utilizadas. Listamos abaixo as funções específicas de cálculo de medidas-resumo para que você possa guardá-las para replicá-las na sua rotina de análise de dados no serviço de vigilância em saúde:

- `mean()`: média
- `median()`: mediana
- `quantile(x, probs)`: quantis. É possível especificar o intervalo de quantil pelo argumento `probs`, e indicar um valor entre 0 e 1.

Na sessão anterior calculamos as idades e os valores máximos e mínimos dos casos de dengue e hepatites de origem desconhecida do Estado de Rosas. Que tal calcular as distribuições de idades de nossos pacientes? Vamos lá!

1. Criaremos uma tabela `{dados_descricao}` a partir da tabela `{dados}` para esta etapa. Lembre-se que criar tabelas para análises específicas é uma boa prática para não perder informações durante as análises no `R`, tornando sua análise reproduzível, inclusive.
2. Calcularemos a média, a mediana e o primeiro quartil (25%) destas idades.

Observe os comandos e os retornos abaixo:

```
# criando uma nova tabela (dataframe) chamada {`dados_descricao`}

dados_descricao <- dados |>

# filtrando os agravos de dengue (código "A90") e hepatite viral
# (código "B19") com a função filter()

filter(ID_AGRAVO %in% c("A90", "B19")) |>

# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
# categorias (levels) em branco após o filtro usando a função droplevels()

mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>

# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)

group_by(ID_AGRAVO) |>

# calculando novos dados a partir das colunas agrupadas com o uso da função
# summarise()

summarise(

  # calculando a média de idade com a função mean() na coluna IDADE_ANOS

  media_idade = mean(IDADE_ANOS, na.rm = TRUE),

  # calculando a mediana de idade com a função median() na coluna IDADE_ANOS

  mediana_idade = median(IDADE_ANOS, na.rm = TRUE),

  # calculando o quantil 25 da distribuição de idade com a função quantile()
  # na coluna IDADE_ANOS e o argumento prob = 0.25

  quantil_25_idade = quantile(IDADE_ANOS, prob = 0.25, na.rm = TRUE)
)
```

Agora, vamos visualizar a tabela criada digitando o nome da tabela e clicando no botão “Run”.

```
# visualizando a tabela {`dados_descricao`}
```

```
dados_descricao
```

```
#> # A tibble: 2 × 4  
#>   ID_AGRAVO media_idade mediana_idade quantil_25_idade  
#>   <fct>         <dbl>         <dbl>         <dbl>  
#> 1 A90          26.9           24           12  
#> 2 B19          42.9           45           31
```

Observe que calculamos a média, mediana e quartil de 25% das idades em anos escrevendo um *script*. O resultado foi que as idades são menores para os casos notificados de dengue se comparados aos casos de hepatite. Além disso, percebemos que, pelo quartil, nos casos notificados de dengue, 25% dos registros são iguais ou menores que 12 anos, e nos casos notificados de hepatite, 25% dos registros são iguais ou menores que 31 anos.



E agora!? Como seria calcular a moda?

Moda é uma medida de posição usada no resumo de dados para identificar o valor mais frequente em um conjunto. Não há função nativa no R específica para o cálculo da moda. Há pacotes que abrangem esta função como, por exemplo, a função `Mode()` do pacote `DescTools`. O algoritmo de identificação do valor mais frequente funciona tanto para valores numéricos, quanto para valores textuais.

Acompanhe o exemplo abaixo em que identificaremos a idade e o sexo mais frequente entre os agravos Dengue (CID A90), Hepatites Virais (CID B19) e Tuberculose (CID A169). Veja os comandos a serem executados e suas respostas. Lembre-se que já carregamos o pacote `DescTools` no início desse módulo.

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90"), hepatite viral  
# (código "B19") e Tuberculose (código "A169")  
filter(ID_AGRAVO %in% c("A90", "B19", "A169")) |>  
  
# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as  
# categorias (levels) em branco após o filtro usando a função droplevels()  
mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>  
  
# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,  
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre  
# data de primeiros sintomas e data de notificação e transformando em  
# número inteiro com a função as.integer(), seguido da divisão por 365.25,  
# e, no final, arredondamento para o menor número inteiro com uso da função  
# floor()  
mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>  
  
# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)  
group_by(ID_AGRAVO) |>  
  
# calculando novos dados a partir das colunas agrupadas com o uso da função  
# summarise()  
summarise(  
  
  # calculando a moda da idade com a função Mode() na coluna IDADE_ANOS  
  moda_idade = Mode(IDADE_ANOS, na.rm = TRUE),  
  
  # calculando a moda da categoria sexo com a função Mode() na coluna CS_SEX0  
  moda_sexo = Mode(CS_SEX0, na.rm = TRUE)  
)
```



```
#> # A tibble: 3 × 3  
#>   ID_AGRAVO moda_idade moda_sexo  
#>   <fct>         <dbl> <fct>  
#> 1 A169             30 M  
#> 2 A90              12 F  
#> 3 B19             58 M
```

Observe que o resultado foi que, nos casos de tuberculose sexo masculino e idade de 30 anos foram os mais frequentes. Nos casos notificados de dengue, sexo feminino e idade de 12 anos foram os mais frequentes e, para os casos notificados de hepatite, os mais frequentes foram sexo masculino e idade de 58 anos.

Com esta etapa de análise exploratória de dados na vigilância em saúde, já é possível compreender a distribuição geral dos seus dados do Sinan Net. Experimente praticar esta etapa para outros agravos notificados!

5. Cálculo de medidas de dispersão (variância e desvio padrão)

Algumas vezes, nas análises epidemiológicas, precisamos avaliar se os valores apresentados em um conjunto de dados estão dispersos ou não e o quão distantes um do outro eles podem estar, ou seja, estamos falando da variância e do desvio padrão que são medidas estratégicas para esta avaliação.

Estes cálculos nos apoiam a responder, por exemplo, se a maior parte dos casos de notificações de Rosas tem idade maior ou menor que a média das idades. Afinal, podem ter médias aritméticas idênticas e, ao mesmo tempo, possuir valores distribuídos de forma diferentes em relação à média. Para isso calculamos estas medidas:

- *Variância*: leva em conta todos os valores de uma distribuição para seu cálculo. Ela é estimada a partir do somatório do quadrado da distância de cada valor em relação à média, dividido pelo total de observações menos um.
- *Desvio Padrão*: é estimativa do quanto, em média, cada valor se distancia da própria média aritmética de uma distribuição. Para calculá-lo, basta extrair a raiz quadrada da fórmula da variância.

No **R**, essas medidas de dispersão podem ser calculadas facilmente utilizando as seguintes funções: `var()` e `sd()`, respectivamente. Observe o *script* abaixo com atenção e faça você também:

```
# criando uma nova tabela (dataframe) chamada {`dados_var`} a partir da tabela {`dados`}

dados_var <- dados |>

# filtrando os agravos de dengue (código "A90") e hepatite viral
# (código "B19") com a função filter()

filter(ID_AGRAVO %in% c("A90", "B19")) |>

# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
# categorias (levels) em branco após o filtro usando a função droplevels()

mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) /
365.25)) |>

# agrupando com a função group_by() os agravos (coluna ID_AGRAVO)

group_by(ID_AGRAVO) |>

# calculando novos dados a partir das colunas agrupadas com o uso da função
# summarise()

summarise(

  # calculando a variância da idade com a função var() na coluna IDADE_ANOS

  variancia_idade = var(IDADE_ANOS, na.rm = TRUE),

  # calculando o desvio padrão da idade com a função sd() na coluna IDADE_ANOS

  desvio_padrao_idade = sd(IDADE_ANOS, na.rm = TRUE)
)
```

Agora, vamos visualizar a tabela criada digitando o nome da tabela e clicando no botão “Run”.

