# Package: iidda.analysis (via r-universe)

October 8, 2026

**Type** Package

**Title** Tools for Analyzing IIDDA Datasets

**Version** 2.0.0

**Author** Steven Walker

**Maintainer** Steven Walker <stevencarlislewalker@gmail.com>

**Description** This package contains tools for working with data obtained from the International Infectious Disease Data Archive.

**Depends** R (>= 3.5.0), iidda

**Imports** stringr, magrittr, EMD, WaveletComp, dplyr, ggforce, ggplot2, janitor, lubridate, purrr, readr, scales, tidyr, tidyselect, patchwork, curl

**Suggests** knitr, rmarkdown, iidda.api, testthat (>= 3.0.0)

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**Roxygen** list(markdown = TRUE)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**StagedInstall** no

**Remotes** iidda.api=canmod/iidda-tools/R/iidda.api, iidda=canmod/iidda-tools/R/iidda

**Config/pak/sysreqs** libfontconfig1-dev libfreetype6-dev libicu-dev libssl-dev libx11-dev

**Repository** https://canmod.r-universe.dev

**Date/Publication** 2025-10-17 02:17:32 UTC

**RemoteUrl** https://github.com/canmod/iidda-tools

**RemoteRef** HEAD

**RemoteSha** 4e958088c4410bba4656575573d4a3f91e13abdf

**RemoteSubdir** R/iidda.analysis

# Contents

add_user_entries . . . 3
basal_group . . . 4
BasalGroupAdder . . . 5
browse_pipeline_dependencies . . . 5
ca_iso_3166_2 . . . 6
check_date . . . 6
clean_canmod_cdi . . . 7
ComputeMovingAverage . . . 7
CountAggregator . . . 8
create_bin_desc . . . 8
CustomDataPrep . . . 9
data_prep_constructors . . . 9
DataDictionaryConverter . . . 10
Deheaper . . . 10
factor_time_scale . . . 11
find_unaccounted_cases . . . 11
generate_disease_df . . . 12
generate_empty_df . . . 12
generate_user_table . . . 13
get_iidda_attr . . . 13
get_implied_zeros . . . 14
get_unit_labels . . . 14
grid_dates . . . 15
handle_missing_values_default . . . 16
HandleMissingValues . . . 19
HandleZeroValues . . . 19
iidda_attach_bar . . . 20
iidda_get_metadata . . . 20
iidda_plot_bar . . . 21
iidda_plot_box . . . 21
iidda_plot_heatmap . . . 22
iidda_plot_highlight . . . 23
iidda_plot_ma . . . 24
iidda_plot_rohani_heatmap . . . 25
iidda_plot_settings . . . 26
iidda_plot_wavelet . . . 27
iidda_prep_bar . . . 28
iidda_prep_box . . . 29
iidda_prep_rohani . . . 30
iidda_prep_seasonal_heatmap . . . 31
iidda_render_plot . . . 32
iidda_series . . . 33
iidda_theme . . . 35
join_lookup_table . . . 35
join_user_table . . . 36
log1p_modified_trans . . . 36

lookup_join . . . 37
lubridate_funcs . . . 37
make_time_trans . . . 38
mid_dates_times . . . 38
mutate_time_vars . . . 39
names_to_join_by . . . 39
normalize_disease_hierarchy . . . 40
normalize_duplicate_sources . . . 41
normalize_location . . . 41
normalize_population . . . 42
normalize_time_scales . . . 42
num_days . . . 43
period_averager . . . 44
PeriodAggregator . . . 45
PeriodDescriber . . . 46
quantile_trans . . . 46
read_iidda_dataset . . . 47
resolve_join . . . 47
SeriesHarmonizer . . . 48
Skipper . . . 48
std_series_variables . . . 48
std_time_units . . . 49
std_time_variables . . . 49
time_extent . . . 49
TimeRangeDescriber . . . 50
TimeScalePicker . . . 50
TimeVariableConverter . . . 50
TitleGuesser . . . 51
titleize . . . 51
TrimSeries . . . 52
union_series . . . 52
unique_entries . . . 53
valid_time_vars . . . 53
wavelet_something . . . 54
year_end_fix . . . 55

**Index** **57**

---

`add_user_entries` *Add entries to user table*

---

## Description

Adds entries to user-defined lookup table Entries should have names or columns from the user lookup table `standards` can be used for entries

## Usage

```
add_user_entries(entries, user_table_path)
```

## Arguments

| | |
|---|---|
| entries | dataframe or named list of entries to add |
| user_table_path | string indicating path to user lookup table |

## Value

user lookup table with added entries in the path

---

# basal_group *Basal Group*

---

## Description

Basal Group

## Usage

```
basal_group(
  value,
  lookup,
  hierarchical_variable,
  nesting_variable,
  encountered_values = character()
)
```

## Arguments

| | |
|---|---|
| value | Value (e.g., hepatitis-A) of the hierarchical_variable (e.g., disease) for which to determine basal value (e.g., hepatitis). |
| lookup | Table with two character-valued columns with names given by hierarchical_variable (e.g., disease) and nesting_variable (e.g., nesting_disease). |
| hierarchical_variable | Name of the hierarchical variable in lookup |
| nesting_variable | Name of the nesting variable in lookup |
| encountered_values | Character vector of values already found. Typically this left at the default value of an empty character vector. |

## Value

The root value that input value maps to in lookup.

## BasalGroupAdder *Basal Group Adder*

### Description

Basal Group Adder

### Usage

```
BasalGroupAdder(lookup)
```

### Arguments

| | |
|---|---|
| lookup | A lookup table with disease and nesting_disease columns that describe a global disease hierarchy that will be applied to find the basal disease of each disease in data. |
| data | A tidy data set with a disease column. |

### Value

tidy dataset with basal disease

## browse_pipeline_dependencies *Browse Pipeline Dependencies*

### Description

Open a browser at the locations of the dependencies associated with a set of datasets.

### Usage

```
browse_pipeline_dependencies(
  dataset_ids,
  dependency_types = c("IsCompiledBy", "IsDerivedFrom", "References"),
  metadata = iidda.api::ops_staging$metadata(dataset_ids = dataset_ids)
)
```

### Arguments

| | |
|---|---|
| dataset_ids | Character vector of dataset identifiers. |
| dependency_types | Vector of types of dependencies to browse. Possible values include "IsCompiledBy", "IsDerivedFrom", and "References". |
| metadata | Optional list giving dataset metadata. The default uses the IIDDA API, which requires the internet. |

## ca_iso_3166_2 *Order Canadian Provinces Geographically*

### Description

Order Canadian Provinces Geographically

### Usage

```
ca_iso_3166_2(data)
```

### Arguments

| | |
|---|---|
| data | Dataset containing an iso_3166_2 field with Canadian province and territory codes. |

## check_date *Test if x is a Date, coerce if not*

### Description

Test if x is a Date, coerce if not

### Usage

```
check_date(x)
```

### Arguments

| | |
|---|---|
| x | vector of putative dates |

### Value

vector with class Date, or error

### Examples

```
d1 <- check_date("1920-01-01")
d1
class(d1)
# returns an error if x can't be coerced to Date easily
# check_date("may 29th")
```

## clean_canmod_cdi *Clean CANMOD CDI Data*

### Description

The important cleaning steps include (1) removing `CA-` from ISO-3166-2 codes (because within Canada this is redundant) and (2) filtering out all time-scales but the 'best'. so that there is no chance of double-counting cases.

### Usage

```
clean_canmod_cdi(canmod_cdi, ...)
```

### Arguments

| | |
|---|---|
| `canmod_cdi` | Dataset from IIDDA of type `CANMOD CDI`. |
| `...` | Arguments to pass on to `normalize_time_scales`. |

## ComputeMovingAverage *Compute Moving Average of Time Series*

### Description

Compute Moving Average of Time Series

### Usage

```
ComputeMovingAverage(ma_window_length = 52)
```

### Arguments

| | |
|---|---|
| `ma_window_length` | length of moving average window, this will depend on the time scale in the data. Defaults to 52, so that weekly data is averaged over years. |

### Value

A function like `compute_moving_average_default` to remove to compute the moving average of a time series variable.

## CountAggregator *Count Aggregator*

### Description

Create a function that aggregates count variables.

### Usage

```
CountAggregator()
```

### Value

A function like `count_aggregator_default` that aggregates count variables.

## create_bin_desc *Create age bin descriptions*

### Description

Create age bin descriptions for joining age_group lookup table

### Usage

```
create_bin_desc(age_df)
```

### Arguments

| | |
|---|---|
| age_df | data frame of data with age_group column |

### Value

data frame of data with bin_desc column

## CustomDataPrep *Custom Data Prep Function*

### Description

Convert a `data_prep_function`, into a function that can be used in an `iidda` data prep pipeline.

