

Package: cstidy (via r-universe)

September 26, 2026

Title Helpful Functions for Cleaning Surveillance Data

Version 2026.8.21

Description Helpful functions for the cleaning and manipulation of surveillance data, especially with regards to the creation and validation of panel data from individual level surveillance data.

Depends R (>= 3.5.0)

License MIT + file LICENSE

URL <https://niphr.github.io/cstidy/>, <https://github.com/niphr/cstidy>

BugReports <https://github.com/niphr/cstidy/issues>

Encoding UTF-8

LazyData true

Imports data.table, ggplot2, csdata, cstime, crayon, digest, stringr, methods

Suggests testthat, knitr, rmarkdown, rstudioapi, glue, gt, dplyr, purrr

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/Needs/website niphr/cstemplate

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-08-26 21:51:22 UTC

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

RemoteRef HEAD

RemoteSha 1bfb159e796a5d4b43db4ab7944960c1dd23cf05

Contents

expand_time_to 2

generate_test_data 3

heal_time_csfmt_rts_data_v1 4

heal_time_csfmt_rts_data_v2 5

identify_data_structure 6

nor_covid19_cases_by_time_location_csfmt_rts_v1 7

nor_covid19_icu_and_hospitalization_csfmt_rts_v1 8

remove_class_csfmt_rts_data 9

set_csfmt_rts_data_v1 10

set_csfmt_rts_data_v2 13

set_csfmt_rts_data_v3 18

unique_time_series 19

Index 21

expand_time_to	<i>Expand time to</i>
----------------	-----------------------

Description

Attempts to expand the dataset to include more time

A time series is defined as a unique combination of:

- granularity_time
- granularity_geo
- country_iso3
- location_code
- border
- age
- sex
- *_id
- *_tag

Usage

```
expand_time_to(  
  x,  
  max_isoyear = NULL,  
  max_isoyearweek = NULL,  
  max_date = NULL,  
  ...  
)
```

Arguments

<code>x</code>	An object of type <code>csfmt_rts_data_v2</code>
<code>max_isoyear</code>	Maximum isoyear to expand each isoyear time series up to.
<code>max_isoyearweek</code>	Maximum isoyearweek to expand each isoyearweek time series up to.
<code>max_date</code>	Maximum date to expand each daily time series up to.
<code>...</code>	Not used.

Value

`data.table`, a larger dataset that includes more rows corresponding to more time. The `csfmt_rts_data_v2` class is dropped from the result. Call `set_csfmt_rts_data_v2()` on it to put the class back.

See Also

The data-format vignette runs this function on a test dataset: `vignette("csfmt_rts_data_v2", package = "cstidy")`.

Other `csfmt_rts_data`: `identify_data_structure()`, `remove_class_csfmt_rts_data()`, `set_csfmt_rts_data_v1()`, `set_csfmt_rts_data_v2()`, `set_csfmt_rts_data_v3()`, `unique_time_series()`

Examples

```
x <- cstidy::generate_test_data() |>
  cstidy::set_csfmt_rts_data_v2()
cstidy::expand_time_to(x, max_isoyearweek = "2022-10")
```

<code>generate_test_data</code>	<i>Generate test data</i>
---------------------------------	---------------------------

Description

Generates some test data

Usage

```
generate_test_data(fmt = "csfmt_rts_data_v2")
```

Arguments

<code>fmt</code>	Data format (<code>csfmt_rts_data_v2</code>)
------------------	--

Value

`data.table`, a dataset containing fake data. The result is a plain `data.table`; pass it to `set_csfmt_rts_data_v2()` to give it the `csfmt_rts_data_v2` class.

See Also

[set_csfmt_rts_data_v2\(\)](#) to classify the result. Two vignettes call this function in a code chunk: [vignette\("cstidy", package = "cstidy"\)](#) and [vignette\("csfmt_rts_data_v2", package = "cstidy"\)](#).

Examples

```
cstidy::generate_test_data("csfmt_rts_data_v2")
```

```
heal_time_csfmt_rts_data_v1
```

Provides corresponding healed times (deprecated)

Description

Looks up the time columns (such as isoyear, isoweek, season and date) that correspond to a vector of dates, isoyearweeks or isoyears. Returns them as a data.table restricted to the requested columns.