```
# visualizando a tabela {`dados_var`}
```

```
dados_var
```

```
#> # A tibble: 2 × 3  
#>   ID_AGRAVO variancia_idade desvio_padrao_idade  
#>   <fct>          <dbl>          <dbl>  
#> 1 A90             344.             18.6  
#> 2 B19             323.             18.0
```

Observe que ao estudar a dengue e as hepatites de origem desconhecidas, o agravo **A90** possui uma variância de 344 casos com idades acima da média (26,9 anos) e 18,6 é o desvio padrão da idade, enquanto o **B19** possui 323 casos com idades acima da média (42,9 anos) e podendo variar +ou- 18 anos nesta amostra (desvio padrão encontrado).



Lembre-se que o desvio padrão, é a raiz quadrada da variância.

6. Avaliando a distribuição dos dados

Já estudamos que todos os cálculos apresentados podem ser feitos com comandos específicos no **R**. Alguns destes cálculos podem ser obtidos com apenas um comando utilizando a função `summary()`. Ela é muito utilizada, pois automatiza a visão geral das variáveis do banco de dados analisado permitindo ao profissional de vigilância conhecer as medidas resumo de uma só vez, ou seja, agiliza o trabalho.

Agora, em todas as avaliações que você fizer da distribuição dos dados analisados, o comando `summary()` se tornará um argumento essencial para análise. Ele resulta em uma espécie de “sumarização” de cada variável conforme seu tipo.

Observe no *script* abaixo a avaliação que faremos da nossa tabela `{dados}`, oriunda do banco de dados `{NINDINET.dbf}` disponível no menu lateral “Arquivos”, do módulo, vamos analisar apenas as variáveis: `ID_AGRAVO`, `DT_NOTIFIC`, `DT_SIN_PRI`, `IDADE_ANOS` e `CS_SEX0`. As variáveis numéricas serão apresentadas segundo as métricas mínimo, máximo, média, mediana, primeiro e terceiro quartis. Entretanto, lembre-se que no caso das variáveis do tipo fator (`fct`), o argumento `summary()` retornará apenas a frequência de cada categoria.

Acompanhe os *script* abaixo com três exemplos para praticar junto ao seu **RStudio**:

- No primeiro exemplo selecionamos algumas variáveis utilizadas anteriormente, mas sem agrupar ou filtrar agravos.
- No segundo exemplo adicionamos ao filtro, a seleção dos registros notificados de dengue.
- E, por fim no terceiro exemplo filtramos apenas os casos notificados por hepatite viral.

Vamos lá!

Observe o *output* do código abaixo quando não filtramos um agravo:

```
# 1º exemplo: sem utilização de filtro pelo agravo

dados |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função
# floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) /
365.25)) |>

# selecionando as colunas ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS
# e CS_SEX0

select(ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS, CS_SEX0) |>

# resumizando o banco utilizando a função summary()

summary()
```

```
#>   ID_AGRAVO      DT_NOTIFIC      DT_SIN_PRI      IDADE_ANOS
#> A90      :12781   Min.      :2007-01-01   Min.      :1939-04-24   Min.      : 0.00
#> W64      : 5438   1st Qu.:2008-03-28   1st Qu.:2008-03-17   1st Qu.: 15.00
#> A169     : 2347   Median :2009-06-18   Median :2009-04-23   Median : 28.00
#> X58      : 1606   Mean      :2009-09-24   Mean      :2009-08-01   Mean      : 30.26
#> B19      :  955   3rd Qu.:2011-04-20   3rd Qu.:2011-04-09   3rd Qu.: 44.00
#> B24      :  675   Max.      :2012-12-30   Max.      :2012-12-29   Max.      :101.00
#> (Other): 3819                                     NA's      :4304
#> CS_SEX0
#> F:13567
#> I:  56
#> M:13998
#>
#>
#>
#>
```

Perceba que no primeiro exemplo acima, a tabela visualizada apresenta as variáveis `ID_AGRAVO` e `CS_SEX0` com suas respectivas frequências e as variáveis `DT_NOTIFIC`, `DT_SIN_PR` e `IDADE_ANOS` com as estatísticas descritivas básicas (mínimo, 1º quartil, mediana, média, 3º quartil e máxima).

Agora, acompanhe o *output* do código abaixo quando filtramos os registros notificados com código CID10 = A90, replique os passos em seu `RStudio`:

```
# 2º exemplo: filtro pelo agravo dengue ("A90")

dados |>

# filtrando os agravos de dengue (código "A90") com a função filter()

filter(ID_AGRAVO == "A90") |>

# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
# categorias (levels) em branco após o filtro usando a função droplevels()

mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>

# selecionando as colunas ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS
# e CS_SEX0

select(ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS, CS_SEX0) |>

# resumizando o banco utilizando a função summary()

summary()
```

```
#> ID_AGRAVO      DT_NOTIFIC      DT_SIN_PRI      IDADE_ANOS      CS_  
SEXO  
#> A90:12781  Min.    :2007-01-03  Min.    :1993-06-23  Min.    : 0.00  F:6827  
#>           1st Qu.:2008-03-13  1st Qu.:2008-03-06  1st Qu.:12.00  I:  11  
#>           Median :2008-04-20  Median :2008-04-11  Median :24.00  M:5943  
#>           Mean   :2009-04-03  Mean   :2009-03-24  Mean   :26.93  
#>           3rd Qu.:2011-03-31  3rd Qu.:2011-03-27  3rd Qu.:39.00  
#>           Max.   :2012-12-23  Max.   :2012-12-21  Max.   :98.00  
#>                                     NA's   :3364
```

Você conseguiu executar? A tabela apresentada em seu **Rstudio** no painel **console**, deve ser igual ao que visualizamos aqui. Perceba que estamos analisando agora as estatísticas descritivas básicas dos casos de Dengue notificados em Rosas:

- Temos 12.781 casos notificados.
- A menor data de notificação foi em 03/01/2007 e a maioria destes casos foram notificados em 20/04/2008.
- A última data de início de sintomas registrada no sistema foi em 21/12/2012, mas a média dos casos foi notificada em 24/03/2009.
- Quanto as idades dos casos, a média (26,9 anos) e a mediana (24 anos), sendo 98 anos a maior idade entre os notificados, e 3.364 dos casos não tiveram a idade relatada.
- A maioria dos notificados eram do sexo feminino (6.827 casos).

Agora vamos praticar com um último exemplo. Observe o output do código abaixo em que filtraremos os registros notificados com código CID10 = B19, replique em seu **RStudio**:


```
# 3º exemplo: filtro pelo agravo hepatite viral ("B19")

dados |>

# filtrando os agravos de hepatite viral (código "B19") com a função filter()

filter(ID_AGRAVO == "B19") |>

# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as
# categorias (levels) em branco após o filtro usando a função droplevels()

mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>

# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre
# data de primeiros sintomas e data de notificação e transformando em
# número inteiro com a função as.integer(), seguido da divisão por 365.25,
# e, no final, arredondamento para o menor número inteiro com uso da função floor()

mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>

# selecionando as colunas ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS
# e CS_SEX0

select(ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS, CS_SEX0) |>

# resumizando o banco utilizando a função summary()

summary()
```

```
#> ID_AGRAVO DT_NOTIFIC DT_SIN_PRI IDADE_ANOS CS_SEX0
#> B19:955 Min. :2007-01-04 Min. :1951-08-13 Min. : 0.00 F:418
#> 1st Qu.:2008-11-09 1st Qu.:2008-06-25 1st Qu.:31.00 I: 0
#> Median :2009-10-23 Median :2009-07-14 Median :45.00 M:537
#> Mean :2009-12-25 Mean :2009-07-01 Mean :42.92
#> 3rd Qu.:2011-04-12 3rd Qu.:2011-01-11 3rd Qu.:56.00
#> Max. :2012-12-27 Max. :2012-12-20 Max. :89.00
#>
```

Observe que visualizamos a tabela com as estatísticas descritivas básicas dos casos notificados com hepatites virais do Estado de Rosas. Tente descrever os dados que visualiza, será um bom exercício para memorização e aprendizagem!



Se houver uma grande quantidade de variáveis contidas na base de dados analisada, a visualização dos *outputs* (resultados) da função `summary()` poderá ficar comprometida.

Nestes casos, você pode utilizar a função de forma individualizada, ou seja, escrevemos os comandos para cada variável de interesse da seguinte forma:

```
# utilizando a função summary() para  
# visualizar a coluna CS_SEX0 da tabela {`dados`}  
summary(dados$CS_SEX0)
```

