# Package: grattanInflators (via r-universe)

October 11, 2026

**Title** Inflators for Australian Policy Analysis

**Version** 0.6.0

**Description** Using Australian Bureau of Statistics indices, provides functions that convert historical, nominal statistics to real, contemporary values without worrying about date input quality, performance, or the ABS catalogue.

**License** GPL-2

**Encoding** UTF-8

**Depends** R (>= 4.0.0)

**Imports** data.table, fy, hutils, tools, utils

**RoxygenNote** 7.3.3

**Suggests** distributional, fable, fabletools, tinytest, withr

**NeedsCompilation** yes

**Author** Hugh Parsonage [aut, cre]

**Maintainer** Hugh Parsonage <hugh.parsonage@gmail.com>

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

**Date/Publication** 2026-09-11 09:51:29 UTC

**RemoteUrl** https://github.com/cran/grattanInflators

**RemoteRef** HEAD

**RemoteSha** 917b5745b1bc1433fe2af1ea08bfb7958a498ea6

## Contents

abs-conn . . . . . 2
awe_inflator . . . . . 3
cpi_inflator . . . . . 5
custom-series . . . . . 6
fast_as_idate . . . . . 7
Inflate . . . . . 8
lf_inflator . . . . . 10
wage_inflator . . . . . 11

**Index** 13

---

abs-conn *ABS Connections*

---

## Description

The package uses the catalogue mirrored at https://github.com/HughParsonage/ABS-Catalogue. These functions expose the guts of the package's method to connect to this mirror.

Each inflator, plus the 'adjustment', is associated with an ABS Series ID.

## Usage

```
content2series_id(
  broad_cat = c("cpi", "lfi", "wpi", "awe", "awote"),
 adjustment = c("original", "seasonal", "trend", "trimmed-mean", "monthly-original",
    "monthly-seasonal", "monthly-excl-volatile")
)

download_data(series_id = NULL)

when_last_updated()

grattanInflators_has_no_data()
```

## Arguments

`broad_cat`, `adjustment`
: Definitions to identify the Series ID. If any are multiple, the result is of the cartesian join, **not** the component-wise values.

`series_id`
: The Series ID desired. For `download_data`, if `NULL`, the default, downloads all files required.

## Value

`content2series_id` A character vector, the Series ID identified by 'broad_cat' and 'adjustment'

`download_data` Called for its side-effect, downloading updated data to the user data directory. A downloaded series takes precedence when its latest observation is at least as recent as the snapshot bundled with the package. Returns an integer vector of the statuses of each download.

`when_last_updated` The date an update was last downloaded, or the string `"Never updated"` if no update has been downloaded. The bundled snapshot does not count as an update. Note that this is the date of *retrieval*, not the ABS release the data came from: the mirror tracks the ABS, so two sessions running the same package version at different times may see different index histories.

`grattanInflators_has_no_data` `TRUE` if neither bundled nor downloaded data are available. A normal installation includes bundled data for every inflator.

awe_inflator *Average weekly earnings inflators and data*

## Description

awe_inflator() uses average weekly total earnings for all employees (AWE); awote_inflator() uses average weekly ordinary time earnings for full-time adult employees (AWOTE). Both cover persons, Australia, private and public sectors combined, in dollars per week, excluding salary sacrifice. Changes reflect workforce composition as well as earnings changes; these are earnings-level measures, unlike the Wage Price Index in wage_inflator().

## Usage

```
awe_inflator(
  from = NULL,
  to = NULL,
  check = 1L,
  series = awe_original(),
  fy_month = 3L,
  x = NULL,
  nThread = getOption("grattanInflators.nThread", 1L)
)

awote_inflator(
  from = NULL,
  to = NULL,
  check = 1L,
  series = awote_original(),
  fy_month = 3L,
  x = NULL,
  nThread = getOption("grattanInflators.nThread", 1L)
)

awe_original(..., FORECAST = FALSE, LEVEL = "mean")

awe_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

awe_trend(..., FORECAST = FALSE, LEVEL = "mean")

awote_original(..., FORECAST = FALSE, LEVEL = "mean")

awote_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

awote_trend(..., FORECAST = FALSE, LEVEL = "mean")
```

