Package ‘grattan’

January 29, 2021

Type Package
Title Australian Tax Policy Analysis
Version 1.9.0.4
Date 2021-01-29
Maintainer Hugh Parsonage <hugh.parsonage@gmail.com>
URL https://github.com/HughParsonage/grattan,
https://hughparsonage.github.io/grattan/
BugReports https://github.com/HughParsonage/grattan/issues
Description Utilities to cost and evaluate Australian tax policy, including fast
projections of personal income tax collections, high-performance tax and
transfer calculators, and an interface to common indices from the Australian
Bureau of Statistics. Written to support Grattan Institute's Australian
Perspectives program, and related projects. Access to the Australian Taxation
Office's sample files of personal income tax returns is assumed.
Depends R (>= 3.5.0)
License GPL-2
Imports data.table, hutils (>= 1.3.0), ineq (>= 0.2-10), fastmatch,
forecast, fy (>= 0.2.0), hutilscpp, assertthat (>= 0.1),
magrittr (>= 1.5), Rcpp (>= 0.12.3), utils, zoo (>= 1.5-5)
LinkingTo Rcpp
RoxygenNote 7.1.1
Suggests curl, dplyr, dtplyr, fst (>= 0.8.4), future, future.apply,
ggplot2, ggrepel, knitr, lattice, mgcv, rlang, rmarkdown,
rmdmx, scales, survey, taxstats, taxstats1516, testthat,
tibble, viridis, yaml, withr, covr
Additional_repositories https://hughparsonage.github.io/tax-drat/
LazyData true
VignetteBuilder knitr
Encoding UTF-8
NeedsCompilation yes
Author  Hugh Parsonage [aut, cre],
        Tim Cameron [aut],
        Brendan Coates [aut],
        Matthew Katzen [aut],
        William Young [aut],
        Ittima Cherastidtham [dlt],
        W. Karsten [ctb],
        M. Enrique Garcia [ctb],
        Matt Cowgill [aut]

Repository  CRAN

Date/Publication  2021-01-29 09:40:05 UTC

R topics documented:

gratan-package ................................................. 4
age_grouper .................................................. 4
age_pension .................................................. 6
age_pension_age ............................................. 7
anyGeq ......................................................... 8
AnyWhich ....................................................... 8
apply_super_caps_and_div293 ................................. 9
aus_pop_qtr .................................................. 11
aus_pop_qtr_age ............................................. 11
awote .......................................................... 12
bto ............................................................. 13
carers_allowance ........................................... 14
carer_payment ............................................... 14
CG_population_inflator ....................................... 16
child_care_subsidy .......................................... 17
compare_avg_tax_rates ...................................... 19
cpi_inflator .................................................. 20
cpi_inflator_general_date .................................. 21
cpi_inflator_quarters ...................................... 22
differentially_uprate_wage ................................ 23
disability_pension .......................................... 24
energy_supplement .......................................... 25
family_tax_benefit ......................................... 26
gdp ............................................................ 28
generic_inflator ............................................. 28
gni ............................................................ 29
IncomeTax ...................................................... 30
income_tax .................................................... 30
income_tax_sapto ........................................... 32
inflator ......................................................... 33
install_taxstats ............................................. 34
inverse_average_rate ....................................... 35
inverse_income .............................................. 35
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>is.fy</td>
<td>36</td>
</tr>
<tr>
<td>lf_inflator</td>
<td>37</td>
</tr>
<tr>
<td>lito</td>
<td>39</td>
</tr>
<tr>
<td>max_super_contr_base</td>
<td>40</td>
</tr>
<tr>
<td>MedicareLevy</td>
<td>41</td>
</tr>
<tr>
<td>medicare_levy</td>
<td>41</td>
</tr>
<tr>
<td>model_child_care_subsidy</td>
<td>42</td>
</tr>
<tr>
<td>model_income_tax</td>
<td>44</td>
</tr>
<tr>
<td>model_new_caps_and_div293</td>
<td>47</td>
</tr>
<tr>
<td>model_rent_assistance</td>
<td>50</td>
</tr>
<tr>
<td>newstart_allowance</td>
<td>51</td>
</tr>
<tr>
<td>new_income_tax</td>
<td>53</td>
</tr>
<tr>
<td>new_medicare_levy</td>
<td>53</td>
</tr>
<tr>
<td>new_sapto</td>
<td>54</td>
</tr>
<tr>
<td>npv</td>
<td>55</td>
</tr>
<tr>
<td>Offset</td>
<td>56</td>
</tr>
<tr>
<td>pension_supplement</td>
<td>56</td>
</tr>
<tr>
<td>pmax3</td>
<td>57</td>
</tr>
<tr>
<td>pmaxC</td>
<td>58</td>
</tr>
<tr>
<td>pmaxV</td>
<td>58</td>
</tr>
<tr>
<td>pminC</td>
<td>59</td>
</tr>
<tr>
<td>pminV</td>
<td>59</td>
</tr>
<tr>
<td>progressivity</td>
<td>60</td>
</tr>
<tr>
<td>prohibit_length0_vectors</td>
<td>60</td>
</tr>
<tr>
<td>prohibit_unequal_length_vectors</td>
<td>61</td>
</tr>
<tr>
<td>rebate_income</td>
<td>64</td>
</tr>
<tr>
<td>rent_assistance</td>
<td>64</td>
</tr>
<tr>
<td>require_taxstats</td>
<td>65</td>
</tr>
<tr>
<td>residential_property_prices</td>
<td>67</td>
</tr>
<tr>
<td>revenue_foregone</td>
<td>68</td>
</tr>
<tr>
<td>sapto</td>
<td>68</td>
</tr>
<tr>
<td>sapto_rcpp</td>
<td>69</td>
</tr>
<tr>
<td>sapto_rcpp_singleton</td>
<td>70</td>
</tr>
<tr>
<td>sapto_rcpp_yr</td>
<td>70</td>
</tr>
<tr>
<td>small_business_tax_offset</td>
<td>71</td>
</tr>
<tr>
<td>student_repayment</td>
<td>72</td>
</tr>
<tr>
<td>unemployment_benefit</td>
<td>73</td>
</tr>
<tr>
<td>validate_date</td>
<td>74</td>
</tr>
<tr>
<td>validate_per</td>
<td>75</td>
</tr>
<tr>
<td>wage_inflator</td>
<td>76</td>
</tr>
<tr>
<td>youth_allowance</td>
<td>77</td>
</tr>
<tr>
<td>youth_unemployment</td>
<td>79</td>
</tr>
</tbody>
</table>
The grattan package.

Description

Grattan package

Details

Tax modelling and other common tasks for Australian policy analysts, in support of the Grattan Institute, Melbourne. <https://grattan.edu.au>

Package options

- `grattan.verbose` (FALSE) Emit diagnostic messages (via `cat()`)
- `grattan.assume1901_2100` (TRUE) Assume `yr2fy` receives an integer >= 1901 and <= 2100.
- `grattan.taxstats.lib` Package library into which `taxstats` packages will be installed. If `NULL`, a temporary directory is used.

Author(s)

<hugh.parsonage+grattanpackage@grattan.edu.au>
<hugh.parsonage@gmail.com>

See Also

Useful links:

- [https://github.com/HughParsonage/grattan](https://github.com/HughParsonage/grattan)
- [https://hughparsonage.github.io/grattan/](https://hughparsonage.github.io/grattan/)
- Report bugs at [https://github.com/HughParsonage/grattan/issues](https://github.com/HughParsonage/grattan/issues)

Age grouper

Description

Age grouper
**Usage**

```r
age_grouper(
    age,
    interval = 10,
    min_age = 25,
    max_age = 75,
    breaks = NULL,
    labels = NULL,
    below = "Below
",
    exp_min_age = 1L,
    exp_max_age = 100L,
    threshold = 10000L
)
```

**Arguments**

- **age**
  A numeric age (in years).

- **interval**
  How big should the age range be. 25-34 means interval = 10.

- **min_age**
  What is the upper bound of the lowest bracket? (min_age = 25 means 'Under 25' will be the lowest bracket.)

- **max_age**
  What is the lower bound of the highest bracket? (max_age = 75 means '75+' will be the bracket.)

- **breaks**
  Specify breaks manually.

- **labels**
  Specify the labels manually.

- **below**
  String giving the prefix for the lowest bin. (Only applicable if breaks and labels are NULL.)

- **exp_min_age, exp_max_age**
  Integers specifying the lowest/highest expected age in age. If any values fall outside this range, ages will still work though perhaps slow when length(age) >> threshold.

- **threshold**
  An integer, the minimum length at which the calculation will be accelerated.

**Value**

An ordered factor giving age ranges (separated by hyphens) as specified.

**Examples**

```r
age_grouper(42)
age_grouper(42, interval = 5, min_age = 20, max_age = 60)
```
Description
Age pension

Usage

```r
age_pension(
    fortnightly_income = 0,
    annual_income = fortnightly_income * 26,
    has_partner = FALSE,
    n_dependants = 0L,
    partner_fortnightly_income = 0,
    partner_annual_income = partner_fortnightly_income * 26,
    partner_pensioner = has_partner,
    Date = NULL,
    fy.year = NULL,
    assets_value = 0,
    financial_assets = 0,
    is_home_owner = FALSE,
    illness_separated_couple = FALSE,
    per = c("year", "fortnight")
)
```

Arguments

- `fortnightly_income`, `annual_income` Income for means-testing purposes. Provide one but not both.
- `has_partner` (logical, default: FALSE) Does the individual have a partner?
- `n_dependants` How many dependants does the individual have? Default is zero.
- `partner_fortnightly_income`, `partner_annual_income` The partner’s income. The sum of this value and the individual’s income gives the income test.
- `partner_pensioner` (logical, default: TRUE) Is the individual’s partner also in receipt of the age pension?
- `Date`, `fy.year` The financial year. Currently only 2015-16 is supported (the most recent survey of income and housing results).
- `assets_value`, `financial_assets` Total value of household assets. Assets which earn incomes for which deeming rates apply.
- `is_home_owner` (logical, default: FALSE) Does the individual own their own home?
age_pension_age

illness_separated_couple
  Is the couple separated by illness? (Affects the assets test.)
per
  Specifies the timeframe in which payments will be made. One of "year" and "fortnight".

Details
  Currently does not include the age pension supplement.

Value
  Returns the age pension payable for each individual defined by the arguments, assuming otherwise eligible.

age_pension_age | Age of eligibility for the Age Pension

Description
  Age of eligibility for the Age Pension

Usage
  age_pension_age(when = Sys.Date(), sex = "male")

Arguments
  when
    Either a Date (or a character vector coercible to such) or a financial year, when the age of eligibility of Age Pension is requested. Defaults to current date.
  sex
    A character vector the same length as when, containing strings "male" and "female". May be abbreviated to "m" or "f" and is case-insensitive.

Value
  A numeric vector, the age of eligibility for the Age Pension for each when.

Source

Examples
  age_pension_age() # Current age of eligibility
  age_pension_age("1995-12-31")
  age_pension_age("2013-14")
**anyGeq**  
*Any without logical creation*

**Description**

Any without logical creation

**Usage**

\[\text{anyGeq}(x, a)\]

**Arguments**

- **x**: An integer vector.
- **a**: An integer.

**Value**

0 if none true or the index of the first match.

---

**AnyWhich**  
*Quickly verify (and locate) the existence of a breach.*

**Description**

Used when a single instance is likely to occur and be important to detect quickly (in a sufficiently large integer vector).

**Arguments**

- **x**: An integer vector.
- **a**: A (single) integer. That which is to be compared.
- **gt, lt, eq**: Booleans, whether or not the comparison is greater than, less than, or equal to. Only \(gt\) and \(lt\) are mutually exclusive.
Superannuation caps and Division 293 calculations

Description
Mutate a sample file to reflect particular caps on concessional contributions and applications of Division 293 tax.

Usage
```r
apply_super_caps_and_div293(
  .sample.file,
  colname_concessional = "concessional_contributions",
  colname_div293_tax = "div293_tax",
  colname_new_Taxable_Income = "Taxable_income_for_ECT",
  div293_threshold = 300000,
  cap = 30000,
  cap2 = 35000,
  age_based_cap = TRUE,
  cap2_age = 59,
  ecc = FALSE,
  use_other_contr = FALSE,
  scale_contr_match_ato = FALSE,
  .lambda = 0,
  reweight_late_lodgers = FALSE,
  .mu = 1.05,
  impute_zero_concess_contr = FALSE,
  .min.Sw.for.SG = 450 * 12,
  .SG_rate = 0.0925,
  warn_if_colnames_overwritten = TRUE,
  drop_helpers = FALSE,
  copyDT = TRUE
)
```

Arguments
- `.sample.file`: A data.table containing at least the variables `sample_file_1314` from the taxstats package.
- `colname_concessional`: The name for concessional contributions.
- `colname_div293_tax`: The name of the column containing the values of Division 293 tax payable for that taxpayer.
- `colname_new_Taxable_Income`: The name of the column containing the new Taxable Income.
apply_super_caps_and_div293

div293_threshold
The Division 293 threshold.
cap
The cap on concessional contributions for all taxpayers if age_based_cap is
FALSE, or for those below the age threshold otherwise.
cap2
The cap on concessional contributions for those above the age threshold. No
effect if age_based_cap is FALSE.
age_based_cap
Is the cap on concessional contributions age-based?
cap2_age
The age above which cap2 applies.
ecc
(logical) Should an excess concessional contributions charge be calculated? (Not
implemented.)
use_other_contr
Make a (poor) assumption that all 'Other contributions' (MCS_Othr_Contr) are
concessional contributions. This may be a useful upper bound should such con-
tributions be considered important.
scale_contr_match_ato
(logical) Should concessional contributions be inflated to match aggregates in
2013-14? That is, should concessional contributions by multiplied by grattan:::super_contribution_inflator_1314,
which was defined to be:
\[
\frac{\text{Total assessable contributions in SMSF and funds}}{\text{Total contributions in 2013-14 sample file}}
\]
.lambda
Scalar weight applied to concessional contributions. \(\lambda = 0\) means no (ex-
tra) weight. \(\lambda = 1\) means contributions are inflated by the ratio of aggregates
to the sample file's total. For \(R = \text{actual/apparent}\) then the contributions are
scaled by \(1 + \lambda(R - 1)\).
reweight_late_lodgers
(logical) Should WEIGHT be inflated to account for late lodgers?
.mu
Scalar weight for WEIGHT. \(w' = \mu w\) No effect if reweight_late_lodgers
is FALSE.
impute_zero_concess_contr
Should zero concessional contributions be imputed using salary?
.min.Sw.for.SG
The minimum salary required for super guarantee to be imputed.
.SG_rate
The super guarantee rate for imputation.
warn_if_colnames_overwritten
(logical) Issue a warning if the construction of helper columns will overwrite
existing column names in .sample.file.
drop_helpers
(logical) Should columns used in the calculation be dropped before the sample
file is returned?
copyDT
(logical) Should the data table be copy()d? If the action of this data table is
being compared, possibly useful.

