Primeiros passos com atlasdasviolenciaR

```
library(atlasviolenciaR)
```

O primeiro passo para usar o `atlasdasviolenciaR` é instalar e carregar o pacote. Até o momento, só é possível instalar através do Github. No futuro, disponibilizaremos um bundle em `.zip` para instalação local.

```
if (!require("devtools")) install.packages("devtools")
devtools::install_github("willbernascimento/atlasviolenciaR")

library(atlasdasviolenciaR)
```

Obtendo os dados

Para obter os dados você só precisa usar as `get_series_values` e `get_series_values_regions` e passar os parâmetros necessários. É importante entender que a API organiza os dados como séries de dados. Nesse sentido, cada tipo de dados está organizado e disponibilizado como uma série, com seu próprio ID e Título.

Para baixar os dados da série *Taxa de óbitos em acidentes de transporte - Faixa etária de 15 - 29 anos* por UF/Estado, precisamos passar o ID da série e a abrangência geográfica.

```
df_serie = get_series_values(serie_id=156,geographic_scope = 3)
#> No encoding supplied: defaulting to UTF-8.
```

```
head(df_serie)
#> # A tibble: 6 x 4
#>   cod   sigla valor periodo
#>   <chr> <chr> <chr> <chr>
#> 1 53    DF    43.43 1989-01-15
#> 2 52    GO    39.26 1989-01-15
#> 3 11    RO    40.47 1989-01-15
#> 4 12    AC    12.67 1989-01-15
#> 5 13    AM    19.15 1989-01-15
#> 6 14    RR    26.44 1989-01-15
```

`get_series_values` retorna um `data.frame` com os dados da série especificada (aqui a `156`) para a área geográfica especificada (`3` = Estado). No momento, a chamada coleta todos os dados disponíveis, no futuro pretendemos permitir que o usuário possa delimitar períodos de tempo.

Para saber informações acerca dessa série, podemos usar usar `get_serie` e passarmos o ID como parâmetro.

```
get_serie(serie_id = 156)
#> # A tibble: 1 x 2
#>      id titulo
#>   <int> <chr>
#> 1   156 Taxa de óbitos em acidentes de transporte - Faixa etária de 15 - 29 anos
```

Agora sabemos o título da série que buscamos.

Note que, nos dois exemplos, precisamos passar um ID válido para a busca funcionar. Além disso, precisamos saber, a priori, o `ID` que desejamos buscar. Para facilitar essa tarefa, podemos usar as demais funções para obtermos informações sobre os dados disponíveis.

Obtendo informações

Podemos buscar informações sobre os temas das séries disponíveis com `get_themes`. Note que não é necessário passar argumentos nessa função.

```
get_themes()
#> # A tibble: 12 x 6
#>       id titulo                                    tema_id imagem    tipo subTe~1
#>    <int> <chr>                                       <int> <chr>    <int> <list>
#>  1     1 Homicídios                                      0 9024-01~     0 <list>
#>  2     2 Juventude Perdida                               0 8201-02~     0 <list>
#>  3     4 Mortes Violentas por Causa Indeterminada         0 1846-05~     0 <list>
#>  4     5 Óbitos por Armas de Fogo                        0 8234-06~     0 <list>
#>  5    15 Óbitos por Causas Externas                      0 7662-08~     0 <list>
#>  6    16 Suicídio                                        0 1351-07~     0 <list>
#>  7    58 Violência Física                                0 1798-11~     0 <list>
#>  8    12 Violência no Trânsito                           0 6631-09~     0 <list>
#>  9    61 Violência por Gênero                            0 7247-04~     0 <list>
#> 10     3 Violência por Raça                              0 6006-03~     0 <list>
#> 11    60 Violência Psicológica                           0 3389-12~     0 <list>
#> 12    59 Violência Sexual                                0 8898-10~     0 <list>
#> # ... with abbreviated variable name 1: subTemas
```

`get_themes` retorna um `data.frame` com IDs de temas disponíveis. As séries estão agrupadas dentro dos temas. Para sabermos as séries disponíveis dentro dos temas podemos usar a função `get_series` e passar o ID do tema desejado. Vamos buscar as séries disponíveis sobre o tema *Óbitos por Armas de Fogo*, que tem como `ID = 5`.

```
get_series(theme_id = 5)
#> # A tibble: 26 x 2
#>       id titulo
#>    <int> <chr>
#>  1   160 Homicídios de Homens por Armas de Fogo
#>  2   162 Homicídios de Mulheres por Armas de Fogo
#>  3    31 Homicídios por Armas de Fogo
#>  4   183 Óbitos de Homens por Armas de Fogo
#>  5   185 Óbitos de Jovens por Armas de Fogo
#>  6   184 Óbitos de Mulheres por Armas de Fogo
#>  7   180 Óbitos por Armas de Fogo
#>  8   193 Proporção Óbitos por Arma de Fogo de Homens ao Total de Óbitos por Arm~
#>  9   195 Proporção Óbitos por Arma de Fogo de Jovens ao Total de Óbitos por Arm~
#> 10   194 Proporção Óbitos por Arma de Fogo de Mulheres ao Total de Óbitos por A~
#> # ... with 16 more rows
```

`get_themes` retorna um `data.frame` com IDs e Títulos de diversas séries sobre o tema pesquisado.

Para coletarmos os dados dessa série, podemos usar novamente `get_series_values` ou `get_series_values_regions` se queremos escolher regiões específicas.