Usage

```
heal_time_csfmt_rts_data_v1(x, cols, granularity_time = "date")
```

Arguments

x	A vector containing either dates, isoyearweek, or isoyear.
cols	Columns to restrict the output to.
granularity_time	date, isoyearweek, or isoyear, depending on the values contained in x.

Value

data.table, a dataset with time columns corresponding to the values given in x.

Deprecated

This lookup is deprecated along with the csfmt_rts_data_v1 format it serves. Nothing warns at run time and nothing has been removed; the mark is a signpost for new work. [heal_time_csfmt_rts_data_v2\(\)](#) replaces it. The two take the same three arguments. v2 accepts every granularity_time this function accepts ("date", "isoyearweek", "isoyear"), plus "season". v2 can also return isoquarter and isoyearquarter. See [set_csfmt_rts_data_v1\(\)](#) for the format itself.

See Also

No vignette covers this function. [set_csfmt_rts_data_v1\(\)](#) calls it while healing a csfmt_rts_data_v1. Other time healing lookups: [heal_time_csfmt_rts_data_v2\(\)](#)

Examples

```
cstidy::heal_time_csfmt_rts_data_v1(
  as.Date(c("2022-01-01", "2022-06-15")),
  cols = c("isoyear", "isoyearweek", "date"),
  granularity_time = "date"
)
```

```
heal_time_csfmt_rts_data_v2
```

Provides corresponding healed times (deprecated)

Description

Looks up the time columns (such as isoyear, isoweek, isoquarter, season and date) that correspond to a vector of dates, isoyearweeks, seasons or isoyears. Returns them as a data.table restricted to the requested columns.

Usage

```
heal_time_csfmt_rts_data_v2(x, cols, granularity_time = "date")
```

Arguments

x	A vector containing dates, isoyearweek, season, or isoyear.
cols	Columns to restrict the output to.
granularity_time	One of "date", "isoyearweek", "season", or "isoyear", matching the values contained in x.

Value

data.table, a dataset with time columns corresponding to the values given in x.

Deprecated

This lookup is deprecated as a public entry point, along with the csfmt_rts_data_v2 format it was written for. Nothing warns at run time, and it is not going away. `heal.csfmt_rts_data_v3()` calls it to derive v3's time columns. It is still the healing engine behind [set_csfmt_rts_data_v3\(\)](#). See [set_csfmt_rts_data_v2\(\)](#) for what replaces the format, and for the three limits of that replacement.

See Also

No vignette covers this function. [set_csfmt_rts_data_v2\(\)](#) and [set_csfmt_rts_data_v3\(\)](#) both call it while healing.

Other time healing lookups: [heal_time_csfmt_rts_data_v1\(\)](#)

Examples

```
cstidy::heal_time_csfmt_rts_data_v2(
  c("2022-01", "2022-02"),
  cols = c("isoyear", "isoweek", "season", "date"),
  granularity_time = "isoyearweek"
)
```

```
identify_data_structure
```

Hash the data structure of a dataset for a given column

Description

Summarises the data structure of a single column inside a dataset. For each combination of `granularity_time`, `granularity_geo`, `age` and `sex` it records the state of the column. The four states are structurally missing, only NA, only data, and a mix of data and NA. Pass the result to `plot()` for a visual overview.

Usage

```
identify_data_structure(x, col, ...)

## S3 method for class 'csfmt_rts_data_v2'
identify_data_structure(x, col, ...)

## S3 method for class '`tbl_Microsoft SQL Server`'
identify_data_structure(x, col, ...)
```

Arguments

<code>x</code>	An object of type <code>csfmt_rts_data_v2</code> .
<code>col</code>	Column name (character) whose data structure is summarised.
<code>...</code>	Arguments passed to or from other methods.

Value

`csfmt_rts_data_structure_hash_v2`, a summary object that can be plotted.

Deprecated

The `csfmt_rts_data_v2` method of `identify_data_structure()` is deprecated along with the format it dispatches on. Nothing warns at run time. See `set_csfmt_rts_data_v2()`. Neither the generic itself nor its `tbl_Microsoft SQL Server` method is deprecated.