Value

A data table comprising the original sample file (.sample.file) with extra superannuation policy-
relevant variables for the policy specified by the function.
**Description**

Australia’s population

**Usage**

```r
 aus_pop_qtr(date_quarter, allow.projections = TRUE)
```

**Arguments**

- `date_quarter`: A character string (YYYY-QQ).
- `allow.projections`: If the date is beyond the ABS’s confirmed data, should a projection be used?

**Value**

The population at `date_quarter`, or at the most recent year in the data if projections are disallowed.

---

**Description**

Australian estimated resident population by age and date

**Usage**

```r
 aus_pop_qtr.Age(
    date = NULL,
    age = NULL,
    tbl = FALSE,
    roll = TRUE,
    roll.beyond = FALSE
)
```
Arguments

- **date**: A vector of dates. If `NULL`, values for all dates are returned in a table. The dates need not be quarters, provided `roll != FALSE`.
- **age**: A vector of (integer) ages from 0 to 100 inclusive. If `NULL`, all ages are returned.
- **tbl**: Should a table be returned? If `FALSE`, a vector is returned.
- **roll**: Should a rolling join be performed?
- **roll.beyond**: Should inputs be allowed to go beyond the limits of data (without a warning)? This is passed to `data.table`'s join, so options other than `TRUE` and `FALSE` are available. See `?data.table`.

Value

A `data.table` or vector with values of the estimated resident population.

Examples

```r
aus_pop_qtr_age(date = as.Date("2016-01-01"), age = 42)
```

---

Description

Adult weekly ordinary-time earnings

Usage

```r
awote(
  Date = NULL,
  fy.year = NULL,
  rollDate = "nearest",
  isMale = NA,
  isAdult = TRUE,
  isOrdinary = TRUE
)
```

Arguments

- **Date, fy.year**: When the AWOTE is desired.
- **rollDate**: How should the `Date` be joined to the source data? Passed to `data.table`.
- **isMale**: (logical, default: NA) `TRUE` for male weekly earnings, `FALSE` for female, NA for the weekly earnings of both sexes.
- **isAdult**: (logical, default: TRUE) Use adult weekly earnings?
- **isOrdinary**: Use ordinary weekly earnings?
Examples

```
awote()  # Current AWOTE
```

---

**bto**

**Beneficiary tax offset**

---

**Description**

Beneficiary tax offset

**Usage**

```
bto(
    benefit_amount,
    fy.year = NULL,
    rate1 = 0.15,
    benefit_threshold = 6000,
    tax_threshold = 37000,
    rate2 = 0.15
)
```

**Arguments**

- `benefit_amount`: The amount of Tax Offsetable benefit received by the taxpayer during the income year.
- `fy.year`: The income year. Not used by default.
- `rate1`: The coefficient in Division 2, section 13(2) of the Income Tax Assessment (1936 Act) Regulation 2015 (the regulations).
- `benefit_threshold`: The amount of benefits above which the offset applies.
- `tax_threshold`: The *threshold at the upper conclusion of the lowest marginal tax rate* in the words of the section 13(3) of the regulations.
- `rate2`: The second coefficient in section 13(3) of the regulations.

**Value**

The beneficiary tax offset.

**WARNING**

This function disagrees with the ATO online calculator.
carers_allowance  Carers allowance

Description
Carers allowance

Usage

```r
carers_allowance(Date = NULL, fy.year = NULL, per = c("year", "fortnight"))
```

Arguments

- `Date`, `fy.year`  The timing of the allowance.
- `per`  Frequency of the payment.

Value

The carer's payment, if eligible.

carer_payment  Carer Payment

Description

Carer payment is available to those who provide constant for a person who has a physical, intellectual, or psychiatric disability. Note that many of the arguments relate to the individual who receives the care (indicated by not starting with 'carer_'). Payment is made to the carer and not to the person receiving the care.

Usage

```r
carer_payment(
  Date = NULL,
  fy.year = NULL,
  carer_fortnightly_income = 0,
  carer_annual_income = carer_fortnightly_income * 26,
  carer_has_partner = FALSE,
  carer_n_dependants = 0L,
  carer_partner_fortnightly_income = 0,
  carer_partner_annual_income = carer_partner_fortnightly_income * 26,
  carer_assets_value = 0,
  carer_is_home_owner = FALSE,
  carer_illness_separated_couple = FALSE,
  dclad_eligible = FALSE,
)```
high_adat = FALSE,
living_at_home = TRUE,
receiving_other_payment = FALSE,
care_receiver_fortnightly_income = 0,
care_receiver_annual_income = care_receiver_fortnightly_income * 26,
care_receiver_asset_value = 0,
partner_fortnightly_income = 0,
partner_annual_income = partner_fortnightly_income * 26,
partner_asset_value = 0,
children_fortnightly_income = 0,
children_annual_income = children_fortnightly_income * 26,
children_asset_value = 0,
parents_fortnightly_income = 0,
parents_annual_income = parents_fortnightly_income * 26,
parents_asset_value = 0
)

Arguments

Date, fy.year  The financial year. Currently only 2015-16 is supported (the most recent survey of income and housing results).
carer_fortnightly_income, carer_annual_income  
Carer’s income for means-testing purposes. Provide one but not both.
carer_has_partner  
(logical, default: FALSE) Does the carer have a partner?
carer_n_dependants  
How many dependants does the carer have? Default is zero.
carer_partner_fortnightly_income, carer_partner_annual_income  
The carer’s partner’s income.
carer_assets_value  
Total value of carer’s household assets.
carer_is_home_owner  
(logical, default: FALSE) Does the carer own their own home?
carer_illness_separated_couple  
Is the couple separated by illness? (Affects the assets test.)
dclad_eligible  
Is the person receiving care a DCLAD (Disability Care Load Assessment) qualifying child as defined in https://guides.dss.gov.au/guide-social-security-law/1/1/q/17?
high_adat  
Does the person receiving care have a high ADAT (Adult Disability Assessment Tool) score as defined in https://guides.dss.gov.au/guide-social-security-law/1/1/a/78?
living_at_home  
Does the person receiving care live at home with their parents?
receiving_other_payment  
Is the care receiver receiving other social security payments?
care_receiver_fortnightly_income  
Care receiver’s fortnightly income
care_receiver_annual_income  
  Care receiver’s annual income

care_receiver_asset_value  
  Care receiver’s asset value

partner_fortnightly_income  
  Care receiver’s partner’s fortnightly income

partner_annual_income  
  Care receiver’s partner’s annual income

partner_asset_value  
  Care receiver’s partner’s asset value

children_fortnightly_income  
  Care receiver’s children’s fortnightly income

children_annual_income  
  Care receiver’s children’s annual income

children_asset_value  
  Care receiver’s children’s asset value

parents_fortnightly_income  
  Care receiver’s parents’ fortnightly income

parents_annual_income  
  Care receiver’s parents’ annual income

parents_asset_value  
  Care receiver’s parents’ asset value

**Author(s)**

Matthew Katzen

---

**CG_population_inflator**

*Forecasting capital gains*

**Description**

Forecasting capital gains

**Usage**

```r
CG_population_inflator(
  x = 1,
  from_fy, to_fy,
  forecast.series = "mean",
  cg.series
)
```

```r
CG_inflator(x = 1, from_fy, to_fy, forecast.series = "mean")
```
child_care_subsidy

Arguments

x To be inflated.
from_fy, to_fy Financial years designating the inflation period.
forecast.series One of "mean", "lower", "upper". What estimator to use in forecasts. "lower" and "upper" give the lower and upper boundaries of the 95% prediction interval.
cg.series (Not implemented.)

Value

For CG_population_inflator, the number of individuals estimated to incur capital gains in fy_year. For CG_inflator, an estimate of the nominal value of (total) capital gains in to_fy relative to the nominal value in from_fy.

Description

Child Care Subsidy paid per child.

Usage

child_care_subsidy(
  family_annual_income = 0,
  activity_level = Inf,
  activity_exemption = FALSE,
  child_age = 3,
  type_of_day_care = c("cbdc", "oshc", "fdc", "ihc"),
  hours_day_care_fortnight = 36,
  cost_hour = 10,
  early_education_program = FALSE,
  cbdc_hourly_cap = 11.77,
  fdc_hourly_cap = 10.9,
  oshc_hourly_cap = 10.29,
  ihc_hourly_cap = 25.48,
  annual_cap_income = 186958,
  annual_cap_subsidy = 10190,
  income_test_bracket_1 = 66958,
  income_test_bracket_2 = 171958,
  income_test_bracket_3 = 251248,
  income_test_bracket_4 = 341248,
  income_test_bracket_5 = 354248,
  taper_1 = 0.85,
  taper_2 = 0.5,
  taper_3 = 0.2,
activity_test_1_brackets = c(0, 8, 16.00001, 48.00001),
activity_test_1_hours = c(0, 36, 72, 100)
)

**Arguments**

- **family_annual_income**
  (numeric) Total income of the family.

- **activity_level**
  (numeric) The total number of activity hours of the parent. Note that if there are two parents the one with the lower activity level will be applied. Common activities include work, leave, and study. A full list can be viewed at [https://guides.dss.gov.au/family-assistance-guide/3/5/2/10](https://guides.dss.gov.au/family-assistance-guide/3/5/2/10).

- **activity_exemption**
  (logical) Whether the parent is exempt from the activity test. Note that in a two parent family both parents must be exempt. A list of exemptions is available at [https://guides.dss.gov.au/family-assistance-guide/3/5/2/10](https://guides.dss.gov.au/family-assistance-guide/3/5/2/10).

- **child_age**
  (numeric) The age of the child in child care.

- **type_of_day_care**
  (character) The type of child care. Acceptable inputs are: "cbdc" Centre Based Day Care, "oshc" Outside School Hours Care, "fdc" Family Day Care, or "ihc" In Home Care. Note that In Home Care can only be claimed once per family.

- **hours_day_care_fortnight**
  (numeric) The number of hours of day care per child per fortnight.

- **cost_hour**
  (numeric) The cost of day care per hour.

- **early_education_program**
  (logical) Whether the child is part of an early education program.

- **cbdc_hourly_cap**, **fdc_hourly_cap**, **oshc_hourly_cap**, **ihc_hourly_cap**
  (numeric) The lower of ‘cost_hour’ or the relevant ‘hourly_cap’ will be used in the calculation of the subsidy.

- **annual_cap_income**
  (numeric) The minimum family income for which the ‘annual_cap_subsidy’ applies from.

- **annual_cap_subsidy**
  (numeric) Amount at which annual subsidies are capped for those who earn more than ‘annual_cap_income’.

- **income_test_bracket_1**, **income_test_bracket_2**, **income_test_bracket_3**, **income_test_bracket_4**, **income_test_bracket_5**

- **taper_1**, **taper_2**, **taper_3**
  (numeric) The proportion of the hourly cap retained. Note that the rate only decreases between each odd bracket.

- **activity_test_1_brackets**
  (numeric vector) The activity levels at which the activity test increases.

- **activity_test_1_hours**
  (numeric vector) The hours corresponding to the step increase in ‘activity_test_1_brackets’.
Value

The annual child care subsidy payable per child.

Examples

```r
close_tax_rate(DT, baseDT, by = "id", ids = NULL)
```
cpi_inflator  

Description

CPI inflator

Usage

cpi_inflator(
  from_nominal_price = 1,
  from_fy = NULL,
  to_fy = NULL,
  adjustment = c("seasonal", "none", "trimmed.mean"),
  useABSConnection = FALSE,
  allow.projection = TRUE,
  accelerate.above = 100000L
)

Arguments

from_nominal_price
  (numeric) the price (or vector of prices) to be inflated

from_fy, to_fy
  (character) a character vector with each element in the form "2012-13" representing the financial years between which the CPI inflator is desired.
  If both from_fy and to_fy are NULL (the default), from_fy is set to the previous financial year and to_fy to the current financial year, with a warning. Setting only one is an error.

adjustment
  What CPI index to use ("none" = raw series, "seasonal", or "trimmed" [mean]).

useABSConnection
  Should the function connect with ABS.Stat via an SDMX connection? If FALSE (the default), a pre-prepared index table is used. This is much faster and more reliable (in terms of errors), though of course relies on the package maintainer to keep the tables up-to-date.
  If the SDMX connection fails, a message is emitted (not a warning) and the function continues as if useABSConnection = FALSE.
  The internal data was updated on 2020-07-02 to 2020-Q1. If using useABSConnection = TRUE, ensure you have rsdmx (>= 0.5-10) up-to-date.

allow.projection
  Should projections beyond the ABS’s data be allowed?

accelerate.above
  An integer setting the threshold for ‘acceleration’. When the maximum length of the arguments exceeds this value, calculate each unique value individually then combine. Set to 100,000 as a rule of thumb beyond which calculation speeds benefit dramatically. Can be set to Inf to disable acceleration.
**Value**

The value of `from_nominal_price` in real (to_fy) dollars.

**Examples**