### Usage

```
CustomDataPrep(
  data_prep_function,
  ign_variables = character(0L),
  opt_variables = character(0L),
  new_variables = character(0L)
)
```

### Arguments

`data_prep_function`
: A standard `R` function that takes a data frame, called `data` as its first argument, and returns another data frame. Optionally, subsequent arguments can be added that give the names of types of variables (e.g., `period_end_variable = "period_end_date"`). Consider using `NULL` as the default of these variable name arguments, which will lead to good guesses about variable names when `data` are obtained from `iidda.api`.

### Value

description A data prep function that maintains `iidda` attributes and guesses at names of types of variables that the `data_prep_function` assumes.

## data_prep_constructors *Data Prep Constructors*

### Description

Data Prep Constructors

DataDictionaryConverter *Data Dictionary Converter*

## Description

Data Dictionary Converter

## Usage

```
DataDictionaryConverter(data_dictionary = iidda_data_dictionary())
```

Deheaper *De-heaping time series*

## Description

Fixes heaping errors in time series. The structure of this function was taken from the function `find_heap_and_deheap` created by Kevin Zhao (https://github.com/davidearn/KevinZhao/blob/main/Report/make_SF_RDa This needs to be better documented.

## Usage

```
Deheaper(
  prefix = "deheaped_",
  first_date = "1830-01-01",
  last_date = "1841-12-31",
  week_start = 45,
  week_end = 5,
  deheaping_scale = 2.7
)
```

## Arguments

| | |
|---|---|
| `first_date` | string containing earliest date to look for heaping errors |
| `last_date` | string containing last date to look for heaping errors |
| `week_start` | numeric value of the first week number to start looking for heaping errors |
| `week_end` | numeric value of the last week number to look for heaping errors |

## Value

A function like `deheaper_default` to fix heaping errors.

## factor_time_scale *Factor Time Scale*

### Description

Factor Time Scale

### Usage

```
factor_time_scale(data)
```

### Arguments

| | |
|---|---|
| data | A tidy data set with a time_scale column. |

### Value

A data set with a factored time_scale column.

## find_unaccounted_cases *Find Unaccounted Cases*

### Description

Make new records for instances when the sum of leaf diseases is less than the reported total for their basal disease. The difference between these counts gets disease name 'basal_disease'_unaccounted'.

### Usage

```
find_unaccounted_cases(data)
```

### Arguments

| | |
|---|---|
| data | A tidy data set with a basal_disease column. |

### Value

A data set containing records that are the difference between a reported total for a basal_disease and the sum of their leaf diseases.

## generate_disease_df *Data for a Particular Disease*

### Description

Data for a Particular Disease

### Usage

```
generate_disease_df(canmod_cdi, disease_name, years = NULL, add_gaps = TRUE)
```

### Arguments

| | |
|---|---|
| canmod_cdi | Dataset from IIDDA of type CANMOD CDI. |
| disease_name | Name to match in the nesting_disease column of a CANMOD CDI dataset. |
| years | If not NULL, a vector of years to keep in the output data. |
| add_gaps | If TRUE, add records with NA in cases_this_period that correspond to time-periods without any data. |

## generate_empty_df *Create empty table with column names*

### Description

Creates an empty table in a specified directory using columns names from another data frame

### Usage

```
generate_empty_df(dir_path, lookup_table, csv_name)
```

### Arguments

| | |
|---|---|
| dir_path | string indicating path to directory |
| lookup_table | data frame with column names to include in table |
| csv_name | string indicating name of the created .csv file |

### Value

empty csv file with columns from lookup_table in the directory if successfully generated

## generate_user_table *Create user-defined lookup table*

### Description

Creates an empty user-defined lookup table in a specified directory

### Usage

```
generate_user_table(path, lookup_table_type)
```

### Arguments

| | |
|---|---|
| path | string indicating path to directory |
| lookup_table_type | string indicating type of lookup table |

### Value

csv file of empty lookup table with columns from `lookup_table_type` in the directory if successful

## get_iidda_attr *Get IIDDA Attribute*

### Description

Get IIDDA Attribute

### Usage

```
get_iidda_attr(data, which)
```

### Arguments

| | |
|---|---|
| data | Data frame that contains a list attribute called `"iidda"`. |
| which | Name of the element in the `"iidda"` list to extract. |

### Value

Value of the element given by `which` in the `"iidda"` attribute.

## get_implied_zeros *Get Implied Zeros*

### Description

Add zeros to data set that are implied by a '0' reported at a coarser timescale.

### Usage

```
get_implied_zeros(data)
```

### Arguments

| | |
|---|---|
| data | A tidy data set with the following minimal set of columns: disease, nesting_disease, year, original_dataset_id, iso_3166_2, basal_disease, time_scale, period_start_date, period_end_date, period_mid_date, days_this_period, dataset_id |

### Value

A tidy data set with inferred 0s.

## get_unit_labels *Get time unit labels*

### Description

Get label of associated time unit

### Usage

```
get_unit_labels(unit)
```

### Arguments

| | |
|---|---|
| unit | time unit, one of iidda.analysis:::time_units |

### Value

label of associated time unit

grid_dates *Create a grid of dates starting at the first day in grid unit*

## Description

Wrapper of `seq.Date()` and `lubridate::floor_date`

## Usage

```
grid_dates(
  start_date = "1920-01-01",
  end_date = "2020-01-01",
  by = "1 week",
  unit = "week",
  lookback = TRUE,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `start_date` | starting date |
| `end_date` | end date |
| `by` | increment of the sequence. Optional. See 'Details'. |
| `unit` | a string, `Period` object or a date-time object. When a singleton string, it specifies a time unit or a multiple of a unit to be rounded to. Valid base units are `second`, `minute`, `hour`, `day`, `week`, `month`, `bimonth`, `quarter`, `season`, `halfyear` and `year`. Arbitrary unique English abbreviations as in the `period()` constructor are allowed. Rounding to multiples of units (except weeks) is supported.<br>When `unit` is a `Period` object, it is first converted to a string representation which might not be in the same units as the constructor. For example `weeks(1)` is converted to "7d 0H 0M 0S". Thus, always check the string representation of the period before passing to this function.<br>When `unit` is a date-time object rounding is done to the nearest of the elements in `unit`. If range of `unit` vector does not cover the range of `x` `ceiling_date()` and `floor_date()` round to the `max(x)` and `min(x)` for elements that fall outside of `range(unit)`. |
| `lookback` | Logical, should the first value start before `start_date` |
| `week_start` | week start day (Default is 7, Sunday. Set `lubridate.week.start` to override). Full or abbreviated names of the days of the week can be in English or as provided by the current locale. |

## Value

vector of Dates at the first of each week, month, year

## Examples

```
grid_dates(start_date = "2023-04-01"
, end_date = "2023-05-16")

grid_dates(start_date = "2023-04-01"
, end_date = "2023-05-16"
, lookback = FALSE)


grid_dates(start_date = "2020-04-01"
, end_date = "2023-05-16"
, by = "2 months"
, unit = "month")
grid_dates(start_date = "2020-04-01"
, end_date = "2023-05-16"
, by = "2 months")
```

---

# handle_missing_values_default

*Data Prep Default Functions*

---

## Description

Data Prep Default Functions

## Usage

```
handle_missing_values_default(data, series_variable = NULL)

handle_zero_values_default(data, series_variable = NULL)

trim_series_default(data, series_variable = NULL, time_variable = NULL)

series_harmonizer_default(data, series_variable = NULL, time_variable = NULL)

deheaper_default(data, series_variable = NULL, time_variable = NULL)

compute_moving_average_default(
  data,
  series_variable = NULL,
  time_variable = NULL
)

period_aggregator_default(
  data,
  time_variable = NULL,
  period_width_variable = NULL,
  count_variable = NULL,
```

