Package 'stats19'

May 9, 2026

Title Work with Open Road Traffic Casualty Data from Great Britain

Version 4.0.0

Description Work with and download road traffic casualty data from Great Britain. Enables access to the UK's official road safety statistics, 'STATS19'. Enables users to specify a download directory for the data, which can be set permanently by adding `STATS19_DOWNLOAD_DIRECTORY=/path/to/a/dir` to your `.Renviron` file, which can be opened with `usethis::edit_r_environ()`. The data is provided as a series of `.csv` files. This package downloads, reads-in and formats the data, making it suitable for analysis. See the stats19 vignette for details. Data available from 1979 to 2024. See the official data series at <https://www.data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-accidents-safety-data>. The package is described in a paper in the Journal of Open Source Software (Lovelace et al. 2019) <doi:10.21105/joss.01181>. See Gilardi et al. (2022) <doi:10.1111/rssa.12823>, Vidal-Tortosa et al. (2021) <doi:10.1016/j.jth.2021.101291>, Tait et al. (2023) <doi:10.1016/j.aap.2022.106895>, and León et al. (2025) <doi:10.18637/jss.v114.i09> for examples of how the data can be used for methodological and empirical research.

Depends R (>= 4.1.0)

License GPL-3

URL https://github.com/ropensci/stats19, https://docs.ropensci.org/stats19/

BugReports https://github.com/ropensci/stats19/issues

Encoding UTF-8

LazyData true

Imports sf, curl (>= 3.2), readr, tibble, dplyr, lubridate, jsonlite, glue

Suggests duckdb, DBI, ggplot2, knitr, rmarkdown, stringr, testthat (>= 2.1.0), tidyr, tidyselect, pkgdown, kableExtra, leaflet, geojsonsf, htmltools, tmap, spatstat.geom, osmdata, covr, readODS, gt, clock, waldo

VignetteBuilder knitr

RoxygenNote 7.3.3

Language en-US

X-schema.org-keywords stats19, road-safety, transport, car-crashes, ropensci, data

NeedsCompilation no

Author Robin Lovelace [aut, cre] (ORCID: <https://orcid.org/0000-0001-5679-6536>), Malcolm Morgan [aut] (ORCID: <https://orcid.org/0000-0002-9488-9183>), Layik Hama [aut] (ORCID: <https://orcid.org/0000-0003-1912-4890>), Mark Padgham [aut] (ORCID: <https://orcid.org/0000-0003-2172-5265>), David Ranzolin [rev], Adam Sparks [rev, ctb] (ORCID: <https://orcid.org/0000-0002-0061-8359>), Ivo Wengraf [ctb], RAC Foundation [fnd], Blaise Kelly [aut] (ORCID: <https://orcid.org/0000-0003-2623-1598>)

Maintainer Robin Lovelace <rob00x@gmail.com>

Repository CRAN

Date/Publication 2026-03-18 15:20:02 UTC

Contents

`accidents_sample` *Sample of stats19 data (2022 collisions)*

Description

Sample of stats19 data (2022 collisions)

Format

A data frame

Note

These were generated using the script in the `data-raw` directory (`misc.Rmd` file).

Examples

```
nrow(accidents_sample_raw)
accidents_sample_raw
```

casualties_sample *Sample of stats19 data (2022 casualties)*

Description

Sample of stats19 data (2022 casualties)

Format

A data frame

Note

These were generated using the script in the `data-raw` directory (`misc.Rmd` file).

Examples

```
nrow(casualties_sample_raw)
casualties_sample_raw
```

check_input_file *Local helper to be reused.*

Description

Local helper to be reused.

Usage

```
check_input_file(filename = NULL, type = NULL, data_dir = NULL, year = NULL)
```

Arguments

`filename`	Character string of the filename of the .csv to read.
`type`	One of 'collision', 'casualty', 'Vehicle'.
`data_dir`	Where sets of downloaded data would be found.
`year`	Single year for which data are to be read.

clean_make *Clean vehicle make*

Description

This function cleans the make of the vehicle.

Usage

```
clean_make(make, extract_make = TRUE)
```

Arguments

make	A character vector of vehicle makes. Can be raw generic make/model strings if extract_make is TRUE.
extract_make	Logical, whether to extract the make from the input string using extract_make_stats19 first. Default is TRUE.