## Arguments

| | |
|---|---|
| from, to | Times for which the inflator is desired. If NULL, a date range close to the previous year is used. |
| check | integer(1) If 0L, no checks are performed, and clearly invalid inputs result in NA in the output. If check = 1L, invalid input errors and extrapolation warns. If check = 2L, dates outside the exact index endpoints error instead of being extrapolated. |
| series | A call to awe_original(), awe_seasonal(), or awe_trend() for awe_inflator(), or the corresponding awote_*() function for awote_inflator(). A custom index accepted by Inflate() may also be used. |
| fy_month | The month to be used in 'series' for financial years. |
| x | (Advanced) A double vector that will be inflated in-place. If NULL, the default, the return vector is simply the inflation factor for 'from'. Since 'x' is modified in place, any other name bound to the same object is modified too; an integer 'x' is coerced to double, which copies, so in that case use the return value. |
| nThread | Number of threads to use. |
| ... | Date-rate pairs or a final annual growth rate for a custom series. |
| FORECAST | Whether the series should be extended via an ETS forecast. |
| LEVEL | If 'FORECAST = TRUE' what prediction interval should be used. ('LEVEL = 20' means the lower end of an 80% prediction interval.) If 'LEVEL = "mean"' (the default), the central estimate is used. |

## Details

The data functions return the ABS half-yearly series for May and November. The bundled May 2026 release contains original observations from November 1994, and seasonal and trend observations from May 2012. Earlier discontinued quarterly series are not spliced into these series.

Inflators use the ratio of the values in the periods containing to and from. May represents May to October; November represents November to April. Dates are matched by month, with exact endpoint checks controlled by check. Integer years represent January, so use an explicit May or November date when that observation is intended. Financial years use fy_month, as in Inflate(). Values between observations are not interpolated.

Custom growth rates are annual rates compounded at half-yearly intervals. Forecasts also retain the half-yearly frequency. Data can be refreshed with download_data() and used directly as the index argument to Inflate().

## Value

The inflators return a numeric vector of earnings ratios, or x multiplied by those ratios. The six data functions return a data.table with date (IDate) and value (dollars per week) columns.

## Source

Australian Bureau of Statistics, Average Weekly Earnings, Australia, May 2026, Tables 1 (trend), 2 (seasonally adjusted), and 3 (original). https://www.abs.gov.au/statistics/labour/earnings-and-working-condit
average-weekly-earnings-australia/may-2026

## Examples

```
awe_original()
awote_seasonal()
awe_inflator("2024-05-15", "2025-05-15")
awote_inflator("2024-05-15", "2025-05-15", series = awote_seasonal())
awote_inflator("2030-05-15", "2031-05-15", series = awote_original("3%"))
```

---

`cpi_inflator` *CPI inflator*

---

## Description

CPI inflator

## Usage

```
cpi_inflator(
  from = NULL,
  to = NULL,
  series = c("seasonal", "original", "trimmed.mean", "monthly-original",
    "monthly-seasonal", "monthly-excl-volatile"),
  fy_month = 3L,
  x = NULL,
  check = 1L,
  nThread = getOption("grattanInflators.nThread", 1L)
)

cpi_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

cpi_original(..., FORECAST = FALSE, LEVEL = "mean")

cpi_trimmed_mean(..., FORECAST = FALSE, LEVEL = "mean")

cpi_monthly_original(..., FORECAST = FALSE, LEVEL = "mean")

cpi_monthly_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

cpi_monthly_excl_volatile(..., FORECAST = FALSE, LEVEL = "mean")
```

## Arguments

| | |
|---|---|
| `from`, `to` | Times for which the inflator is desired. If `NULL`, a date range close to the previous year is used. |
| `series` | Which CPI series to use. |

| | |
|---|---|
| fy_month | An integer 1-12 used to locate financial-year inputs in the index. Ordinary integer years are represented by January and are unaffected by fy_month. For financial years, the month is the month of the year, so for example fy_month = 9 and "2015-16" means Sep-2015, while fy_month = 6 means Jun-2016. |
| x | (Advanced) A double vector that will be inflated in-place. If NULL, the default, the return vector is simply the inflation factor for 'from'. Since 'x' is modified in place, any other name bound to the same object is modified too; an integer 'x' is coerced to double, which copies, so in that case use the return value. |
| check | integer(1) If 0L, no checks are performed, and clearly invalid inputs result in NA in the output. If check = 1L, invalid input errors and extrapolation warns. If check = 2L, dates outside the exact index endpoints error instead of being extrapolated. |
| nThread | Number of threads to use. |
| ... | Set of date-rate pairs for custom CPI series in the future. |
| FORECAST | Whether the series should be extended via an ETS forecast. |
| LEVEL | If 'FORECAST = TRUE' what prediction interval should be used. ('LEVEL = 20' means the lower end of an 80% prediction interval.) If 'LEVEL = "mean"' (the default), the central estimate is used. |

