

Package: econiodatajp (via r-universe)

October 21, 2026

Type Package

Title Ready-to-Use Japanese Input-Output Tables

Version 0.1.3

Description Provides ready-to-use input-output (IO) tables for Japan, built from data published by Japan's national government (e-Stat), individual prefectures, and the Research Institute of Economy, Trade and Industry (RIETI). Tables are archived and rebuilt on demand via the 'tarchives' and 'targets' packages, and returned as 'econio' input-output table objects for direct use with its analysis functions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dibble, dplyr, econio, purrr, rlang, stringr, tarchives (>= 0.2.0.9000), targets, tibble

Suggests adverbial, conflicted, econioread, forcats, tarchetypes, testthat (>= 3.0.0), tidy

Remotes UchidaMizuki/adverbial, UchidaMizuki/econio, UchidaMizuki/econioread, UchidaMizuki/tarchives

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Depends R (>= 3.5)

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev

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

Date/Publication 2026-09-21 04:21:27 UTC

RemoteUrl <https://github.com/UchidaMizuki/econiodatajp>

RemoteRef HEAD

RemoteSha 26c74c670335f7e0929387e1d6c768908a26e0b3

Contents

io_margin_available	2
io_margin_get	3
io_margin_target	4
io_sector_available	6
io_sector_conversion_get	7
io_sector_conversion_target	8
io_sector_get	9
io_sector_jsic_get	10
io_sector_jsic_target	12
io_sector_target	13
io_table_available	15
io_table_get	15
io_table_target	17
Index	20

io_margin_available	List available trade margin/transport margin tables
---------------------	---

Description

One row per table available via [io_margin_get()]/[io_margin_target()]. Useful for discovering which ‘year’s have margin data at all (only benchmark years up to 2015, see [io_margin_get()]) and which ‘sector_class’ each offers, rather than assuming ‘”medium”‘ is the only one that will ever exist. Precomputed and shipped as package data for the same reason as [io_table_available].

Usage

io_margin_available

Format

A tibble with one row per available table, with columns ‘region_type’, ‘region_class’, ‘year’, ‘region’ (matching [io_table_available]’s; always ‘”nation”‘ as of writing, since that’s the only ‘region_class’ with margin data today), ‘sector_class’, ‘margin_type’ (‘”trade_margin”‘ or ‘”transport_margin”‘, matching [io_margin_get()]’s ‘margin_type’), ‘language’, and ‘name’ (the archive target name, for [io_margin_target()]’s ‘..’/ ‘tarchives::tar_read_archive_raw()’).

Source

Computed from the tarchive pipelines bundled with this package by ‘data-row/io_margin_available.R’; re-run that script and reinstall whenever ‘inst/tarchives/’ changes.

io_margin_get

Get a trade margin or transport margin table

Description

The trade margin ("") and transport margin ("", i.e. domestic freight) tables published alongside an [io_table_get()] benchmark year. Both record, for each commodity, how much of its buyer-price markup over the producer price is trade margin (wholesale/retail) versus transport margin – the two components that, combined with a producer-price table, yield a buyer-price one. Only published as standalone tables for benchmark years up to 2015; from 2020 on, e-stat folds this information into the "input-output table" ("") instead and no longer publishes these as separate tables (see [io_margin_available] for which 'year's have data).

Usage

```
io_margin_get(
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  margin_type = c("trade_margin", "transport_margin"),
  sector_class,
  language = c("ja", "en")
)
```

Arguments

region_type	Table "shape": "regional" for a single-region table (see [econio::io_table_regional()]) or "multiregional" for a table with an explicit region dimension (see [econio::io_table_multiregional()]).
region_class	Region granularity: "nation" (the default) for the single national table; "pref" for prefecture-level, either the 47-prefecture interregional table ('region_type = "multiregional"') or one specific prefecture's own table ('region_type = "regional"', see 'region'); or "block" for the official 9-region block interregional table ('region_type = "multiregional"' only).
year	Year of the data.
region	Which single region to select within 'region_class'. Only applicable, and required, when 'region_class = "pref"' and 'region_type = "regional"', where it selects one prefecture by the code_name fragment used in the archive (e.g. "01_hokkaido", "13_tokyo" – see [io_table_available()]) to list them). Defaults to 'NULL' in every other case, where it resolves to "nation": 'region_class = "nation"' always covers the whole country as a single scope, and 'region_class = "pref"/"block"' under 'region_type = "multiregional"' also cover the whole country as their scope (they're broken into prefecture/block detail internally – that's 'region_class', not

a different ‘region’). If ‘region’ is omitted where it’s actually required (‘region_class = "pref"’, ‘region_type = "regional"’), it likewise resolves to “nation”, which doesn’t match any real per-prefecture archive name, so the call fails with the same generic “can’t find it, here’s what’s available” error as an invalid ‘region’ rather than a dedicated missing-argument error.