Examples

```
clean_make(c("VW", "Mercedez"))
clean_make(c("FORD FIESTA", "LAND ROVER DISCOVERY"), extract_make = TRUE)
```

clean_make_model *Clean vehicle make and model*

Description

This function returns a combined cleaned make and model string. It uses `clean_make` and `clean_model` to standardize both parts.

Usage

```
clean_make_model(generic_make_model)
```

Arguments

generic_make_model	A character vector of generic make/model strings

Examples

```
clean_make_model(c("FORD FIESTA", "BMW 3 SERIES"))
```

clean_model *Clean vehicle model*

Description

This function cleans the model of the vehicle. It extracts the make using `extract_make_stats19` and removes it from the string, returning the remaining text as the model in title case.

Usage

```
clean_model(model)
```

Arguments

`model`	A character vector of generic make/model strings

Examples

```
clean_model(c("FORD FIESTA", "BMW 3 SERIES"))
```

dl_stats19 *Download STATS19 data for a year*

Description

Download STATS19 data for a year

Usage

```
dl_stats19(
  year = NULL,
  type = NULL,
  data_dir = get_data_directory(),
  file_name = NULL,
  ask = FALSE,
  silent = FALSE,
  timeout = 600
)
```

Arguments

year	Single year for which data are to be read
type	One of 'collision', 'casualty', 'Vehicle'; defaults to 'collision'.
data_dir	Where sets of downloaded data would be found.
file_name	Character string of a specific STATS19 CSV filename to download/read. If NULL, filenames are inferred from year and type.
ask	Should you be asked whether or not to download the files? TRUE by default.
silent	Boolean. If FALSE (default value), display useful progress messages on the screen.
timeout	Timeout in seconds for the download if current option is less than this value. Defaults to 600 (10 minutes).

Examples

```
if (curl::has_internet()) {
  # type by default is collisions table
  dl_stats19(year = 2022)
}
```

---

extract_make_stats19 *Extract vehicle make from generic make/model string*

Description

This function extracts the make from a generic make/model string, handling multi-word makes.

Usage

```
extract_make_stats19(generic_make_model)
```

Arguments

generic_make_model	A character vector of generic make/model strings

Examples

```
extract_make_stats19(c("FORD FIESTA", "LAND ROVER DISCOVERY"))
```

file_names *stats19 file names for easy access*

Description

URL decoded file names. Currently there are 52 file names released by the DfT (Department for Transport) and the details include how these were obtained and would be kept up to date.

Format

A named list

Note

These were generated using the script in the `data-raw` directory (`misc.Rmd` file).

Examples

```
head(file_names)
```

find_file_name *Find file names within stats19::file_names.*

Description

Find file names within stats19::file_names.

Usage

```
find_file_name(years = NULL, type = NULL)
```

Arguments

`years`	Year for which data are to be found
`type`	One of 'collisions', 'casualty' or 'vehicles' ignores case.

Examples

```
find_file_name(2016)
```

format_casualties *Format STATS19 casualties*

Description

Format STATS19 casualties

Usage

```
format_casualties(x)
```

Arguments

x	Data frame created with `read_casualties()`

Details

This function formats raw STATS19 data

Examples

```
if(curl::has_internet()) {
dl_stats19(year = 2022, type = "casualty")
x = read_casualties(year = 2022)
casualties = format_casualties(x)
}
```

format_collisions *Format STATS19 'collisions' data*

Description

Format STATS19 'collisions' data

Usage

```
format_collisions(x)
```

Arguments

x	Data frame created with `read_collisions()`

Details

This is a helper function to format raw STATS19 data

Examples

```
if(curl::has_internet()) {
  dl_stats19(year = 2022, type = "collision")
}
```

`format_column_names` *Format column names of raw STATS19 data*

Description

This function takes messy column names and returns clean ones that work well with R by default. Names that are all lower case with no R-unfriendly characters such as spaces and - are returned.

Usage

```
format_column_names(column_names)
```

Arguments

`column_names`	Column names to be cleaned

Value

Column names cleaned.

Examples

```
if(curl::has_internet()) {
crashes_raw = read_collisions(year = 2022)
column_names = names(crashes_raw)
column_names
format_column_names(column_names = column_names)
}
```

`format_ppp` *Convert STATS19 data into ppp (spatstat) format.*

Description

This function is a wrapper around the `spatstat.geom::ppp()` function and it is used to transform STATS19 data into a ppp format.

