

Package: cs9example (via r-universe)

October 17, 2026

Title A skeleton implementation of cs9 (Core Surveillance 9)

Version 26.8.23

Description A skeleton implementation of cs9 (Surveillance Core 9).

License MIT + file LICENSE

URL <https://www.csids.no/cs9/>

BugReports <https://github.com/csids/cs9/issues>

Depends R (>= 4.1.0)

Imports data.table, dplyr, csdata, fs, ggplot2, rmarkdown, cs9, csdb,
csmaps, cstdy, cstime, glue, httr, plnr, progress, progressr,
stringr

Suggests testthat, usethis

Remotes niphr/cs9, raubreywhite/plnr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/Needs/website niphr/cstemplate

Config/roxygen2/version 8.0.0

Config/pak/sysreqs
cmake git make libgit2-dev libicu-dev libuv1-dev unixodbc-dev libssl-dev libx11-dev

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

Date/Publication 2026-08-26 21:50:59 UTC

RemoteUrl <https://github.com/niphr/cs9example>

RemoteRef HEAD

RemoteSha 6f35b50a44fec98ff3ab22162456bb6bc4702596

Contents

global	2
make_skeleton_date	3
make_skeleton_isoyearweek	4
set_definitions	5
weather_clean_data_action	5
weather_clean_data_data_selector	6
weather_download_and_import_rawdata_action	7
weather_download_and_import_rawdata_data_selector	7
weather_export_plots_action	8
weather_export_plots_data_selector	9

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

global	<i>Declaration of environments that can be used globally</i>
--------	--

Description

cs9example creates global empty, and fills it when the package loads. It holds two things every task needs. global\$border is the Norwegian border year the package is configured for. global\$ss is a [cs9::SurveillanceSystem_v9](#) object with every database table and every task already registered.

Usage

global

See Also

[cs9::SurveillanceSystem_v9](#) for the class global\$ss is an instance of, and [make_skeleton_date\(\)](#) for the skeleton builder the tasks use. cs9example has no vignettes of its own; the framework is documented in the cs9 package.

Examples

```
# The border year, set when the package loads.
global$border

# The two tables and the three tasks the package registers.
names(global$ss$tables)
names(global$ss$tasks)
```

make_skeleton_date	Create Date-Based Data Skeleton
--------------------	---------------------------------

Description

Creates a data skeleton with complete date sequences and location codes for the geographic granularities you specify.

Usage

```
make_skeleton_date(  
  date_min,  
  date_max,  
  granularity_geo,  
  location_reference = csdata::nor_locations_names()  
)
```

Arguments

date_min	Start date (Date or character in YYYY-MM-DD format)
date_max	End date (Date or character in YYYY-MM-DD format)
granularity_geo	Vector of granularity_geo values to include
location_reference	Location reference data, defaults to csdata::nor_locations_names()

Value

data.table with complete skeleton

See Also

[weather_clean_data_action\(\)](#), the task action that starts from this skeleton, and [csdata::nor_locations_names\(\)](#) for the default location reference. `cs9example` has no vignettes of its own; the framework is documented in the `cs9` package.

Other skeleton builders: [make_skeleton_isoyearweek\(\)](#)

Examples

```
# Every county, every day of the first week of 2024, with no data merged in  
# yet: 15 counties x 7 days.  
skeleton <- make_skeleton_date(  
  date_min = "2024-01-01",  
  date_max = "2024-01-07",  
  granularity_geo = "county"  
)  
nrow(skeleton)  
head(skeleton)
```

make_skeleton_isoyearweek

Create ISO Year-Week Data Skeleton

Description

Creates a data skeleton with complete ISO year-week sequences and location codes for the geographic granularities you specify.

Usage

```
make_skeleton_isoyearweek(
  isoyearweek_min,
  isoyearweek_max,
  granularity_geo,
  location_reference = csdata::nor_locations_names()
)
```

Arguments

`isoyearweek_min`
Start ISO year-week (character in YYYY-WW format)

`isoyearweek_max`
End ISO year-week (character in YYYY-WW format)

`granularity_geo`
Vector of `granularity_geo` values to include

`location_reference`
Location reference data, defaults to `csdata::nor_locations_names()`

Value

data.table with complete skeleton

See Also

[make_skeleton_date\(\)](#) for the daily equivalent, and [csdata::nor_locations_names\(\)](#) for the default location reference. `cs9example` has no vignettes of its own; the framework is documented in the `cs9` package.