```
  norm_variable = NULL
)

count_aggregator_default(
  data,
  total_count_variable = NULL,
  count_variable = NULL,
  grouping_variable = NULL
)

period_describer_default(
  data,
  period_start_variable = NULL,
  period_end_variable = NULL,
  period_mid_time_variable = NULL,
  period_mid_date_variable = NULL,
  period_days_variable = NULL
)
```

## Arguments

| | |
|---|---|
| data | Data frame that likely comes from IIDDA. |
| series_variable | Name of variable that can be used on the y-axis of a time-series. |
| time_variable | Name of variable that characterizes the temporal location of the time period. |
| period_width_variable | Name of variable that characterizes the width of the time period. |
| count_variable | Name of variable containing a count (e.g., cases, births, deaths, population). |
| norm_variable | Name of variable that can be used to normalize another variable (e.g., population normalized cases). |
| total_count_variable | Name of variable containing a marginal total of a set of counts. |
| grouping_variable | Name of variable containing |
| period_start_variable | Name of variable containing |
| period_end_variable | Name of variable containing |
| period_mid_time_variable | Name of variable containing |
| period_mid_date_variable | Name of variable containing |
| period_days_variable | Name of variable containing |
| cases_variable | Name of variable containing unstandardized reported incidence. |

`population_variable`
: Name of variable containing population numbers.

`birth_variable`
: Name of variable containing numbers of births.

`death_variable`
: Name of variable containing numbers of deaths.

`median_cases_variable`
: Name of variable containing the median of a set of count variables.

`period_mid_variable`
: Name of variable containing

`date_variable`
: Name of variable containing

`integer_time_variable`
: Name of variable containing

`numeric_time_variable`
: Name of variable containing

`time_scale_variable`
: Name of variable containing

`time_group_variable`
: Name of variable containing

`time_grouping_variable`
: Name of variable containing

`disease_variable`
: Name of variable containing

`hierarchical_variable`
: Name of variable containing

`nesting_variable`
: Name of variable containing

`basal_variable`
: Name of variable containing

`title_variable`
: Name of variable containing

`among_panel_variable`
: Name of variable containing

`within_panel_variable`
: Name of variable containing

`colour_variable`
: Name of variable containing

`category_variable`
: Name of variable containing

`categorical_variable`
: Name of variable containing

## HandleMissingValues *Handle Missing Values in Series Variable*

### Description

Construct a function that takes a data frame and returns another data frame with NA values either removed or replaced.

### Usage

```
HandleMissingValues(na_remove = FALSE, na_replace = NULL)
```

### Arguments

| | |
|---|---|
| na_remove | boolean value, if TRUE remove NAs in series variable |
| na_replace | numeric value to replace NAs in series variable, if NULL no replacement is performed |

### Value

A function like handle_missing_values_default that removes or replaces missing values.

## HandleZeroValues *Handle Zero Values in Series Variable*

### Description

Construct a function that takes a data frame and returns another data frame with 0 values either removed or replaced.

### Usage

```
HandleZeroValues(zero_remove = FALSE, zero_replace = NULL)
```

### Arguments

| | |
|---|---|
| zero_remove | boolean value, if TRUE remove zeroes in series variable |
| zero_replace | numeric value to replace zeroes in series variable, if NULL no replacement is performed |

### Value

A function like handle_zero_values_default to remove or replace zero values.

## iidda_attach_bar *Attach Bar Plot to Dataset*

### Description

Attach Bar Plot to Dataset

### Usage

```
iidda_attach_bar(
  data,
  initial_ggplot_object = ggplot(),
  series_variable = NULL,
  time_unit = NULL,
  aggregated = NULL
)
```

### Arguments

| | |
|---|---|
| `data` | Data frame, probably containing an IIDDA dataset. |
| `initial_ggplot_object` | Plot object that will be used to add a bar geom. |

## iidda_get_metadata *Get IIDDA metadata*

### Description

Get starting time period, ending time period and mortality cause name from the data set for use in axis and main plot titles.

### Usage

```
iidda_get_metadata(data, time_variable = NULL, descriptor_variable = NULL)
```

### Arguments

| | |
|---|---|
| `data` | data frame containing time series data |
| `time_variable` | column name of time variable in `data`, default is "period_end_date" |
| `descriptor_variable` | column name of the descriptor variable in `data`, default is "cause" for mortality data sets. |

### Value

a list in order containing minimum time period, maximum time period and cause name.

## iidda_plot_bar *Plot Bar Graph*

### Description

Add a bar plot to an exiting ggplot plot object. Graphical choices were made to closely reflect plots generated with `LBoM::monthly_bar_graph` and `LBoM::weekly_bar_graph`.

### Usage

```
iidda_plot_bar(
  plot_object,
  data = NULL,
  series_variable = NULL,
  time_unit = "month_factor_abbr"
)
```

### Arguments

| | |
|---|---|
| `plot_object` | a `ggplot2` plot object |
| `data` | data frame containing data prepped for bar plotting, typically output from `iidda_prep_bar()`. If `NULL` data is inherited from `plot_object` |
| `series_variable` | column name of series variable in `data`, default is "deaths" |
| `time_unit` | time unit to display bar graphs on the x-axis. Defaults to "week" or one of iidda.analysis:::time_units that starts with "month". Should generalize at some point to be able to take any time_unit argument. |

### Value

a ggplot2 plot object containing a bar graphs of time series data

## iidda_plot_box *Plot Box Plot*

### Description

Add a box plot to an exiting ggplot plot object. Graphical choices were made to closely reflect plots generated with `LBoM::monthly_box_plot`.

### Usage

```
iidda_plot_box(
  plot_object,
  data = NULL,
  series_variable = "deaths",
  time_unit = "week",
  ...
)
```

### Arguments

| | |
|---|---|
| plot_object | a ggplot2 plot object |
| data | data frame containing data prepped for box plotting, typically output from iidda_prep_box(). If NULL data is inherited from plot_object |
| series_variable | column name of series variable in data, default is "deaths" |
| time_unit | time unit to display box plots on the x-axis. Defaults to "week", should be able to handle any time_unit from iidda.analysis:::time_units. |
| ... | other arguments to be passed to scale_x_discrete |

### Value

a ggplot2 plot object containing a box plots of time series data

---

## iidda_plot_heatmap *Plot Heatmap*

---

### Description

Add a yearly vs. weekly heatmap to an exiting ggplot plot object. Graphical choices were made to closely reflect plots generated withLBoM::seasonal_heat_map.

### Usage

```
iidda_plot_heatmap(
  plot_object,
  data = NULL,
  series_variable = "deaths",
  start_year_variable = "Year",
  end_year_variable = "End Year",
  start_day_variable = "Day of Year",
  end_day_variable = "End Day of Year",
  colour_trans = "log2",
  NA_colour = "black",
  palette_colour = "RdGy",
  ...
)
```

### Arguments

| | |
|---|---|
| plot_object | a ggplot2 plot object |
| data | data frame containing data prepped for yearly vs. weekly heatmaps, typically output from iidda_prep_heatmap(). If NULL data is inherited from plot_object. |
| series_variable | column name of series variable in data, default is "deaths" |
| start_year_variable | column name of time variable containing the year of the starting period, defaults to "Year" |
| end_year_variable | column name of time variable containing the year of the ending period, defaults to "End Year" |
| start_day_variable | column name of time variable containing the day of the starting period, defaults to "Day of Year" |
| end_day_variable | column name of time variable containing the day of the ending period, defaults to "End Day of Year" |
| colour_trans | string indicating colour transformation, one of "log2", "sqrt" or "linear" |
| NA_colour | colour for NA values, defaults to "black" |
| palette_colour | colour of heatmap palette, defaults to "RdGy". Should specify what type of palette colours are accepted by this argument. |
| ... | Not currently used. |

### Value

a ggplot2 plot object containing a yearly vs. weekly heatmap of time series data

---

## iidda_plot_highlight *Add Plot Highlight*

---

### Description

Add a rectangular highlighted region to an existing ggplot2 plot object

### Usage

```
iidda_plot_highlight(
  plot_object,
  data = NULL,
  series_variable = "deaths",
  time_variable = "period_end_date",
  filter_variable = "period_end_date",
  filter_start = "1700-01-01",
  filter_end = "1800-01-01",
  ...
)
```

### Arguments

| | |
|---|---|
| plot_object | a ggplot2 plot object |
| data | data frame containing time series data. If NULL data is inherited from plot_object. This has only been tested with data output from iidda_plot_ma. |
| series_variable | column name of series variable in data, default is "deaths" |
| time_variable | column name of time variable in data, default is "period_end_date" |
| filter_variable | column name of variable to filter on in data, default is "period_end_date" |
| filter_start | value of filter_variable for starting range, default is "1700-01-01" |
| filter_end | value of filter_variable for ending range, default is "1800-01-01" |
| ... | other arguments to be passed to ggforce::geom_mark_rect, for example annotating with text |

### Value

a ggplot2 plot object a rectangular plot highlight

---

## iidda_plot_ma *Plot Moving Average Time Series*

---

### Description

Add a moving average time series line to an exiting ggplot plot object. Graphical choices were made to closely reflect plots generated with LBoM::plot.LBoM.

### Usage

```
iidda_plot_ma(
  plot_object,
  data = NULL,
  series_variable = NULL,
  time_variable = NULL
)
```

### Arguments

| | |
|---|---|
| plot_object | a ggplot2 plot object |
| data | data frame containing moving average time series data, typically output from iidda_prep_ma(). If NULL data is inherited from plot_object |
| series_variable | column name of series variable in data, default is "deaths" |
| time_variable | column name of time variable in data, default is "period_end_date" |

### Value

a ggplot2 plot object containing a moving average time series

iidda_plot_rohani_heatmap *Plot Rohani Heatmap*

## Description

Add a rohani heatmap to an exiting ggplot plot object. Possibly to be extended to include time series in a separate facet.

## Usage