See Also

Two vignettes run this function and plot the result: `vignette("cstidy", package = "cstidy")` and `vignette("csfmt_rts_data_v2", package = "cstidy")`.

Other `csfmt_rts_data`: [expand_time_to\(\)](#), [remove_class_csfmt_rts_data\(\)](#), [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v2\(\)](#), [set_csfmt_rts_data_v3\(\)](#), [unique_time_series\(\)](#)

Examples

```
cstidy::generate_test_data() |>
  cstidy::set_csfmt_rts_data_v2() |>
  cstidy::identify_data_structure("deaths_n") |>
  plot()
```

`nor_covid19_cases_by_time_location_csfmt_rts_v1`

Covid-19 data for PCR-confirmed cases in Norway (nation and county)

Description

This data comes from the Norwegian Surveillance System for Communicable Diseases (MSIS). The date corresponds to when the PCR-test was taken.

Usage

```
nor_covid19_cases_by_time_location_csfmt_rts_v1
```

Format

A `csfmt_rts_data_v1` with 11028 rows and 18 variables:

granularity_time day/isoweek

granularity_geo nation, county

country_iso3 nor

location_code norge, 11 counties

border 2020

age total

isoyear Isoyear of event

isoweek Isoweek of event

isoyearweek Isoyearweek of event

season Season of event

seasonweek Seasonweek of event

calyear Calyear of event

calmonth Calmonth of event

calyearmonth Calyearmonth of event

date Date of event

covid19_cases_testdate_n Number of confirmed covid19 cases

covid19_cases_testdate_pr100000 Number of confirmed covid19 cases per 100.000 population

Details

The raw number of cases and cases per 100.000 population are recorded.

This data was extracted on 2022-05-04.

Source

https://github.com/folkehelseinstituttet/surveillance_data/blob/master/covid19/_DOCUMENTATION_data_covid19_msis_by_time_location.txt

Examples

```
head(cstidy::nor_covid19_cases_by_time_location_csfmt_rts_v1)
```

```
nor_covid19_icu_and_hospitalization_csfmt_rts_v1
```

Norwegian Covid-19 data for ICU and hospitalization

Description

This data was extracted on 2022-05-04.

Usage

```
nor_covid19_icu_and_hospitalization_csfmt_rts_v1
```

Format

A csfmt_rts_data_v1 with 919 rows and 18 variables:

granularity_time day/isoweek

granularity_geo nation

country_iso3 nor

location_code norge

border 2020

age total

isoyear Isoyear of event

isoweek Isoweek of event

isoyearweek Isoyearweek of event
season Season of event
seasonweek Seasonweek of event
calyear Calyear of event
calmonth Calmonth of event
calyearmonth Calyearmonth of event
date Date of event
icu_with_positive_pcr_n Number of new admissions to the ICU with a positive PCR test
hospitalization_with_covid19_as_primary_cause_n Number of new hospitalizations with Covid-19 as the primary cause

Source

https://github.com/folkehelseinstituttet/surveillance_data/blob/master/covid19/_DOCUMENTATION_data_covid19_hospital_by_time.txt

Examples

```
head(cstidy::nor_covid19_icu_and_hospitalization_csfmt_rts_v1)
```

```
remove_class_csfmt_rts_data
      Remove class csfmt_rts_data_*
```

Description

Remove class csfmt_rts_data_*

Usage

```
remove_class_csfmt_rts_data(x)
```

Arguments

x data.table

Value

x, modified by reference, invisibly. It returns x rather than nothing, which is what lets it sit in the middle of a pipe.

See Also

Two vignettes run this function inside a pipe: `vignette("cstidy", package = "cstidy")` and `vignette("csfmt_rts_data_v2", package = "cstidy")`.

