Package ‘rfars’

May 5, 2023

Type Package
Title Download and Analyze Fatal Crash Data
Version 0.3.0
Description
   Download raw data from the Fatality Analysis Reporting System and prepare it for research.
License CC0
Encoding UTF-8
LazyData true
Imports data.table, downloader, dplyr, haven, janitor, lubridate, magrittr, purrr, readr, rlang, sas7bdat, stringr, tidyselect, zoo
RoxygenNote 7.2.3
Depends R (>= 3.5.0)
Suggests knitr, rmarkdown, leaflet, leaflet.extras, ggplot2, scales, stargazer, viridis
VignetteBuilder knitr
URL https://github.com/s87jackson/rfars
BugReports https://github.com/s87jackson/rfars/issues
NeedsCompilation no
Author Steve Jackson [aut, cre] (<https://orcid.org/0000-0002-3337-7846>)
Maintainer Steve Jackson <steve.jackson@toxcel.com>
Repository CRAN
Date/Publication 2023-05-05 09:40:02 UTC

R topics documented:

   alcohol .......................................................... 2
   appendRDS .................................................... 3
   bicyclist ....................................................... 3
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>alcohol</td>
<td>26</td>
</tr>
<tr>
<td>(Internal) Find crashes involving alcohol</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.
**Usage**

```r
alcohol(df)
```

**Arguments**

`df` The FARS or GESCRSS data object to be searched.

---

**Description**

(Internal) Append RDS files

**Usage**

```r
appendRDS(object, file, wd)
```

**Arguments**

`object` The object to save or append

`file` The name of the file to be saved

`wd` The directory to check

---

**Description**

(Internal) Find crashes involving bicyclists

**Usage**

```r
bicyclist(df)
```

**Arguments**

`df` The FARS or GESCRSS data object to be searched.

---

These internal functions take the FARS object created by `use_fars` and look for various cases, such as distracted or drowsy drivers.
compare_counts

Description

Compare counts generated by `counts()`

Usage

```r
compare_counts(
  df,
  interval = c("year", "month")[[1]],
  what = c("crashes", "fatalities", "injuries", "people")[[1]],
  where = list(states = "all", region = c("all", "ne", "mw", "s", "w")[[1]], urb = c("all", "rural", "urban")[[1]],
    who = c("all", "drivers", "passengers", "bicyclists", "pedestrians")[[1]],
    involved = NULL,
    what2 = what,
    where2 = where,
    who2 = who,
    involved2 = involved
)
```

Arguments

- `df` The input FARS object.
- `interval` The interval in which to count: months or years.
- `what` What to count: crashes, fatalities, or people involved.
- `where` Where to count, a list with up to three elements: states ("all" by default), region ("all"), urb ("all")
- `who` The type of person to count: all (default) drivers, passengers, pedestrians, or bicyclists.
- `involved` Factors involved with the crash. Can be any of: distracted driver, drowsy driver, police pursuit, motorcycle, pedalcyclist, bicyclist, pedestrian, pedbike, young driver, older driver, speeding, alcohol, drugs, hit and run, roadway departure, rollover, or large trucks.
- `what2` Comparison point for 'what' (set to 'what' unless specified).
- `where2` Comparison point for 'where' (set to 'where' unless specified).
- `who2` Comparison point for 'who' (set to 'who' unless specified).
- `involved2` Comparison point for 'involved' (set to 'involved' unless specified).

Value

A tibble of counts.
counts

Examples

get_fars(years = 2020, states="Virginia") %>%
  compare_counts(
    where = list(urb="rural"),
    where2 = list(urb="urban")
  )

get_fars(years = 2020, states="FL") %>%
  compare_counts(
    involved = "pedestrians",
    involved2 = "bicyclists"
  )

counts  Generate counts

Description

Use FARS or GES/CRSS data to generate commonly requested counts.

