Package 'rfars'

October 22, 2025

Type Package

Title Download and Analyze Crash Data

Version 2.0.2

Description Easily Download Analysis-Ready Crash Data from the U.S. National Highway Traffic Safety Administration.

License CC0

Encoding UTF-8

LazyData true

Imports data.table, downloader, dplyr, haven, janitor, lubridate, magrittr, purrr, readr, rlang, stringr, tidyr, tidyselect

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, leaflet, leaflet.extras, ggplot2, scales, stargazer, viridis, lme4, tidyverse, tidytext, DT, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://github.com/s87jackson/rfars, https://s87jackson.github.io/rfars/

BugReports https://github.com/s87jackson/rfars/issues

Config/testthat/edition 3

NeedsCompilation no

Author Steve Jackson [aut, cre] (ORCID: <https://orcid.org/0000-0002-3337-7846>)

Maintainer Steve Jackson <steve.jackson@toxcel.com>

Repository CRAN

Date/Publication 2025-10-22 07:40:02 UTC

Contents

alcohol *(Internal) Find crashes involving alcohol*

Description

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

Usage

```
alcohol(df)
```

Arguments

df	The FARS or GESCRSS data object to be searched.

annual_counts *Annual Crash Counts by Risk Factors*

Description

Pre-computed annual crash counts from FARS (fatal crashes) and CRSS (general crash estimates) databases for 2014-2023, broken down by various risk factors and vulnerable road user categories.

Usage

```
annual_counts
```

Format

A tibble with 340 rows and 9 variables:

year Year (2014-2023)

month Month, if included in interval, as the three-letter abbreviation and an ordered factor (Jan=1, Feb=2, etc.)

what Count unit - currently only "crashes"

states Geographic scope - "all" for national-level data

region Regional scope - "all" for national-level data

urb Urban/rural classification - "all" for combined data

who Person type - "all" for all person types

involved Risk factor or crash type. Options include:

- **"any"** All crashes (general counts)
- **"each"** Each factor listed below, separately

"alcohol" Alcohol-involved crashes

"bicyclist" Crashes involving bicyclists

"distracted driver" Distracted driving crashes

"drugs" Drug-involved crashes

"hit and run" Hit-and-run crashes

"large trucks" Large truck-involved crashes

"motorcycle" Motorcycle crashes

"older driver" Crashes involving older drivers

"pedalcyclist" Crashes involving pedalcyclists

"pedbike" Pedestrian and bicyclist crashes combined

"pedestrian" Pedestrian crashes

"police pursuit" Police pursuit-related crashes

"roadway departure" Roadway departure crashes

"rollover" Rollover crashes

"speeding" Speed-related crashes

"young driver" Crashes involving young drivers

n Count of crashes. FARS counts represent actual fatal crashes; CRSS counts represent weighted estimates of all crashes

Details

This dataset provides quick access to national-level annual crash counts without needing to download and process the full datasets. It combines data from two NHTSA databases:

FARS Fatal crashes (actual counts)

CRSS General crashes (weighted estimates)

The data can be reproduced using the `counts()` function on downloaded FARS and CRSS data with `involved = "any"` and `involved = "each"` parameters.

See Also

`counts` for generating custom counts from downloaded data

Examples

```
## Not run:
# View total crashes over time by data source
library(dplyr)
library(ggplot2)

annual_counts %>%
  filter(involved == "any") %>%
  ggplot(aes(x = year, y = n, fill = source)) +
  geom_col(position = "dodge") +
  labs(title = "Annual Crash Counts by Data Source",
       x = "Year", y = "Number of Crashes")
```

```
# Compare risk factor trends in fatal crashes
annual_counts %>%
  filter(source == "FARS",
         involved %in% c("alcohol", "speeding", "distracted driver")) %>%
  ggplot(aes(x = year, y = n, color = involved)) +
  geom_line() +
  labs(title = "Fatal Crash Trends by Risk Factor",
       x = "Year", y = "Fatal Crashes")

## End(Not run)
```

---

appendRDS *(Internal) Append RDS files*

Description

(Internal) Append RDS files

Usage

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

Arguments

object	The object to save or append
file	The name of the file to be saved to be saved
wd	The directory to check

---

bicyclist *(Internal) Find crashes involving 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

```
bicyclist(df)
```

Arguments

df	The FARS or GESCRSS data object to be searched.

check_internet_connection *(Internal) Check internet connection*

Description

Test if internet connection is available by attempting to reach a reliable host. This function is used to gracefully handle cases where internet resources are not available.