margin_type	Which table to get, “trade_margin” or “transport_margin”.
sector_class	Sector classification granularity. No default – must be specified. As of writing, every available table uses “medium” (a direct translation of the official “ ” tier name) – e-stat only ever published these two tables at that one granularity – but see [io_margin_available] to confirm what’s actually available rather than assuming “medium” is the only value that will ever exist.
language	Language of the table’s sector names, “ja” (the default) or “en”. Japanese is the authoritative source (sector *type* classification is derived from the Japanese sector classification workbook); English names are pre-translated from it when each tarchive is built (see ‘translate_iotable_sector()’ in ‘inst/tarchives/R/translate.R’), and a sector with no English source name keeps its Japanese name. Only supported when ‘region_class = "nation"’; a non-default value errors otherwise.

Value

An input-output table object for the specified year.

io_margin_target	<i>Declare a target that reads a trade margin or transport margin table</i>
-------------------------	---

Description

For use inside a ‘_targets.R’ pipeline, e.g. ‘tar_plan(io_margin_target(...))’. ‘io_margin_target()’ uses non-standard evaluation to capture ‘name’, mirroring [tarchives::tar_target_archive()]; ‘io_margin_target_raw()’ takes ‘name’ as a string instead, for programmatic use such as building several targets in a loop, mirroring [tarchives::tar_target_archive_raw()].

Usage

```
io_margin_target(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  margin_type = c("trade_margin", "transport_margin"),
  sector_class,
  language = c("ja", "en"),
```

```

    ...
)

io_margin_target_raw(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  margin_type = c("trade_margin", "transport_margin"),
  sector_class,
  language = c("ja", "en"),
  ...
)

```

Arguments

name	Symbol (<code>'io_margin_target()'</code>) or string (<code>'io_margin_target_raw()'</code>), name of the target.
region_type	Table "shape": <code>"regional"</code> for a single-region table (see <code>[econio::io_table_regional()]</code>) or <code>"multiregional"</code> for a table with an explicit region dimension (see <code>[econio::io_table_multiregional()]</code>).
region_class	Region granularity: <code>"nation"</code> (the default) for the single national table; <code>"pref"</code> for prefecture-level, either the 47-prefecture interregional table (<code>'region_type = "multiregional"'</code>) or one specific prefecture's own table (<code>'region_type = "regional"'</code> , see <code>'region'</code>); or <code>"block"</code> for the official 9-region block interregional table (<code>'region_type = "multiregional"'</code> only).
year	Year of the data.
region	Which single region to select within <code>'region_class'</code> . Only applicable, and required, when <code>'region_class = "pref"'</code> and <code>'region_type = "regional"'</code> , where it selects one prefecture by the code_name fragment used in the archive (e.g. <code>"01_hokkaido"</code> , <code>"13_tokyo"</code> – see <code>[io_table_available()]</code> to list them). Defaults to <code>'NULL'</code> in every other case, where it resolves to <code>"nation"</code> : <code>'region_class = "nation"'</code> always covers the whole country as a single scope, and <code>'region_class = "pref"/"block"'</code> under <code>'region_type = "multiregional"'</code> also cover the whole country as their scope (they're broken into prefecture/block detail internally – that's <code>'region_class'</code> , not a different <code>'region'</code>). If <code>'region'</code> is omitted where it's actually required (<code>'region_class = "pref"'</code> , <code>'region_type = "regional"'</code>), it likewise resolves to <code>"nation"</code> , which doesn't match any real per-prefecture archive name, so the call fails with the same generic <code>"can't find it, here's what's available"</code> error as an invalid <code>'region'</code> rather than a dedicated missing-argument error.
margin_type	Which table to get, <code>"trade_margin"</code> or <code>"transport_margin"</code> .
sector_class	Sector classification granularity. No default – must be specified. As of writing, every available table uses <code>"medium"</code> (a direct translation of the official <code>" "</code> tier name) – e-stat only ever published these two tables at

	that one granularity – but see <code>[io_margin_available]</code> to confirm what’s actually available rather than assuming “medium” is the only value that will ever exist.
language	Language of the table’s sector names, “ja” (the default) or “en”. Japanese is the authoritative source (sector <i>*type*</i> classification is derived from the Japanese sector classification workbook); English names are pre-translated from it when each tarchive is built (see <code>translate_iotable_sector()</code> in <code>inst/tarchives/R/translate.R</code>), and a sector with no English source name keeps its Japanese name. Only supported when <code>region_class = "nation"</code> ; a non-default value errors otherwise.
...	Arguments passed to <code>[tarchives::tar_target_archive_raw()]</code> .