Usage

counts(
  df,
  what = c("crashes", "fatalities", "injuries", "people")[1],
  interval = c("year", "month")[1],
  where = list(states = "all", region = c("all", "ne", "mw", "s", "w")[1], urb = c("all", "rural", "urban")[1]),
  who = c("all", "drivers", "passengers", "bicyclists", "pedestrians")[1],
  involved = NULL,
  filterOnly = FALSE
)

Arguments

df The input data object (must be of class 'FARS' or 'GESCRSS' as is produced by get_fars() and get_gescrss()).
what What to count: crashes (the default), fatalities, injuries, or people involved.
interval The interval in which to count: months or years (the default).
where Where to count. Must be a list with any of the elements: states (can be 'all', full or abbreviated state names, or FIPS codes), region ('all', 'ne', 'mw', 's', or 'w'; short for northeast, midwest, south, and west), urb ('all', 'rural', or 'urban'). Any un-specified elements are set to 'all' by default.
who The type of person to count: 'all' (default) 'drivers', 'passengers', 'pedestrians', or 'bicyclists'.
Factors involved with the crash. Can be any of: 'distracted driver', 'drowsy driver', 'police pursuit', 'motorcycle', 'pedalicyclist', 'bicyclist', 'pedestrian', 'pedbike', 'young driver', 'older driver', 'speeding', 'alcohol', 'drugs', 'hit and run', 'roadway departure', 'rollover', or 'large trucks'. NULL by default.

Logical, whether to only filter data or reduce to counts (FALSE by default).

Either a filtered tibble (filterOnly=TRUE) or a tibble of counts (filterOnly=FALSE). If filterOnly=TRUE, the tibble that is returned is the 'flat' tibble from the input FARS object, filtered according to other parameters.

If 'df' is a GESCRSS object, the counts returned are the sum of the appropriate weights.

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

The FARS or GESCRSS data object to be searched.
**download_fars**

*(Internal) Download FARS data files*

---

**Description**

Download files from NHTSA, unzip, and prepare them.

**Usage**

`download_fars(years, dest_raw, dest_prepd, states)`

**Arguments**

- `years` (Years to be downloaded, in yyyy (character or numeric formats))
- `dest_raw` (Directory to store raw CSV files)
- `dest_prepd` (Directory to store prepared CSV files)
- `states` (Optional) Inherits from `get_fars()`

**Details**

Raw files are downloaded from NHTSA.

**Value**

Nothing directly to the current environment. Various CSV files are stored either in a temporary directory or dir as specified by the user.

---

**download_gescrss**

*(Internal) Download GES/CRSS data files*

---

**Description**

Download files from NHTSA, unzip, and prepare them.

**Usage**

`download_gescrss(years, dest_raw, dest_prepd, regions)`

**Arguments**

- `years` (Years to be downloaded, in yyyy (character or numeric formats))
- `dest_raw` (Directory to store raw CSV files)
- `dest_prepd` (Directory to store prepared CSV files)
- `regions` (Optional) Inherits from `get_gescrss()`
Details

Raw files are downloaded directly from NHTSA.

Value

Nothing directly to the current environment. Various CSV files are stored either in a temporary directory or dir as specified by the user.

---

**driver_age**

*(Internal) Find crashes involving drivers of a given age*

---

Description

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage

```
driver_age(df, age_min, age_max)
```

Arguments

- `df`: The FARS or GESCRSS data object to be searched.
- `age_min`: Lower bound on driver age (inclusive).
- `age_max`: Upper bound on driver age (inclusive).

---

**drowsy_driver**

*(Internal) Find crashes involving drowsy drivers*

---

Description

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage

```
drowsy_driver(df)
```

Arguments

- `df`: The FARS or GESCRSS data object to be searched.
Drugs

(Internal) Find crashes involving drugs

Description

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage

drugs(df)

Arguments

df The FARS or GESCRSS data object to be searched.

FARS Codebook

Description

A table describing each FARS variable name, value, and corresponding value label.

Usage

fars_codebook

Format

A data frame with 132,454 rows and 8 variables:

source  The source of the data (either FARS or GES/CRSS)
year    Year of the data element definition.
file    The data file that contains the given variable.
nname_ncsa The original name of the data element.
nname_rfars The modified data element name used in rfars
label   The label of the data element itself (not its constituent values).
value   The original value of the data element.
value_label The de-coded value label.
Details

This codebook serves as a useful reference for researchers using FARS data. The ‘source’ variable is intended to help combine with the gescrss_codebook. Data elements are relatively stable but are occasionally discontinued, created anew, or modified. The ‘year’ variable helps indicate the availability of data elements, and differentiates between different definitions over time. Users should always check for discontinuities when tabulating cases.

The ‘file’ variable indicates the file in which the given data element originally appeared. Here, files refers to the SAS files downloaded from NHTSA. Most data elements stayed in their original file. Those that did not were moved to the multi_files. For example, ‘weather’ originates from the ‘accident’ file, but appears in the multi_acc data object created by rfars.