```
iidda_plot_rohani_heatmap(
  plot_object,
  data = NULL,
  series_variable = "deaths",
  start_year_variable = "Year",
  end_year_variable = "End Year",
  start_day_variable = "Day of Year",
  end_day_variable = "End Day of Year",
  grouping_variable = "cause",
  colour_trans = log1p_modified_trans(),
  n_colours = (scales::brewer_pal(palette = "YlOrRd"))(9),
  NA_colour = "black",
  palette_colour = "YlOrRd"
)
```

## Arguments

| | |
|---|---|
| plot_object | a ggplot2 plot object |
| data | data frame containing data prepped for yearly vs. weekly heatmaps, typically output from iidda_prep_heatmap(). If NULL data is inherited from plot_object. |
| series_variable | column name of series variable in data, default is "deaths" |
| start_year_variable | column name of time variable containing the year of the starting period, defaults to "Year" |
| end_year_variable | column name of time variable containing the year of the ending period, defaults to "End Year" |
| start_day_variable | column name of time variable containing the day of the starting period, defaults to "Day of Year" |
| end_day_variable | column name of time variable containing the day of the ending period, defaults to "End Day of Year" |

grouping_variable
: column name of grouping variable to appear on the y-axis of the heatmap.

colour_trans
: function to scale colours, to be supplied to trans argument of scale_fill_gradientn()

n_colours
: vector of colours to be supplied to scale_fill_gradientn()

NA_colour
: colour for NA values, defaults to "black"

palette_colour
: colour of heatmap palette, defaults to "RdGy". Should specify what type of palette colours are accepted by this argument.

## Value

a ggplot2 plot object containing a yearly vs. weekly heatmap of time series data

---

# iidda_plot_settings *LBoM plot settings*

---

## Description

Add basic features to a ggplot2 plot object including title, subtitle and classic `ggplot2::theme_bw` theme.

## Usage

```
iidda_plot_settings(
  plot_object,
  data = data.frame(),
  min_time = "min_time",
  max_time = "max_time",
  descriptor_name = "descriptor_name",
  theme = iidda_theme
)
```

## Arguments

plot_object
: a ggplot2 plot object

data
: list containing metadata. If NULL data is inherited from `plot_object`.

min_time
: name of field in data containing the minimum time period range, defaults to "min_time".

max_time
: name of field in data containing the minimum time period range, defaults to "max_time".

descriptor_name
: either the name of a field in data containing the descriptor or a string to be used as the plot title. If there are too more than 3 elements in the descriptor field, then `descriptor_variable` is used as the plot title.

theme
: ggplot theme

## Value

a ggplot2 plot object with title, subtitle and adjusted theme.

---

`iidda_plot_wavelet` *Plot Wavelet*

---

## Description

Plot wavelet to look similar to base R plot of `WaveletComp::wt.image` using ggplot2 functionality.
Some visual choices were made to reflect work done by Steven Lee (https://github.com/davidearn/StevenLee) and Kevin Zhao (https://github.com/davidearn/KevinZhao).

## Usage

```
iidda_plot_wavelet(
  plot_object,
  data = NULL,
  wavelet_data,
  contour_data,
  y_variable_name = "Period (years)",
  fill_variable_name = "Power",
  max_period = 10,
  colour_levels = 250,
  start_hue = 0,
  end_hue = 0.7,
  sig_lvl = 0.05
)
```

## Arguments

| | |
|---|---|
| `plot_object` | a `ggplot2` plot object |
| `data` | data frame containing wavelet data prepped for use in `ggplot2::geom_tile`. The output from `iidda_prep_wavelet` produces a data set prepped for this argument, named `tile_data_to_plot` in the returned list. If `NULL` data is inherited from `plot_object`. |
| `wavelet_data` | list containing raw wavelet transformed data, typically output from `WaveletComp::analyze.wavelet`. The output from `iidda_prep_wavelet` produces a data set prepped for this argument, named `transformed_data` in the returned list. |
| `contour_data` | data set containing contour data prepped for use in `ggplot2::geom_contour`. The output from `iidda_prep_wavelet` produces a data set prepped for this argument, named `cont_data_to_plot` in the returned list. |
| `y_variable_name` | name of y variable in plot, defaults to "Period (years)". |
| `fill_variable_name` | name of colour fill variable in plot, defaults to "Power". |

| | |
|---|---|
| max_period | maximum period to appear on the plot, defaults to 10 years. |
| colour_levels | number of colours to pass to scale_fill_gradientn. |
| start_hue | starting hue colour to pass to scale_fill_gradientn, default taken from WaveletComp::wt.image. |
| end_hue | ending hue colour to pass to scale_fill_gradientn, default taken from WaveletComp::wt.image. |
| sig_lvl | significance level for white contours |

## Value

a ggplot2 object of a wavelet

---

iidda_prep_bar *Prep Data for Bar Graph*

---

## Description

Prep data for plotting bar graphs. Prep steps were taken from LBoM::monthly_bar_graph and LBoM::weekly_bar_graph and they include handling missing values and aggregating series data by time unit grouping variable.

## Usage

```
iidda_prep_bar(
  data,
  series_variable = NULL,
  time_variable = NULL,
  time_unit = NULL,
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
  handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL)
)
```

## Arguments

| | |
|---|---|
| data | data frame containing time series data |
| series_variable | column name of series variable in data, default is "deaths" |
| time_variable | column name of time variable in data, default is "period_end_date" |
| time_unit | time unit to sum series data over, must be one of iidda.analysis:::time_units, defaults to "week". |
| handle_missing_values | function to handle missing values, defaults to HandleMissingValues |
| handle_zero_values | function to handle zero values, defaults to HandleZeroValues |

## Value

data with records prepped for plotting bar graphs with series_variable and time_unit field. The name of the resulting time_unit field will be named from lubridate_funcs.

---

iidda_prep_box *Prep Data for Box plot*

---

## Description

Prep data for plotting box plots. Prep steps were taken from LBoM::monthly_box_plot and they include handling missing values and creating additional time unit fields.

## Usage

```
iidda_prep_box(
  data,
  series_variable = NULL,
  time_variable = NULL,
  time_unit = "month_factor_abbr",
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
  handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL)
)
```

## Arguments

| | |
|---|---|
| data | data frame containing time series data |
| series_variable | column name of series variable in data, default is "deaths" |
| time_variable | column name of time variable in data, default is "period_end_date" |
| time_unit | time unit to create field from time_variable. Must be one of iidda.analysis:::time_units, defaults to "week". |
| handle_missing_values | function to handle missing values, defaults to HandleMissingValues |
| handle_zero_values | function to handle zero values, defaults to HandleZeroValues |

## Value

all fields indata with records prepped for plotting box plots. The name of the new time_unit field will be named from lubridate_funcs.

iidda_prep_rohani *Prep Data for Rohani Plot*

## Description

Prep data for rohani plots. Prep steps include creating additional time unit fields, summarizing the series variable by time unit and grouping variable (the x and y axis variables) ,and optionally normalizing series data to be in the range (0,1). By default, the grouping variable is ranked in order of the summarized series variable. Needs to be generalized more, might need to handle the case where the desired y-axis is a second time unit, as in the seasonal heatmap plot and therefore making use of the year_end_fix function.

## Usage

```
iidda_prep_rohani(
  data,
  series_variable = NULL,
  time_variable = "period_end_date",
  start_time_variable = "period_start_date",
  time_unit = "year",
  grouping_variable = "cause",
  ranking_variable = NULL,
  normalize = FALSE,
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
 handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL),
  create_nonexistent = FALSE
)
```

## Arguments

| | |
|---|---|
| data | data frame containing time series data |
| series_variable | column name of series variable in data, default is "deaths" |
| time_variable | column name of time variable in data, default is "period_end_date" |
| start_time_variable | column name of time variable in data, default is "period_end_date" |
| time_unit | a vector of new time unit fields to create from start_time_variable and end_time_variable. Defaults to "c("year")". The currently functionality expects that "year" is included, should be made more general to incorporate any of iidda.analysis:::time_units. |
| grouping_variable | column name of grouping variable to appear on the y-axis of the heatmap. |
| ranking_variable | column name of variable used to rank the grouping variable. |
| normalize | boolean flag to normalize series_variable data to be between 0 and 1. |

handle_missing_values
: function to handle missing values, defaults to HandleMissingValues

handle_zero_values
: function to handle zero values, defaults to HandleZeroValues

create_nonexistent
: boolean flag to create NA records for non-existent time_unit and grouping_variable. This creates all combinations of time_unit and grouping_variable to ensure there are no missing records.

## Value

all fields indata with records prepped for plotting rohani heatmaps. The name of the new time_unit fields will be named from lubridate_funcs.

---

iidda_prep_seasonal_heatmap
*Prep Data for seasonal heatmap*

---

## Description

Prep data for seasonal heatmap plots. Prep steps were taken from LBoM::seasonal_heat_map and they include creating additional time unit fields, splitting weeks that cover the year end, and optionally normalizing series data to be in the range (0,1).

## Usage

