

Package: geslaR (via r-universe)

October 4, 2026

Title Get and Manipulate the GESLA Dataset

Version 1.0-3

Description Promote access to the GESLA

<<https://gesla787883612.wordpress.com>> (Global Extreme Sea Level Analysis) dataset, a higher-frequency sea-level record data from all over the world. It provides functions to download it entirely, or query subsets directly into R, without the need of downloading the full dataset. Also, it provides a built-in web-application, so that users can apply basic filters to select the data of interest, generating informative plots, and showing the selected sites.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL <https://github.com/EireExtremes/geslaR>,
<https://eireextremes.github.io/geslaR/>

BugReports <https://github.com/EireExtremes/geslaR/issues>

Depends arrow, dplyr, R (>= 4.1.0)

Imports cli

Suggests bslib, DT, knitr, leaflet, leafpop, lubridate, patchwork,
plotly, rmarkdown, shiny, shinyalert, shinycssloaders,
shinyWidgets, testthat (>= 3.0.0), tidyverse

SystemRequirements arrow

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs cmake libssl-dev

Repository <https://eireextremes.r-universe.dev>

Date/Publication 2026-08-05 02:01:42 UTC

RemoteUrl <https://github.com/eireextremes/geslar>

RemoteRef HEAD
RemoteSha 0ef3090c2c87f4f3f7f8f372d60e37ad8ac08e0f

Contents

download_gesla	2
query_gesla	3
read_gesla	5
run_gesla_app	7
write_gesla	9

Index	11
--------------	-----------

download_gesla	<i>Download the GESLA dataset</i>
----------------	-----------------------------------

Description

This function will download the entire GESLA dataset to the specified folder. Note that the full dataset is about 7GB in size, so the total download time may take a few minutes, as it will depend on internet connection. If you don't need the whole dataset, you can use the [query_gesla\(\)](#) function, to directly import a subset of it.

Usage

```
download_gesla(  
  dest = "./gesla_dataset",  
  ask = TRUE,  
  messages = TRUE,  
  overwrite = FALSE  
)
```

Arguments

dest	The directory to download the files to. If the directory doesn't exist, it will be created. Defaults to a folder called gesla_dataset in the current working directory.
ask	Ask for confirmation before downloading? Defaults to TRUE.
messages	Show informative messages? Defaults to TRUE.
overwrite	Overwrite the whole dataset (i.e. download again)? Defaults to FALSE. Note that, if TRUE, it will only overwrite if the function is called in the same directory where dest is.

Details

This function should only be useful if you want to deal with all the files from the GESLA dataset. If you need only a subset, you can use the [query_gesla\(\)](#) function, or the GESLA Shiny app interface, from the [run_gesla_app\(\)](#) function.

Value

The whole GESLA dataset, consisting of 5119 files (with .parquet extension). It should have approximately 7GB in size.

Author(s)

Fernando Mayer <fernando.mayer@ufpr.br>

Examples

```
if(interactive()) {
  ## Create a temporary directory for downloaded files
  dest <- paste0(tempdir(), "/gesla_dataset")
  ## Download to 'gesla_dataset' folder in the temporary directory
  download_gesla(dest = dest)
  ## To overwrite (download again) on the same location
  download_gesla(dest = dest, overwrite = TRUE)
  ## Don't ask for confirmation before download
  download_gesla(dest = dest, overwrite = TRUE, ask = FALSE)
  ## Don't show informative messages
  download_gesla(dest = dest, overwrite = TRUE, messages = FALSE)
  ## Don't ask for confirmation neither show messages
  download_gesla(dest = dest, overwrite = TRUE,
    ask = FALSE, messages = FALSE)
  ## Remove temporary directory
  unlink(dest, recursive = TRUE)
}
```

query_gesla

Query the GESLA dataset

Description

This function will make a query to fetch a subset of the GESLA dataset. At least a country code and one year must be specified. Site names can also be specified, but are optional. By default, the resulting subset will contain only data that were revised and recommended for analysis, by the GESLA group of researchers.