Other skeleton builders: [make_skeleton_date\(\)](#)

Examples

```
# Every county, the first three ISO weeks of 2024: 15 counties x 3 weeks.
skeleton <- make_skeleton_isoyearweek(
  isoyearweek_min = "2024-01",
  isoyearweek_max = "2024-03",
  granularity_geo = "county"
```

```
)
nrow(skeleton)
head(skeleton)
```

set_definitions	Set global definitions
-----------------	------------------------

Description

Set global definitions

Usage

```
set_definitions()
```

weather_clean_data_action	<i>weather_clean_data (action)</i>
---------------------------	------------------------------------

Description

Lays out a complete municipality-by-date skeleton with [make_skeleton_date\(\)](#). Merges the raw weather onto that skeleton. Aggregates it up to county and to nation. Aggregates the whole lot up again to ISO year-week. The action formats both granularities as csfmt_rts_data_v2, stacks them, and writes them to anon_example_weather_data. The write replaces whatever was there.

Usage

```
weather_clean_data_action(data, argset, tables)
```

Arguments

data	Data
argset	Argset
tables	DB tables

See Also

[weather_clean_data_data_selector\(\)](#), which produces this action’s data argument, and [make_skeleton_date\(\)](#) for the skeleton it starts from. cs9example has no vignettes of its own; how cs9 pairs an action with a data selector is documented in [cs9::SurveillanceSystem_v9](#).
Other cs9 task actions: [weather_download_and_import_rawdata_action\(\)](#), [weather_export_plots_action\(\)](#)

Examples

```
## Not run:
# Needs a live cs9 PostgreSQL database: the action ends by dropping every row
# of anon_example_weather_data and inserting the cleaned data in its place.
global$ss$run_task("weather_clean_data")

## End(Not run)
```

```
weather_clean_data_data_selector
      weather_clean_data (data selector)
```

Description

Pulls the daily, municipality-level, total-age, total-sex rows out of anon_example_weather_rawdata. Returns them under the name date_municipip, ordered by location and date.

Usage

```
weather_clean_data_data_selector(argset, tables)
```

Arguments

argset	Argset
tables	DB tables

Value

A named list with one element, date_municipip: a data.table of granularity_time, location_code, date, temp_max, temp_min and precip.

See Also

[weather_clean_data_action\(\)](#), which consumes this list, and [cs9::mandatory_db_filter\(\)](#) for the filter it applies. cs9example has no vignettes of its own; how cs9 pairs a data selector with an action is documented in [cs9::SurveillanceSystem_v9](#).

Other cs9 task data selectors: [weather_download_and_import_rawdata_data_selector\(\)](#), [weather_export_plots_data_selector\(\)](#)

Examples

```
## Not run:
# Needs a live cs9 PostgreSQL database: it reads the
# anon_example_weather_rawdata table.
weather_clean_data_data_selector(
  argset = global$ss$shortcut_get_argset("weather_clean_data"),
  tables = global$ss$shortcut_get_tables("weather_clean_data")
)

## End(Not run)
```

```
weather_download_and_import_rawdata_action
      weather_download_and_import_rawdata (action)
```

Description

Keeps the 00, 06, 12 and 18 o'clock timepoints of the forecast that [weather_download_and_import_rawdata_data_selector](#) returned. Reduces them to one row per date: maximum air temperature, minimum air temperature and total precipitation. Upserts those rows into anon_example_weather_rawdata.

Usage

```
weather_download_and_import_rawdata_action(data, argset, tables)
```

Arguments

data	Data
argset	Argset
tables	DB tables

See Also

[weather_download_and_import_rawdata_data_selector\(\)](#), which produces this action's data argument. cs9example has no vignettes of its own; how cs9 pairs an action with a data selector is documented in [cs9::SurveillanceSystem_v9](#).

Other cs9 task actions: [weather_clean_data_action\(\)](#), [weather_export_plots_action\(\)](#)

Examples

```
## Not run:
# Needs a live cs9 PostgreSQL database: the last thing the action does is
# upsert into the anon_example_weather_rawdata table.
global$ss$run_task("weather_download_and_import_rawdata")

## End(Not run)
```

```
weather_download_and_import_rawdata_data_selector
      weather_download_and_import_rawdata (data selector)
```

Description

Takes the mean latitude and longitude of the municipality named in `argset$location_code`. Downloads the MET Norway location forecast for that point. Returns it under the name data.

Usage

```
weather_download_and_import_rawdata_data_selector(argset, tables)
```

Arguments

argset	Argset
tables	DB tables

Value

A named list with one element, data: the parsed forecast.

See Also

[weather_download_and_import_rawdata_action\(\)](#), which consumes this list. cs9example has no vignettes of its own; how cs9 pairs a data selector with an action is documented in [cs9::SurveillanceSystem_v9](#).