The ‘name_ncsa’ variable describes the data element’s name as assigned by NCSA (the organization within NHTSA that manages the database). To maximize compatibility between years and ease of use for programming, ‘name_rfars’ provides a cleaned naming convention (via janitor::clean_names()). Both names are provided here to help users find the corresponding entry in the Analytical User’s Manual but only the latter are used in the data produced by get_fars().

Each data element has a ‘label’, a more human-readable version of the element names. For example, the label for ‘road_fnc’ is ‘Roadway Function Class’. These are not definitions but may provide enough information to help users conduct their analysis. Consult the Analytical User’s Manual for definitions and further details.

Each data element has multiple ‘value’-‘value_label’ pairs: ‘value’ represents the original, non-human-readable value (usually a number), and ‘value_label’ represents the corresponding text value. For example, for ‘road_fnc’, 1 (the ‘value’) corresponds to ‘Rural-Principal Arterial-Interstate’ (the ‘value_label’), 2 corresponds to ‘Rural-Principal Arterial-Other’, etc.

See Also

"gescrss_codebook"

table

geo_relations

Description

A dataset providing different ways to refer to states and counties.

Usage

geo_relations

Format

A data frame with 3,142 rows and 6 variables:

- **fips_state** 2-digit FIPS code indicating a state
- **fips_county** 3-digit FIPS code indicating a county within a state
**fips_tract** 6-digit FIPS code indicating a tract within a county

**state_name_abbr** 2-character, capitalized state abbreviation

**state_name_full** fully spelled and case-sensitive state name

**county_name_abbr** abbreviated county name (usually minus the word 'County')

**county_name_full** fully spelled and case-sensitive county name

**region** fully spelled out and case-sensitive NHTSA region and constituent states

**region_abbr** abbreviated NHTSA region (ne, mw, s, w)

---

**Source**


---

**gescrss_codebook**

**GESCRSS Codebook**

---

**Description**

A table describing each GESCRSS variable name, value, and corresponding value label.

**Usage**

gescrss_codebook

**Format**

A data frame with 85,907 rows and 8 variables:

- **source** The source of the data (either FARS or GESCRSS)
- **year** Year of the data element definition.
- **file** The data file that contains the given variable.
- **name_ncsa** The original name of the data element.
- **name_rfars** The modified data element name used in rfars
- **label** The label of the data element itself (not its constituent values).
- **value** The original value of the data element.
- **value_label** The de-coded value label.
Details

This codebook serves as a useful reference for researchers using GES/CRSS data. The ‘source’ variable is intended to help combine with the fars_codebook. Data elements are relatively stable but are occasionally discontinued, created anew, or modified. The ‘year’ variable helps indicate the availability of data elements, and differentiates between different definitions over time. Users should always check for discontinuities when tabulating cases.

The ‘file’ variable indicates the file in which the given data element originally appeared. Here, files refers to the SAS files downloaded from NHTSA. Most data elements stayed in their original file. Those that did not were moved to the multi_files. For example, ‘weather’ originates from the ‘accident’ file, but appears in the multi_acc data object created by rfars.

The ‘name_ncsa’ variable describes the data element’s name as assigned by NCSA (the organization within NHTSA that manages the database). To maximize compatibility between years and ease of use for programming, ‘name_rfars’ provides a cleaned naming convention (via janitor::clean_names()). Both names are provided here to help users find the corresponding entry in the CRSS User Manual but only the latter are used in the data produced by get_gescrss().

Each data element has a ‘label’, a more human-readable version of the element names. For example, the label for ‘harm_ev’ is ‘First Harmful Event’. These are not definitions but may provide enough information to help users conduct their analysis. Consult the CRSS User Manual for definitions and further details.

Each data element has multiple ‘value’-‘value_label’ pairs: ‘value’ represents the original, non-human-readable value (usually a number), and ‘value_label’ represents the corresponding text value. For example, for ‘harm_ev’, 1 (the ‘value’) corresponds to ‘Rollover/Overturn’ (the ‘value_label’), 2 corresponds to ‘Fire/Explosion’, etc.

See Also

"fars_codebook"

---

get_fars  
*Get FARS data*

Description

Bring FARS data into the current environment, whether by downloading it anew or by using pre-existing files.