```
iidda_prep_seasonal_heatmap(
  data,
  series_variable = NULL,
  start_time_variable = "period_start_date",
  end_time_variable = "period_end_date",
  time_unit = c("yday", "year"),
  prepend_string = "End ",
  normalize = FALSE,
  ...
)
```

## Arguments

data
: data frame containing time series data

series_variable
: column name of series variable in data, default is "deaths"

start_time_variable
: column name of time variable in data, default is "period_start_date"

end_time_variable
: column name of time variable in data, default is "period_end_date"

| | |
|---|---|
| time_unit | a vector of new time unit fields to create from start_time_variable and end_time_variable. Defaults to "c("yday","year")". The currently functionality expects that both "yday" and "year" are included, should be made more general to incorporate any of iidda.analysis:::time_units. |
| prepend_string | string to prepend to newly created time_unit fields to distinguish between time_unit fields corresponding to starting versus ending time periods. Defaults to "End ". For example, a time_unit of "year" will create a field name "Year" from start_time_variable and a field called "End Year" created from end_time_variable. |
| normalize | boolean flag to normalize series_variable data to be between 0 and 1. |
| ... | optional arguments to year_end_fix() |

## Value

all fields indata with records prepped for plotting seasonal heatmaps. The name of the new time_unit fields will be named from lubridate_funcs.

---

# iidda_render_plot *Render a Plot*

---

## Description

Extracts a plot object attached to a data set, applies a title, subtitle, and theme, and returns the plot.

## Usage

```
iidda_render_plot(
  data,
  title = TitleGuesser(),
  subtitle = TimeRangeDescriber(),
  theme = iidda_theme
)
```

## Arguments

| | |
|---|---|
| data | A data frame with an attached plot object. |
| title | A plot title, or a function that takes data and returns a title string. |
| subtitle | A plot subtitle, or a function that takes data and returns a subtitle string. |
| theme | A function that returns a ggplot2::theme object. |

iidda_series *Plot a time series*

## Description

Prepare a data set for plotting (trimming, handling missing/zero values, converting the time variable), build a line chart, and apply title, subtitle, and theme.

## Usage

```
iidda_series(
  data,
  series_variable = NULL,
  time_variable = NULL,
  trim_series = TrimSeries(zero_lead = FALSE, zero_trail = FALSE),
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
 handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL),
  time_variable_converter = TimeVariableConverter(),
  title = TitleGuesser(),
  subtitle = TimeRangeDescriber(),
  theme = iidda_theme
)

iidda_attach_series(
  data,
  initial_ggplot_object = ggplot(),
  series_variable = NULL,
  time_variable = NULL
)

iidda_prep_ma(
  data,
  series_variable = NULL,
  time_variable = NULL,
  trim_series = TrimSeries(zero_lead = FALSE, zero_trail = FALSE),
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
 handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL),
  compute_moving_average = ComputeMovingAverage(ma_window_length = 52),
  time_variable_converter = TimeVariableConverter()
)

iidda_prep_series(
  data,
  series_variable = NULL,
  time_variable = NULL,
  trim_series = TrimSeries(zero_lead = FALSE, zero_trail = FALSE),
 handle_missing_values = HandleMissingValues(na_remove = FALSE, na_replace = NULL),
```

```
 handle_zero_values = HandleZeroValues(zero_remove = FALSE, zero_replace = NULL),
  time_variable_converter = TimeVariableConverter()
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame containing the time-series data (typically output from `iidda_prep_series()`). |
| `series_variable` | Name of the series column in `data` (e.g., `"deaths"`). |
| `time_variable` | Name of the time column in `data` (e.g., `"period_end_date"`). |
| `trim_series` | A `TrimSeries(...)` specification controlling removal of leading/trailing zeros. Default: `TrimSeries(zero_lead = FALSE, zero_trail = FALSE)`. |
| `handle_missing_values` | A `HandleMissingValues(...)` specification controlling NA handling. Default: `HandleMissingValues(na_remove = FALSE, na_replace = NULL)`. |
| `handle_zero_values` | A `HandleZeroValues(...)` specification controlling zero handling. Default: `HandleZeroValues(zero_remove = FALSE, zero_replace = NULL)`. |
| `time_variable_converter` | A `TimeVariableConverter(...)` specifying how to parse/convert the time variable. Default: `TimeVariableConverter()`. |
| `title` | A plot title, or a function that takes `data` and returns a title string. |
| `subtitle` | A plot subtitle, or a function that takes `data` and returns a subtitle string. |
| `theme` | A function that returns a `ggplot2::theme` object. |
| `compute_moving_average` | function to compute the moving average of `series_variable` |

## Value

A `ggplot2` plot object containing the time-series line chart with title, subtitle, and theme applied.

## Functions

- `iidda_attach_series()`: Attach a time-series plot to a data frame containing the plotted data.
- `iidda_prep_ma()`: Prepare a dataset so that it can be used to produce a time-series plot of a moving average.
- `iidda_prep_series()`: Prepare a dataset so that it can be used to produce a time-series plot.

## iidda_theme *Themes for ggplot2*

### Description

Themes for ggplot2

### Usage

```
iidda_theme()

iidda_theme_time()

iidda_theme_heat()

iidda_theme_above()
```

### Functions

- `iidda_theme_time()`: Theme for plots where the x-axis represents time. No x-axis titles will be plotted with this theme, because the meaning of a time axis is obvious.
- `iidda_theme_heat()`: Theme for heatmaps where the x-axis represents time. No x-axis titles will be plotted with this theme, because the meaning of a time axis is obvious. Grid lines are not plotted with this theme because interpretation can be compromised when grid lines are visible through the colours of the heatmap.
- `iidda_theme_above()`: Theme for plots where the x-axis represents time, but for which time information is not displayed because there are vertically aligned plots below with the same time axis.

## join_lookup_table *Join lookup table*

### Description

Joins lookup table in API to data

### Usage

```
join_lookup_table(raw_data, lookup_type, api_hook)
```

### Arguments

| | |
|---|---|
| `raw_data` | data frame of table to be harmonized |
| `lookup_type` | string indicating type of lookup table from API to join |
| `api_hook` | API operations list |

### Value

data frame of harmonized data with keys from API

---

## join_user_table *Join user-defined lookup table*

---

### Description

Joins user-defined lookup table to data

### Usage

```
join_user_table(raw_data, user_table_path, lookup_type, join_by)
```

### Arguments

| | |
|---|---|
| raw_data | data frame of table to be harmonized |
| user_table_path | string indicating path to user-defined lookup table |
| lookup_type | string indicating type of lookup table (disease, location, sex). Used to determine columns to join by if join_by not specified |
| join_by | vector of strings indicating columns to join by (optional if lookup_type is disease, location, or sex) |

### Value

data frame of harmonized data with user-defined keys

---

## log1p_modified_trans *Log1p Scale Transformation*

---

### Description

Slight modification of log1p_trans() to include better breaks that are log1p-based (log-based and shifted 1 so that breaks can be computed in the presence of zeroes.)

### Usage

```
log1p_modified_trans(n = 10)
```

### Arguments

| | |
|---|---|
| n | number of desired breaks |

### Value

a scales::trans_new function

## lookup_join *Left join for lookup tables*

### Description

Left joins lookup table to data frame of data.