Usage

```
check_internet_connection()
```

Value

Logical indicating whether internet connection is available

compare_counts *Compare counts*

Description

Compare counts generated by counts()

Usage

```
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, 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.

Examples

```
  ## Not run:
    compare_counts(
      get_fars(years = 2020, states="Virginia"),
      where = list(urb="rural"),
      where2 = list(urb="urban")
      )

## End(Not run)
```

---

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 = c("any", "each", "alcohol", "bicyclist", "distracted driver", "drugs",
    "hit and run", "large trucks", "motorcycle", "older driver", "pedalcyclist",
    "pedbike", "pedestrian", "police pursuit", "roadway departure", "rollover",
    "speeding", "young driver")[1],
  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'.
involved	Factors involved with the crash: 'any' (the default, produces general counts), 'each' (produces separate counts for each factor), 'distracted driver', 'police pursuit', 'motorcycle', 'pedalcyclist', 'bicyclist', 'pedestrian', 'pedbike', 'young driver', 'older driver', 'speeding','alcohol', 'drugs', 'hit and run', 'roadway departure', 'rollover', or 'large trucks'.
filterOnly	Logical, whether to only filter data or reduce to counts (FALSE by default).

Value

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.

Examples

```
  ## Not run:
    counts(get_fars(years = 2019), where = list(states="Virginia", urb="rural"))

## End(Not run)
```

distracted_driver *(Internal) Find crashes involving distracted 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

```
distracted_driver(df)
```

Arguments

df	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).

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 *FARS Codebook*

Description

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

Usage

```
fars_codebook
```

Format

A data frame with 15,951 rows and 19 variables:

source The source of the data (either FARS or GES/CRSS).

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).

Definition The data element's definition, pulled from the Analytical User Manual.

Additional Information Additional information on the data element, pulled from the Analytical User Manual.

value The original value of the data element.

value_label The de-coded value label.

2014 Indicator: 1 if valid for 2014, NA otherwise.

2015 Indicator: 1 if valid for 2015, NA otherwise.

2016 Indicator: 1 if valid for 2016, NA otherwise.

2017 Indicator: 1 if valid for 2017, NA otherwise.

2018 Indicator: 1 if valid for 2018, NA otherwise.

2019 Indicator: 1 if valid for 2019, NA otherwise.

2020 Indicator: 1 if valid for 2020, NA otherwise.

2021 Indicator: 1 if valid for 2021, NA otherwise.

2022 Indicator: 1 if valid for 2022, NA otherwise.

2023 Indicator: 1 if valid for 2023, NA otherwise.

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()).

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.

'Definition' and 'Additional Information' were extracted from the Analytical User's Manual.

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.

@source Codebooks are automatically generated by extracting SAS format catalogs (.sas7bcat files) and VALUE statements from .sas files during data processing, then consolidating variable names, labels, and value-label mappings across all years into searchable reference tables. Source files are published by NHTSA and available here.

See Also

"gescrss_codebook"

Examples

```
head(rfars::fars_codebook)
```

geo_relations *Synonym table for various geographical scales*

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

https://www.census.gov/geographies/reference-files/2015/demo/popest/2015-fips.html

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 34,662 rows and 8 variables:

source The source of the data (either FARS or GESCRSS).

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).

Definition The data element's definition, pulled from the Analytical User Manual

Additional Information Additional information on the data element, pulled from the Analytical User Manual.

value The original value of the data element.

value_label The de-coded value label.

2014 Indicator: 1 if valid for 2014, NA otherwise.

2015 Indicator: 1 if valid for 2015, NA otherwise.

2016 Indicator: 1 if valid for 2016, NA otherwise.

2017 Indicator: 1 if valid for 2017, NA otherwise.

2018 Indicator: 1 if valid for 2018, NA otherwise.

2019 Indicator: 1 if valid for 2019, NA otherwise.

2020 Indicator: 1 if valid for 2020, NA otherwise.

2021 Indicator: 1 if valid for 2021, NA otherwise.

2022 Indicator: 1 if valid for 2022, NA otherwise.

2023 Indicator: 1 if valid for 2023, NA otherwise.

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()).

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.

'Definition' and 'Additional Information' were extracted from the Analytical User's Manual.

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.

@source Codebooks are automatically generated by extracting SAS format catalogs (.sas7bcat files) and VALUE statements from .sas files during data processing, then consolidating variable names, labels, and value-label mappings across all years into searchable reference tables. Source files are published by NHTSA and available here.

See Also

"fars_codebook"

Examples