Usage

```
format_ppp(data, window = NULL, ...)
```

Arguments

data	A STATS19 dataframe to be converted into ppp format.
window	A windows of observation, an object of class owin(). If window = NULL (i.e. the default) then the function creates an approximate bounding box covering the whole UK. It can also be used to filter only the events occurring in a specific region of UK (see the examples of get_stats19).
...	Additional parameters that should be passed to spatstat.geom::ppp() function. Read the help page of that function for a detailed description of the available parameters.

Value

A ppp object.

See Also

format_sf for an analogous function used to convert data into sf format and spatstat.geom::ppp() for the original function.

Examples

```
if (requireNamespace("spatstat.geom", quietly = TRUE)) {
  x_ppp = format_ppp(accidents_sample)
  x_ppp
}
```

---

format_sf *Format convert STATS19 data into spatial (sf) object*

Description

Format convert STATS19 data into spatial (sf) object

Usage

```
format_sf(x, lonlat = FALSE)
```

Arguments

x	Data frame created with read_collisions()
lonlat	Should the results be returned in longitude/latitude? By default FALSE, meaning the British National Grid (EPSG code: 27700) is used.

Examples

```
x_sf = format_sf(accidents_sample)
sf:::plot.sf(x_sf)
```

format_vehicles *Format STATS19 vehicles data*

Description

Format STATS19 vehicles data

Usage

```
format_vehicles(x)
```

Arguments

x	Data frame created with `read_vehicles()`

Details

This function formats raw STATS19 data

Examples

```
if(curl::has_internet()) {
dl_stats19(year = 2022, type = "vehicle", ask = FALSE)
x = read_vehicles(year = 2022, format = FALSE)
vehicles = format_vehicles(x)
}
```

get_data_directory *Get data download dir*

Description

Get data download dir

Usage

```
get_data_directory()
```

`get_MOT` *Download vehicle data from the DVSA MOT API using VRM.*

Description

Download vehicle data from the DVSA MOT API using VRM.

Usage

```
get_MOT(vrm, apikey)
```

Arguments

`vrm`	A list of VRMs as character strings.
`apikey`	Your API key as a character string.

Details

This function takes a a character vector of vehicle registrations (VRMs) and returns vehicle data from MOT records. It returns a data frame of those VRMs which were successfully used with the DVSA MOT API.

Information on the DVSA MOT API is available here: https://dvsa.github.io/mot-history-api-documentation/

The DVSA MOT API requires a registration. The function therefore requires the API key provided by the DVSA. Be aware that the API has usage limits. The function will therefore limit lists with more than 150,000 VRMs.

Examples

```
vrm = c("1RAC","P1RAC")
apikey = Sys.getenv("MOTKEY")
if(nchar(apikey) > 0) {
  get_MOT(vrm = vrm, apikey = apikey)
}
```

`get_stats19` *Download, read and format STATS19 data in one function.*

Description

Download, read and format STATS19 data in one function.

Usage

```
get_stats19(
  year = NULL,
  type = "collision",
  data_dir = get_data_directory(),
  file_name = NULL,
  format = TRUE,
  ask = FALSE,
  silent = FALSE,
  output_format = "tibble",
  engine = "readr",
  where = NULL,
  ...
)
```

Arguments

year	Single year for which data are to be read
type	One of 'collision', 'casualty', 'Vehicle'; defaults to 'collision'.
data_dir	Where sets of downloaded data would be found.
file_name	Character string of a specific STATS19 CSV filename to download/read. If NULL, filenames are inferred from year and type.
format	Switch to return raw read from file, default is TRUE.
ask	Should you be asked whether or not to download the files? TRUE by default.
silent	Boolean. If FALSE (default value), display useful progress messages on the screen.
output_format	A string that specifies the desired output format. The default value is "tibble". Other possible values are "data.frame", "sf" and "ppp", that, respectively, returns objects of class data.frame, sf::sf and spatstat.geom::ppp. Any other string is ignored and a tibble output is returned. See details and examples.
engine	CSV reader backend. Defaults to "readr". Set to "duckdb" to query files via DuckDB before loading into R.
where	Optional SQL predicate appended to the WHERE clause when engine = "duckdb", e.g. "longitude > -1.9 AND longitude < -1.2". Ignored when engine = "readr".
...	Other arguments be passed to format_sf() or format_ppp() functions. Read and run the examples.