Vamos baixar os dados da série *ID=185* que é sobre *Óbitos de Jovens por Armas de Fogo*. Vamos baixar a série para todas as regiões geográficas do país.

```
df_serie_regiao = get_series_values(serie_id=185,geographic_scope = 2)

head(df_serie_regiao)
#> # A tibble: 6 x 4
#>   cod   sigla valor periodo
#>   <chr> <chr> <chr> <chr>
#> 1 5     CO    563   1989-01-15
#> 2 3     SE    4063  1989-01-15
#> 3 4     S     885   1989-01-15
#> 4 2     NE    1904  1989-01-15
#> 5 1     N     450   1989-01-15
#> 6 5     CO    526   1990-01-15
```

geographic_scope e regions:

Como você pode ver na documentação das funções, `geographic_scope` pode receber `1` para País; `2` para Região; `3` para UF (Estado) e `4` para Município. Ao usar `get_series_values` você deve passar apenas a abrangência e os dados serão retornados.

Contudo, ao usar `get_series_values_regions` será necessário passar o argumento para `regions` e será necessário que seja uma entrada válida, ou será retornado um erro.

`regions` recebe o código de cada unidade regional sendo buscada. Para `geographic_scope` 3 e 4, esse código refere-se ao identificador usado pelo IBGE para identificar cada unidade. Para os estados/UF, teremos 27 códigos e para os municípios, teremos aproximadamente 5.600. Você pode buscar esses códigos em https://www.ibge.gov.br/explica/codigos-dos-municipios.php

OBS É importante notar que os códigos de `regions` depende dos valores de `geographic_scope`.

Quando `geographic_scope` é 1: apenas dados do Brasil estão disponíveis. O código de `regions` é 1076.

Quando `geographic_scope` é 2: podemos usar os identificadores de 1 a 5 para cada região. Sendo 1: Norte; 2: Nordeste e assim por diante

Quando `geographic_scope` é 3: podemos usar os identificadores do IBGE para as UF/Estados. Sendo 12: Acre; 27: Alagoas e assim por diante.

Quando `geographic_scope` é 4: podemos usar os identificadores do IBGE para os munnicípios. Sendo 1200013: Acrelândia; 1200054: Assis Brasil e assim por diante.

Você pode buscar esses códigos em https://www.ibge.gov.br/explica/codigos-dos-municipios.php

Vamos baixar os dados da série 185 para o município de Acrelândia:

```
get_series_values_regions(serie_id=185,geographic_scope = 4, regions = 1200013)
#> No encoding supplied: defaulting to UTF-8.
#> # A tibble: 20 x 4
#>    cod     sigla      valor periodo
#>    <chr>   <chr>      <chr> <chr>
#>  1 1200013 Acrelândia 0     2000-01-15
#>  2 1200013 Acrelândia 0     2001-01-15
#>  3 1200013 Acrelândia 0     2002-01-15
#>  4 1200013 Acrelândia 0     2003-01-15
```

```
#>  5 1200013 Acrelândia 0     2004-01-15
#>  6 1200013 Acrelândia 1     2005-01-15
#>  7 1200013 Acrelândia 0     2006-01-15
#>  8 1200013 Acrelândia 0     2007-01-15
#>  9 1200013 Acrelândia 0     2008-01-15
#> 10 1200013 Acrelândia 0     2009-01-15
#> 11 1200013 Acrelândia 0     2010-01-15
#> 12 1200013 Acrelândia 0     2011-01-15
#> 13 1200013 Acrelândia 0     2012-01-15
#> 14 1200013 Acrelândia 0     2013-01-15
#> 15 1200013 Acrelândia 0     2014-01-15
#> 16 1200013 Acrelândia 0     2015-01-15
#> 17 1200013 Acrelândia 2     2016-01-15
#> 18 1200013 Acrelândia 0     2017-01-15
#> 19 1200013 Acrelândia 4     2018-01-15
#> 20 1200013 Acrelândia 2     2019-01-15
```

Demais funções `get_` no plural

As funções que terminam no plural e não possuem parâmetros, tais como , tais como `get_themes` ou `get_fonts` retornam um data.frame com o ID e o Título da informação acerca de um tema, fonte de dados, ou periodicidade disponível.

Por exemplo,

```
get_fonts()
#> # A tibble: 6 x 2
#>      id titulo
#>   <int> <chr>
#> 1     3 Banco Mundial
#> 2     7 Instituto Brasileiro de Geografia e Estatística - IBGE
#> 3     1 IPEA
#> 4     5 Ministério da Saúde
#> 5     6 Organização Mundial da Saúde - OMS
#> 6     2 Polícia Rodoviária Federal
```

E agora periodicidades

```
get_periodicities()
#> # A tibble: 5 x 2
#>      id titulo
#>   <int> <chr>
#> 1     1 Anual
#> 2    11 Diária
#> 3     2 Mensal
#> 4     4 Semestral
#> 5     3 Trimestral
```

Demais funções `get_` no singular

As funções que terminam no singular, tais como `get_serie` ou `get_periodicity` retornam um data.frame com o ID e o Título da informação da série.

```
get_serie(serie_id = 156)
#> # A tibble: 1 x 2
#>      id titulo
#>   <int> <chr>
#> 1   156 Taxa de óbitos em acidentes de transporte - Faixa etária de 15 - 29 anos
```

Agora a periodicidade

```
get_periodicity(periodicity_id = 3)
#> # A tibble: 1 x 2
#>      id titulo
#>   <int> <chr>
#> 1     3 Trimestral
```