```r
  cpi_inflator(100, from_fy = "2005-06", to_fy = "2014-15")
```

---

**cpi_inflator_general_date**

*CPI for general dates*

**Description**

CPI for general dates

**Usage**

```r
  cpi_inflator_general_date(from_nominal_price = 1, from_date, to_date, ...)
```

**Arguments**

- `from_nominal_price` (numeric) the nominal prices to be converted to a real price
- `from_date` (character, date-like) the 'date' contemporaneous to `from_nominal_price`. The acceptable forms are 'YYYY', 'YYYY-YY' (financial year), 'YYYY-MM-DD', and 'YYYY-Q[1-4]' (quarters). Note a vector cannot contain a mixture of date forms.
- `to_date` (character, date-like) the date at which the real price is valued (where the nominal price equals the real price). Same forms as for `from_date`
- `...` other arguments passed to `cpi_inflator_quarters`

**Value**

A vector of real prices in to_date dollars.
CPI inflator when dates are nice

Usage

cpi_inflator_quarters(
  from_nominal_price,  
  from_qtr, 
  to_qtr, 
  adjustment = c("seasonal", "trimmed", "none"), 
  useABSConnection = FALSE 
)

Arguments

from_nominal_price
  (numeric) the nominal prices to be converted to a real price

from_qtr
  (date in quarters) the dates contemporaneous to the prices in from_nominal_price. Must be of the form "YYYY-Qq" e.g. "1066-Q2". Q1 = Mar, Q2 = Jun, Q3 = Sep, Q4 = Dec.

to_qtr
  (date in quarters) the date to be inflated to, where nominal price = real price. Must be of the form "YYYY-Qq" e.g. "1066-Q2".

adjustment
  Should there be an adjustment made to the index? Adjustments include 'none' (no adjustment), 'seasonal', or 'trimmed' [referring to trimmed mean]. By default, seasonal.

useABSConnection
  Should the function connect with ABS.Stat via an SDMX connection? By default set to FALSE in which case a pre-prepared index table is used. This is much faster and more reliable (in terms of errors), though of course relies on the package maintainer to keep the tables up-to-date. The internal data was updated on 2020-07-02 to 2020-Q1. If using useABSConnection = TRUE, ensure you have rsdmx (>= 0.5-10) up-to-date.

Value

A vector of real prices.
differentially_uprate_wage

Differential uprating

Description

Apply differential uprating to projections of the Sw_amt variable.

Usage

differentially_uprate_wage(wage = 1, from_fy, to_fy, ...)

Arguments

- **wage**: A numeric vector to be uprated.
- **from_fy**: The financial year contemporaneous to wage, which must be a financial year of an available sample file – in particular, not after 2016-17.
- **to_fy**: The target of the uprating. Passed to `wage_inflator`.
- **...**: Other arguments passed to `wage_inflator`.

Details

See vignette("differential-uprating").

Value

The vector `wage` differentially uprated to `to_fy`.

Author(s)

Hugh Parsonage and William Young

Examples

```r
ws <- c(20e3, 50e3, 100e3)
from <- "2013-14"
to <- "2016-17"
differentially_uprate_wage(ws, from, to)
differentially_uprate_wage(ws, from, to) / wage_inflator(ws, from, to)

# Use a wage series:
if (requireNamespace("taxstats", quietly = TRUE)) {
  library(data.table)
  library(taxstats)
  WageGrowth <- data.table(fy_year = c("2017-18", "2018-19"),
                           r = c(0.0, 0.1))
  Wage201314 <- sample_file_1314["Sw_amt"]
```
disability_pension

Description

Identical to the age_pension except for those under 21.

Usage
disability_pension(
  fortnightly_income = 0,
  annual_income = 26 * fortnightly_income,
  assets_value = 0,
  fy.year = NULL,
  Date = NULL,
  age = 21L,
  has_partner = FALSE,
  n_dependants = 0L,
  lives_at_home = FALSE,
  independent = FALSE,
  per = c("year", "fortnight"),
  ...)

Arguments

fortnightly_income, annual_income
  Income for the means test

assets_value
  Value of assets for the assets test.

fy.year, Date
  Either the financial year and Date in which the pension is paid. Only ‘fy.year = "2015-16"’ is implemented.

age
  Age of the individual, only relevant for those under 21.

has_partner
  (logical, default: FALSE) Is the individual a member of a couple?

n_dependants
  Integer number of dependent children.

lives_at_home
  (logical, default: FALSE) Does the individual live at home with their parents? Only relevant if age < 21.

independent
  (logical, default: FALSE) Is the person independent? Only relevant if age < 21.
energy_supplement

per One of "fortnight", "year" to return either the fortnightly pension or the annual amount.

... Other arguments passed to age_pension.

energy_supplement Energy supplement

Description

The energy supplement (ES) is a supplementary payment that commenced on 20 September 2014. It was previously known as the clean energy supplement (CES). It is a fixed nominal amount; the supplement is neither indexed nor increased each year. There is no means testing.

Usage

energy_supplement(
  qualifying_payment,
  has_partner = FALSE,
  n_dependants = 0L,
  age = 21,
  lives_at_home = FALSE,
  independent = FALSE,
  is.jspceoalfcoahodeoc = FALSE,
  long_term = FALSE,
  per = c("year", "fortnight", "quarter")
)

Arguments

  qualifying_payment
    A character vector designating the payment type the individual is entitled to. Valid strings are
    pension All pensions and bereavement allowance
    seniors health card Commonwealth Seniors Health Card
    disability pension Disability support pension (over 21)
    allowance All allowances not elsewhere described, viz. Newstart allowance,
      Widow allowance, Partner allowance, Sickness allowance
    parenting Parenting payments
    youth allowance Youth allowance (but not receiving youth disability supplement)
    youth disability Youth allowance but also receiving youth disability supplement
    austudy Austudy recipients
  has_partner (logical, default: FALSE) Does the individual have a partner? For persons with partners but separated due to the partner’s illness or imprisonment, this may be true or false depending on the eligibility of the qualifying payment.
n_depends How many dependants does the individual have? Default is zero.
age The age of the individual.
lives_at_home (logical, default: FALSE) Does the individual live at home?
independent (logical, default: FALSE) For persons under 21, is the person ‘independent’?
isjspacealfofcoahodeoc Is the recipient a single job seeker principal carer, either of large family or foster child/ren, or who is a home or distance educator of child/ren?long_term Is the individual a long-term welfare recipient?per Dictates whether the result is per year, per fortnight, or per quarter. By default, yearly payments are returned, with a message. Payments are generally made each fortnight though recipients can elect to have them paid quarterly.

Value
The energy supplement for each individual. Arguments are recycled, but only if length-one.

Source

Description
Family tax benefit

Usage
family_tax_benefit(
  .data = NULL,
  id_hh = NULL,
  id = NULL,
  age = NULL,
  income = NULL,
  in_secondary_school = NULL,
  single_parent = NULL,
  other_allowance_benefit_or_pension = NULL,
  maintenance_income = NULL,
  maintenance_children = NULL,
  income_test_ftbA_1_bound = 51027,
  income_test_ftbA_2_bound = 94316,
  income_test_ftbB_bound = 5402,
  taper_ftbA_1 = 0.2,
taper_ftbA_2 = 0.3,
taper_ftbB = 0.2,
per = "year",
copy = TRUE
)

Arguments

.data data.table input. Each row is an individual. Columns must be have the same names
id_hh household identifier, used to group households to determine eligibility and number of children
id individual identifier
age numeric: age of each id
income numeric: income of each id
in_secondary_school logical column: does id attend secondary school?
single_parent logical column: is id (a parent) single?
other_allowance_benefit_or_pension logical column: does the individual receive a pension, benefit, or labour market program payment such as Youth Allowance?
maintenance_income numeric: the amount of maintenance income the individual receives for the care of a child/children from a previous relationship
maintenance_children integer: the number of children in the care of id for whom id receives maintenance
income_test_ftbA_1_bound Lower bound for which reduction in FTB A max payment occurs at rate taper_ftbA_1.
income_test_ftbA_2_bound Lower bound for which reduction in FTB A base payment occurs at rate taper_ftbA_1.
income_test_ftbB_bound Lower bound for which reduction in FTB B payment occurs at rate taper_ftbB.
taper_ftbA_1 The amount at which ftb A max payment is reduced for each dollar earned above income_test_ftbA_1_bound.
taper_ftbA_2 The amount at which ftb A base payment is reduced for each dollar earned above income_test_ftbA_2_bound.
taper_ftbB The amount at which ftb B payment is reduced for each dollar earned above income_test_ftbB_bound.
per How often the payment will be made. At present, payments can only be annually.
copy (logical, default: TRUE) Should a copy of .data be made before the calculation? If FALSE, intermediate values will be assigned by reference to .data (if not NULL).
**generic_inflator**

**Author(s)**
Matthew Katzen

---

**gdp**

*Gross Domestic Product, Australia*

---

**Description**

Gross domestic product, at contemporaneous prices (called ‘current prices’ by the ABS).

**Usage**

- `gdp_qtr(date, roll = "nearest")`
- `gdp_fy(fy_year)`

**Arguments**

- **date**: A Date vector or character coercible thereto.
- **roll**: Passed to `data.table` when joining.
- **fy_year**: Character vector of financial years.

**Value**

For `gdp_qtr`, the quarterly GDP for the quarter date nearest (or otherwise using `roll`). For `gdp_fy` the sum over the quarters in the financial year provided. If `fy_year` would provide incomplete data (i.e. only sum three or fewer quarters), a warning is issued. Dates or `fy_year` outside the available data is neither a warning nor an error, but NA.

**Source**

Australian Bureau of Statistics, Catalogue 5206.0. Series A2304350J.

---

**generic_inflator**

*Generic inflator*

**Description**

Used to inflate variables in the sample file when there is no clear existing index. Note this is an unexported function: it is not available to the end-user.
Usage

generic_inflator(
    vars,
    h,
    fy.year.of.sample.file = "2012-13",
    nonzero = FALSE,
    estimator = "mean",
    pred_interval = 80
)

Arguments

vars A character vector of those variables within .sample_file for which forecasts are desired.
h An integer, how many years ahead should the inflator be targeted.
fy.year.of.sample.file A string representing the financial year of .sample_file.
nonzero Should the forecast be taken on all values, or just nonzero values?
estimator What forecast element should be used: the point estimate ("mean"), or the upper or lower endpoint of a prediction interval?
pred_interval If estimator is upper or lower, what prediction interval are these the end points of?

Value

A data table of two columns: variable containing vars and inflator equal to the inflator to be applied to that variable to inflate it ahead h years.

---

gni Gross National Income, Australia

Description

Gross national income, at contemporaneous prices (called ’current prices’ by the ABS).

Usage

gni_qtr(date, roll = "nearest")

gni_fy(fy_year)

Arguments

date A Date vector or character coercible there to.
roll Passed to data.table when joining.
fy_year Character vector of financial years.
Value
For gni_qtr, the quarterly GNI for the nearest quarter date. For gni_fy the sum over the quarters in the financial year provided. If fy_year would provide incomplete data (i.e. only sum three or fewer quarters), a warning is issued. Dates or fy_year outside the available data is neither a warning nor an error, but NA.

Source
Australian Bureau of Statistics, Catalogue 5206.0. Series A2304354T.

---

| IncomeTax | IncomeTax |
---|---|

Description
Calculates the ordinary tax payable given income and tax thresholds and rates. Basic, designed for performance.

Arguments
- **x**: Taxable income.
- **thresholds**: Lower brackets of the tax tables.
- **rates**: Marginal rates

---

| income_tax | Income tax payable |
---|---|

Description
Income tax payable

Usage
```r
income_tax(
  income,
  fy_year = NULL,
  age = NULL,
  family_status = "individual",
  n_dependants = 0L,
  .dots.ATO = NULL,
  return.mode = c("numeric", "integer"),
  allow.forecasts = FALSE,
  .debug = FALSE
)
```
Arguments

- **income**: The individual assessable income.
- **fy.year**: The financial year in which the income was earned. Tax years 2000-01 to 2018-19 are supported, as well as the tax year 2019-20, for convenience. If `fy.year` is not given, the current financial year is used by default.
- **age**: The individual’s age. Ignored if `.dots.ATO` is provided (and contains an age variable such as `age_range` or `Birth_year`).
- **family_status**: For Medicare and SAPTO purposes.
- **n_dependants**: An integer for the number of children of the taxpayer (for the purposes of the Medicare levy).
- **.dots.ATO**: A data.frame that contains additional information about the individual’s circumstances, with columns the same as in the ATO sample files. Age variables in `.dots.ATO` take precedence over `age` and providing both is a warning.
- **return.mode**: The mode (numeric or integer) of the returned vector.
- **allow.forecasts**: should dates beyond 2019-20 be permitted? Currently, not permitted.
- **.debug**: (logical, default: FALSE) If TRUE, returns a data.table containing the components of income tax calculated. (This argument and its result is liable to change in future versions, possibly without notice.)

Details

The function is inflexible by design. It is designed to return the correct tax payable in a year, not to model the tax payable under different tax settings. (Use `model_income_tax` for that purpose.)

The function aims to produce the personal income tax payable for the inputs given in the tax year `fy.year`. The function is specified to produce the most accurate calculation of personal income tax given the variables in the ATO’s 2% sample files. However, many components are absent from these files, while other components could not be computed reliably.

For the 2018-19 tax year, the function calculates

**tax on ordinary taxable income** The tax as specified in Schedule 7 of the *Income Tax Rates Act 1986* (Cth).

**Medicare levy** See `medicare_levy` for details.

**LITO** See `lito` for details.

**SAPTO** See `sapto`. For years preceding the introduction of SAPTO, the maximum offset is assumed to apply to those above age 65 (since the sample files only provide 5-year age groups).

**SBTO** See `small_business_tax_offset` for details.

**Historical levies** The flood levy and the temporary budget repair levy.

Notably, when used with a 2% sample file, the function will not be able to correctly account for different tax rates and offsets among taxpayers with dependants since the sample files (as of 2015-16) do not have this information.
Value

The total personal income tax payable.

Author(s)

Tim Cameron, Brendan Coates, Matthew Katzen, Hugh Parsonage, William Young

Examples