```
#>      F      I      M  
#> 13567    56 13998
```

Você também poderá utilizar o pacote `summarytools` para melhorar a visualização do comando executado. Este pacote possui a função `dfSummary()`. Ela apresenta um resumo dos dados em forma de tabela, contendo os nomes, tipos, frequência, resumo numérico, gráfico histograma, a porcentagem de registros em branco e registros preenchidos (válidos). Isso torna a leitura dos resultados mais agradável.

Para utilizá-la você deve utilizar o argumento `view()`, pois ele permitirá a visualização do relatório elaborado pelo comando na aba **Viewer** do **RStudio**, no formato de relatório em HTML (página da web). A função também é simples, tendo como argumento apenas a base de dados.

Acompanhe o *script* abaixo e replique em seu **RStudio**:

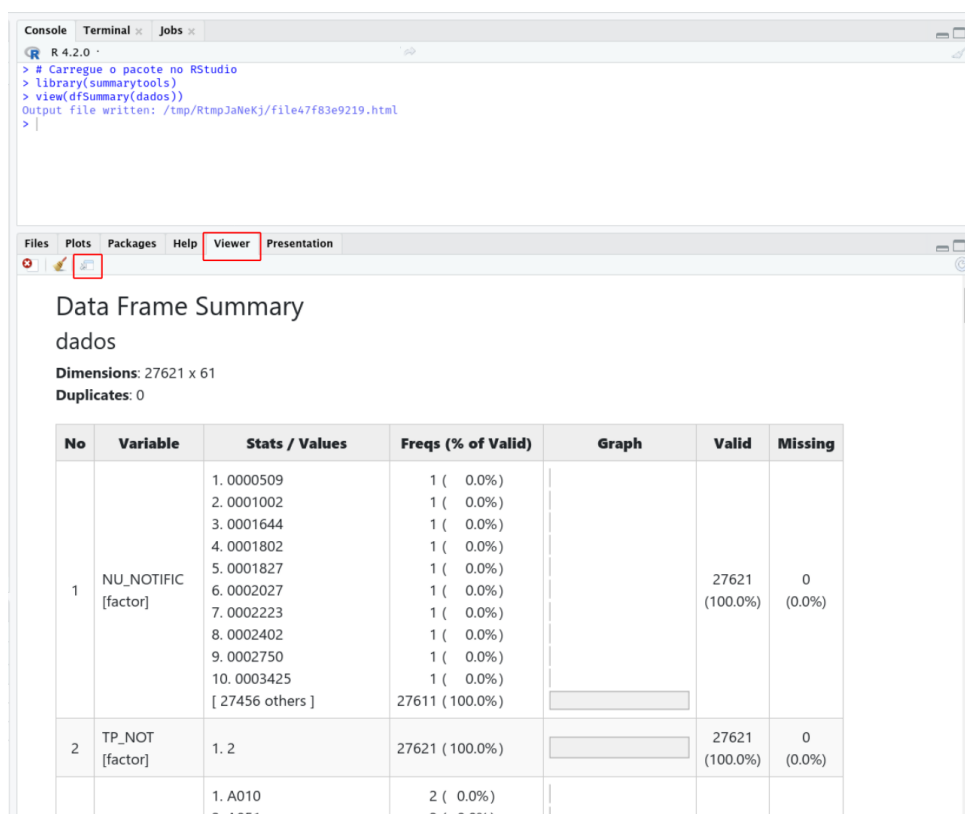
```
dados |>
```

```
# criando uma nova coluna chamada IDADE_ANOS com a função mutate() e, nela,  
# calculando a idade em anos. Primeiro, fazendo a diferença em dias entre  
# data de primeiros sintomas e data de notificação e transformando em  
# número inteiro com a função as.integer(), seguido da divisão por 365.25,  
# e, no final, arredondamento para o menor número inteiro com uso da função  
# floor()  
mutate(IDADE_ANOS = floor(as.integer(DT_SIN_PRI - DT_NASC) / 365.25)) |>  
  
# selecionando as colunas ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS  
# e CS_SEX0  
select(ID_AGRAVO, DT_NOTIFIC, DT_SIN_PRI, IDADE_ANOS, CS_SEX0) |>  
  