Details

This function gets STATS19 data. Behind the scenes it uses dl_stats19() and read_* functions, returning a tibble (default), data.frame, sf or ppp object, depending on the output_format parameter.

By default, stats19 downloads files to a temporary directory. You can change this behavior to save the files in a permanent directory. This is done by setting the STATS19_DOWNLOAD_DIRECTORY environment variable. A convenient way to do this is by adding STATS19_DOWNLOAD_DIRECTORY=/path/to/a/dir to your .Renviron file, which can be opened with usethis::edit_r_environ().

The function returns data for a specific year (e.g. `year = 2022`)

Note: for years before 2016 the function may return data from more years than are requested due to the nature of the files hosted at data.gov.uk.

As this function uses `dl_stats19` function, it can download many MB of data, so ensure you have a sufficient disk space.

If `output_format = "data.frame"` or `output_format = "sf"` or `output_format = "ppp"` then the output data is transformed into a data.frame, sf or ppp object using the `as.data.frame()` or `format_sf()` or `format_ppp()` functions, as shown in the examples.

See Also

`dl_stats19()`

`read_collisions()`

Examples

```
if(curl::has_internet()) {
col = get_stats19(year = 2022, type = "collision")
cas = get_stats19(year = 2022, type = "casualty")
veh = get_stats19(year = 2022, type = "vehicle")
class(col)
# data.frame output
x = get_stats19(2022, silent = TRUE, output_format = "data.frame")
class(x)

# # Get 5-years worth of data (commented-out due to large response size):
# col_5 = get_stats19(year = 5, type = "collision")
# cas_5 = get_stats19(year = 5, type = "casualty")
# veh_5 = get_stats19(year = 5, type = "vehicle")


# Run tests only if endpoint is alive:
if(nrow(x) > 0) {

# use duckdb engine
col_duck = get_stats19(year = 2022, type = "collision", engine = "duckdb")

# use duckdb with where clause
col_where = get_stats19(year = 2022, type = "collision", engine = "duckdb",
                        where = "speed_limit = 30")

# sf output
x_sf = get_stats19(2022, silent = TRUE, output_format = "sf")

# sf output with lonlat coordinates
x_sf = get_stats19(2022, silent = TRUE, output_format = "sf", lonlat = TRUE)
sf::st_crs(x_sf)

if (requireNamespace("spatstat.geom", quietly = TRUE)) {
# ppp output
x_ppp = get_stats19(2022, silent = TRUE, output_format = "ppp")
```

```
# We can use the window parameter of format_ppp function to filter only the
# events occurred in a specific area. For example we can create a new bbox
# of 5km around the city center of Leeds

leeds_window = spatstat.geom::owin(
xrange = c(425046.1, 435046.1),
yrange = c(428577.2, 438577.2)
)

leeds_ppp = get_stats19(2022, silent = TRUE, output_format = "ppp", window = leeds_window)
spatstat.geom::plot.ppp(leeds_ppp, use.marks = FALSE, clipwin = leeds_window)
}
}
}
```

`get_stats19_adjustments` *Download and read-in severity adjustment factors*

Description

See the DfT's documentation on adjustment factors Annex: Update to severity adjustments methodology.

Usage

```
get_stats19_adjustments(
  data_dir = get_data_directory(),
  u = paste0("https://data.dft.gov.uk/road-accidents-safety-data/",
    "dft-road-casualty-statistics-casualty-adjustment-lookup_",
    "2004-latest-published-year.csv")
)
```

Arguments

`data_dir`	Where sets of downloaded data would be found.
`u`	The URL of the zip file with adjustments to download

Details

See Estimating and adjusting for changes in the method of severity reporting for road accidents and casualty data: final report for details.

Examples

```
## Not run:
if(curl::has_internet()) {
adjustment = get_stats19_adjustments()
}

## End(Not run)
```

`get_ULEZ` *Download DVLA-based vehicle data from the TfL API using VRM.*

Description

Download DVLA-based vehicle data from the TfL API using VRM.

Usage

```
get_ULEZ(vrm)
```

Arguments

`vrm`	A list of VRMs as character strings.

Details

This function takes a character vector of vehicle registrations (VRMs) and returns DVLA-based vehicle data from TfL's API, included ULEZ eligibility. It returns a data frame of those VRMs which were successfully used with the TfL API. Vehicles are either compliant, non-compliant or exempt. ULEZ-exempt vehicles will not have all vehicle details returned - they will simply be marked "exempt".