```r
## Income tax payable on a taxable income of 50,000
## for the 2013-14 tax year
income_tax(50e3, "2013-14")

## Calculate tax for each lodger in the 2013-14 sample file.
if (requireNamespace("taxstats", quietly = TRUE)) {
  library(data.table)
  library(taxstats)

  s1314 <- as.data.table(sample_file_1314)
  s1314[, tax := income_tax(Taxable_Income, "2013-14", .dots.ATO = s1314)]
}
```

income_tax_sapto  

Income tax payable as a function of SAPTO

Description

Income tax payable as a function of SAPTO

Usage

```r
income_tax_sapto(
  income,
  fy.year = NULL,
  age = 42,
  family_status = "individual",
  n_dependants = 0L,
  return.mode = c("numeric", "integer"),
  .dots.ATO = NULL,
  allow.forecasts = FALSE,
  sapto.eligible,
  medicare.sapto.eligible,
  new_sapto_tbl = NULL
)
```
Arguments

income
The individual assessable income.

fy.year
The financial year in which the income was earned. Only tax years from 2000-01 to 2016-17 are available. If fy.year is not given, the current financial year is used by default.

age
The individual’s age.

family_status
For Medicare and SAPTO purposes.

n_dependants
An integer for the number of children of the taxpayer (for the purposes of the Medicare levy).

return.mode
The mode (numeric or integer) of the returned vector.

.dots.ATO
A data.frame that contains additional information about the individual’s circumstances, with columns the same as in the ATO sample files. If .dots.ATO is a data.table, I recommend you enclose it with copy().

allow.forecasts
should dates beyond 2016-17 be permitted? Currently, not permitted.

sapto.eligible
Specify explicitly the eligibility for SAPTO. If missing, defaults to ages over 65.

medicare.sapto.eligible
Specify explicitly the eligibility for SAPTO with respect to the Medicare levy for low-income earners. If missing, defaults to ages over 65.

new_sapto_tbl
If not NULL, supplied to new_sapto. Otherwise, fy.year is passed to sapto.

Details

Used to cost simple changes to SAPTO.

inflator

Description

Inflate using a general index

Usage

inflator(
    x = 1,
    from,
    to,
    inflator_table,
    index.col = "Index",
    time.col = "Time",
    roll = NULL,
    max.length = NULL
)
Arguments

- **x**: The vector to be inflated.
- **from**: The contemporaneous time of x.
- **to**: The target time (in units of the `inflator_table`) to which x is to be inflated.
- **inflator_table**: A `data.table` having columns `index.col` and `time.col`.
- **index.col**: The column in `inflator_table` containing the index used for inflation.
- **time.col**: The column in `inflator_table` by which times are mapped.
- **roll**: If `NULL`, inflation is calculated only on exact matches in `inflator_table`. Otherwise, uses a rolling join. See `data.table::data.table`.
- **max.length**: (Internal use only). If not `NULL`, the maximum length of x, from, and to known in advance. May be provided to improve the performance if known.

Value

A vector of inflated values. For example, `inflator_table = grattan::cpi_seasonal_adjustment`, `index.col = "obsValue", time.col = "obsTime",` gives the CPI inflator.

install_taxstats

Install 'taxstats' files

Description

The taxstats packages provide the sample files as released by the ATO. These packages are used for testing, but are not available through CRAN as they are too large.

Usage

```r
install_taxstats(pkg = c("taxstats"), ...)
```

Arguments

- **pkg**: The package to install such as "taxstats" or "taxstats1516".
- **...**: Arguments passed to `install.packages`.
inverse_average_rate  

*Inverse average tax rate*

**Description**

Inverse average tax rate

**Usage**

inverse_average_rate(average_rate, ..., .max = 100000000)

**Arguments**

- `average_rate`: The average tax rate \( \frac{\text{tax}}{\text{income}} \)
- `...`: Parameters passed to `income_tax`.
- `.max`: The maximum income to test before ending the search. (Used only to prevent infinite loops.)

**Value**

The minimum income at which the average tax rate exceeds `average_rate`.

**Examples**

inverse_average_rate(0.2, fy.year = "2014-15")

---

inverse_income  

*Inverse income tax functions*

**Description**

Inverse income tax functions

**Usage**

inverse_income(
  tax,
  fy.year = "2012-13",
  zero.tax.income = c("maximum", "zero", "uniform", numeric(1)),
  ...)

---
Arguments

- **tax**: The tax payable.
- **fy.year**: The relevant financial year.
- **zero.tax.income**: A character vector, ("maximum", "zero", "uniform", numeric(1)) Given that many incomes map to zero taxes, the `income_tax` function is not invertible there. As a consequence, the inverse function's value must be specified for tax = 0. "maximum" returns the maximum integer income one can have with a zero tax liability; "zero" returns zero for any tax of zero; "uniform" provides a random integer from zero to the maximum income with a zero tax. The value can also be specified explicitly.

... Other arguments passed to `income_tax`. If `tax` or `fy.year` are vectors, these should be named vectors.

Details

This function has an error of $2.

Value

The approximate taxable income given the tax payable for the financial year. See Details.

---

**is.fy**

*Convenience functions for dealing with financial years*

Description

From grattan v1.7.1.4, these are reexports from the *fy-package*.

Arguments

- **yr_ending**: An integer representing a year.
- **fy.yr**: A string suspected to be a financial year.
- **date**: A string or date for which the financial year is desired. Note that `yr2fy` does not check its argument is an integer.
- **assume1901_2100**: For `yr2fy`, assume that `yr_ending` is between 1901 and 2100, for performance. By default, set to `getOption("grattan.assume1901_2100",TRUE)`.

Details

The following forms are permitted: 2012–13, 201213, 2012 13, only. However, the 2012–13 form is preferred and will improve performance.
Value

For is.fy, a logical, whether its argument is a financial year. The following forms are allowed: 2012-13, 2012-13, 2012 13, only. For fy.year, yr2fy, and date2fy, the financial year. For the inverses, a numeric corresponding to the year.

fy.year is a deprecated alias for yr2fy, the latter is slightly more efficient, as well as more declarative.

fy2yr converts a financial year to the year ending: fy2yr("2016-17") returns 2017. yr2fy is the inverse: yr2fy(fy2yr("2016-17")) == "2016-17".

date2fy converts a financial year to the 30 June of the financial year ending.

Examples

is.fy("2012-13")
is.fy("2012-14")
yr2fy(2012)
fy2yr("2015-16")
date2fy("2014-08-09")

lf_inflator

Labour force inflators

Description

Labour force inflators

Usage

lf_inflator_fy(
    labour_force = 1,
    from_fy = NULL,
    to_fy = NULL,
    useABSConnection = FALSE,
    allow.projection = TRUE,
    use.month = 1L,
    forecast.series = c("mean", "upper", "lower", "custom"),
    forecast.level = 95,
    lf.series = NULL,
    .lf_indices = NULL,
    accelerate.above = 100000L
)

lf_inflator(
    labour_force = 1,
    from_date = "2013-06-30",
    to_date,


useABSConnection = FALSE

Arguments

labour_force A numeric vector.

from_fy, to_fy (character) a character vector with each element in the form "2012-13" representing the financial years between which the labour force inflator is desired. If both from_fy and to_fy are NULL (the default), from_fy is set to the previous financial year and to_fy to the current financial year, with a warning. Setting only one is an error.

useABSConnection Should the function connect with ABS.Stat via an SDMX connection? If FALSE (the default), a pre-prepared index table is used. This is much faster and more reliable (in terms of errors), though of course relies on the package maintainer to keep the tables up-to-date. If the SDMX connection fails, a message is emitted (not a warning) and the function continues as if useABSConnection = FALSE. The internal data was updated on 2020-07-02 to 2020-05-01.

allow.projection Logical. Should projections be allowed?

use.month An integer (corresponding to the output of data.table::month) representing the month of the series used for the inflation.

forecast.series Whether to use the forecast mean, or the upper or lower boundaries of the prediction intervals.

forecast.level The prediction interval to be used if forecast.series is upper or lower.

lf.series If forecast.series = 'custom', a data.table with two variables, fy_year and r. The variable fy_year consists of all financial years between the last financial year in the (known) labour force series and to_fy inclusive. The variable r consists of rates of labour force growth assumed in each fy_year, which must be 1 in the first year (to connect with the original labour force series).

.lf_indices (Internal use only.) A data.table sent directly to inflator without any checks.

accelerate.above An integer setting the threshold for 'acceleration'. When the maximum length of the arguments exceeds this value, calculate each unique value individually then combine. Set to 100,000 as a rule of thumb beyond which calculation speeds benefit dramatically. Can be set to Inf to disable acceleration.

from_date The date of labour_force.

to_date Dates as a character vector.

Details

lf_inflator is used on dates. The underlying data series is available every month.
Value

The relative labour force between to_date and for_date or to_fy and from_fy, multiplied by labour_force.

Author(s)

Tim Cameron, Matthew Katzen, and Hugh Parsonage

Source

ABS Cat 6202.0

Examples

lf_inflator_fy(labour_force = 1, from_fy = "2012-13", to_fy = "2013-14")

library(data.table)
# Custom 1% growth over 2018-19 -> 2019-20
lf_inflator_fy(from_fy = "2018-19",
    to_fy = "2019-20",
    forecast.series = "custom",
    lf.series = data.table(fy_year = c("2018-19", "2019-20"),
                          r = c(0, 0.01)))

## Not run:
lf_inflator(labour_force = 1, from_date = "2013-06-30", to_date = "2014-06-30")

## End(Not run)

lito

Low Income Tax Offset

Description

The Low Income Tax Offset (LITO) is a non-refundable tax offset to reduce ordinary personal income tax for low-income earners.

Usage

.lito(input)

lito(income, max_lito = 445, lito_taper = 0.015, min_bracket = 37000)

Arguments

input A keyed data.table containing the financial year and the input of every observation for which the LITO should be calculated. The input must have the following structure. The structure will not be checked.
max_super_contr_base

fy_year  The financial year the LITO parameters should be obtained. This must be the key of the data.table.
income  The Taxable Income of the individual.
ordering  An integer sequence from 1 to nrow(input) which will be the order of the output.
income  Income of taxpayer
max_lito  The maximum LITO available.
lito_taper  The amount by which LITO should be shaded out or reduced for every additional dollar of taxable income.
min_bracket  The income at which the lito_taper applies.

Value

For .lito, the a numeric vector equal to the offset for each income and each financial year in input. For lito, a numeric vector equal to the offset for each income given the LITO parameters.

max_super_contr_base  Maximum superannuation contribution base

Description

Data maximum super contribution base.

Usage

max_super_contr_base

Format

A data frame with 25 rows and 2 variables:

fy_year  The financial year.
max_sv_per_qtr  Maximum superannuation guarantee per quarter.

Source

ATO.
Medicare Levy

Medicare Levy

Medicare levy in C++

Description

Medicare levy. Experimental function in C++, equivalent to medicare_levy.

Arguments

income, SpouseIncome, isFamily, NDependants, lowerThreshold, upperThreshold, lowerFamilyThreshold, upperFamilyThreshold, lowerUpForEachChild

As in medicare_levy.

rate, taper The parameters for the specific year or hypothetical requested.

Details

For yr > 2018, the 2017-18 values are used.

medicare_levy

Description

The (actual) amount payable for the Medicare levy.

Usage

medicare_levy(
  income,
  fy.year = "2013-14",
  Spouse_income = 0,
  sapto.eligible = FALSE,
  sato = NULL,
  pto = NULL,
  family_status = "individual",
  n_dependants = 0,
  .checks = TRUE
)

Arguments

income The taxable income. A vector of numeric values.
fy.year The financial year. A character vector satisfying is.fy.
Spouse_income The spouse’s adjusted income.
sapto.eligible (logical) Is the taxpayer eligible for SAPTO? See Details.
Details

The Medicare levy for individuals is imposed by the *Medicare Levy Act 1986* (Cth). The function only calculates the levy for individuals (not trusts). It includes the *s 7 Levy in cases of small incomes*, including the differences for those eligible for *sapto. s 8 Amount of levy—person who has spouse or dependants* (though the number of dependants is not a variable in the sample files).

The function does not include the Medicare levy surcharge; it assumes that all persons (who would potentially be liable for it) avoided it.#'

The Seniors and Pensioners Tax Offset was formed in 2012-13 as an amalgam of the Senior Australians Tax Offset and the Pensions Tax Offset. Medicare rates before 2012-13 were different based on these offsets. For most taxpayers, eligibility would be based on whether your age is over the pension age (currently 65). If sato and pto are NULL, sapto.eligible stands for eligibility for the sato and not pto. If sato or pto are not NULL for such years, only sato is currently considered. Supplying pto independently is currently a warning.


Value

The Medicare levy payable for that taxpayer.

---

**model_child_care_subsidy**

*Model Child Care Subsidy*

---

Description

The child care subsidy if thresholds and rates are changed. (See *child_care_subsidy*.)

Usage

```r
model_child_care_subsidy(
    sample_file,
    Cbdc_hourly_cap = NULL,
    Fdc_hourly_cap = NULL,
    Oshc_hourly_cap = NULL,
    Ihc_hourly_cap = NULL,
    Annual_cap_income = NULL,
)```
model_child_care_subsidy

Annual_cap_subsidy = NULL,
Income_test_bracket_1 = NULL,
Income_test_bracket_2 = NULL,
Income_test_bracket_3 = NULL,
Income_test_bracket_4 = NULL,
Income_test_bracket_5 = NULL,
Taper_1 = NULL,
Taper_2 = NULL,
Taper_3 = NULL,
Activity_test_1_brackets = NULL,
Activity_test_1_hours = NULL,
calc_baseline_ccs = TRUE,
return. = c("sample_file", "new_ccs", "sample_file.int")
)

Arguments

sample_file A sample file having the same variables as the data.frame in the example.