Usage

```
query_gesla(
  country,
  year = NULL,
  site_name = NULL,
  use_flag = 1,
  as_data_frame = FALSE
)
```

Arguments

country	A character vector specifying the selected countries, using the three-letter ISO 3166-1 alpha-3 code. See Details.
year	A numeric vector specifying the selected years. If NULL (the default), all available years will be selected.
site_name	Optional character vector of site names.
use_flag	The default is 1, which means to use only the data that was revised and useful for analysis. Can be 0, to fetch only revised and not recommend for analysis, or c(0, 1) to fetch all the data. See Details.
as_data_frame	If FALSE (default), the data will be imported as an <code>arrow_dplyr_query</code> object. Otherwise, the data will be in a <code>tbl_df</code> (<code>data.frame</code>) format. See Details.

Details

The country codes must follow the three-letter **ISO 3166-1 alpha-3** code. However, note that not all countries are available at the GESLA dataset. If in doubt, use the `run_gesla_app()` function to check the GESLA Shiny app interface (`geslaR-app`) locally.

The `use_flag` argument must be 1 or 0, or `c(0, 1)`. The `use_flag` is a column at the GESLA dataset that indicates whether the data should be used for analysis or not. The 1 (default) indicates it should, and 0 the otherwise. In a data analysis scenario, the user must only be interested in using the recommended data, so this argument shouldn't be changed. However, in some cases, one must be interested in the non-recommended data, therefore this option is available. Also, you can specify `c(0, 1)` to fetch all the data (usable and not usable). In any case, the `use_flag` column will always be present, and it can be used for any post-processing. Please, see the **GESLA format** documentation for more details.

The default argument `as_data_frame = FALSE` will result in an object of the `arrow_dplyr_query` class. The advantage is that, regardless of the size of the resulting dataset, the object will be small in (memory) size. Also, as it happens with the Arrow Table class, it can be manipulated with `dplyr` verbs. Please, see the documentation at the **Arrow website**.

Note that, if the `as_data_frame` argument is set to TRUE, the imported R object will vary in size, according to the size of the subset. In many situations, this can take a long time and may even be infeasible, since the object can result in a "larger-than-memory" size, and possibly will make R operations slow or even a session crash. Therefore, we always recommend to start with `as_data_frame = FALSE`, and work with the dataset from there.

Please, see `vignette("intro-to-geslaR")` for a detailed example.

Value

An object of class `arrow_dplyr_query` or a `tbl_df` (`data.frame`).

Author(s)

Fernando Mayer <fernando.mayer@ufpr.br>

Examples

```
if(interactive()) {  
  ## Simple query  
  da <- query_gesla(country = "IRL")  
  
  ## Select one specific year  
  da <- query_gesla(country = "IRL", year = 2015)  
  
  ## Multiple years  
  da <- query_gesla(country = "IRL", year = c(2015, 2017))  
  da <- query_gesla(country = "IRL", year = 2010:2017)  
  da <- query_gesla(country = "IRL", year = c(2010, 2012, 2015))  
  da |>  
    count(year) |>  
    collect()  
  
  ## Multiple countries  
  da <- query_gesla(country = c("IRL", "ATA"), year = 2015)  
  da <- query_gesla(country = c("IRL", "ATA"), year = 2010:2017)  
  da |>  
    count(country, year) |>  
    collect()  
  
  ## Specifying a site name  
  da <- query_gesla(country = "IRL", year = c(2015, 2017),  
    site_name = "Dublin_Port")  
  da |>  
    count(year) |>  
    collect()  
}
```

read_gesla

Read a GESLA dataset

Description

Read a CSV or Parquet file, as exported from the GESLA Shiny app interface (geslaR-app). A "GESLA dataset file" is a subset of the GESLA dataset, fetched from the geslaR-app. When using that app, you can choose to download the selected subset in CSV or Parquet file formats. Whichever option is chosen this function will automatically identify the file type and use the appropriate functions to import the dataset to R.

This function can be used for exported files from the geslaR-app interface, run locally with the [run_gesla_app\(\)](#) function.

Usage

```
read_gesla(file, as_data_frame = FALSE, ...)
```

Arguments

<code>file</code>	The file name (must end in <code>.csv</code> or <code>.parquet</code> only)
<code>as_data_frame</code>	If <code>FALSE</code> (default), the data will be imported as an Arrow Table format. Otherwise, the data will be in a <code>tbl_df</code> (<code>data.frame</code>) format. See Details.
<code>...</code>	Other arguments from <code>arrow::read_csv_arrow()</code> , and <code>arrow::read_parquet()</code> , from the <code>arrow</code> package.