# criando o sumário com a função dfSummary()  
dfSummary() |>  
  
# visualizando o relatório em formato HTML com função view()  
view()
```

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Grafo	Faltante
ID_AGRAVO [factor]	1. A010 2. A051 3. A059 4. A09 5. A169 6. A279 7. A309 8. A35 9. A379 10. A509 [53 outros]	2 (0.0%) 2 (0.0%) 2 (0.0%) 4 (0.0%) 2347 (8.5%) 49 (0.2%) 266 (1.0%) 4 (0.0%) 26 (0.1%) 329 (1.2%) 24590 (89.0%)		0 (0.0%)
DT_NOTIFIC [Date]	mín : 2007-01-01 mediana : 2009-06-18 máx : 2012-12-30 range : 5y 11m 29d	2103 valores distintos		0 (0.0%)
DT_SIN_PRI [Date]	mín : 1939-04-24 mediana : 2009-04-23 máx : 2012-12-29 range : 73y 8m 5d	2478 valores distintos		0 (0.0%)
IDADE_ANOS [numeric]	Média (dp) : 30.3 (19.5) mín < mediana < máx: 0 < 28 < 101 IQE (CV) : 29 (0.6)	99 valores distintos		4304 (15.6%)
CS_SEXO [factor]	1. F 2. I 3. M	13567 (49.1%) 56 (0.2%) 13998 (50.7%)		

O relatório gerado é apresentado no formato HTML, e pode ser visualizado em um navegador de internet, clicando no ícone do painel **Viewer** conforme a Figura 4:

Figura 3: Tela de visualização da aba Viewer.



Observe que a tabela acima apresentou a visualização das seguintes análises em cada coluna:

- **Variável:** indicando qual é a variável estudada;
- **Estatísticas ou Valores:** indicando a frequência absoluta de cada registro da variável estudada,
- **Freqs (% de válidos):** indicando a frequência absoluta e relativa de cada registro da variável estudada,
- **Graph:** visualização gráfica da distribuição dos dados da variável estudada,
- **Valid:** indicando a frequência absoluta e relativa dos valores considerados validos da variável estudada e
- **Faltante:** indicando a frequência absoluta e relativa dos valores considerados nulos ou em branco da variável estudada.

O que você achou depois de executar os comandos? Esta visualização é mais agradável, não é mesmo? Pratique com os bancos de dados que você utiliza na vigilância.



Para geração de tabelas que possuam variáveis calculadas incluindo cálculos de porcentagens e médias, o pacote `gtsummary` é muito útil. Afinal, com poucas linhas de comando, será possível gerar uma bela tabela para análises. Veja e acompanhe os passos abaixo:

1. Primeiro, filtramos com a função `filter()` os casos notificados de dengue e hepatite virais, notificados no Estado de Rosas.
2. Segundo, eliminamos os valores nulos com a função `droplevels()`.
3. No terceiro passo, selecionamos com a função `select()` as variável (`CS_SEX0`) que utilizaremos para a análise dos agravos.
4. Por fim, utilizamos a função `tbl_summary()`, incluindo o argumento `by = CS_SEX0` que permitirá que os agravos selecionados no passo 1, sejam cruzados com a variável sexo.

Assim, teremos como resultado uma tabela com estatísticas descritivas já calculadas. Observe o *script* abaixo:

```
dados |>
```

```
# filtrando os agravos de dengue (código "A90") e hepatite viral  
# (código "B19") com a função filter()  
filter(ID_AGRAVO %in% c("A90", "B19")) |>  
  
# utilizando a função mutate() para modificar a coluna ID_AGRAVO, removendo as  
# categorias (levels) em branco após o filtro usando a função droplevels()  
mutate(ID_AGRAVO = droplevels(ID_AGRAVO)) |>  
  
# selecionando apenas as colunas de agravo (ID_AGRAVO) e sexo (CS_SEX0)  
select(ID_AGRAVO, CS_SEX0) |>  
  
# gerando uma tabela com resumo das informações cruzando  
# do agravo (ID_AGRAVO) pelo sexo (CS_SEX0)  
tbl_summary(by = CS_SEX0)
```

Características	F, N = 7,245 ¹	I, N = 11 ¹	M, N = 6,480 ¹
ID_AGRAVO			
A90	6,827 (94%)	11 (100%)	5,943 (92%)
B19	418 (5.8%)	0 (0%)	537 (8.3%)
¹ n (%)			

A tabela gerada poderá ser visualizada no painel **Viewer** do seu **RStudio** (ela não será visualizada no painel **Console**). Agora vamos interpretar os resultados:

- **N** é o número de valores totais por categorias de sexo, está localizado no cabeçalho da tabela.
- **n** é o número total de casos por agravo, está sinalizado com ¹ no rodapé da tabela.
- (%) é a porcentagem total por agravo, está sinalizado com ¹ no rodapé da tabela.

Para a visualização dessas informações apresentadas na tabela no idioma português, precisamos definir o idioma português como padrão. Siga o *script* com o comando abaixo configure esse padrão.

```
# definindo o idioma das tabelas geradas em pacote "gtsummary" como português ("pt")
theme_gtsummary_language("pt")
```



Próximo módulo

Parabéns, você chegou ao final do nosso quarto módulo! Agora você já conhece todas as funções básicas para manipular banco de dados e fazer cálculos epidemiológicos utilizando a linguagem **R**. No próximo módulo você irá colocar em prática cálculos de indicadores de saúde necessários para estabelecer rotinas de trabalho para o seu dia a dia na vigilância em saúde.

Até lá.