Cbdc_hourly_cap, Fdc_hourly_cap, Oshc_hourly_cap, Ihc_hourly_cap
  (numeric) The lower of ‘cost_hour’ or the relevant ‘hourly_cap’ will be used in
  the calculation of the subsidy.

Annual_cap_income
  (numeric) The minimum family income for which the ‘Annual_cap_subsidy’
  applies from.

Annual_cap_subsidy
  (numeric) Amount at which annual subsidies are capped for those who earn
  more than ‘Annual_cap_income’.

Income_test_bracket_1, Income_test_bracket_2, Income_test_bracket_3, Income_test_bracket_4, Income_test_bracket_5
  (numeric) The steps at which income test 1 changes rates. Note the strange struc-

Taper_1, Taper_2, Taper_3
  (numeric) The proportion of the hourly cap retained. Note that the rate only
decreases between each odd bracket.

Activity_test_1_brackets
  (numeric vector) The activity levels at which the activity test increases.

Activity_test_1_hours
  (numeric vector) The hours corresponding to the step increase in ‘activity_test_1_brackets’.

calc_baseline_ccs
  (logical, default: TRUE) Should the current child care subsidy be included as a
  column in the result?

return. What should the function return? One of subsidy, sample_file, or sample_file.int.
  If subsidy, the subsidy received under the settings; if sample_file, the sample_file,
  but with variables subsidy and possibly new_subsidy; if sample_file.int, same
  as sample_file but new_subsidy is coerced to integer.
**Description**

The income tax payable if tax settings are changed.

**Usage**

```r
model_income_tax(
  sample_file,
  baseline_fy,
  n_dependants = 0L,
  elasticity_of_taxable_income = NULL,
  ordinary_tax_thresholds = NULL,
  ordinary_tax_rates = NULL,
  medicare_levy_taper = NULL,
  medicare_levy_rate = NULL,
  medicare_levy_lower_threshold = NULL,
  medicare_levy_upper_threshold = NULL,
  medicare_levy_lower_sapto_threshold = NULL,
  medicare_levy_upper_sapto_threshold = NULL,
  medicare_levy_lower_family_threshold = NULL,
  medicare_levy_upper_family_threshold = NULL,
  medicare_levy_lower_family_sapto_threshold = NULL,
  medicare_levy_upper_family_sapto_threshold = NULL,
  medicare_levy_lower_up_for_each_child = NULL,
  lito_max_offset = NULL,
  lito_taper = NULL,
  lito_min_bracket = NULL,
  lito_multi = NULL,
  Budget2018_lamington = FALSE,
  Budget2019_lamington = NA,
  Budget2018_lito_202223 = FALSE,
  Budget2018_watr = FALSE,
  Budget2019_watr = FALSE,
  sapto_eligible = NULL,
  sapto_max_offset = NULL,
  sapto_lower_threshold = NULL,
  sapto_taper = NULL,
  sapto_max_offset_married = NULL,
  sapto_lower_threshold_married = NULL,
  sapto_taper_married = NULL,
  sbto_discount = NULL,
  cgt_discount_rate = NULL,
  calc_baseline_tax = TRUE,
  return. = c("sample_file", "tax", "sample_file.int"),
```
model_income_tax

clear_tax_cols = TRUE,
warn_upper_thresholds = TRUE,
.debug = FALSE
)

Arguments

sample_file A sample file having at least as many variables as the 2012-13 sample file.
baseline_fy If a parameter is not selected, the parameter's value in this tax year is used.
    Must be a valid tax year and one for which income_tax has been programmed.
n_dependants The number of dependants for each entry in sample_file.
elasticity_of_taxable_income Either NULL (the default), or a numeric vector the same length of sample_file (or length-1) providing the elasticity of taxable income for each observation in sample_file;
    \[
    \frac{\Delta z}{z} = \frac{\Delta \tau}{(1 - \tau)}
    \]
    where \(z\) is taxable income and \(\tau\) is tax payable.
    For example, if, for a given taxpayer, the tax settings would otherwise result in a 2% decrease of disposable income under the tax settings to be modelled, and elasticity_of_taxable_income is set to 0.1, the Taxable_Income is reduced by 0.2% before the tax rates are applied.
    If NULL, an elasticity of 0 is used.
ordinary_tax_thresholds A numeric vector specifying the lower bounds of the brackets for "ordinary tax" as defined by the Regulations. The first element should be zero if there is a tax-free threshold.
ordinary_tax_rates The marginal rates of ordinary tax. The first element should be zero if there is a tax-free threshold. Since the temporary budget repair levy was imposed on a discrete tax bracket when it applied, it is not included in this function.
medicare_levy_taper The taper that applies between the _lower and _upper thresholds.
medicare_levy_rate The ordinary rate of the Medicare levy for taxable incomes above medicare_levy_upper_threshold.
medicare_levy_lower_threshold Minimum taxable income at which the Medicare levy will be applied.
medicare_levy_upper_threshold Minimum taxable income at which the Medicare levy will be applied at the full Medicare levy rate (2% in 2015-16). Between this threshold and the medicare_levy_lower_threshold, a tapered rate applies, starting from zero and climbing to medicare_levy_rate.
medicare_levy_lower_sapto_threshold, medicare_levy_upper_sapto_threshold The equivalent values for SAPTO-eligible individuals (not families).
medicare_levy_lower_family_threshold, medicare_levy_upper_family_threshold The equivalent values for families.
medicare_levy_lower_family_sapto_threshold, medicare_levy_upper_family_sapto_threshold
The equivalent values for SAPTO-eligible individuals in a family.

medicare_levy_lower_up_for_each_child
The amount to add to the _family_thresholds for each dependant child.

lito_max_offset
The maximum offset available for low incomes.

lito_taper
The taper to apply beyond lito_min_bracket.

lito_min_bracket
The taxable income at which the value of the offset starts to reduce (from lito_max_offset).

lito_multi
A list of two components, named x and y, giving the value of a replacement for lito at specified points, which will be linked by a piecewise linear curve between the points specified. For example, to mimic LITO in 2015-16 (when the offset was $445 for incomes below $37,000, and afterwards tapered off to $66,667), one would use lito_multi = list(x = c(-Inf, 37e3, 200e3/3, Inf), y = c(445, 445, 0, 0)). The reason the argument ends with multi is that it is intended to extend the original parameters of LITO so that multiple kinks (including ones of positive and negative gradients) can be modelled.

Budget2018_lamington
logical; default is ‘FALSE’. If set to ‘TRUE’, calculates the amount that taxpayers would be entitled to under the Low and Middle Income Tax Offset as contained in the 2018 Budget.

Budget2019_lamington
logical. If set to ‘TRUE’, calculates the amount that taxpayers would be entitled to under the Low and Middle Income Tax Offset as amended by the 2019 Budget.

The default, ‘NA’, means ‘TRUE’ if ‘baseline_fy’ is set to a year where the LMITO is in effect, viz. 2017-18, 2018-19, 2019-20 or 2020-21, and ‘FALSE’ otherwise.

Budget2018_lito_202223
The LITO proposed to start in 2022-23 as announced in the 2018 Budget.

Budget2018_watr
logical; default is ‘FALSE’. If set to ‘TRUE’, calculates the "Working Australian Tax Refund" as proposed in the Labor Opposition Leader’s Budget Reply Speech 2018.

Budget2019_watr
logical; default is ‘FALSE’. If set to ‘TRUE’, calculates the "Working Australian Tax Refund" as revised in the Labor Opposition Leader’s Budget Reply Speech 2019.

sapto_eligible
Whether or not each taxpayer in sample_file is eligible for SAPTO. If NULL, the default, then eligibility is determined by age_range in sample_file; i.e., if age_range <= 1 then the taxpayer is assumed to be eligible for SAPTO.

sapto_max_offset
The maximum offset available through SAPTO.

sapto_lower_threshold
The threshold at which SAPTO begins to reduce (from sapto_max_offset).

sapto_taper
The taper rate beyond sapto_lower_threshold.
Model changes to the contributions cap, Division 293 threshold and related modelling. Note: defaults are relevant to pre-2017 for compatibility.
Usage

```r
model_new_caps_and_div293(
  .sample.file,
  fy.year,
  new_cap = 30000,
  new_cap2 = 35000,
  new_age_based_cap = TRUE,
  new_cap2_age = 49,
  new_ecc = FALSE,
  new_contr_tax = "15%",
  new_div293_threshold = 300000,
  use_other_contr = FALSE,
  scale_contr_match_ato = FALSE,
  .lambda = 0,
  reweight_late_lodgers = TRUE,
  .mu = 1.05,
  impute_zero_concess_contr = TRUE,
  .min.Sw.for.SG = 450 * 12,
  .SG_rate = 0.0925,
  prv_cap = 30000,
  prv_cap2 = 35000,
  prv_age_based_cap = TRUE,
  prv_cap2_age = 49,
  prv_ecc = FALSE,
  prv_div293_threshold = 300000
)
```

`n_affected_from_new_cap_and_div293(..., adverse_only = TRUE)`

`revenue_from_new_cap_and_div293(...)`

Arguments

- `.sample.file` A data.table whose variables include those in `taxstats::sample_file_1314`.
- `fy.year` The financial year tax scales.
- `new_cap` The **proposed** cap on concessional contributions for all taxpayers if `age_based_cap` is FALSE, or for those below the age threshold otherwise.
- `new_cap2` The **proposed** cap on concessional contributions for those above the age threshold. No effect if `age_based_cap` is FALSE.
- `new_age_based_cap` Is the **proposed** cap on concessional contributions age-based?
- `new_cap2_age` The age above which `new_cap2` applies.
- `new_ecc` (logical) Should an excess concessional contributions charge be calculated? (Not implemented.)
- `new_contr_tax` A string to determine the contributions tax.
- `new_div293_threshold` The **proposed** Division 293 threshold.
Should MCS_Othr_CONTR be used to calculate Division 293 liabilities?

(.logical) Should concessional contributions be inflated to match aggregates in 2013-14? That is, should the concessional contributions be multiplied by the internal constant grattan::super_contribution_inflator_1314, which was defined to be:

\[
\frac{\text{Total assessable contributions in SMSF and funds}}{\text{Total contributions in 2013-14 sample file}}
\]

\(\lambda\) Scalar weight applied to concessional contributions. \(\lambda = 0\) means no (extra) weight. \(\lambda = 1\) means contributions are inflated by the ratio of aggregates to the sample file’s total. For \(R = \text{actual/apparent}\) then the contributions are scaled by \(1 + \lambda(R - 1)\).

(reweight_late_lodgers) Should WEIGHT be inflated to account for late lodgers?

\(\mu\) Scalar weight for WEIGHT. \((w' = \mu w)\) No effect if reweight_late_lodgers is FALSE.

Should zero concessional contributions be imputed using salary?

The minimum salary required for super guarantee to be imputed.

The super guarantee rate for imputation.

The comparator cap on concessional contributions for all taxpayers if age_based_cap is FALSE, or for those below the age threshold otherwise.

The comparator cap on concessional contributions for those above the age threshold. No effect if age_based_cap is FALSE.

Is the comparator cap on concessional contributions age-based?

The age above which new_cap2 applies.

(.logical) Should an excess concessional contributions charge be calculated? (Not implemented.)

The comparator Division 293 threshold.

Passed to model_new_caps_and_div293.

Count only individuals who are adversely affected by the change.

Value

For model_new_caps_and_div293, a data.frame, comprising the variables in .sample.file, the superannuation variables generated by apply_super_caps_and_div293, and two variables, prv_revenue and new_revenue, which give the tax (income tax, super tax, and division 293 tax) payable by that taxpayer in the comparator scenario and the proposed scenario, respectively.

For affected_from_new_cap_and_div293, the number of individuals affected by the proposed changes.