```
head(rfars::gescrss_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

```
get_fars(
  years = 2014:2023,
  states = NULL,
  source = c("zenodo", "nhtsa"),
  proceed = FALSE,
  dir = NULL,
  cache = NULL
)
```

Arguments

`years`	Years to be downloaded, in yyyy (character or numeric formats, defaults to last 10 years).
`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).

source	The source of the data: 'zenodo' (the default) pulls the prepared dataset from Zenodo, 'nhtsa' pulls the raw files from NHTSA's FTP site and prepares them on your machine. 'zenodo' is much faster and provides the same dataset produced by using source='nhtsa'.
proceed	Logical, whether or not to proceed with downloading files without asking for user permission (defaults to FALSE, thus asking permission)
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. Ignored if source = 'zenodo'.
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. Ignored if source = 'zenodo'.

Details

This function provides the FARS database for the specified years and states. By default, it pulls from a Zenodo repository for speed and memory efficiency. It can also pull the raw files from NHTSA and process them in memory, or use an RDS file saved on your machine.

If source = 'nhtsa' and 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, otherwise 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

Value

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

Examples

```
## Not run:
  # Use defaults to get 10 years of national data
  myFARS <- get_fars()

  # Get latest year of data
  myFARS <- get_fars(2023)

  # Get data for one state
  myFARS <- get_fars(states = "VA")

## End(Not run)
```

`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 = 2014:2023,
  regions = c("mw", "ne", "s", "w"),
  source = c("zenodo", "nhtsa"),
  proceed = FALSE,
  dir = NULL,
  cache = NULL
)
```

Arguments

`years`	Years to be downloaded, in yyyy (character or numeric formats, defaults to last 10 years).
`regions`	(Optional) Regions to keep: mw=midwest, ne=northeast, s=south, w=west.
`source`	The source of the data: 'zenodo' (the default) pulls the prepared dataset from Zenodo, 'nhtsa' pulls the raw files from NHTSA's FTP site and prepares them on your machine. 'zenodo' is much faster and provides the same dataset produced by using source='nhtsa'.
`proceed`	Logical, whether or not to proceed with downloading files without asking for user permission (defaults to FALSE, thus asking permission)
`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. Ignored if source = 'zenodo'.

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. Ignored if source = 'zenodo'.

Details

This function provides the GES/CRSS database for the specified years and regions By default, it pulls from a Zenodo repository for speed and memory efficiency. It can also pull the raw files from NHTSA and process them in memory, or use an RDS file saved on your machine.

If source = 'nhtsa' and 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 'GESCRSS data' folder. If not found, it will be created and populated with raw and prepared SAS and RDS files, otherwise 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 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.

The codebook tibble serves as a searchable codebook for all files of any given year.

Please review the CRSS Analytical User's Manual

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

```
## Not run:
  # Use defaults to get 10 years of national data
  myCRSS <- get_gescrss()

  # Get latest year of data
  myCRSS <- get_gescrss(2023)
```

```
    # Get data for one region
    myCRSS <- get_gescrss(regions = "s")

## End(Not run)
```

`get_sas_attrs` *(Internal) Check SAS attributes*

Description

(Internal) Check SAS attributes

Usage

```
get_sas_attrs(data)
```

Arguments

`data`	An object produced by haven::read_sas()

`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

```
hit_and_run(df)
```

Arguments

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

`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

```
make_all_numeric(df)
```

Arguments

`df`	The input dataframe

make_id *(Internal) Generate an ID variable*

Description

(Internal) Generate an ID variable

Usage

```
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

```
motorcycle(df)
```

Arguments

df	The FARS or GESCRSS data object to be searched.

parse_sas_format *(Internal) Parse formats.sas instead of using a .sas7bcat file*

Description

(Internal) Parse formats.sas instead of using a .sas7bcat file

Usage

```
parse_sas_format(file_path)
```

Arguments

file_path	The path of the formats.sas file

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 pedstrians 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

```
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_veh.rds, 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

```
read_basic_sas(x, wd, rawfiles, catfile, imps = NULL, omits = NULL)
```

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
imps	A named list to be passed to use_imp(). Each item's name represents the non-imputed variable name; the item itself represents the related imputed variable.
omits	Character vector of columns to omit

See Also

read_basic_sas_nocat

`road_depart` *(Internal) Find crashes involving road departures*

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