Be aware that the API has usage limits. The function will therefore limit API calls to below 50 per minute - this is the maximum rate before an API key is required.

Examples

```
if(curl::has_internet()) {
vrm = c("1RAC","P1RAC")
get_ULEZ(vrm = vrm)
}
```

get_url *Convert file names to urls*

Description

Convert file names to urls

Usage

```
get_url(
  file_name = "",
  domain = "https://data.dft.gov.uk",
  directory = "road-accidents-safety-data"
)
```

Arguments

file_name	Optional file name to add to the url returned (empty by default)
domain	The domain from where the data will be downloaded
directory	The subdirectory of the url

Examples

```
# get_url(find_file_name(1985))
```

locate_files *Locate a file on disk*

Description

Locate a file on disk

Usage

```
locate_files(
  data_dir = get_data_directory(),
  type = NULL,
  years = NULL,
  quiet = FALSE
)
```

Arguments

data_dir	Where sets of downloaded data would be found.
type	One of 'collision', 'casualty', 'Vehicle'; defaults to 'collision'.
years	Single year or vector of years for which data are to be read.
quiet	Print out messages (files found)

locate_one_file *Pin down a file on disk from parameters.*

Description

Pin down a file on disk from parameters.

Usage

```
locate_one_file(
  filename = NULL,
  data_dir = get_data_directory(),
  year = NULL,
  type = NULL
)
```

Arguments

filename	Character string of the filename of the .csv to read.
data_dir	Where sets of downloaded data would be found.
year	Single year for which data are to be read.
type	One of 'collision', 'casualty', 'Vehicle'; defaults to 'collision'.

Examples

```
locate_one_file()
```

match_tag *Match STATS19 collisions with TAG (RAS4001) cost estimates*

Description

Downloads and processes the UK Department for Transport **TAG Data Book** table **RAS4001**, and joins estimated collision costs to STATS19 data.

Three matching modes are available:

- `"severity"` — cost varies only by collision severity
- `"severity_road"` — cost varies by severity *and* road type
- `"severity_road_bua"` — cost varies by severity & road type, where road type is determined using **ONS Built-Up Area (BUA)** polygons (2022)

BUA polygons are downloaded automatically from: https://open-geography-portalx-ons.hub.arcgis.com/api/download/v1/items/ad30b234308f4b02b4bb9b0f4766f7bb/geoPackage?layers=0

Optionally, total costs may be summarised by severity and/or road type.

Usage

```
match_tag(
  crashes,
  shapes_url =
   paste0("https://open-geography-portalx-ons.hub.arcgis.com/api/download/v1/",
    "items/ad30b234308f4b02b4bb9b0f4766f7bb/geoPackage?layers=0"),
  costs_url = paste0("https://assets.publishing.service.gov.uk/media/",
    "68d421cc275fc9339a248c8e/ras4001.ods"),
  match_with = "severity",
  include_motorway_bua = FALSE,
  summarise = FALSE
)
```

Arguments

crashes	A STATS19 collision data frame. May be sf or non-sf, but spatial geometry is required when using "severity_road_bua".
shapes_url	URL to download the ONS Built-Up Areas geopackage. Defaults to the official ONS 2022 dataset.
costs_url	URL to download the TAG Data Book **RAS4001** table (ODS format). Defaults to the Department for Transport asset link.
match_with	Character string specifying the matching mode. One of: • "severity" • "severity_road" • "severity_road_bua" The default uses match_with = c("severity", "severity_road", "severity_road_bua"), so users get tab-completion and help in RStudio.
include_motorway_bua	Logical; if TRUE, motorways inside a built-up area polygon are treated as "built_up" rather than "Motorway".
summarise	Logical; if TRUE, returns a summary table of total costs (in millions) grouped by severity and/or road type.

Details

The function:

1. Downloads and parses **RAS4001** cost tables
2. Computes road type using STATS19 fields or optional BUA polygons
3. Joins appropriate cost estimates
4. Optionally aggregates totals

When crashes is an sf object and summarise = TRUE, geometry is automatically dropped.

Value

A data frame (or `sf` object if input is `sf`) with added columns for estimated collision costs. If `summarise = TRUE`, a summary table of total costs (in millions) is returned.