Other csfmt_rts_data: `expand_time_to()`, `identify_data_structure()`, `set_csfmt_rts_data_v1()`, `set_csfmt_rts_data_v2()`, `set_csfmt_rts_data_v3()`, `unique_time_series()`

Examples

```
x <- cstdy::generate_test_data() |>
  cstdy::set_csfmt_rts_data_v2()
class(x)
cstdy::remove_class_csfmt_rts_data(x)
class(x)
```

set_csfmt_rts_data_v1 *Convert data.table to csfmt_rts_data_v1 (deprecated)*

Description

set_csfmt_rts_data_v1 converts a data.table to csfmt_rts_data_v1 by reference. csfmt_rts_data_v1 creates a new csfmt_rts_data_v1 (not by reference) from a data.table. Both stop with an error when x is not a data.table; call data.table::setDT() first.

Usage

```
set_csfmt_rts_data_v1(x, create_unified_columns = TRUE, heal = TRUE)
```

```
csfmt_rts_data_v1(x, create_unified_columns = TRUE, heal = TRUE)
```

Arguments

x	The data.table to be converted to csfmt_rts_data_v1
create_unified_columns	Do you want it to create unified columns?
heal	Derive the missing time and geography columns on creation? These are deterministically looked up from the time and location columns you supply (see cstime and csdata). Nothing is statistically imputed and no count is invented. Time healing reads granularity_time to decide which time column the others are derived from, so supply it.

Value

An extended data.table, which has been modified by reference and returned (invisibly).

Returns a duplicated csfmt_rts_data_v1.

Smart assignment

csfmt_rts_data_v1 contains the smart assignment feature for time and geography.

When the **variables in bold** are assigned using `:=`, the listed variables are automatically re-derived from it. This is deterministic derivation from a calendar and a geography lookup, not statistical imputation.

location_code:

- granularity_geo

- country_iso3

isoyear:

- granularity_time
- isoweek
- isoyearweek
- season
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

isoyearweek:

- granularity_time
- isoyear
- isoweek
- season
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

date:

- granularity_time
- isoyear
- isoweek
- isoyearweek
- season
- seasonweek
- calyear
- calmonth
- calyearmonth

Unified columns

csfmt_rts_data_v1 contains 16 unified columns:

- granularity_time
- granularity_geo
- country_iso3
- location_code
- border
- age
- sex
- isoyear
- isoweek
- isoyearweek
- season
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

Deprecated

csfmt_rts_data_v1 is deprecated. The format still works and nothing warns at run time; the mark is a signpost for new work, not a removal notice. [set_csfmt_rts_data_v2\(\)](#) is the replacement. Two properties of that move were measured, and both come out clean. All 16 of v1's unified columns are among v2's 18, the two extra ones being isoquarter and isoyearquarter. v2 heals from every granularity_time v1 heals from ("date", "isoyearweek", "isoyear") and from "season" as well. csdb exports a field-type validator for both formats, so either can be written to the database.

v2 is itself deprecated, in favour of [set_csfmt_rts_data_v3\(\)](#). That further step is not lossless. Read the Deprecated section of [set_csfmt_rts_data_v2\(\)](#) before going past v2.

See Also

No vignette runs this function. The benchmarks vignette reports its run time, but that vignette is precompiled and carries no runnable code: `vignette("benchmarks", package = "cstidy")`.

Other csfmt_rts_data: [expand_time_to\(\)](#), [identify_data_structure\(\)](#), [remove_class_csfmt_rts_data\(\)](#), [set_csfmt_rts_data_v2\(\)](#), [set_csfmt_rts_data_v3\(\)](#), [unique_time_series\(\)](#)

Other csfmt format converters: [set_csfmt_rts_data_v2\(\)](#), [set_csfmt_rts_data_v3\(\)](#)

Examples

```
# granularity_time names the time column that the others are derived from.
d <- data.table::data.table(
  granularity_time = "isoyearweek",
  isoyearweek = c("2022-01", "2022-02"),
  location_code = "county_nor03",
  deaths_n = c(3L, 5L)
)

# set_csfmt_rts_data_v1() converts d in place and returns it invisibly.
cstidy::set_csfmt_rts_data_v1(d)
d[]
class(d)

# csfmt_rts_data_v1() copies instead, so e is left as it was.
e <- data.table::data.table(
  granularity_time = "isoyearweek",
  isoyearweek = c("2022-01", "2022-02"),
  location_code = "county_nor03",
  deaths_n = c(3L, 5L)
)
y <- cstidy::csfmt_rts_data_v1(e)
class(y)
class(e)
names(e)
```

set_csfmt_rts_data_v2 *Convert data.table to csfmt_rts_data_v2 (deprecated)*

Description

set_csfmt_rts_data_v2 converts a data.table to csfmt_rts_data_v2 by reference. csfmt_rts_data_v2 creates a new csfmt_rts_data_v2 (not by reference) from a data.table. Both stop with an error when x is not a data.table; call data.table::setDT() first.