### Usage

```
lookup_join(raw_data, lookup_table, join_by, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| raw_data | Data frame of data to be harmonized. |
| lookup_table | Data frame of lookup table. |
| join_by | Vector of strings indicating columns to left_join by (can use names_to_join_by or specify manually). |
| verbose | Print information about the lookup. |

### Value

Data frame of newly harmonized and resolved data. Note that all entries in the returned data frame are strings.

## lubridate_funcs *Lubridate functions*

### Description

lubridate functions with desired interpretable labels

### Usage

```
lubridate_funcs
```

### Format

An object of class character of length 10.

## make_time_trans *Get time transformation*

### Description

Get associated lubridate function to compute time unit.

### Usage

```
make_time_trans(unit = unname(time_units))
```

### Arguments

| | |
|---|---|
| unit | time unit, one or more of iidda.analysis:::time_units |

### Value

function to compute time unit

## mid_dates_times *Period Mid-Dates and Mid-Times*

### Description

Compute a vector giving the mid-points of a vector of temporal periods, defined by start dates and one of either a vector of end dates or a vector of period lengths in days (see num_days). You can either return a date, with mid_dates, or a date-time, with mid_times. In addition to the type of return value (date vs time), the former rounds down to the nearest date whereas the latter is accurate to the nearest hour and so can account for uneven

### Usage

```
mid_dates(start_date, end_date, period_length)

mid_times(start_date, end_date, period_length)
```

### Arguments

| | |
|---|---|
| start_date | Vector of period starting dates |
| end_date | Vector of period ending dates. If missing then period_length is used to define the ends of the periods. |
| period_length | Vector of integers giving the period length in days. If missing then it is calculated using num_days. |

## mutate_time_vars *Mutate time variables*

### Description

Create new time unit fields

### Usage

```
mutate_time_vars(
  data,
  unit = unname(time_units),
  input_nm = "period_end_date",
  output_nm = get_unit_labels(unit)
)
```

### Arguments

| | |
|---|---|
| data | data set containing an input time field |
| unit | time unit, one of iidda.analysis:::time_units |
| input_nm | field name in data containing input time field |
| output_nm | field name of newly created time unit field, by default uses get_unit_labels(). |

### Value

all fields in data with additional time unit field

## names_to_join_by *Column names to join by*

### Description

Defines column names to join by for a type of lookup table

### Usage

```
names_to_join_by(lookup_type)
```

### Arguments

| | |
|---|---|
| lookup_type | string indicating type of lookup table (disease, location, sex, age group) |

### Value

vector of column names to join by for the type of lookup table

normalize_disease_hierarchy
*Normalize Disease Hierarchy*

## Description

Take a tidy data set with a potentially complex disease hierarchy and flatten this hierarchy so that, at any particular time and location (or some other context), all diseases in the `disease` column have the same `nesting_disease`.

## Usage

```
normalize_disease_hierarchy(
  data,
  disease_lookup,
  grouping_columns = c("period_start_date", "period_end_date", "location"),
  basal_diseases_to_prune = character(),
  find_unaccounted_cases = TRUE,
  specials_pattern = "_unaccounted$"
)
```

## Arguments

| | |
|---|---|
| `data` | A tidy data set with the following minimal set of columns: `disease`, `nesting_disease`, `basal_disease`, `period_start_date`, `period_end_date`, and `location`. Note that the latter three can be modified with `grouping_columns`. |
| `disease_lookup` | A lookup table with `disease` and `nesting_disease` columns that describe a global disease hierarchy that will be applied locally to flatten disease hierarchy at each point in time and space in the tidy data set in the `data` argument. |
| `grouping_columns` | Character vector of column names to use when grouping to determine the context. |
| `basal_diseases_to_prune` | Character vector of `diseases` to remove from `data`. |
| `find_unaccounted_cases` | Make new records for instances when the sum of leaf diseases is less than the reported total for their basal disease. |
| `specials_pattern` | Optional regular expression to use to match `disease` names in `data` that should be added to the lookup table. This is useful for disease names that are not historical and produced for harmonization purposes. The most common example is `"_unaccounted$"`, which is the default. Setting this argument to `NULL` avoids adding any special disease names to the lookup table. |

## normalize_duplicate_sources *Normalize Duplicate Sources*

### Description

Filter out overlapping sources for the same `disease/nesting_disease/basal_disease`, `period_start_date`, `period_end_date` , and `iso_3166_2`, with the choice to keep either national level data (i.e. from Statistics Canada / Dominion Bureau of Statistics / Health Canada) or provincial level data (from a provincial ministry of Health).

### Usage

```
normalize_duplicate_sources(data, preferred_jurisdiction = "national")
```

### Arguments

| | |
|---|---|
| `data` | A tidy data set with columns `dataset_id` , `period_start_date`, `period_end_date` , `disease` , `nesting_disease` , `basal_disease`, and `time_scale`. |
| `preferred_jurisdiction` | 'national' or 'provincial', indicating which jurisdiction level will be kept if these sources overlap. |

### Value

A data set with no overlapping sources.

## normalize_location *Normalize Location*

### Description

Set geographic order of provinces and territories and remove country-level data.

### Usage

```
normalize_location(data)
```

### Arguments

| | |
|---|---|
| `data` | Tidy dataset with an iso_3166_2 column. |

### Value

Tidy dataset without country-level data and with provinces and territories geographically ordered.

## normalize_population *Normalize Population*

### Description

Normalize Population

### Usage

```
normalize_population(data, harmonized_population)
```

### Arguments

| | |
|---|---|
| data | Tidy dataset with columns period_start_date, period_end_date iso_3166_2. |
| harmonized_population | Harmonized population data with columns date, iso_3166_2, and population (other columns will be dropped). |

### Value

Tidy dataset joined with harmonized population.

## normalize_time_scales *Normalize Time Scales*

### Description

Choose a single best `time_scale` for each year in a dataset, grouped by nesting disease. This best `time_scale` is defined as the longest of the shortest time scales in each location and sub-disease.

### Usage

```
normalize_time_scales(
  data,
  initial_group = c("year", "iso_3166", "iso_3166_2", "disease", "nesting_disease",
    "basal_disease"),
  final_group = c("basal_disease"),
  get_implied_zeros = TRUE,
  aggregate_if_unavailable = TRUE
)
```

### Arguments

| | |
|---|---|
| data | A tidy data set with columns time_scale, period_start_date and period_end_date. |
| initial_group | Character vector naming columns for defining the initial grouping used to compute the shortest time scales. |
| final_group | Character vector naming columns for defining the final grouping used to compute the longest of the shortest time scales. |
| get_implied_zeros | Add zeros that are implied by a '0' reported at a coarser timescale. |
| aggregate_if_unavailable | If a location is not reporting for the determined 'best timescale', but is reporting at a finer timescale, aggregate this finer timescale to the 'best timescale'. |

### Value

A data set only containing records with the optimal time scale.

---

## num_days *Numbers of Days*

---

### Description

Compute a vector giving the number of days in a set of periods, given equal length vectors of the start date and end date of these periods. This

### Usage

```
num_days(start_date, end_date)

num_days_util(start_date, end_date)
```

### Arguments

| | |
|---|---|
| start_date | Vector of period starting dates |
| end_date | Vector of period ending dates |

### Functions

- num_days_util(): Low-level interface for num_days.

`period_averager` *Obtain period midpoints and average daily rates for count data*

## Description

Obtain period midpoints and average daily rates for count data

## Usage

```
period_averager(
  data,
  count_col = "cases_this_period",
  start_col = "period_start_date",
  end_col = "period_end_date",
  norm_col = NULL,
  norm_const = 1e+05,
  keep_raw = TRUE,
  keep_cols = names(data)
)
```

## Arguments

| | |
|---|---|
| `data` | Data frame with rows at minimum containing period start and end dates and a count variable. |
| `count_col` | Character, name of count data column. |
| `start_col` | Character, name of start date column. |
| `end_col` | Character, name of end date column. |
| `norm_col` | Character, name of column giving data for normalization. A good option is often `population_reporting`, which is a column in many datasets containing the total size of the reference population for the count data. To avoid normalization set `norm_col` to `NULL`, which is the default. |
| `norm_const` | Numeric value for multiplying the `daily_rate` column if a `norm_col` is supplied. By default this is `1e5`, which corresponds to `daily_rate` having units of `count per day per 100,000 individuals` if the `norm_col` represents the reference population size. |
| `keep_raw` | Logical value indicating whether to force all `*_col` columns in the output, even if they are not specified in `keep_cols`, and to place them at the beginning of the columns list. The default is `TRUE`. |
| `keep_cols` | Character vector containing the names of columns in the input `data` to retain in the output. All columns are retained by default. |

## Value

Data frame containing the following fields.

- Columns from the original dataset specified using `keep_raw` and `keep_cols`.
- `year` : Year of the `period_start_date`.
- `num_days` : Length of the period in days from the beginning of the `period_start_date` to the end of the `period_end_date`.
- `period_mid_time` : Timestamp of the middle of the period.
- `period_mid_date` : Date containing the `period_mid_time`.
- `daily_rate` : Daily count rate, which by default is given by `daily_rate = count_col / num_days`. If the name of `norm_col` is specified then `daily_rate = norm_const * count_col / num_days / norm_col`. When interpreting these formulas, please keep in mind that `norm_const` is a numeric constant, `num_days` is a derived numeric column, and `count_col` and `norm_col` are columns supplied within the input `data` object.

## Examples