For revenue_from_new_cap_and_div293, the extra revenue expected from the proposed changes.
Examples

```r
if (requireNamespace("taxstats", quietly = TRUE)) {
  library(data.table)
  s1314 <- taxstats::sample_file_1314
  s1314[, WEIGHT := 50L]
  revenue_from_new_cap_and_div293(s1314, new_cap = 12e3, "2016-17")
  revenue_from_new_cap_and_div293(s1314, new_contr_tax = "mr - 15\%", "2016-17")
}
```

---

**model_rent_assistance**  
**Model Rent Assistance**

**Description**

Model Rent Assistance

**Usage**

```r
model_rent_assistance(
  sample_file,
  baseline_fy = NULL,
  baseline_Date = NULL,
  Per = "fortnight",
  .Prop_rent_paid_by_RA = NULL,
  Max_rate = NULL,
  Min_rent = NULL,
  calc_baseline_ra = TRUE,
  return. = c("sample_file", "new_ra", "sample_file.int")
)
```

**Arguments**

- **sample_file**: A sample file having the same variables as the data.frame in the example.
- **baseline_fy**, **baseline_Date**
  - (character) The financial year/date over which the baseline rent assistance is to be calculated. Only one can be provided.
- **Per**
  - Specifies the timeframe in which payments will be made. Can either take value "fortnight" or "annual".
- **.Prop_rent_paid_by_RA**
  - The proportion of the rent above the minimum threshold paid by rent assistance.
- **Max_rate**
  - If not NULL, a numeric vector indicating for each individual the maximum rent assistance payable.
- **Min_rent**
  - If not NULL, a numeric vector indicating for each individual the minimum fortnightly rent above which rent assistance is payable. max_rate and min_rent
newstart_allowance

newstart_allowance

Description

Newstart allowance

Usage

newstart_allowance(
  fortnightly_income = 0,
  annual_income = 0,
  has_partner = FALSE,
  partner_pensioner = FALSE,
  n_dependants = 0,
  nine_months = FALSE,
  isjspceoalfofcoahodeoc = FALSE,
  principal_carer = FALSE,
  fortnightly_partner_income = 0,
  annual_partner_income = 0,
  age = 22,
  fy.year = "2015-16",
  assets_value = 0,
  homeowner = FALSE,
)

Examples

library(data.table)
sample <-
  CJ(rent = 1:500,
    n_dependants = 0:3,
    has_partner = 0:1 > 0,
    is_homeowner = 0:1 > 0,
    lives_in_sharehouse = 0:1 > 0)
model_rent_assistance(sample,
  baseline_fy = "2018-19",
  .Prop_rent_paid_by_RA = 0.75,
  Max_rate = 500,
  Min_rent = 100)

newstart_allowance

newstart_allowance

Description

Newstart allowance

Usage

newstart_allowance(
  fortnightly_income = 0,
  annual_income = 0,
  has_partner = FALSE,
  partner_pensioner = FALSE,
  n_dependants = 0,
  nine_months = FALSE,
  isjspceoalfofcoahodeoc = FALSE,
  principal_carer = FALSE,
  fortnightly_partner_income = 0,
  annual_partner_income = 0,
  age = 22,
  fy.year = "2015-16",
  assets_value = 0,
  homeowner = FALSE,
)

Examples

library(data.table)
sample <-
  CJ(rent = 1:500,
    n_dependants = 0:3,
    has_partner = 0:1 > 0,
    is_homeowner = 0:1 > 0,
    lives_in_sharehouse = 0:1 > 0)
model_rent_assistance(sample,
  baseline_fy = "2018-19",
  .Prop_rent_paid_by_RA = 0.75,
  Max_rate = 500,
  Min_rent = 100)

newstart_allowance

newstart_allowance

Description

Newstart allowance

Usage

newstart_allowance(
  fortnightly_income = 0,
  annual_income = 0,
  has_partner = FALSE,
  partner_pensioner = FALSE,
  n_dependants = 0,
  nine_months = FALSE,
  isjspceoalfofcoahodeoc = FALSE,
  principal_carer = FALSE,
  fortnightly_partner_income = 0,
  annual_partner_income = 0,
  age = 22,
  fy.year = "2015-16",
  assets_value = 0,
  homeowner = FALSE,
)
lower = 102,
upper = 252,
taper_lower = 0.5,
taper_upper = 0.6,
taper_principal_carer = 0.4,
per = c("year", "fortnight")
)

Arguments

fortnightly_income
'Ordinary income' received fortnightly within the meaning of s. 1068-G1 of the

annual_income 'Ordinary income' received annually.

has_partner Does the individual have a partner?

partner_pensioner Does the partner receive a pension?

n_dependants How many dependent children does the individual have?

nine_months If the person is over 60 years old, have they been receiving payments for over 9
continuous months?

principal_carer
Is the individual the parent with most of the day-to-day care of child/ren, or who is a home or distance educator of child/ren?

fortnightly_partner_income
Partner's 'Ordinary income' received fortnightly.

annual_partner_income
Partner's Ordinary income' received annually.

age The individual's age.

fy.year Financial year. Default is "2015-16".

assets_value Total value of household assets. Details can be found at https://www.servicesaustralia.gov.au/individuals/topics/asset-types/30621.

homeowner Is the individual a homeowner?

lower Lower bound for which reduction in payment occurs at rate taper_lower (taper_principal_carer
for principal carers).

upper Upper bound for which reduction in payment occurs at rate taper_lower. Lower
bound for which reduction in payment occurs at rate taper_upper. Note that
for principal carers there is no upper bound.

taper_lower The amount at which the payment is reduced for each dollar earned between the
lower and upper bounds for non-principal carers.

taper_upper The amount at which the payment is reduced for each dollar earned above the
upper bound for non-principal carers.
**taper_principal_carer**

The amount at which the payment is reduced for each dollar earned above the lower bound for principal carers.

**per**

Specifies the timeframe in which payments will be made. Can either take value "fortnight" or "annual".

**Source**


---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_income_tax</td>
<td>New income tax payable Income tax payable with new tax brackets, tax rates etc</td>
</tr>
</tbody>
</table>

**Usage**

```
new_income_tax(income, new_tax_tbl)
```

**Arguments**

- **income**
  A vector of taxable incomes.

- **new_tax_tbl**
  A data.table with columns `lower_bracket` and `marginal_rate` for the new brackets and marginal rates.

**Value**

The income according to the new parameters.

---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_medicare_levy</td>
<td>New medicare levy</td>
</tr>
</tbody>
</table>

**Usage**

```
new_medicare_levy(parameter_table)
```

**Description**

Use a different way to calculate medicare levy.
Arguments

parameter_table

A data.table containing
switches The value in a row specifying which different medicare function is
to apply.
lower_threshold What is the lower medicare threshold, below which no medi-
care levy is applied, above which a tapering rate applies.
taper What is the taper above lower_threshold.
rate The medicare levy applicable above the medicare thresholds.
lower_up_for_each_child How much the lower threshold should increase
with each n_dependants.
lower_family_threshold The threshold as applied to families (i.e. couples)

Value

A function similar to medicare_levy.

---

**new_sapto**

*SAPTO with user-defined thresholds*

---

Description

SAPTO with user-defined thresholds

Usage

```
new_sapto(
  rebate_income,
  new_sapto_tbl,
  sapto.eligible = TRUE,
  Spouse_income = 0,
  fill = 0,
  family_status = "single"
)
```

Arguments

rebate_income The rebate income of the individual.
new_sapto_tbl Having the same columns as grattan:::sapto_tbl, keyed on family_status.
sapto.eligible Is the individual eligible for sapto?
Spouse_income Spouse income whose unutilized SAPTO may be added to the current taxpayer.
  Must match family_status; i.e. can only be nonzero when family_status != "single".
fill If SAPTO was not applicable, what value should be used?
family_status Family status of the individual.
Description

Financial functions from Excel. These functions are equivalent to the Excel functions of the same name (in uppercase).

Usage

\[
\text{npv(rate, values)} \\
\text{irr(x, start = 0.1)} \\
\text{fv(rate, nper, pmt, pv = 0, type = 0)} \\
\text{pv(rate, nper, pmt, fv = 0, type = 0)} \\
\text{pmt(rate, nper, pv, fv = 0, type = 0)}
\]

Arguments

- **rate**: Discount or interest rate.
- **values**: Income stream.
- **x**: Cash flow.
- **start**: Initial guess to start the iterative process.
- **nper**: Number of periods
- **pmt**: Payments.
- **pv**: Present value.
- **type**: Factor.
- **fv**: Future value.

Author(s)

Enrique Garcia M.  <egarcia@egm.as>
Karsten W.  <k.weinert@gmx.net>

Examples

\[
\text{npv(0.07, c(1, 2))} \\
\text{irr(x = c(1, -1), start = 0.1)} \\
\text{fv(0.04, 7, 1, pv = 0.0, type = 0)} \\
\text{pv(rate = 0.08, nper = 7, pmt = 1, fv = 0.0, type = 0)} \\
\text{pmt(rate = 0.025, nper = 7, pv = 0, fv = 0.0, type = 0)}
\]
Offset | General offset in C++
---|---

**Description**

Calculate the offset given a threshold, a maximum offset, and a taper.

**Arguments**

- **x**
  A vector of incomes etc.
- **y**
  The maximum offset available; the offset when x is zero.
- **a**
  The maximum value of x at which the maximum offset is available.
- **m**
  The taper rate (the negative slope).

**pension_supplement** | Pension Supplement
---|---

**Description**

The Pension Supplement gets added to the max rate of payment before income reduction tests are applied. Note that if the individual is part of a couple, the rate indicates the payment amount per person, not for the couple. Can be claimed by those receiving Age Pension, Carer Payment, Wife Pension, Widow B Pension, Bereavement Allowance, or Disability Support Pension (except if under 21 and have no children). Can also be claimed if over age pension age and are receiving ABSTUDY, Austudy, Parenting Payment, Partner Allowance, Special Benefit, or Widow Allowance. Can still claim the basic amount if single, under age pension age, and receive the Parenting Payment.

**Usage**

```r
pension_supplement(
  has_partner = FALSE,
  age = 70,
  n_dependants = 0,
  parenting_payment = FALSE,
  Date = NULL,
  fy.year = NULL,
  qualifying_payment = "age_pension",
  per = c("year", "fortnight", "quarter"),
  overseas_absence = FALSE,
  separated_couple = FALSE
)
```
Arguments

- **has_partner**: Does the individual have a partner?
- **age**: The individual’s age. Default is 70 years.
- **n_dependants**: How many dependent children does the individual have?
- **parenting_payment**: Is the individual receiving parenting payment?
- **Date**: Date. Default is "2016/03/01" if fy.year is not present.
- **fy.year**: Financial year. Default is "2015-16" if Date is not present.
- **qualifying_payment**: What is the payment that the supplement is being applied to?
- **per**: How often the payment will be made. Default is to return the annual payment, with a message.
- **overseas_absence**: Will the individual be living outside of Australia for more than 6 weeks of the upcoming year?
- **separated_couple**: Is the individual part of an illness separated couple, respite care couple, or partner imprisoned?

Author(s)

Matthew Katzen

---

**pmax3**

*Threeway parallel maximum*

**Description**

Returns the parallel maximum of three

**Arguments**

- **x, y, z**: Numeric vectors of identical lengths.

**Value**

The parallel maximum of the vectors.
Description

A faster \texttt{pmax()}.

Arguments

\begin{itemize}
  \item \texttt{x}: A numeric vector.
  \item \texttt{a}: A single numeric value.
\end{itemize}

Value

The parallel maximum of the input values. \texttt{pmax0(x)} is shorthand for \texttt{pmaxC(x,0)}, i.e. convert negative values in \texttt{x} to 0.

Note

This function will always be faster than \texttt{pmax(x,a)} when \texttt{a} is a single value, but can be slower than \texttt{pmax.int(x,a)} when \texttt{x} is short. Use this function when comparing a numeric vector with a single value.
**pminC**  
*Parallel maximum*

---

**Description**  
A faster `pmin()`.

**Arguments**
- **x**: A numeric vector.
- **a**: A single numeric value.

**Value**  
The parallel minimum of the input values. The 0 versions are shortcuts for `a = 0`.

**Note**  
This function will always be faster than `pmin(x, a)` when `a` is a single value, but can be slower than `pmin.int(x, a)` when `x` is short. Use this function when comparing a numeric vector with a single value.

---

**pminV**  
*Parallel maximum*

---

**Description**  
A faster `pmin()`.

**Arguments**
- **x**: A numeric vector.
- **y**: A numeric vector, the same length as `x`.

**Value**
The parallel maximum of the input values.
### progressivity

**Description**

Compute the progressivity

**Usage**

```r
progressivity(income, tax, measure = c("Reynolds-Smolensky", "Kakwani"))
```

**Arguments**

- **income**: Pre-tax income.
- **tax**: Tax paid.
- **measure**: Currently, only "Reynolds-Smolensky" progressivity is calculated:

\[
G_Y - G_Z
\]

where \(G_Y\) is the Gini coefficient of income and \(G_X\) is the Gini coefficient of post-tax income.

**Value**

The progressivity measure. Positive for progressive tax systems, and higher the value the more progressive the system.

**Examples**

```r
I <- c(10e3, 20e3, 50e3, 100e3, 150e3)
progressivity(I, 0.3 * I) # zero
progressivity(I, income_tax(I, "2017-18"))
```

---

**prohibit_length0_vectors**

**Description**

Tests whether any vectors have zero length.

**Usage**

```r
prohibit_length0_vectors(.)
```
Arguments
    ... A list of vectors

Value
    An error message if any of the vectors ... have zero length.

---

prohibit_unequal_length_vectors

Prohibit unequal length vectors

Description
    Tests whether all vectors have the same length.

Usage
    prohibit_unequal_length_vectors(...)

Arguments
    ... Vectors to test.

Value
    An error message unless all of ... have the same length in which case NULL, invisibly.

---

project

Simple projections of the annual 2% samples of Australian Taxation Office tax returns.

Description
    Simple projections of the annual 2% samples of Australian Taxation Office tax returns.

Usage
    project(
        sample_file,
        h = 0L,
        fy.year.of.sample.file = NULL,
        WEIGHT = 50L,
        excl_vars = NULL,
        forecast.dots = list(estimator = "mean", pred_interval = 80),
        wage.series = NULL,
lf.series = NULL,
use_age_pop_forecast = FALSE,
.recalculate.inflators = NA,
.copyDT = TRUE,
check_fy_sample_file = TRUE,
differentially_uprate_Sw = NA,
r_super_balance = 1.05
)

Arguments

sample_file A data.table matching a 2% sample file from the ATO. See package taxstats for an example.

h An integer. How many years should the sample file be projected?

fy.year.of.sample.file The financial year of sample_file. If NULL, the default, the number is inferred from the number of rows of sample_file to be one of 2012-13, 2013-14, 2014-15, 2015-16, or 2016-17.

WEIGHT The sample weight for the sample file. (So a 2% file has WEIGHT = 50.)
excl_vars A character vector of column names in sample_file that should not be inflated. Columns not present in the 2013-14 sample file are not inflated and nor are the columns Ind, Gender, age_range, Occ_code, Partner_status, Region, Lodgment_method, and PHI_Ind.

forecast.dots A list containing parameters to be passed to generic_inflator.
wage.series See wage_inflator. Note that the Sw_amt will uprated by differently_uprate_wage (if requested).

lf.series See lf_inflator_fy.
use_age_pop_forecast Should the inflation of the number of taxpayers be moderated by the number of resident persons born in a certain year? If TRUE, younger ages will grow at a slightly higher rate beyond 2018 than older ages.