Usage

```r
get_fars(
    years = 2011:2021,
    states = NULL,
    dir = NULL,
    proceed = FALSE,
    cache = NULL
)
```
Arguments

- **years**: Years to be downloaded, in yyyy (character or numeric formats), currently limited to 2011-2021 (the default).
- **states**: States to keep. Leave as NULL (the default) to keep all states. Can be specified as full state name (e.g. "Virginia"), abbreviation ("VA"), or FIPS code (51).
- **dir**: Directory in which to search for or save a 'FARS data' folder. If NULL (the default), files are downloaded and unzipped to temporary directories and prepared in memory.
- **proceed**: Logical, whether or not to proceed with downloading files without asking for user permission (defaults to FALSE, thus asking permission)
- **cache**: The name of an RDS file to save or use. If the specified file (e.g., 'myFARS.rds') exists in 'dir' it will be returned; if not, an RDS file of this name will be saved in 'dir' for quick use in subsequent calls.

Details

This function downloads raw data from NHTSA. If no directory (dir) is specified, SAS files are downloaded into a tempdir(), where they are also prepared, combined, and then brought into the current environment. If you specify a directory (dir), the function will look there for a 'FARS data' folder. If not found, it will be created and populated with raw and prepared SAS and RDS files. If the directory is found, the function makes sure all requested years are present and asks permission to download any missing years.

The object returned is a list with class 'FARS'. It contains six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook.

Flat files are wide-formatted and presented at the person level. All crashes involve at least one motor vehicle, each of which may contain one or multiple people. These are the three entities of crash data. The flat files therefore repeat some data elements across multiple rows. Please conduct your analysis with your entity in mind.

Some data elements can include multiple values for any data level (e.g., multiple weather conditions corresponding to the crash, or multiple crash factors related to vehicle or person). These elements have been collected in the yyyy_multi_[acc/veh/per].rds files in long format. These files contain crash, vehicle, and person identifiers, and two variables labelled name and value. These correspond to variable names from the raw data files and the corresponding values, respectively.

The events tibble provides a sequence of events for all vehicles involved in the crash. See Crash Sequences vignette for an example.

Finally, the codebook tibble serves as a searchable codebook for all files of any given year.

Please review the [FARS Analytical User’s Manual](https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813417) for more information.

Value

A FARS data object (list of six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook), described below.
get_gescrss

Examples

myFARS <- get_fars(years = 2019:2021, states = "51")
myFARS <- get_fars(years = 2021, states = "NC")

get_gescrss

Get GES/CRSS data

Description

Bring GES/CRSS data into the current environment, whether by downloading it anew or by using pre-existing files.

Usage

get_gescrss(
  years = 2011:2021,
  regions = c("mw", "ne", "s", "w"),
  dir = NULL,
  proceed = FALSE,
  cache = NULL
)

Arguments

years
  Years to be downloaded, in yyyy (character or numeric formats), currently limited to 2011-2021.

regions
  (Optional) Regions to keep: mw=midwest, ne=northeast, s=south, w=west.

dir
  Directory in which to search for or save a 'GESCRSS data' folder. If NULL (the default), files are downloaded and unzipped to temporary directories and prepared in memory.

proceed
  Logical, whether or not to proceed with downloading files without asking for user permission (defaults to FALSE, thus asking permission)

cache
  The name of an RDS file to save or use. If the specified file (e.g., 'myFARS.rds') exists in 'dir' it will be returned; if not, an RDS file of this name will be saved in 'dir' for quick use in subsequent calls.

Details

This function downloads raw data from the GES and CRSS crash databases. If no directory (dir) is specified, raw CSV files are downloaded into a tempdir(), where they are also prepared, combined, and then brought into the current environment. If you specify a directory (dir), the function will look there for a 'GESCRSS data' folder. If not found, it will be created and populated with raw and prepared SAS and RDS files. If the directory is found, the function makes sure all requested years are present and asks permission to download any missing years.
The object returned is a list with class `GESCRSS`. It contains six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook.

Flat files are wide-formatted and presented at the person level. All *crashes* involve at least one motor *vehicle*, each of which may contain one or multiple *people*. These are the three entities of crash data. The flat files therefore repeat some data elements across multiple rows. Please conduct your analysis with your entity in mind.

Some data elements can include multiple values for any data level (e.g., multiple weather conditions corresponding to the crash, or multiple crash factors related to vehicle or person). These elements have been collected in the `yyy_multi_[acc/veh/per].rds` files in long format. These files contain crash, vehicle, and person identifiers, and two variables labelled *name* and *value*. These correspond to variable names from the raw data files and the corresponding values, respectively.

The events tibble provides a sequence of events for all vehicles involved in the crash. See Crash Sequences vignette for an example.

The codebook tibble serves as a searchable codebook for all files of any given year. Please review the [CRSS Analytical User’s Manual](https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813436) for more information.