Other cs9 task data selectors: [weather_clean_data_data_selector\(\)](#), [weather_export_plots_data_selector\(\)](#)

Examples

```
## Not run:
# Needs internet access to api.met.no, and the csmaps package for the
# municipality's coordinates.
weather_download_and_import_rawdata_data_selector(
  argset = list(location_code = "municip_nor0301"),
  tables = NULL
)

## End(Not run)
```

```
weather_export_plots_action
      weather_export_plots (action)
```

Description

Creates `argset$output_dir`, draws `data$data` as a ribbon between `temp_min` and `temp_max` over date, and saves it with [ggplot2::ggsave\(\)](#) to `argset$output_absolute_path`. That path is a `glue::glue()` template evaluated once against `argset`, so `"{argset$output_dir}"` resolves but a brace left inside `argset$output_filename` does not.

Usage

```
weather_export_plots_action(data, argset, tables)
```

Arguments

data	Data
argset	Argset
tables	DB tables

Details

This is the one action that touches no database table, so tables is unused and the example below passes NULL for it.

See Also

[weather_export_plots_data_selector\(\)](#), which produces this action's data argument. cs9example has no vignettes of its own; how cs9 pairs an action with a data selector is documented in [cs9::SurveillanceSystem_v9](#).

Other cs9 task actions: [weather_clean_data_action\(\)](#), [weather_download_and_import_rawdata_action\(\)](#)

Examples

```
# An argset shaped like the one cs9 builds for this task, and a stand-in for
# what the data selector would have returned.
d <- list(data = data.table::data.table(
  date = as.Date("2024-01-01") + 0:29,
  temp_min = seq(-8, 6, length.out = 30),
  temp_max = seq(-2, 12, length.out = 30)
))
argset <- list(
  location_code = "county_nor03",
  output_dir = fs::path(tempdir(), "cs9example-plots"),
  output_filename = "weather_county_nor03.png",
  output_absolute_path = fs::path("{argset$output_dir}", "{argset$output_filename}"),
  first_analysis = TRUE,
  last_analysis = TRUE
)

weather_export_plots_action(data = d, argset = argset, tables = NULL)
fs::dir_ls(argset$output_dir)
```

```
weather_export_plots_data_selector
```

```
  weather_export_plots (data selector)
```

Description

Pulls every row of anon_example_weather_data for the location named in argset\$location_code. Returns the date, temp_max and temp_min columns under the name data, ordered by date.

Usage

```
weather_export_plots_data_selector(argset, tables)
```

Arguments

argset	Argset
tables	DB tables

Value

A named list with one element, data: a data.table of date, temp_max and temp_min.

See Also

[weather_export_plots_action\(\)](#), which consumes this list, and [cs9::mandatory_db_filter\(\)](#) for the filter it applies. cs9example has no vignettes of its own; how cs9 pairs a data selector with an action is documented in [cs9::SurveillanceSystem_v9](#).

Other cs9 task data selectors: [weather_clean_data_data_selector\(\)](#), [weather_download_and_import_rawdata_data_selector\(\)](#)

Examples

```
## Not run:
# Needs a live cs9 PostgreSQL database: it reads the
# anon_example_weather_data table.
weather_export_plots_data_selector(
  argset = global$ss$shortcut_get_argset("weather_export_plots"),
  tables = global$ss$shortcut_get_tables("weather_export_plots")
)

## End(Not run)
```

Index

- * **cs9 task actions**
 - weather_clean_data_action, [5](#)
 - weather_download_and_import_rawdata_action, [6, 7, 10](#)
 - weather_export_plots_action, [8](#)
 - weather_export_plots_action(), [5, 7, 10](#)
 - weather_export_plots_data_selector, [9](#)
 - weather_export_plots_data_selector(), [6, 8, 9](#)
- * **cs9 task data selectors**
 - weather_clean_data_data_selector, [6](#)
 - weather_download_and_import_rawdata_data_selector, [7](#)
 - weather_export_plots_data_selector, [9](#)
- * **skeleton builders**
 - make_skeleton_date, [3](#)
 - make_skeleton_isoyearweek, [4](#)
- cs9::mandatory_db_filter(), [6, 10](#)
- cs9::SurveillanceSystem_v9, [2, 5–10](#)
- csdata::nor_locations_names(), [3, 4](#)
- ggplot2::ggsave(), [8](#)
- global, [2](#)
- make_skeleton_date, [3](#)
- make_skeleton_date(), [2, 4, 5](#)
- make_skeleton_isoyearweek, [4](#)
- make_skeleton_isoyearweek(), [3](#)
- set_definitions, [5](#)
- weather_clean_data_action, [5](#)
- weather_clean_data_action(), [3, 6, 7, 9](#)
- weather_clean_data_data_selector, [6](#)
- weather_clean_data_data_selector(), [5, 8, 10](#)
- weather_download_and_import_rawdata_action, [7](#)
- weather_download_and_import_rawdata_action(), [5, 8, 9](#)
- weather_download_and_import_rawdata_data_selector, [7](#)