.recalculate.inflators (logical, default: NA). Should generic_inflator() or CG_inflator be called to project the other variables? Adds time. Default NA means TRUE if the pre-calculated inflators are available, FALSE otherwise.

.copyDT (logical, default: TRUE) Should a copy() of sample_file be made? If set to FALSE, will update sample_file in place, which may be necessary when memory is constrained, but is dangerous as it modifies the original data and its projection. (So if you run the same code twice you may end up with a projection 2h years ahead, not h years.)

check_fy_sample_file (logical, default: TRUE) Should fy.year.of.sample_file be checked against sample_file? By default, TRUE, an error is raised if the base is not 2012-13, 2013-14, 2014-15, 2015-16, 2016-17, or 2017-18, and a warning is raised if the number of rows in sample_file is different to the known number of rows in the sample files.
differentially_uprate_Sw
(logical, default: NA) Should the salary and wage column (Sw_amt) be differentially uprated using (differentially_uprate_wage)? Default of NA means use differential uprating is used when fy.year.of.sample.file <= "2016-17". It is known that the Treasury stopped using differential uprating by 2019. Selecting TRUE for fy.year.of.sample.file > "2016-17" is an error as the precalculated values are not available.

r_super_balance
The factor to inflate super balances by (annualized). Set to 1.05 for backwards compatibility. The annual superannuation bulletin of June 2019 from APRA reported 7.3% growth of funds with more than fund members over the previous 5 years and 7.9% growth over the previous ten years.

Details
Currently components of taxable income are individually inflated based on their historical trends in the ATO sample files, with the exception of:

inflated using differently_uprate_wage. Sw_amt
inflated using wage_inflator Allow_ben_amt,ETP_txbl_amt,Rptbl_Empr_spr_cont_amt,Non_emp_spr_amt, MCS_Emplr_Contr,MCS_Prsn1_Contr,MCS_Othr_Contr
inflated using cpi_inflator WRE_car_amt,WRE_trvl_amt,WRE_uniform_amt,WRE_self_amt, WRE_other_amt
inflated by lf_inflator_fy WEIGHT
inflated by CG_inflator Net_CG_amt,Tot_CY_CG_amt

Superannuation balances are inflated by a fixed rate of 5% p.a.
We recommend you use sample_file_1213 over sample_file_1314, unless you need the superannuation variables, as the latter suggests lower-than-recorded tax collections. However, more recent data is of course preferable.

Value
A sample file with the same number of rows as sample_file but with inflated values as a forecast for the sample file in to_fy. If WEIGHT is not already a column of sample_file, it will be added and its sum will be the predicted number of taxpayers in to_fy.

Examples

# install_taxstats()
if (requireNamespace("taxstats", quietly = TRUE) && requireNamespace("data.table", quietly = TRUE)) {
library(taxstats)
library(data.table)
sample_file <- copy(sample_file_1314)
sample_file_1617 <- project(sample_file, h = 3L, # to "2016-17"
fy.year.of.sample.file = "2013-14")
}
**project_to**

*Simple projections of the annual 2% samples of Australian Taxation Office tax returns.*

**Description**

Simple projections of the annual 2% samples of Australian Taxation Office tax returns.

**Usage**

```r
project_to(sample_file, to_fy, fy.year.of.sample.file = NULL, ...)
```

**Arguments**

- `sample_file` A `data.table` matching a 2% sample file from the ATO. See package `taxstats` for an example.
- `to_fy` A string like "1066-67" representing the financial year for which forecasts of the sample file are desired.
- `fy.year.of.sample.file` The financial year of `sample_file`. See `project` for the default.
- `...` Other arguments passed to `project`.

**Value**

A sample file with the same number of rows as `sample_file` but with inflated values as a forecast for the sample file in `to_fy`. If `WEIGHT` is not already a column of `sample_file`, it will be added and its sum will be the predicted number of taxpayers in `to_fy`.

---

**rebate_income**

*Rebate income*

**Description**

Rebate income

**Usage**

```r
rebate_income(  
  Taxable_Income,  
  Rptbl_Empr_spr_cont_amt = 0,  
  All_deductible_super_contr = 0,  
  Net_fincl_invstmt_lss_amt = 0,  
  Net_rent_amt = 0,  
  Rep_frng_ben_amt = 0)
```


Arguments

- **Taxable_Income**: the taxable income
- **Rptbl_Empr_spr_cont_amt**: The reportable employer superannuation contributions amount
- **All_deductible_super_contr**: deductible personal superannuation contributions
- **Net_fincl_investmt_lss_amt**: Net financial investment loss
- **Net_rent_amt**: (for Rental deductions)
- **Rep_frng_ben_amt**: Reportable fringe-benefits

Source


---

**rent_assistance**  
**Rent assistance**

Description

The rent assistance to each individual payable by financial year.

Usage

```R
rent_assistance(
  fortnightly_rent = Inf,
  per = "fortnight",
  fy.year = NULL,
  Date = NULL,
  n_dependants = 0L,
  has_partner = FALSE,
  .prop_rent_paid_by_RA = 0.75,
  max_rate = NULL,
  min_rent = NULL,
  sharers_provision_applies = FALSE,
  is_homeowner = FALSE,
  lives_in_sharehouse = FALSE
)
```
Arguments

`fortnightly_rent`

The fortnightly rent paid by each individual. By default, infinity, so the maximum rent assistance is returned by default, since rent assistance is capped at a maximum rate. Note the criteria for board and lodging which can be found at [https://guides.dss.gov.au/guide-social-security-law/3/8/1/70](https://guides.dss.gov.au/guide-social-security-law/3/8/1/70)

`per`

Specifies the timeframe in which payments will be made. Can either take value "fortnight" or "annual".

`fy.year`

(character) The financial year over which rent assistance is to be calculated. When left as NULL, defaults to the user's financial year, unless `max_rate` and `min_rent` are both set. If `fy.year` is set, the annual payment is provided.

`Date`

(Date vector or coercible to such) An alternative to `fy.year`. If both `fy.year` and `Date` are provided, `fy.year` is ignored, with a warning. If `Date` is used, the fortnightly rent assistance is provided.

`n_dependants`

(integer) Number of dependent children. By default, 0L, so no children.

`has_partner`

(logical) Is each individual married? By default, FALSE.

`.prop_rent_paid_by_RA`

The proportion of the rent above the minimum threshold paid by rent assistance. Since it so happens that this value is constant over the period, it is set here rather than being added to the internal table.

`max_rate`

If not NULL, a numeric vector indicating for each individual the maximum rent assistance payable.

`min_rent`

If not NULL, a numeric vector indicating for each individual the minimum fortnightly rent above which rent assistance is payable. `max_rate` and `min_rent` must not be used when `fy.year` is set.

`sharers_provision_applies`


`is_homeowner`

(logical, default: FALSE) Does the individual own their own home?

`lives_in_sharehouse`

(logical, default: FALSE) Does the individual live in a sharehouse?

Value

If `fy.year` is used, the annual rent assistance payable for each individual; if `Date` is used, the *fortnightly* rent assistance payable. If the arguments cannot be recycled safely, the function errors.

Examples

```r
# current annual rent assistance
rent_assistance()

# current fortnightly payment
rent_assistance(Date = Sys.Date())
```
# zero since no rent
rent_assistance(0, Date = "2016-01-02")

# Rent assistance is payable at 75c for every dollar over min rent
rent_assistance(101, max_rate = 500, min_rent = 100)
rent_assistance(500, max_rate = 500, min_rent = 100)

---

**require_taxstats**  
*Attach a 'taxstats' package*

**Description**  
Used in lieu of simply `library(taxstats)` to handle cases where it is not installed, but should not be installed to the user’s default library (as during CRAN checks).

**Usage**  
```r
require_taxstats()

require_taxstats1516()
```

**Value**  
TRUE, invisibly, for success. Used for its side-effect: attaching the taxstats package.

---

**residential_property_prices**  
*Residential property prices in Australia*

**Description**  
Residential property prices indexes for the capital cities of Australia, and a weighted average for the whole country. Last updated 2018-07-06.

**Usage**  
```r
residential_property_prices
```

**Format**  
A data.table of three columns and 522 observations:

- **Date**  
  Date of the index

- **City**  
  Capital city (or Australia (weighted average))

- **Residential_property_price_index**  
  An index (100 = 2011-12-01) measuring the price change in all residential dwellings.
**Source**


---

**revenue_foregone**

Revenue foregone from a modelled sample file

---

**Description**

Revenue foregone from a modelled sample file

**Usage**

```r
revenue_foregone(dt, revenue_positive = TRUE, digits = NULL)
```

**Arguments**

- **dt**: A data.table from `model_income_tax`
- **revenue_positive**: If TRUE, the default, tax increase (revenue) is positive and tax cuts are negative.
- **digits**: If not NULL, affects the print method of the value.

---

**sapto**

Seniors and Pensioner Tax Offset

---

**Description**

Seniors and Pensioner Tax Offset

**Usage**

```r
sapto(
  rebate_income,
  fy.year,
  fill = 0,
  sapto.eligible = TRUE,
  Spouse_income = 0,
  family_status = "single",
  .check = TRUE
)
```
Arguments

- rebate_income: The rebate income of the individual.
- fy.year: The financial year in which sapto is to be calculated.
- fill: If SAPTO was not applicable, what value should be used?
- sapto.eligible: Is the individual eligible for sapto?
- Spouse_income: Spouse income whose unutilized SAPTO may be added to the current taxpayer. Must match family_status; i.e. can only be nonzero when family_status != "single".
- family_status: Family status of the individual.
- .check: Run checks for consistency of values. For example, ensuring no single individuals have positive Spouse_income.

Description

SAPTO done in Rcpp

Usage

```r
sapto_rcpp(
  RebateIncome,
  MaxOffset,
  LowerThreshold,
  TaperRate,
  SaptoEligible,
  SpouseIncome,
  IsMarried
)
```

Arguments

RebateIncome, MaxOffset, LowerThreshold, TaperRate, SaptoEligible, SpouseIncome, IsMarried

Arguments as in `sapto`.
Description

Length-one version of SAPTO in C++.

Usage

```c++
sapto_rcpp_singleton(
    rebate_income,
    max_offset,
    lower_threshold,
    taper_rate,
    sapto_eligible,
    Spouse_income,
    is_married
)
```

Arguments

rebate_income, max_offset, lower_threshold, taper_rate, sapto_eligible, Spouse_income, is_married

As in sapto.

Description

Fast way to calculate SAPTO for multiple people when the year is known in advance. Speed is by cheating and entering in the year’s parameters literally.

Arguments

RebateIncome, IsMarried, SpouseIncome

As in sapto.
Small Business Tax Offset

Description

Small Business Tax Offset

Usage

```r
call(small_business_tax_offset(taxable_income, basic_income_tax_liability, .dots.ATO = NULL, aggregated_turnover = NULL, total_net_small_business_income = NULL, fy_year = NULL, tax_discount = NULL))
```

Arguments

- **taxable_income**: Individual’s assessable income.
- **basic_income_tax_liability**: Tax liability (in dollars) according to the method in the box in s 4.10(3) of the *Income Tax Assessment Act 1997* (Cth). In general, `basic_income_tax_liability` is the ordinary tax minus offsets. In particular, it does not include levies (such as the Medicare levy or the Temporary Budget Repair Levy).

\[
\text{Income Tax} = \text{Taxable income} \times \text{Rate} - \text{Tax offsets}
\]

For example, in 2015-16, an individual with an assessable income of $100,000 had a basic tax liability of approximately $25,000.

- **.dots.ATO**: A data.table of tax returns. If provided, it must contain the variables `Total_PP_BE_amt`, `Total_PP_BI_amt`, `Total_NPP_BE_amt`, `Total_NPP_BI_amt`. If both `.dots.ATO` and either `aggregated_turnover` or `total_net_small_business_income` are provided, `.dots.ATO` takes precedence, with a warning.

If `.dots.ATO` contains the variable `Tot_net_small_business_inc`, it is used instead of the income variables.

- **aggregated_turnover**: A numeric vector the same length as `taxable_income`. Only used to determine whether or not the offset is applicable; that is, the offset only applies if aggregated turnover is less than $2M.

Aggregated turnover of a taxpayer is the sum of the following:

- the taxpayer’s annual turnover for the income year,
student_repayment

- the annual turnover of any entity connected with the taxpayer’s, for that part of the income year that the entity is connected with the taxpayer’s
- the annual turnover of any entity that is an affiliate of the taxpayer, for that part of the income year that the entity is affiliated with the taxpayer’s
- When you calculate aggregated turnover for an income year, do not include either:
  - the annual turnover of other entities for any period of time that the entities are either not connected with the taxpayer or are not the taxpayer’s affiliate, or
  - amounts resulting from any dealings between these entities for that part of the income year that the entity is connected or affiliated with the taxpayer.


total_net_small_business_income
Total net business income within the meaning of the Act. For most taxpayers, this is simply any net income from a business they own (or their share of net income from a business in which they have an interest). The only difference being in the calculation of the net business income of some minors (vide Division 6AA of Part III of the Act).

fy_year
The financial year for which the small business tax offset is to apply.

tax_discount
If you do not wish to use the legislated discount rate from a particular fy_year, you can specify it via tax_discount. If both are provided, tax_discount prevails, with a warning.

Source

student_repayment HELP / HECS repayment amounts

Description
HELP / HECS repayment amounts

Usage
student_repayment(repayment_income, fy.year, debt)
Arguments

- **repayment_income**: The repayment income of the individual, equal to Taxable Income + Total net investment loss (incl Net rental loss) + reportable fringe benefits amounts + Reportable super contributions + exempt foreign income
- **fy.year**: The financial year repayment_income was earned.
- **debt**: The amount of student debt held.