Regions are as follows: mw = Midwest = OH, IN, IL, MI, WI, MN, ND, SD, NE, IA, MO, KS ne = Northeast = PA, NJ, NY, NH, VT, RI, MA, ME, CT s = South = MD, DE, DC, WV, VA, KY, TN, NC, SC, GA, FL, AL, MS, LA, AR, OK, TX w = West = MT, ID, WA, OR, CA, NV, NM, AZ, UT, CO, WY, AK, HI

**Value**

A GESCRSS data object (a list with six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook).

**Examples**

```r
myGESCRSS <- get_gescrss(years = 2019:2021, regions = "s")
myGESCRSS <- get_gescrss(years = 2021)
```

**hit_and_run**

*(Internal) Find hit and run crashes*

**Description**

These internal functions take the FARS object created by `use_fars` and look for various cases, such as distracted or drowsy drivers.

**Usage**

```r
hit_and_run(df)
```
import_multi

(Internal) Import the multi_files

Description
An internal function that imports the multi_files

Usage
import_multi(filename, where)

Arguments
filename The filename (e.g. "multi_acc.csv") to be imported
where The directory to search within

large_trucks

(Internal) Find crashes involving large trucks

Description
These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage
large_trucks(df)

Arguments
df The FARS or GESCRSS data object to be searched.
**make_all_numeric**

*(Internal) Make id and year numeric*

**Description**

*(Internal) Make id and year numeric*

**Usage**

```r
make_all_numeric(df)
```

**Arguments**

- `df` The input dataframe

---

**make_id**

*(Internal) Generate an ID variable*

**Description**

*(Internal) Generate an ID variable*

**Usage**

```r
make_id(df)
```

**Arguments**

- `df` The dataframe from which to make the id

---

**motorcycle**

*(Internal) Find crashes involving motorcycles*

**Description**

These internal functions take the FARS object created by `use_fars` and look for various cases, such as distracted or drowsy drivers.

**Usage**

```r
motorcycle(df)
```

**Arguments**

- `df` The FARS or GESCRSS data object to be searched.
pedalcyclist  
(Internal) Find crashes involving pedalcyclists

Description
These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage
pedalcyclist(df)

Arguments

df The FARS or GESCRSS data object to be searched.

pedbike  
(Internal) Find crashes involving pedestrians or bicyclists

Description
These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage
pedbike(df)

Arguments

df The FARS or GESCRSS data object to be searched.

pedestrian  
(Internal) Find crashes involving pedestrians

Description
These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage
pedestrian(df)

Arguments

df The FARS or GESCRSS data object to be searched.
police_pursuit  
(Internal) Find crashes involving police pursuits

Description
These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage
police_pursuit(df)

Arguments
- df: The FARS or GESCRSS data object to be searched.

prep_fars  
Prepare downloaded FARS files for use

Description
Prepare downloaded FARS files for use

Usage
prep_fars(y, wd, rawfiles, prepared_dir, states)

Arguments
- y: year, to be passed from prep_fars
- wd: working directory, to be passed from prep_fars
- rawfiles: dataframe translating filenames into standard terms, to be passed from prep_fars
- prepared_dir: the location where prepared files will be saved, to be passed from prep_fars
- states: (Optional) Inherits from get_fars()

Value
Produces six files: yyyy_flat.rds, yyyy_multi_acc.rds, yyyy_multi_veh.rds, yyyy_multi_per.rds, yyyy_events.rds, and codebook.rds
**prep_gescrss**

Prepare downloaded GES/CRSS files for use

**Description**

Prepare downloaded GES/CRSS files for use

**Usage**

```r
prep_gescrss(y, wd, rawfiles, prepared_dir, regions)
```

**Arguments**

- `y`: year, to be passed from `prep_gescrss`
- `wd`: working directory, to be passed from `prep_gescrss`
- `rawfiles`: dataframe translating filenames into standard terms, to be passed from `prep_gescrss`
- `prepared_dir`: the location where prepared files will be saved, to be passed from `prep_gescrss`
- `regions`: (Optional) Inherits from `get_gescrss()`

**Value**

Produces six files: `yyyy_flat.rds`, `yyyy_multi_acc.rds`, `yyyy_multi_vehrds`, `yyyy_multi_per.rds`, `yyyy_events.rds`, and `codebook.rds`