Details

We highly recommend to export subsets of the GESLA dataset from the `geslaR`-app in the Parquet file format. This format has a much smaller file size when compared to the CSV format.

In any case, the only difference between CSV and Parquet files will be the file size. However, when importing these data to R, both file types have the option to be imported as an Arrow Table format, which is the default (argument `as_data_frame = FALSE`). This way, the object created in R will have a very small size, independent of how big the file size is. To deal with this type of object, you can use `dplyr` verbs, in the same way as a normal `data.frame` (or `tbl_df`). Some examples can be found in the [Arrow documentation](#).

If the `as_data_frame` argument is set to `TRUE`, the imported R object will vary in size, according to the size of the dataset, and regardless of the file type. In many situations, this can be infeasible, since the object can result in a "larger-than-memory" size, and possibly will make R operations slow or even a session crash. Therefore, we always recommend to start with `as_data_frame = FALSE`, and work with the dataset from there.

See **Examples** below.

Value

An Arrow Table object, or a `tbl_df` (`data.frame`)

Author(s)

Fernando Mayer <fernando.mayer@ufpr.br>

Examples

```
##-----
## Import an internal example Parquet file
tmp <- tempdir()
file.copy(system.file(
  "extdata", "ireland.parquet", package = "geslaR"), tmp)
da <- read_gesla(paste0(tmp, "/ireland.parquet"))
## Check size in memory
object.size(da)

##-----
## Import an internal example CSV file
tmp <- tempdir()
file.copy(system.file(
  "extdata", "ireland.csv", package = "geslaR"), tmp)
```

```

da <- read_gesla(paste0(tmp, "/ireland.csv"))
## Check size in memory
object.size(da)

##-----
## Import an internal example Parquet file as data.frame
tmp <- tempdir()
file.copy(system.file(
  "extdata", "ireland.parquet", package = "geslaR"), tmp)
da <- read_gesla(paste0(tmp, "/ireland.parquet"),
  as_data_frame = TRUE)
## Check size in memory
object.size(da)

##-----
## Import an internal example CSV file as data.frame
tmp <- tempdir()
file.copy(system.file(
  "extdata", "ireland.csv", package = "geslaR"), tmp)
da <- read_gesla(paste0(tmp, "/ireland.csv"),
  as_data_frame = TRUE)
## Check size in memory
object.size(da)

## Remove files from temporary directory
unlink(paste0(tmp, "/ireland.parquet"))
unlink(paste0(tmp, "/ireland.csv"))

```

run_gesla_app

Run the GESLA Shiny app

Description

Run the GESLA Shiny app (geslaR-app) locally. The first time this function is called, it will check if the GESLA dataset is present. If not, it will prompt to download it or not. Please note that the entire GESLA dataset is about 7GB in size, so make sure there is enough space for it. The Shiny app will only work with the entire dataset downloaded locally.

Note, however, that the dataset needs to be downloaded only once, so the next time this function is called, the app will open instantly.

Usage

```

run_gesla_app(
  app_dest = "./gesla_app",
  dest = paste0(app_dest, "/gesla_dataset"),
  overwrite = FALSE,
  open = TRUE
)

```

Arguments

app_dest	The destination directory that will host the app and the database. It will be created if it doesn't exist. By default, it will create a directory called gesla_app in the current working directory.
dest	The destination directory that will host the GESLA dataset files. By default, it will create a subdirectory under the directory defined in app_dest. It's not recommended to change this argument. If needed, change only the app_dest argument.
overwrite	Overwrite the current dataset? If TRUE and called on the same directory as the app, it will overwrite (i.e. download again) the whole dataset. This is usually not necessary, unless the dataset has really changed.
open	Should the app open in the default browser? Defaults to TRUE.

Details

The geslaR-app Shiny interface relies on a set of packages, defined in the Suggests fields of the package DESCRIPTION file. When called for the first time, the function will check if all the packages are available. If one or more are not installed, a message will show which one of them should be installed. Alternatively, you can install all of them at once by reinstalling the geslaR package with `devtools::install_github("EireExtremes/geslaR", dependencies = TRUE)`. In this case, you will need to restart your R session.