```
set.seed(666)
data <- data.frame(disease = "senioritis"
 , period_start_date = seq(as.Date("2023-04-03"), as.Date("2023-06-05"), by = 7)
 , period_end_date = seq(as.Date("2023-04-09"), as.Date("2023-06-11"), by = 7)
 , cases_this_period = sample(0:100, 10, replace = TRUE)
 , location = "college"
)

period_averager(data, keep_raw = TRUE, keep_cols = c("disease", "location"))
```

---

`PeriodAggregator` *Period Aggregator*

---

## Description

Create function that aggregates information over time periods, normalizes a count variable, and creates new fields to summarize this information.

## Usage

```
PeriodAggregator(rate_variable, norm_exponent = 5)
```

## Arguments

| | |
|---|---|
| `rate_variable` | Name of variable to be used to store the normalized count variable. |
| `norm_exponent` | Exponent to use in normalization. The default is `5`, which means `per 100,000`. |

## Value

A function like `period_aggregator_default` that aggregates data so that each time period is represented by exactly one record.

## PeriodDescriber *Period Describer*

### Description

Create a function that takes a data set containing at least two of the following variables, `period_start_variable`, `period_end_variable`, `period_days_variable`, and returning a data set with all of these three variables and other variables describing the middle of the period with either or both of `period_mid_time_variable` and `period_mid_date_variable`. These two period middle descriptors will only differ (by exactly 12 hours) for periods with odd numbers of days.

### Usage

```
PeriodDescriber(mid_types = c("time", "date"))
```

### Arguments

| | |
|---|---|
| `mid_types` | Compute mid-times and/or mid-dates? |

### Value

A function like `period_describer_default` that adds variables to describe the time period represented by each record.

## quantile_trans *Quantile Transformation*

### Description

Quantile transformation, adapted from https://stackoverflow.com/questions/38874741/transform-color-scale-to-probability-transformed-color-distribution-with-scale-f

### Usage

```
quantile_trans(x)
```

### Arguments

| | |
|---|---|
| `x` | vector to be transformed |

### Value

a `scales::trans_new` function

## read_iidda_dataset *Read IIDDA Dataset into a Dataframe*

### Description

Read IIDDA Dataset into a Dataframe

### Usage

```
read_iidda_dataset(dataset_id)
```

### Arguments

| | |
|---|---|
| dataset_id | ID for a dataset in the IIDDA |

## resolve_join *Resolve left_join*

### Description

Resolves any duplicate columns that results after left_join due to shared columns between data frames. Rule: Keeps old values if all newly joined values are NA. Keeps new values otherwise (even if some entries are empty)

### Usage

```
resolve_join(df)
```

### Arguments

| | |
|---|---|
| df | data frame with duplicate columns ending in .x and .y |

### Value

data frame with one remaining column for duplicates

## SeriesHarmonizer *Series Harmonizer*

### Description

Harmonizes the series variable in `data` so there is one data value for each time unit in time variable (to account for different variations in disease/cause name)

### Usage

```
SeriesHarmonizer(sum_fn = base::sum)
```

### Arguments

| | |
|---|---|
| `time_variable` | column name of time variable in `data`, default is "period_end_date" |
| `series_variable` | column name of series variable in `data`, default is "deaths" |

### Value

A function like `series_harmonizer_default` to harmonize disease/cause names.

## Skipper *Skip Pipeline Step*

### Description

Skip Pipeline Step

### Usage

```
Skipper()
```

## std_series_variables *Series variables*

### Description

List of column names that contain numerical information that could represent a time series if processed appropriately.

### Usage

```
std_series_variables()
```

## std_time_units *Time units*

### Description

Vector of all possible time units, most or all are derived from lubridate functions

### Usage

```
std_time_units
```

### Format

An object of class `character` of length 29.

## std_time_variables *Time variables*

### Description

List of column names that contain information locating a point or interval of time.

### Usage

```
std_time_variables()
```

## time_extent *Time Extent*

### Description

Length of time in days representated by an object

### Usage

```
time_extent(x, time_id)
```

### Arguments

| | |
|---|---|
| `x` | an object |
| `time_id` | identifier for finding time axis information in the object |

## TimeRangeDescriber *Time Range Desciber*

### Description

Time Range Desciber

### Usage

```
TimeRangeDescriber(cutoff = 50)
```

## TimeScalePicker *Time Scale Picker*

### Description

Time Scale Picker

### Usage

```
TimeScalePicker()
```

### Value

A function like `time_scale_picker_default` that

## TimeVariableConverter *Time Variable Converter*

### Description

Construct a function that takes a data frame and returns another data frame with a time variable converted so that it has the correct format, class, and/or type.

### Usage

```
TimeVariableConverter(
  as_date = as.Date,
  as_integer = as.integer,
  as_numeric = as.numeric
)
```

### Arguments

| | |
|---|---|
| as_date | Function that takes a vector and converts it to a date vector if possible. Used only if time_variable is in std_date_variables(). |
| as_integer | Function that takes a vector and converts it to an integer vector if possible. Used only if time_variable is in std_integer_variables(). |
| as_numeric | Function that takes a vector and converts it to a numeric vector if possible. Used only if time_variable is in std_numeric_variables(). |

**Returned Function:**

- Arguments :
  - data : Data frame containing a time variable.
  - time_variable : Column name of time variable in data. The default is "period_end_date".
- Return : A version of data with time_variable column converted to

---

## TitleGuesser *Title Guesser*

---

### Description

Title Guesser

### Usage

```
TitleGuesser(custom = NULL, prefer = std_title_variables())
```

### Arguments

| | |
|---|---|
| custom | Custom string for the title |
| prefer | List of variables that could contain title information in an order that will be used to find variables that will be used to guess at a title. The first variable found in the data is the one that is chosen. |

---

## titleize *Titleize*

---

### Description

Convert a character vector (i.e. a character column) into a title for a plot.

### Usage

```
titleize(title_info, max_items = 3L, max_chars = 15L)
```

### Arguments

| | |
|---|---|
| title_info | Character vector to be summarized into a title |
| max_items | TODO |
| max_chars | TODO |

---

## TrimSeries *Trim Time Series*

---

### Description

Remove leading or trailing zeros in a time series data set.

### Usage

```
TrimSeries(zero_lead = FALSE, zero_trail = FALSE)
```

### Arguments

| | |
|---|---|
| zero_lead | boolean value, if TRUE remove leading zeroes in data |
| zero_trail | boolean value, if TRUE remove trailing zeroes in data |

### Value

A function like trim_series_default to remove to remove leading and/or trailing zeroes.

---

## union_series *Union Time Series*

---

### Description

Combine two time series data sets with the option to handle overlapping time periods. This is particularly useful for data sets that come from two sources (ex. LBoM and RG). Assumes both data sets have the same number of columns with the same names.

### Usage

```
union_series(x, y, overlap = TRUE, time_variable = "period_end_date")
```

### Arguments

| | |
|---|---|
| x | first data frame containing time series data |
| y | second data frame containing time series data |
| overlap | boolean to indicate if x should get priority with overlapping time periods in y. If TRUE the returned data frame will contain all data from x, and the filtered y data that does not overlap with x. If FALSE, a union between x and y is returned. |
| time_variable | column name of time variable in x and y, default is "period_end_date" |

### Value

combined x and y data frames with optional filtering for overlaps

---

## unique_entries *Get unique tokens from iidda metadata*

---

### Description

Get unique tokens from iidda metadata

### Usage

```
unique_entries(entries, metadata_search)
```

### Arguments

| | |
|---|---|
| entries | List returned by iidda.api::ops_staging$metadata |
| metadata_search | Character, field from which unique tokens are desired |

### Value

Character vector of unique tokens for a given field from all iidda datasets

---

## valid_time_vars *Validate time variables*

---

### Description

Validate if variable is a date data type in the data set.

### Usage