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

`io_sector_available` *List available sector classifications*

Description

One row per sector classification, conversion, or JSIC correspondence table available via `[io_sector_get()]/[io_sector_conversion_get()]/[io_sector_jsic_get()]/[io_sector_target()]/[io_sector_conversion_target()]/[io_sector_jsic_target()]`. Useful for discovering which years have sector data, since not every `io_table_get()` pipeline does. Precomputed and shipped as package data (rather than a function) for the same reason as `[io_table_available]`: introspecting every tarchive pipeline is too slow to redo on every use.

Usage

`io_sector_available`

Format

A tibble with one row per available sector table, with columns `region_type`, `region_class`, `year`, `region` (matching `[io_table_available]`’s; always “nation” as of writing, since that’s the only `region_class` with sector data today), `type` (the accessor function it backs, “sector” for `[io_sector_get()]`, “sector_conversion” for `[io_sector_conversion_get()]`, or “sector_jsic” for `[io_sector_jsic_get()]` – always exactly `io_<type>_get()`’s `<type>`), `axis` (“input” or “output”; always “output” for `type = "sector_jsic"`, since `[io_sector_jsic_get()]`’s own `axis` argument only ever accepts “output” – a known, fixed fact about the data, not a real choice), `language` (“ja” or “en”, matching `[io_table_available]`’s), and `name` (the archive target name, for `[io_sector_target()]`’s / `[io_sector_conversion_target()]`’s / `[io_sector_jsic_target()]`’s `.../tarchives::tar_read_archive_raw()` – derivable from `region/` `type/` `axis/` `language` via the same logic `[io_sector_get()]`’s family uses internally, see `io_sector_name_archive()`).

Source

Computed from the tarchive pipelines bundled with this package by ‘data-raw/io_sector_available.R’; re-run that script and reinstall whenever ‘inst/tarchives/’ changes.

```
io_sector_conversion_get
```

Get a sector classification conversion table

Description

The crosswalk between sector classification tiers (“basic” -> “small”/“medium”/“large”/“template”) that [io_table_get()] uses internally to reclassify a table from its “basic” sector classification to a coarser one. See [io_sector_get()] for the sector names and tiers themselves.

Usage

```
io_sector_conversion_get(
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en")
)
```

Arguments

region_type	Table “shape” whose sector classification to fetch, “regional” (the default) or “multiregional” – see [io_table_get()].
region_class	Region granularity whose sector classification to fetch, “nation” (the default), “pref”, or “block” – see [io_table_get()]. Not every ‘region_type’/‘region_class’/‘year’ combination has sector data; see [io_sector_available] to list which do.
year	Year of the data.
region	Which single region to select within ‘region_class’, matching [io_table_get()]’s ‘region’. Defaults to ‘NULL’, which resolves to “nation” – as of writing, sector data only exists for ‘region_class = “nation”’ pipelines, so there’s no other value to pass yet; the argument exists so a future ‘region_class = “pref”/“block”’ sector archive (keyed per-prefecture/per-block the same way [io_table_get()]’s tables are) wouldn’t need a signature change to support it.
axis	Which axis of the table’s classification to fetch, “input” (rows) or “output” (columns). No default – must be specified: the two return different sector sets (endogenous plus gross-value-added sectors on the input axis, endogenous plus final-demand sectors on the output axis), and even the shared endogenous sectors use different codes at the “basic” tier.

language Language of the sector names, “ja” (the default) or “en”, matching `[io_table_get()]`’s ‘language’.

Value

A tibble with one row per ‘sector_class_from’/‘sector_class_to’ sector pair, with columns ‘sector_type’, ‘sector_class_from’, ‘sector_name_from’, ‘sector_class_to’, and ‘sector_name_to’.

io_sector_conversion_target

Declare a target that reads a sector classification conversion table

Description

For use inside a ‘_targets.R’ pipeline, e.g. ‘tar_plan(io_sector_conversion_target(...))’. ‘io_sector_conversion_target()’ uses non-standard evaluation to capture ‘name’, mirroring `[tarchives::tar_target_archive()]`; ‘io_sector_conversion_target_raw()’ takes ‘name’ as a string instead, for programmatic use such as building several targets in a loop, mirroring `[tarchives::tar_target_archive_raw()]`.