Usage

```
set_csfmt_rts_data_v2(x, create_unified_columns = TRUE, heal = TRUE)
```

```
csfmt_rts_data_v2(x, create_unified_columns = TRUE, heal = TRUE)
```

Arguments

x The data.table to be converted to csfmt_rts_data_v2

create_unified_columns Do you want it to create unified columns?

heal Derive the missing time and geography columns on creation? These are deterministically looked up from the time and location columns you supply (see `cstime` and `csdata`). Nothing is statistically imputed and no count is invented. Time healing reads `granularity_time` to decide which time column the others are derived from, so supply it.

Details

For more details see the vignette: `vignette("csfmt_rts_data_v2", package = "cstidy")`

Value

An extended `data.table`, which has been modified by reference and returned (invisibly).

Returns a duplicated `csfmt_rts_data_v2`.

Smart assignment

`csfmt_rts_data_v2` contains the smart assignment feature for time and geography.

When the **variables in bold** are assigned using `:=`, the listed variables are automatically re-derived from it. This is deterministic derivation from a calendar and a geography lookup, not statistical imputation.

location_code:

- `granularity_geo`
- `country_iso3`

isoyear:

- `granularity_time`
- `isoweek`
- `isoyearweek`
- `isoquarter`
- `isoyearquarter`
- `season`
- `seasonweek`
- `calyear`
- `calmonth`
- `calyearmonth`
- `date`

isoyearweek:

- `granularity_time`
- `isoyear`
- `isoweek`
- `isoquarter`

- isoyearquarter
- season
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

season:

- granularity_time
- isoyear
- isoweek
- isoyearweek
- isoquarter
- isoyearquarter
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

date:

- granularity_time
- isoyear
- isoweek
- isoyearweek
- isoquarter
- isoyearquarter
- season
- seasonweek
- calyear
- calmonth
- calyearmonth

Unified columns

csfmt_rts_data_v2 contains 18 unified columns:

- granularity_time
- granularity_geo
- country_iso3
- location_code
- border
- age
- sex
- isoyear
- isoweek
- isoyearweek
- isoquarter
- isoyearquarter
- season
- seasonweek
- calyear
- calmonth
- calyearmonth
- date

Deprecated

csfmt_rts_data_v2 is deprecated as a direction of travel, not because a finished replacement exists. The format still works, nothing warns at run time, and nothing has been removed. [set_csfmt_rts_data_v3\(\)](#) is what new work should target, subject to three limits that were measured rather than estimated.

First, v3 derives fewer columns than v2, but drops none. Counted on the unified set, v2 derives 18 columns and v3 derives 11. The unified set is `names(attr(x, "format_unified"))`, the columns each format creates when the input lacks them. The seven columns in v2's unified set and not in v3's are `granularity_time`, `border`, `isoquarter`, `isoyearquarter`, `calyear`, `calmonth` and `calyearmonth`. Deriving and keeping are different things. [set_csfmt_rts_data_v3\(\)](#) removes no column. An existing v2 object converted to v3 keeps every column it had, `granularity_time` and `border` included. v3 healing still refreshes `isoquarter` and `isoyearquarter` when the columns are present. What a move to v3 costs is the guarantee. A v3 table built from an input that lacks those columns will not have them, where the same input under v2 would.

Second, v3 is weekly-only, and that is what costs you the calendar columns. `heal.csfmt_rts_data_v3()` derives from `isoyearweek` alone. The `isoyearweek` lookup holds no calendar values at all. `calyear`, `calmonth` and `calyearmonth` are NA for all 7829 rows of the `isoyearweek` lookup, under v2 as much as under v3. So no v3 table can populate them. Heal from `date` under v2 if you aggregate by calendar month or calendar year. A daily table converted to v3 keeps its `date` values and gets nothing else healed.