```
valid_time_vars(var_nm, data)
```

### Arguments

| | |
|---|---|
| var_nm | string of variable name |
| data | data frame |

### Value

boolean of validation status

wavelet_something *Prep Data for Wavelet Plot*

## Description

Prep data for wavelet plot. Prep steps were taken from code provided by Steven Lee (https://github.com/davidearn/StevenLee) and Kevin Zhao (https://github.com/davidearn/KevinZhao).

## Arguments

| | |
|---|---|
| data | data frame containing time series data |
| trend_data | data frame containing time series trend data |
| time_variable | column name of time variable in data, default is "period_end_date" |
| series_variable | column name of series variable in data, default is "deaths_series" |
| trend_variable | column name of series variable in data, default is "deaths_trend" |
| series_suffix | suffix to be appended to series data fields |
| trend_suffix | suffix to be appended to trend data fields |
| wavelet_variable | name of the field in data to be wavelet transformed |
| output_emd_trend | name of output field for the empirical mode decomposition applied to trend_variable |
| output_norm | name of output field for the series_variable normalized by output_emd_trend |
| output_sqrt_norm | name of output field for the square root of output_norm |
| output_log_norm | name of output field for the logarithm of (output_norm + eps) |
| output_emd_norm | name of output field for the empirical mode decomposition applied to output_norm |
| output_emd_sqrt | name of output field for the empirical mode decomposition applied to output_sqrt_norm |
| output_emd_log | name of output field for the empirical mode decomposition applied to output_log_norm |
| output_detrend_norm | name of output field for the computed field output_norm-output_emd_norm |
| output_detrend_sqrt | name of output field for the computed field output_sqrt_norm-output_emd_sqrt |
| output_detrend_log | name of output field for the computed field output_log_norm-output_emd_log |
| data_harmonizer | function that harmonizes time scales and series names so there is one data point per time unit |

| | |
|---|---|
| trend_data_harmonizer | function that harmonizes time scales and trend names so there is one data point per time unit |
| data_deheaper | function that fixes heaping errors on series data |
| trend_deheaper | function that fixes heaping errors on trend data |
| joiner | function that joins series and trend data sets |
| interpolator | function that linearly interpolates series and trend data |
| normalizer | function that computes normalized fields |
| transformer | function that computes wavelet transform |

## Value

list containing: * transforemd_data - wavelet transformed data * tile_data_to_plot - data set of the wavelet transformed data prepped for plotting with ggplot2::geom_tile * contour_data_to_plot - data set of the transformed wavelet data prepped for plotting with ggplot2::geom_contour

---

# year_end_fix *Year End Fix*

---

## Description

Weeks covering the year end are split into two records. The first week is adjusted to end on day 365 (or 366 in leap years), and the second week starts on the first day of the year. This was adapted from LBoM::edge_fix which keeps the same series variable value for both of the newly created weeks. This doesn't seem to make much difference when viewing the heatmap, however it might make sense to do something sensible like dividing the series variable value in half and allocating each week to have half of the values.

Weeks covering the year end are split into two records. The first week is adjusted to end on day 365 (or 366 in leap years), and the second week starts on the first day of the year. This was adapted from LBoM::edge_fix which keeps the same series variable value for both of the newly created weeks. This doesn't seem to make much difference when viewing the seasonal heatmap, however it might make sense to do something sensible like dividing the series variable value in half and allocating each week to have half of the values.

## Usage

```
year_end_fix(
  data,
  series_variable = "deaths",
  start_year_variable = "Year",
  end_year_variable = "End Year",
  start_day_variable = "Day of Year",
  end_day_variable = "End Day of Year",
  temp_year_variable = "yr"
)
```

```
year_end_fix(
  data,
  series_variable = "deaths",
  start_year_variable = "Year",
  end_year_variable = "End Year",
  start_day_variable = "Day of Year",
  end_day_variable = "End Day of Year",
  temp_year_variable = "yr"
)
```

## Arguments

| | |
|---|---|
| `data` | data frame containing time series data |
| `series_variable` | column name of series variable in `data`, default is "deaths" |
| `start_year_variable` | column name of time variable containing the year of the starting period, defaults to "Year" |
| `end_year_variable` | column name of time variable containing the year of the ending period, defaults to "End Year" |
| `start_day_variable` | column name of time variable containing the day of the starting period, defaults to "Day of Year" |
| `end_day_variable` | column name of time variable containing the day of the ending period, defaults to "End Day of Year" |
| `temp_year_variable` | temporary variable name when pivoting the data frame |

## Value

all fields in `data` with only records corresponding to year end weeks that have been split

all fields in `data` with only records corresponding to year end weeks that have been split

# Index


∗ **data_prep_constructors**
ComputeMovingAverage, 7
CountAggregator, 8
data_prep_constructors, 9
Deheaper, 10
HandleMissingValues, 19
HandleZeroValues, 19
PeriodAggregator, 45
PeriodDescriber, 46
SeriesHarmonizer, 48
Skipper, 48
TimeRangeDescriber, 50
TimeVariableConverter, 50
TrimSeries, 52
∗ **datasets**
lubridate_funcs, 37
std_time_units, 49
∗ **dates**
year_end_fix, 55
∗ **harmonization**
add_user_entries, 3
create_bin_desc, 8
generate_empty_df, 12
generate_user_table, 13
join_lookup_table, 35
join_user_table, 36
lookup_join, 37
names_to_join_by, 39
resolve_join, 47
∗ **normalization**
ca_iso_3166_2, 6
factor_time_scale, 11
find_unaccounted_cases, 11
get_implied_zeros, 14
grid_dates, 15
normalize_disease_hierarchy, 40
normalize_duplicate_sources, 41
normalize_location, 41
normalize_population, 42
normalize_time_scales, 42
∗ **periods**
mid_dates_times, 38
num_days, 43
period_averager, 44
∗ **plotting_functions**
iidda_plot_bar, 21
iidda_plot_box, 21
iidda_plot_heatmap, 22
iidda_plot_highlight, 23
iidda_plot_ma, 24
iidda_plot_rohani_heatmap, 25
iidda_plot_settings, 26
iidda_plot_wavelet, 27
iidda_series, 33
∗ **prep_data_for_plotting**
iidda_prep_bar, 28
iidda_prep_box, 29
iidda_prep_rohani, 30
iidda_prep_seasonal_heatmap, 31
wavelet_something, 54
∗ **time_periods**
period_averager, 44

add_user_entries, 3

basal_group, 4
BasalGroupAdder, 5
browse_pipeline_dependencies, 5

ca_iso_3166_2, 6
check_date, 6
clean_canmod_cdi, 7
compute_moving_average_default, *7*
compute_moving_average_default *(*handle_missing_values_default*)*, 16
ComputeMovingAverage, 7
count_aggregator_default, *8*

count_aggregator_default
(handle_missing_values_default), 16
CountAggregator, 8
create_bin_desc, 8
CustomDataPrep, 9

data_prep_constructors, 9
data_prep_default
(handle_missing_values_default), 16
DataDictionaryConverter, 10
Deheaper, 10
deheaper_default, *10*
deheaper_default
(handle_missing_values_default), 16

factor_time_scale, 11
find_unaccounted_cases, 11

generate_disease_df, 12
generate_empty_df, 12
generate_user_table, 13
get_iidda_attr, 13
get_implied_zeros, 14
get_unit_labels, 14
grid_dates, 15

handle_missing_values_default, 16, *19*
handle_zero_values_default, *19*
handle_zero_values_default
(handle_missing_values_default), 16
HandleMissingValues, 19
HandleZeroValues, 19

iidda_attach_bar, 20
iidda_attach_series (iidda_series), 33
iidda_get_metadata, 20
iidda_plot_bar, 21
iidda_plot_box, 21
iidda_plot_heatmap, 22
iidda_plot_highlight, 23
iidda_plot_ma, 24
iidda_plot_rohani_heatmap, 25
iidda_plot_settings, 26
iidda_plot_wavelet, 27
iidda_prep_bar, 28
iidda_prep_box, 29
iidda_prep_ma (iidda_series), 33
iidda_prep_rohani, 30
iidda_prep_seasonal_heatmap, 31
iidda_prep_series (iidda_series), 33
iidda_render_plot, 32
iidda_series, 33
iidda_theme, 35
iidda_theme_above (iidda_theme), 35
iidda_theme_heat (iidda_theme), 35
iidda_theme_time (iidda_theme), 35

join_lookup_table, 35
join_user_table, 36

log1p_modified_trans, 36
lookup_join, 37
lubridate_funcs, 37

make_time_trans, 38
mid_dates (mid_dates_times), 38
mid_dates_times, 38
mid_times (mid_dates_times), 38
mutate_time_vars, 39

names_to_join_by, *37*, 39
normalize_disease_hierarchy, 40
normalize_duplicate_sources, 41
normalize_location, 41
normalize_population, 42
normalize_time_scales, *7*, 42
num_days, *38*, 43
num_days_util (num_days), 43

period(), *15*
period_aggregator_default, *45*
period_aggregator_default
(handle_missing_values_default), 16
period_averager, 44
period_describer_default, *46*
period_describer_default
(handle_missing_values_default), 16
PeriodAggregator, 45
PeriodDescriber, 46

quantile_trans, 46

read_iidda_dataset, 47

resolve_join, 47

series_harmonizer_default, *48*
series_harmonizer_default
        *(*handle_missing_values_default*)*,
        16
SeriesHarmonizer, 48
Skipper, 48
std_series_variables, 48
std_time_units, 49
std_time_variables, 49

time_extent, 49
time_scale_picker_default, *50*
TimeRangeDescriber, 50
TimeScalePicker, 50
TimeVariableConverter, 50
TitleGuesser, 51
titleize, 51
trim_series_default, *52*
trim_series_default
        *(*handle_missing_values_default*)*,
        16
TrimSeries, 52

union_series, 52
unique_entries, 53

valid_time_vars, 53

wavelet_something, 54

year_end_fix, 55