## Value

If 'x' is 'NULL', the default, a numeric vector matching the lengths of 'from' and 'to' equal to the inflators by which nominal prices dated 'from' must be multiplied so that they are in 'to' real terms.

If 'x' is numeric, it is taken to be prices dated 'from' and the value returned is 'x' in 'to' real terms.

## Examples

```
cpi_inflator(1995, 2019)  # Inflation from 1995 to 2019
cpi_inflator("2015-16", "2016-17")
cpi_inflator("2015-01-01", "2016-01-01")

# A custom series, holding CPI growth at 10% a year from 2030
cpi_inflator("2030-01-01", "2031-01-01", series = cpi_original(2030, 0.1))
cpi_inflator("2030-01-01", "2031-01-01", series = cpi_original("10%"))
cpi_inflator("2030-01-01", "2032-01-01",
             series = cpi_original(2030, 0.1, 2031, 0.1, 2032, 0))
```

---

# custom-series *Custom series*

---

## Description

Used when the true series is not appropriate, as when a forecast is desired and the series is required beyond the original series.

### Usage

```
dr2index(index, d1, r1, ...)
```

### Arguments

| | |
|---|---|
| index | An index (i.e. a data.table with columns date and value, where date is a regular sequence of monthly, quarterly, half-yearly, or annual dates), and value is the indexed value for that date. |
| d1 | A single date or value representing a date. |
| r1 | The desired rate of increase for the index from the last date in index to the end of d1. For example, d1 = 2025 and r1 = 0 applied to a monthly index would keep value constant until 2025-12-01.<br>Rates are annual and may be given as a number (0.05) or as a percentage string ("5%", "-2.5%"). |
| ... | A set of date-rate pairs. |

### Value

index with dates extended until the last supported date. The final rate supplied is the rate for all dates after the final date.

---

## fast_as_idate *Faster conversion to IDate for common dates*

---

### Description

Faster conversion to IDate for common dates

### Usage

```
fast_as_idate(
  x,
  incl_day = TRUE,
  check = 0L,
  nThread = 1L,
  format = "%Y-%m-%d"
)
```

### Arguments

| | |
|---|---|
| x | The character vector to convert, in YYYY-mm-dd form only. |
| incl_day | Whether or not the day is necessary to convert. Set to FALSE when the day component does not matter (or is constantly -01); the day component in the output will be -01. |

| | |
|---|---|
| check | integer: 0, 1, or 2 Level of check to perform. 0 for no checks; 1 errors on any element that cannot be parsed; 2 additionally rejects impossible days of the month (such as 30 February). |
| nThread | Number of threads to use. |
| format | The expected format of the input: one of "%Y-%m-%d", "%d/%m/%Y", "%d-%m-%Y" or "%d%b%Y". An unrecognised format is an error rather than being reinterpreted as another. guess_format() returns a format that this function accepts. |

## Details

A 10M vector of dates was observed to be parsed in 0.1s whereas as.IDate took 9.0s, and lubridate::ymd, 1.6s. Note that false dates (such as Feb 30) will be naively parsed without warning or error (unless 'check' is changed from its default argument).

## Value

A vector of class IDate, Date the same length as x.