Third, csdb cannot CHECK a v3 table, though it can store one. csdb exports `validator_field_types_csfmt_rts_data_v1` and `validator_field_types_csfmt_rts_data_v2()` and has no v3 equivalent, so a v3 column set fails both. The validator is an ordinary argument to `csdb::DBTable_v9$new()`: pass `validator_field_types_blank()` or a function of your own, and the table writes. What you lose is the column check, not the ability to store.

So continue with v2 in any of these three cases:

- The data covers a granularity other than isoyearweek.
- The data must derive a calendar or quarterly column that the input does not carry.
- csdb must validate the shape of the data on the way in.

See Also

Two vignettes run `set_csfmt_rts_data_v2()` in a code chunk: `vignette("cstidy", package = "cstidy")` and `vignette("csfmt_rts_data_v2", package = "cstidy")`. Neither of them runs `csfmt_rts_data_v2()`.

Other `csfmt_rts_data`: [expand_time_to\(\)](#), [identify_data_structure\(\)](#), [remove_class_csfmt_rts_data\(\)](#), [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v3\(\)](#), [unique_time_series\(\)](#)

Other `csfmt` format converters: [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v3\(\)](#)

Examples

```
# Create some fake data as data.table
d <- cstidy::generate_test_data(fmt = "csfmt_rts_data_v2")
d <- d[1:5]

# convert to csfmt_rts_data_v2 by reference
cstidy::set_csfmt_rts_data_v2(d, create_unified_columns = TRUE)

#
d[1, isoyearweek := "2021-01"]
d
d[2, isoyear := 2019]
d
d[3, date := as.Date("2020-01-01")]
d
d[4, c("isoyear", "isoyearweek") := .(2021, "2021-01")]
d
d[5, c("location_code") := .("norge")]
d

# Investigating the data structure of one column inside a dataset
cstidy::generate_test_data() |>
  cstidy::set_csfmt_rts_data_v2() |>
  cstidy::identify_data_structure("deaths_n") |>
  plot()
# Investigating the data structure via summary
cstidy::generate_test_data() |>
  cstidy::set_csfmt_rts_data_v2() |>
  summary()
```

`set_csfmt_rts_data_v3` *Convert a data.table to csfmt_rts_data_v3 (clean csfmt; explicit healing)*

Description

Eleven unified columns, taken from the 18 of [set_csfmt_rts_data_v2](#): `granularity_geo`, `country_iso3`, `location_code`, `age`, `sex`, `isoyear`, `isoweek`, `isoyearweek`, `season`, `seasonweek` and `date`. It drops the self-healing `[]` override (healing is explicit) and gives `time_series_id` a content hash.

Usage

```
set_csfmt_rts_data_v3(x, create_unified_columns = TRUE, heal = TRUE)
```

```
csfmt_rts_data_v3(x, create_unified_columns = TRUE, heal = TRUE)
```

Arguments

<code>x</code>	The data.table to convert (by reference).
<code>create_unified_columns</code>	Create the unified columns?
<code>heal</code>	Derive the missing time and geography columns on creation? These are deterministically looked up from <code>isoyearweek</code> and <code>location_code</code> ; nothing is statistically imputed and no count is invented.

Value

`x`, modified by reference, invisibly.
 A new `csfmt_rts_data_v3` (not by reference).

See Also

No vignette covers `csfmt_rts_data_v3`. The data-format vignette documents the `csfmt_rts_data_v2` columns that this format takes its own from: `vignette("csfmt_rts_data_v2", package = "cstidy")`.