Examples

```
## Not run:
  # Simple severity-based matching
  match_tag(stats19_df, match_with = "severity")

  # Severity + road type
  match_tag(stats19_df, match_with = "severity_road")

  # Using ONS Built-Up Areas, with motorway override
  match_tag(
    stats19_df,
    match_with = "severity_road_bua",
    include_motorway_bua = TRUE
  )

  # Summarised totals
  match_tag(stats19_df, match_with = "severity", summarise = TRUE)

## End(Not run)
```

---

phrase *Generate a phrase for data download purposes*

Description

Generate a phrase for data download purposes

Usage

```
phrase()
```

---

police_boundaries *Police force boundaries in England (2016)*

Description

This dataset represents the 43 police forces in England and Wales. These are described on the Wikipedia page. on UK police forces.

Format

An sf data frame

Details

The geographic boundary data were taken from the UK government's official geographic data portal. See http://geoportal.statistics.gov.uk/

Note

These were generated using the script in the `data-raw` directory (`misc.Rmd` file) in the package's GitHub repo: github.com/ITSLeeds/stats19.

Examples

```
nrow(police_boundaries)
police_boundaries[police_boundaries$pfa16nm == "West Yorkshire", ]
sf:::plot.sf(police_boundaries)
```

---

read_casualties *Read in STATS19 road safety data from .csv files downloaded.*

Description

Read in STATS19 road safety data from .csv files downloaded.

Usage

```
read_casualties(
  year = NULL,
  filename = "",
  data_dir = get_data_directory(),
  format = TRUE
)
```

Arguments

`year`	Single year for which data are to be read
`filename`	Character string of the filename of the .csv to read, if this is given, type and years determine whether there is a target to read, otherwise disk scan would be needed.
`data_dir`	Where sets of downloaded data would be found.
`format`	Switch to return raw read from file, default is `TRUE`.

`read_collisions` *Read in STATS19 road safety data from .csv files downloaded.*

Description

Read in STATS19 road safety data from .csv files downloaded.

Usage

```
read_collisions(
  year = NULL,
  filename = "",
  data_dir = get_data_directory(),
  format = TRUE,
  silent = FALSE
)
```

Arguments

`year`	Single year for which data are to be read
`filename`	Character string of the filename of the .csv to read, if this is given, type and years determine whether there is a target to read, otherwise disk scan would be needed.
`data_dir`	Where sets of downloaded data would be found.
`format`	Switch to return raw read from file, default is `TRUE`.
`silent`	Boolean. If `FALSE` (default value), display useful progress messages on the screen.

Details

This is a wrapper function to access and load stats 19 data in a user-friendly way. The function returns a data frame, in which each record is a reported incident in the STATS19 data.

Examples

```
if(curl::has_internet()) {
dl_stats19(year = 2024, type = "collision")
ac = read_collisions(year = 2024)
}
```

read_vehicles *Read in stats19 road safety data from .csv files downloaded.*

Description

Read in stats19 road safety data from .csv files downloaded.

Usage

```
read_vehicles(
  year = NULL,
  filename = "",
  data_dir = get_data_directory(),
  format = TRUE
)
```

Arguments

year	Single year for which data are to be read
filename	Character string of the filename of the .csv to read, if this is given, type and years determine whether there is a target to read, otherwise disk scan would be needed.
data_dir	Where sets of downloaded data would be found.
format	Switch to return raw read from file, default is TRUE.

schema_original *Schema for stats19 data (UKDS)*

Description

Schema for stats19 data (UKDS)

Format

A data frame

select_file *Interactively select from options*

Description

Interactively select from options

Usage

```
select_file(fnames)
```

Arguments

fnames	Character vector of filenames to select from.

set_data_directory *Set data download dir*

Description

Set data download dir

Usage

```
set_data_directory(data_path)
```

Arguments

data_path	valid existing path to save downloaded files in.

stats19_schema *Stats19 schema and variables*

Description

`stats19_schema` and `stats19_variables` contain metadata on **stats19** data. `stats19_schema` is a look-up table matching codes provided in the raw stats19 dataset with character strings.

Note

The schema data can be (re-)generated using the script in the `data-raw` directory.

vehicles_sample *Sample of stats19 data (2022 vehicles)*

Description

Sample of stats19 data (2022 vehicles)

Format

A data frame

Note

These were generated using the script in the `data-raw` directory (`misc.Rmd` file).

Examples

```
nrow(vehicles_sample_raw)
vehicles_sample_raw
```

Index