---

**read_basic_sas**

(Internal) Takes care of basic SAS file reading

**Description**

(Internal) Takes care of basic SAS file reading

**Usage**

```r
read_basic_sas(x, wd, rawfiles, catfile = paste0(wd, "formats.sas7bcat"))
```

**Arguments**

- `x`: The cleaned name of the data table (SAS7BDAT).
- `wd`: The working directory for these files
- `rawfiles`: The data frame connecting raw filenames to cleaned ones.
- `catfile`: The location of the sas7bcat file

**See Also**

`read_basic_sas_nocat`
read_basic_sas_nocat  

(Internal) Takes care of basic SAS file reading when the bcat file creates an issue

Description

(Internal) Takes care of basic SAS file reading when the bcat file creates an issue

Usage

read_basic_sas_nocat(x, wd, rawfiles)

Arguments

x       The cleaned name of the data table (SAS7BDAT).
wd      The working directory for these files
rawfiles The data frame connecting raw filenames to cleaned ones.

rm_cols.f  

(Internal) rm_cols.f

Description

An internal function that removes variables that are unnecessarily duplicated across FARS tables.

Usage

rm_cols.f(df, a, b)

Arguments

df       The input data frame.
a        The original, non-imputed variable.
b        The imputed variable (often with an _im suffix).
Description

An internal function that removes variables that are unnecessarily duplicated across GES/CRSS tables.

Usage

rm_cols.g(df, a, b)

Arguments

df The input data frame.
a The original, non-imputed variable.
b The imputed variable (often with an _im suffix).

Description

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

Usage

road_depart(df)

Arguments

df The FARS or GESCRSS data object to be searched.
**rollover**  
*(Internal) Find crashes involving rollovers*

**Description**

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

**Usage**

`rollover(df)`

**Arguments**

- `df`  
The FARS or GESCRSS data object to be searched.

---

**speeding**  
*(Internal) Find crashes involving speeding*

**Description**

These internal functions take the FARS object created by use_fars and look for various cases, such as distracted or drowsy drivers.

**Usage**

`speeding(df)`

**Arguments**

- `df`  
The FARS or GESCRSS data object to be searched.
use_fars  
(Internal) Use FARS data files

Description
Compile multiple years of prepared FARS data.

Usage
use_fars(dir, prepared_dir, cache)

Arguments
- dir: Inherits from get_fars().
- prepared_dir: Inherits from get_fars().
- cache: Inherits from get_fars().

Value
Returns an object of class 'FARS' which is a list of six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook.

use_gescrss  
(Internal) Use GESCRSS data files

Description
Compile multiple years of prepared GESCRSS data.

Usage
use_gescrss(dir, prepared_dir, cache)

Arguments
- dir: Inherits from get_gescrss().
- prepared_dir: Inherits from get_gescrss().
- cache: Inherits from get_gescrss().

Value
Returns an object of class 'GESCRSS' which is a list of six tibbles: flat, multi_acc, multi_veh, multi_per, events, and codebook.
use_imp

(Internal) use_imp

Description

An internal function that uses imputed variables (present in many GES/CRSS tables)

Usage

use_imp(df, original, imputed, show = FALSE)

Arguments

df The input data frame.
original The original, non-imputed variable.
imputed The imputed variable (often with an _im suffix).
show Logical (FALSE by default) Show differences between original and imputed values.

validate_states

(Internal) Validate user-provided list of states

Description

(Internal) Validate user-provided list of states

Usage

validate_states(states)

Arguments

states States specified in get_fars, prep_fars, or counts
Index

* datasets
  fars_codebook, 9
  geo_relations, 10
  gescrss_codebook, 11
  alcohol, 2
  appendRDS, 3
  bicyclist, 3
  compare_counts, 4
  counts, 5
  distracted_driver, 6
  download_fars, 7
  download_gescrss, 7
  driver_age, 8
  drowsy_driver, 8
  drugs, 9
  fars_codebook, 9
  geo_relations, 10
  gescrss_codebook, 11
  get_fars, 12
  get_gescrss, 14
  hit_and_run, 15
  import_multi, 16
  large_trucks, 16
  make_all_numeric, 17
  make_id, 17
  motorcycle, 17
  pedalcyclist, 18
  pedbike, 18
  pedestrian, 18
  police_pursuit, 19
  prep_fars, 19
  prep_gescrss, 20
  read_basic_sas, 20
  read_basic_sas_nocat, 21
  rm_cols.f, 21
  rm_cols.g, 22
  road_depart, 22
  rollover, 23
  speeding, 23
  use_fars, 24
  use_gescrss, 24
  use_imp, 25
  validate_states, 25