Other `csfmt_rts_data`: [expand_time_to\(\)](#), [identify_data_structure\(\)](#), [remove_class_csfmt_rts_data\(\)](#), [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v2\(\)](#), [unique_time_series\(\)](#)

Other `csfmt` format converters: [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v2\(\)](#)

Examples

```
# v3 is weekly-only: the other time columns come from isoyearweek alone.
d <- data.table::data.table(
  isoyearweek = c("2020-34", "2020-35"),
  location_code = "nation_nor",
  deaths_n = c(1L, 2L)
)
```

```

# set_csfmt_rts_data_v3() converts d in place and returns it invisibly.
cstidy::set_csfmt_rts_data_v3(d)
d[]
class(d)

# csfmt_rts_data_v3() copies instead, so e is left as it was.
e <- data.table::data.table(
  isoyearweek = c("2020-34", "2020-35"),
  location_code = "nation_nor",
  deaths_n = c(1L, 2L)
)
y <- cstidy::csfmt_rts_data_v3(e)
class(y)
class(e)
names(e)

```

unique_time_series	<i>Unique time series</i>
--------------------	---------------------------

Description

Attempts to identify the unique time series that exist in this dataset.

A time series is defined as a unique combination of:

- granularity_time
- granularity_geo
- country_iso3
- location_code
- border
- age
- sex
- *_id
- *_tag

Usage

```
unique_time_series(x, set_time_series_id = FALSE, ...)
```

Arguments

x	An object of type <code>csfmt_rts_data_v2</code>
set_time_series_id	If TRUE, then x will have a new column called 'time_series_id'
...	Not used.

Value

data.table, a dataset that lists all the unique time series in x.

See Also

The data-format vignette runs this function on a test dataset: `vignette("csfmt_rts_data_v2", package = "cstidy")`.

Other `csfmt_rts_data`: [expand_time_to\(\)](#), [identify_data_structure\(\)](#), [remove_class_csfmt_rts_data\(\)](#), [set_csfmt_rts_data_v1\(\)](#), [set_csfmt_rts_data_v2\(\)](#), [set_csfmt_rts_data_v3\(\)](#)

Examples

```
x <- cstidy::generate_test_data() |>
  cstidy::set_csfmt_rts_data_v2()
cstidy::unique_time_series(x)
```

Index

- * **csfmt format converters**
 - set_csfmt_rts_data_v1, 10
 - set_csfmt_rts_data_v2, 13
 - set_csfmt_rts_data_v3, 18
- * **csfmt_rts_data**
 - expand_time_to, 2
 - identify_data_structure, 6
 - remove_class_csfmt_rts_data, 9
 - set_csfmt_rts_data_v1, 10
 - set_csfmt_rts_data_v2, 13
 - set_csfmt_rts_data_v3, 18
 - unique_time_series, 19
- * **datasets**
 - nor_covid19_cases_by_time_location_csfmt_rts_v1, 7
 - nor_covid19_icu_and_hospitalization_csfmt_rts_v1, 8
- * **time healing lookups**
 - heal_time_csfmt_rts_data_v1, 4
 - heal_time_csfmt_rts_data_v2, 5

csfmt_rts_data_v1
(set_csfmt_rts_data_v1), 10

csfmt_rts_data_v2, 3, 6, 19

csfmt_rts_data_v2
(set_csfmt_rts_data_v2), 13

csfmt_rts_data_v3
(set_csfmt_rts_data_v3), 18

expand_time_to, 2

expand_time_to(), 7, 9, 12, 17, 18, 20

generate_test_data, 3

heal_time_csfmt_rts_data_v1, 4

heal_time_csfmt_rts_data_v1(), 5

heal_time_csfmt_rts_data_v2, 4, 5

heal_time_csfmt_rts_data_v2(), 4

identify_data_structure, 6

identify_data_structure(), 3, 9, 12, 17, 18, 20

nor_covid19_cases_by_time_location_csfmt_rts_v1, 7

nor_covid19_icu_and_hospitalization_csfmt_rts_v1, 8

remove_class_csfmt_rts_data, 9

remove_class_csfmt_rts_data(), 3, 7, 12, 17, 18, 20

set_csfmt_rts_data_v1, 4, 10

set_csfmt_rts_data_v1(), 3, 7, 9, 17, 18, 20

set_csfmt_rts_data_v2, 3–6, 12, 13, 18

set_csfmt_rts_data_v2(), 3, 7, 9, 12, 18, 20

set_csfmt_rts_data_v3, 5, 12, 16, 18

set_csfmt_rts_data_v3(), 3, 7, 9, 12, 17, 20

unique_time_series, 19

unique_time_series(), 3, 7, 9, 12, 17, 18