When downloading the GESLA dataset for the first time, it may take a few minutes, since it depends on your internet connection and on the traffic on an Amazon AWS server. Don't stop the process before it ends completely. Note that this will be needed only the first time. Once the dataset is downloaded, the other time this function is called on the same directory, the interface should open in your browser instantly.

Value

The geslaR-app Shiny interface will open in your default browser.

Author(s)

Fernando Mayer <fernando.mayer@ufpr.br>

Examples

```
if(interactive()) {
  ##-----
  ## This will create a directory called `geslaR_app` on the current
  ## working directory and import the necessary files for the app.
  ## Also, it will create a subdirectory `gesla_app/gesla_dataset`,
  ## where the dataset will be downloaded.
  tmp <- paste0(tempdir(), "/gesla_app")
  run_gesla_app(app_dest = tmp)

  ##-----
  ## This function call on the same directory where the app is hosted,
  ## will overwrite the whole dataset (i.e. it will be downloaded
```

```

## again). A prompt for confirmation will be issued.
run_gesla_app(app_dest = tmp, overwrite = TRUE)

## Remove files from temporary directory
unlink(tmp, recursive = TRUE)
}

```

write_gesla

Write a GESLA dataset

Description

Write a CSV or Parquet file. Given an object `x`, this function will write a file in the appropriate format to store this object in the hard drive, facilitating it's reading in any other session.

The only accepted classes of `x` are `ArrowObject` or `data.frame`. If `x` is an `ArrowObject`, then the resulting file will have the `.parquet` extension, in the **Apache Parquet** file format. If `x` is a `data.frame`, the file will have a standard `.csv` extension.

This function is useful to save objects created by the `query_gesla()` function, for example. However, it may be used in any case where saving a (possible subset) of the GESLA dataset may be needed.

Usage

```
write_gesla(x, file_name = "gesla-data", ...)
```

Arguments

<code>x</code>	An object of class <code>ArrowObject</code> or <code>data.frame</code>
<code>file_name</code>	The name of the file to be created. Must be provided without extension, as this will be determined by the class of <code>x</code> .
<code>...</code>	Other arguments from <code>arrow::write_csv_arrow()</code> , and <code>arrow::write_parquet()</code> , from the arrow package.

Details

We highly recommend to always use the `ArrowObject` class, as it will be much more efficient for dealing with it in R. Also, the resulting file (with `.parquet` extension) from objects of this type will be much smaller than CSV files created from `data.frame` objects.

Value

A file with extension `.csv`, if `x` is a `data.frame`, or a file with extension `.parquet`, if `x` is an `ArrowObject`

Author(s)

Fernando Mayer <fernando.mayer@ufpr.br>

Examples

```
##-----
## Import an internal example Parquet file
## Reading file
tmp <- tempdir()
file.copy(system.file(
  "extdata", "ireland.parquet", package = "geslaR"), tmp)
da <- read_gesla(paste0(tmp, "/ireland.parquet"))
## Generates a subset by filtering
db <- da |>
  filter(day == 1) |>
  collect()
## Save filtered data as file
write_gesla(db, file_name = paste0(tmp, "/gesla-data"))

##-----
## Querying some data
## Make the query
if(interactive()) {
  da <- query_gesla(country = "IRL", year = 2019,
    site_name = "Dublin_Port")
  ## Save the resulting query to file
  write_gesla(da, file_name = paste0(tmp, "/gesla-data"))
}

## Remove files from temporary directory
unlink(paste0(tmp, "/gesla-data.csv"))
unlink(paste0(tmp, "/gesla-data.parquet"))
unlink(paste0(tmp, "/ireland.parquet"))
```

Index

`arrow::read_csv_arrow()`, [6](#)
`arrow::read_parquet()`, [6](#)
`arrow::write_csv_arrow()`, [9](#)
`arrow::write_parquet()`, [9](#)

`download_gesla`, [2](#)

`query_gesla`, [3](#)
`query_gesla()`, [2](#)

`read_gesla`, [5](#)
`run_gesla_app`, [7](#)
`run_gesla_app()`, [2](#), [4](#), [5](#)

`write_gesla`, [9](#)