## Examples

```
# For ABS data, we only need to care (and check)
# the year and month
fast_as_idate("2015-12-13", incl_day = FALSE)
```

---

# Inflate *Generic inflator*

---

## Description

Generic inflator

## Usage

```
Inflate(
  from,
  to,
  index,
  x = NULL,
  fy_month = 3L,
  check = 2L,
  nThread = getOption("grattanInflators.nThread", 1L)
)
```

## Arguments

| | |
|---|---|
| from, to | Times for which the inflator is desired. If NULL, a date range close to the previous year is used. |
| index | A table of at least two columns, named date and value. date is the column of times to which from, to will be matched. value is the values that determine the inflation factor.<br>The dates must be a strictly increasing, regular annual, half-yearly, quarterly, or monthly sequence, and the values finite and nonzero: the underlying implementation locates an observation by arithmetic on the first date, not by a lookup, so an irregular or unsorted table would silently give the wrong answer. Inflate validates this before use. For checked inputs, period-based matching is used only between the first and last observed dates. A date outside those exact endpoints is treated as extrapolation, even if it falls in the same month, quarter, or anchored year as an endpoint. With check = 1L, a date after the final observation but within its calculation period carries forward the terminal index value; a date in a later period causes the index to be projected. Both cases warn. Half-yearly periods are anchored to the first observation's month: a May observation represents May to October, and November represents November to April. Matching uses the month, subject to the exact endpoint checks above. |
| x | (Advanced) A double vector that will be inflated in-place. If NULL, the default, the return vector is simply the inflation factor for 'from'.<br>Because x is modified in place, any other name bound to the same object is modified too. An integer x is coerced to double, which necessarily copies, so in that case the result must be taken from the return value rather than from x. |
| fy_month | An integer 1-12 used to locate financial-year inputs in the index. Ordinary integer years are represented by January and are unaffected by fy_month. For financial years, the month is the month of the year, so for example fy_month = 9 and "2015-16" means Sep-2015, while fy_month = 6 means Jun-2016. |
| check | integer(1) If 0L, no checks are performed, and clearly invalid inputs result in NA in the output. If check = 1L, invalid input errors and extrapolation warns. If check = 2L, dates outside the exact index endpoints error instead of being extrapolated.<br>Note that check governs how loudly invalid *dates* are reported. It never governs whether the index is bounds-checked: an out-of-range date always yields NaN rather than reading beyond the index. |
| nThread | Number of threads to use. |

## Value

If 'x' is 'NULL', the default, a numeric vector matching the lengths of 'from' and 'to' equal to the ratio between the corresponding values in the column value.

If 'x' is numeric, those values are multiplied by the inflators, in-place.

`lf_inflator` *Labour force inflator*

## Description

Uses the Labour Force Index to provide equivalent sizes of the labour force over different times by multiplying by the simple ratio of the sizes on those dates.

## Usage

```
lf_inflator(
  from = NULL,
  to = NULL,
  check = 1L,
  series = lfi_original(),
  fy_month = 3L,
  x = NULL,
  nThread = getOption("grattanInflators.nThread", 1L)
)

lfi_original(..., FORECAST = FALSE, LEVEL = "mean")

lfi_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

lfi_trend(..., FORECAST = FALSE, LEVEL = "mean")
```

## Arguments

| | |
|---|---|
| `from`, `to` | Times for which the inflator is desired. If `NULL`, a date range close to the previous year is used. |
| `check` | `integer(1)` If `0L`, no checks are performed, and clearly invalid inputs result in `NA` in the output. If `check = 1L`, invalid input errors and extrapolation warns. If `check = 2L`, dates outside the exact index endpoints error instead of being extrapolated. |
| `series` | A call to ‘lfi_original()‘, ‘lfi_seasonal()‘, or ‘lfi_trend()‘. |
| `fy_month` | The month to be used in ‘series‘ for financial years. |
| `x` | (Advanced) A double vector that will be inflated in-place. If `NULL`, the default, the return vector is simply the inflation factor for ‘from‘. Since ‘x‘ is modified in place, any other name bound to the same object is modified too; an integer ‘x‘ is coerced to double, which copies, so in that case use the return value. |
| `nThread` | Number of threads to use. |
| `...` | Set of date-rate pairs for custom labour force series in the future. |
| `FORECAST` | Whether the series should be extended via an ETS forecast. |
| `LEVEL` | If ‘FORECAST = TRUE‘ what prediction interval should be used. (‘LEVEL = 20‘ means the lower end of an 80% prediction interval.) If ‘LEVEL = "mean"‘ (the default), the central estimate is used. |

## Value

If ‘x‘ is ‘NULL‘, the default, a numeric vector matching the lengths of ‘from‘ and ‘to‘ equal to the relative size of the labour force of ‘from‘ and ‘to‘.