Details

The student repayments for fy.year = '2018-19' assume the measures in Budget 2017 will pass.

Value

The repayment amount.

Author(s)

Ittima Cherastidtham and Hugh Parsonage

Source


Examples

```r
student_repayment(50e3, "2013-14", debt = 10e3)
# 0 since below the threshold

student_repayment(60e3, "2013-14", debt = 10e3)
# above the threshold

student_repayment(60e3, "2013-14", debt = 0)
# above the threshold, but no debt
```

unemployment_benefit  
*Unemployment benefit*

Description

Calculates the unemployment benefit (Newstart Allowance) payable for individuals in the specified financial year(s), given each individual’s income and assets, and whether they are married, have children, or own their own home.
Usage

```r
unemployment_benefit(
    income = 0,
    assets = 0,
    fy.year = NULL,
    Date = NULL,
    has_partner = FALSE,
    has_dependant = FALSE,
    is_home_owner = FALSE
)
```

Arguments

- **income**: Numeric vector of fortnightly income for the income test.
- **assets**: Numeric vector of the value of assets. By default, `income` and `assets` are both zero, thus returning the maximum benefit payable.
- **fy.year**: A character vector of valid financial years between "2000-01" and "2020-21" specifying which financial year the allowance is to be calculated.
- **Date**: (Date vector or coercible to such). An alternative to `fy.year` to specify the period over which the allowance is calculated.
- **has_partner**: (logical vector, default: `FALSE`) Does the individual have a partner?
- **has_dependant**: (logical vector, default: `FALSE`) Does the individual have any dependant children?
- **is_home_owner**: (logical vector, default: `FALSE`) Does the individual own their own home?

Details

The income test for long-term employed persons above 60 happens to be the same as that for singles with dependants, so calculating the benefit payable for such individuals can be performed by setting `has_partner = FALSE, has_dependant = TRUE`.

Value

The fortnightly unemployment benefit payable for each entry. The function is vectorized over its arguments, with any length-1 argument recycled. (Other vector recycling is not supported and will result in an error.)

---

validate_date  Verifying validity of dates

Description

Many functions expect Dates. Determining that they are validly entered is often quite computationally costly, relative to the core calculations. These internal functions provide mechanisms to check validity quickly, while still providing clear, accurate error messages.
validate_per

Usage

validate_date(date_to_verify, from = NULL, to = NULL, deparsed = "Date")

Arguments

date_to_verify (character) A user-provided value, purporting to be character vector of dates.
from, to Indicating the range of years valid for date_to_verify. Default set to -Inf and
Inf respectively (i.e. there is no bound)
deparsed The name of variable to appear in error messages.

Value

date_to_verify as a Date object, provided it can be converted to a Date and all elements are within
the bounds from and to.

validate_per Valid per

Description

Checks whether a valid input of ‘per’ is used and outputs the amount which yearly payments are
divided by to get the desired rate.

Usage

validate_per(per, missing_per, .fortnights_per_yr = 26)

Arguments

per How often are payments made? Can only take values 'year', 'fortnight', or
'month'.
missing_per Is ‘per’ missing in the outer function? If so the default for that function will be
used. Essentially, you should always pass missing(per) to this argument.
.fortnights_per_yr What is the ratio of the fortnightly payment amount to the yearly payment
amount? By default, 26. (Some payments expect 26; others expect 364/14.)

Details

For examples, see rent_assistance function code.
wage_inflator

Inflation using the Wage Price Index.

Description

Predicts the inflation of hourly rates of pay, between two financial years.

Usage

wage_inflator(
  wage = 1,
  from_fy = NULL,
  to_fy = NULL,
  useABSConnection = FALSE,
  allow.projection = TRUE,
  forecast.series = c("mean", "upper", "lower", "custom"),
  forecast.level = 95,
  wage.series = NULL,
  accelerate.above = 100000L
)

Arguments

wage

The amount to be inflated (1 by default).

from_fy, to_fy

(character) a character vector with each element in the form "2012-13" representing the financial years between which the CPI inflator is desired. If both from_fy and to_fy are NULL (the default), from_fy is set to the previous financial year and to_fy to the current financial year, with a warning. Setting only one is an error.

useABSConnection

Should the function connect with ABS.Stat via an SDMX connection? If FALSE (the default), a pre-prepared index table is used. This is much faster and more

Examples

## Not run:
# Typical use-case
# attach(asNamespace("grattan"))
z <- function(per = "year") 52 / validate_per(per, missing(per))
z() # message
z(per = "year")          # same, no message
z(per = "fortnight")    # in fortights
z(per = "sidfh")        # error

## End(Not run)
reliable (in terms of errors), though of course relies on the package maintainer to keep the tables up-to-date.

If the SDMX connection fails, a message is emitted (not a warning) and the function continues as if useABSConnection = FALSE.

The internal data was updated on 2020-07-02 to 2020-Q1.

allow.projection

If set to TRUE the forecast package is used to project forward, if required.

forecast.series

Whether to use the forecast mean, or the upper or lower boundaries of the prediction intervals. A fourth option custom allows manual forecasts to be set.

forecast.level

The prediction interval to be used if forecast.series is upper or lower.

wage.series

If forecast.series = 'custom', how future years should be inflated. The future wage series can be provided in two ways: (1) a single value, to be the assumed rate of wage inflation in years beyond the known series, or (2) a data.table with two variables, fy_year and r. If (2), the variable fy_year must be a vector of all financial years after the last financial year in the (known) wage series and the latest to_fy inclusive. The variable r consists of rates of wage growth assumed in each fy_year.

accelerate.above

An integer setting the threshold for 'acceleration’. When the maximum length of the arguments exceeds this value, calculate each unique value individually then combine. Set to 100,000 as a rule of thumb beyond which calculation speeds benefit dramatically. Can be set to Inf to disable acceleration.

Value

The wage inflation between the two years.

Examples

# Wage inflation
wage_inflator(from_fy = "2013-14", to_fy = "2014-15")

# Custom wage inflation
wage_inflator(from_fy = "2016-17",
              to_fy = "2017-18",
              forecast.series = "custom",
              wage.series = 0.05)
Usage

```r
youth_allowance(
  fortnightly_income = 0,
  annual_income = 0,
  fy.year = NULL,
  include_ES = TRUE,
  age = 18L,
  eligible_if_over22 = FALSE,
  has_partner = FALSE,
  lives_at_home = FALSE,
  n_dependants = 0L,
  isjspacecoalfcoahodeoc = FALSE,
  is_student = TRUE,
  per = c("fortnight", "year"),
  max_rate = NULL,
  es = NULL,
  taper1 = NULL,
  taper2 = NULL,
  FT_YA_student_lower = NULL,
  FT_YA_student_upper = NULL,
  FT_YA_jobseeker_lower = NULL,
  FT_YA_jobseeker_upper = NULL,
  partner_fortnightly_income = 0,
  partner_is_pensioner = FALSE,
  partner_taper = 0.6
)
```

Arguments

- **fortnightly_income, annual_income**
  Individual's income. Default is zero. You may provided both; providing both when the ratio is not 26 is an error.

- **fy.year**
  Financial year. Default is current financial year.

- **include_ES**
  (logical, default: TRUE) If FALSE do not include the energy supplement.

- **age**
  The individual’s age. Default is 18 years. If type double will be coerced to integer via truncation (i.e. 17.9 becomes 17).

- **eligible_if_over22**
  To be eligible for Youth Allowance while over 22, recipients must either commence full-time study or an Australian apprenticeship having been in receipt of an income support payment for at least 6 out of the last 9 months since turning 22, or study an approved course in English where English is not their first language.

- **has_partner**
  Does the individual have a partner?

- **lives_at_home**
  Does the individual live at home with their parents?

- **n_dependants**
  How many dependant children does the individual have?
is_jspcoalfcoahodeoc
Is the recipient a single job seeker principal carer, either of large family or foster child/ren, or who is a home or distance educator of child/ren?

is_student
Is the individual a student? Note that apprentices are considered students.

per
How often the payment will be made. Default is fortnightly. At present payments can only be fortnightly.

max_rate
If not NULL, a length-1 double representing the maximum *fortnightly* rate for youth allowance.

es
If not NULL, a length-1 double as the energy supplement.

taper1
The amount at which the payment is reduced for each dollar earned between the lower and upper bounds.

taper2
The amount at which the payment is reduced for each dollar earned above the upper bound.

FT_YA_student_lower
Student and apprentice lower bound for which reduction in payment occurs at rate taper1.

FT_YA_student_upper
Student and apprentice upper bound for which reduction in payment occurs at rate taper1. Student and apprentice lower bound for which reduction in payment occurs at rate taper2.

FT_YA_jobseeker_lower
Jobseeker lower bound for which reduction in payment occurs at rate taper1

FT_YA_jobseeker_upper
Jobseeker upper bound for which reduction in payment occurs at rate taper1. Student and apprentice lower bound for which reduction in payment occurs at rate taper2.

partner_fortnightly_income
The partner’s fortnightly income (or zero if no partner).

partner_is_pensioner
(logical, default: FALSE) Is the individual’s partner in receipt of a *pension* (or benefit)?

partner_taper
The amount by which the payment is reduced for each dollar earned by the individual’s partner. (See https://guides.dss.gov.au/guide-social-security-law/4/2/8/40.)
youth_unemployment

Usage

```r
youth_unemployment(
    income = 0,
    assets = 0,
    fy.year = NULL,
    Date = NULL,
    has_partner = FALSE,
    has_dependant = FALSE,
    age = 23,
    lives_at_home = FALSE,
    independent = TRUE,
    unemployed = FALSE
)
```

Arguments

- **income**: Numeric vector of fortnightly income for the income test.
- **assets**: Numeric vector of the value of assets. By default, income and assets are both zero, thus returning the maximum benefit payable.
- **fy.year**: A character vector of valid financial years between "2000-01" and "2020-21" specifying which financial year the allowance is to be calculated.
- **Date**: (Date vector or coercible to such). An alternative to fy.year to specify the period over which the allowance is calculated.
- **has_partner**: (logical, default: FALSE) Does the individual have a partner?
- **has_dependant**: (logical, default: FALSE) Does the individual have any dependant children?
- **age**: Age (only determines whether the 16-17 age or 18 or over rates will apply).
- **lives_at_home**: (logical, default: FALSE) Is the individual a dependant who lives at home?
- **independent**: (logical, default: TRUE) Should the individual be considered independent.
- **unemployed**: (logical, default: FALSE) Is the individual unemployed?

Value

The fortnightly unemployment benefit payable for each entry. The function is vectorized over its arguments, with any length-1 argument recycled. (Other vector recycling is not supported and will result in an error.)
Index

* datasets
  max_super_contri_base, 40
  residential_property_prices, 67

* package
  grattan-package, 4
  .lito (lito), 39
  _PACKAGE (grattan-package), 4

age_grouper, 4
age_pension, 6, 24, 25
age_pension_age, 7
anyEq, 8
AnyWhich, 8
apply_super_caps_and_div293, 9
aus_pop_qtr, 11
aus_pop_qtr_age, 11
awote, 12

bto, 13

carer_payment, 14
carers_allowance, 14
CG_inflator, 63
CG_inflator (CG_population_inflator), 16
CG_population_inflator, 16
child_care_subsidy, 17, 42
compare_avg_tax_rates, 19
cpi_inflator, 20, 63
cpi_inflator_general_date, 21
cpi_inflator_quarters, 21, 22
date2fy (is.fy), 36
differentially_uprate_wage, 23, 62, 63
disability_pension, 24

disability_benefit, 25
family_tax_benefit, 26
fy (npv), 55
fy.year (is.fy), 36
fy2date (is.fy), 36

fy2yr (is.fy), 36
gdp, 28
gdp_fy (gdp), 28
gdp_qtr (gdp), 28
generic_inflator, 28
gni, 29
gni_fy (gni), 29
gni_qtr (gni), 29
grattan (grattan-package), 4
grattan-package, 4

income_tax, 30, 35
income_tax_sapto, 32
IncomeTax, 30
inflator, 33
install.packages, 34
install_taxstats, 34
inverse_average_rate, 35
inverse_income, 35
irr (npv), 55
is.fy, 36

l_f_inflator, 37
l_f_inflator, 62, 63
l_f_inflator (l_f_inflator), 37
lito, 31, 39

max_super_contri_base, 40
medicare_levy, 31, 41, 41
MedicareLevy, 41
model_child_care_subsidy, 42
model_income_tax, 31, 44, 68
model_new_caps_and_div293, 47
model_rent_assistance, 50

n_affected_from_new_cap_and_div293
  (model_new_caps_and_div293), 47
new_income_tax, 53
new_medicare_levy, 53
new_sapto, 33, 54
newstart_allowance, 51
npv, 55

Offset, 56

pension_supplement, 56
pmax, 57
pmaxC, 58
pmaxV, 58
pminC, 59
pminV, 59
pmt (npv), 55
progressivity, 60
prohibit_length0_vectors, 60
prohibit_unequal_length_vectors, 61
project, 61, 64
project_to, 64
pv (npv), 55

rebate_income, 64
rent_assistance, 65, 75
require_taxstats, 67
require_taxstats1516
  (require_taxstats), 67
residential_property_prices, 67
revenue_forgone, 68
revenue_from_new_cap_and_div293
  (model_new_caps_and_div293), 47

sapto, 31, 33, 42, 68, 69, 70
sapto_rcpp, 69
sapto_rcpp_singleton, 70
sapto_rcpp_yr, 70
small_business_tax_offset, 31, 47, 71
student_repayment, 72

unemployment_benefit, 73
validate_date, 74
validate_per, 75

wage_inflator, 23, 62, 63, 76

youth_allowance, 77
youth_unemployment, 79
yr2fy(is.fy), 36