If ‘x‘ is numeric, it is taken to be the sizes of the labour force on dates ‘from‘ and the value returned is the equivalent size of ‘x‘ on dates ‘to‘ (by simple multiplication).

## Examples

```
# The relative size of the labour force in FY 2016-17
# compared to FY 2015-16
lf_inflator("2015-16", "2016-17")
```

---

`wage_inflator` *Wage inflator*

---

## Description

Uses the Wage Price Index

## Usage

```
wage_inflator(
  from = NULL,
  to = NULL,
  check = 1L,
  series = wpi_original(),
  fy_month = 3L,
  x = NULL,
  nThread = getOption("grattanInflators.nThread", 1L)
)

wpi_original(..., FORECAST = FALSE, LEVEL = "mean")

wpi_seasonal(..., FORECAST = FALSE, LEVEL = "mean")

wpi_trend(..., FORECAST = FALSE, LEVEL = "mean")
```

## Arguments

| | |
|---|---|
| `from`, `to` | Times for which the inflator is desired. If `NULL`, a date range close to the previous year is used. |
| `check` | `integer(1)` If `0L`, no checks are performed, and clearly invalid inputs result in `NA` in the output. If `check = 1L`, invalid input errors and extrapolation warns. If `check = 2L`, dates outside the exact index endpoints error instead of being extrapolated. |

| | |
|---|---|
| series | A call to ‘wpi_original()‘, ‘wpi_seasonal()‘, or ‘wpi_trend()‘, defining which wage price index series to use. |
| fy_month | The month to be used in ‘series‘ for financial years. |
| x | (Advanced) A double vector that will be inflated in-place. If NULL, the default, the return vector is simply the inflation factor for ‘from‘. Since ‘x‘ is modified in place, any other name bound to the same object is modified too; an integer ‘x‘ is coerced to double, which copies, so in that case use the return value. |
| nThread | Number of threads to use. |
| ... | Set of date-rate pairs for custom WPI series in the future. |
| FORECAST | Whether the series should be extended via an ETS forecast. |
| LEVEL | If ‘FORECAST = TRUE‘ what prediction interval should be used. (‘LEVEL = 20‘ means the lower end of an 80% prediction interval.) If ‘LEVEL = "mean"‘ (the default), the central estimate is used. |

## Value

If ‘x‘ is ‘NULL‘, the default, a numeric vector matching the lengths of ‘from‘ and ‘to‘ equal to the inflators by which nominal wages dated ‘from‘ must be multiplied so that they are in ‘to‘ real terms.

If ‘x‘ is numeric, it is taken to be wages dated ‘from‘ and the value returned is ‘x‘ in ‘to‘ real terms.

# Index


abs-conn, 2
awe_inflator, 3
awe_original *(awe_inflator)*, 3
awe_seasonal *(awe_inflator)*, 3
awe_trend *(awe_inflator)*, 3
awote_inflator *(awe_inflator)*, 3
awote_original *(awe_inflator)*, 3
awote_seasonal *(awe_inflator)*, 3
awote_trend *(awe_inflator)*, 3

content2series_id *(abs-conn)*, 2
cpi_inflator, 5
cpi_monthly_excl_volatile *(cpi_inflator)*, 5
cpi_monthly_original *(cpi_inflator)*, 5
cpi_monthly_seasonal *(cpi_inflator)*, 5
cpi_original *(cpi_inflator)*, 5
cpi_seasonal *(cpi_inflator)*, 5
cpi_trimmed_mean *(cpi_inflator)*, 5
custom-series, 6

download_data *(abs-conn)*, 2
download_data(), *4*
dr2index *(custom-series)*, 6

fast_as_idate, 7

grattanInflators_has_no_data *(abs-conn)*, 2

Inflate, 8
Inflate(), *4*

lf_inflator, 10
lfi_original *(lf_inflator)*, 10
lfi_seasonal *(lf_inflator)*, 10
lfi_trend *(lf_inflator)*, 10

wage_inflator, 11
wage_inflator(), *3*
when_last_updated *(abs-conn)*, 2
wpi_original *(wage_inflator)*, 11
wpi_seasonal *(wage_inflator)*, 11
wpi_trend *(wage_inflator)*, 11