Usage

```
io_sector_conversion_target(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en"),
  ...
)

io_sector_conversion_target_raw(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en"),
  ...
)
```

Arguments

name	Symbol (<code>'io_sector_conversion_target()'</code>) or string (<code>'io_sector_conversion_target_raw()'</code>), name of the target.
region_type	Table "shape" whose sector classification to fetch, "regional" (the default) or "multiregional" – see <code>[io_table_get()]</code> .
region_class	Region granularity whose sector classification to fetch, "nation" (the default), "pref", or "block" – see <code>[io_table_get()]</code> . Not every 'region_type'/'region_class'/'year' combination has sector data; see <code>[io_sector_available]</code> to list which do.
year	Year of the data.
region	Which single region to select within 'region_class', matching <code>[io_table_get()]</code> 's 'region'. Defaults to 'NULL', which resolves to "nation" – as of writing, sector data only exists for 'region_class = "nation"' pipelines, so there's no other value to pass yet; the argument exists so a future 'region_class = "pref"/"block"' sector archive (keyed per-prefecture/per-block the same way <code>[io_table_get()]</code> 's tables are) wouldn't need a signature change to support it.
axis	Which axis of the table's classification to fetch, "input" (rows) or "output" (columns). No default – must be specified: the two return different sector sets (endogenous plus gross-value-added sectors on the input axis, endogenous plus final-demand sectors on the output axis), and even the shared endogenous sectors use different codes at the "basic" tier.
language	Language of the sector names, "ja" (the default) or "en", matching <code>[io_table_get()]</code> 's 'language'.
...	Arguments passed to <code>[tarchives::tar_target_archive_raw()]</code> .

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

<code>io_sector_get</code>	<i>Get a sector classification</i>
----------------------------	------------------------------------

Description

The sector names and classification tiers ("basic", "small", "medium", "large", "template") that `[io_table_get()]`'s tables are classified by. See `[io_sector_conversion_get()]` for the crosswalk between tiers.

Usage

```
io_sector_get(
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en")
)
```

Arguments

region_type	Table "shape" whose sector classification to fetch, "regional" (the default) or "multiregional" – see [io_table_get()].
region_class	Region granularity whose sector classification to fetch, "nation" (the default), "pref", or "block" – see [io_table_get()]. Not every 'region_type'/'region_class'/'year' combination has sector data; see [io_sector_available] to list which do.
year	Year of the data.
region	Which single region to select within 'region_class', matching [io_table_get()]’s 'region'. Defaults to 'NULL', which resolves to "nation" – as of writing, sector data only exists for 'region_class = "nation"' pipelines, so there’s no other value to pass yet; the argument exists so a future 'region_class = "pref"/"block"' sector archive (keyed per-prefecture/per-block the same way [io_table_get()]’s tables are) wouldn’t need a signature change to support it.
axis	Which axis of the table’s classification to fetch, "input" (rows) or "output" (columns). No default – must be specified: the two return different sector sets (endogenous plus gross-value-added sectors on the input axis, endogenous plus final-demand sectors on the output axis), and even the shared endogenous sectors use different codes at the "basic" tier.
language	Language of the sector names, "ja" (the default) or "en", matching [io_table_get()]’s 'language'.

Value

A tibble with one row per sector, with columns 'sector_type', 'sector_class', and 'sector_name'.

io_sector_jsic_get	<i>Get the correspondence between IO sectors and JSIC</i>
---------------------------	---

Description

The official crosswalk between a table's basic-classification industry sectors and the Japan Standard Industrial Classification (JSIC), as published by e-stat/MIC alongside a benchmark year's sector classification. The correspondence is many-to-many: some IO sectors map to several JSIC codes, and some JSIC codes are themselves split across several IO sectors, in which case 'note' carries the source's own description of the split (e.g. " ", "of which: wheat"). JSIC names and notes are Japanese-only in the source and have no English variant; only 'sector_name' is translated by 'language'.

Usage

```
io_sector_jsic_get(
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis = c("output"),
  language = c("ja", "en")
)
```

Arguments

region_type , region_class	Currently only 'region_type = "regional"', 'region_class = "nation"' tables have JSIC correspondence data; see [io_sector_available] to confirm which 'year's do.
year	Year of the data.
region	Which single region to select within 'region_class', matching [io_table_get()]'s 'region'. Defaults to 'NULL', which resolves to "nation" – as of writing, sector data only exists for 'region_class = "nation"' pipelines, so there's no other value to pass yet; the argument exists so a future 'region_class = "pref"/"block"' sector archive (keyed per-prefecture/per-block the same way [io_table_get()]'s tables are) wouldn't need a signature change to support it.
axis	Always "output" – see Details.
language	Language of the sector names, "ja" (the default) or "en", matching [io_table_get()]'s 'language'.

Details

JSIC has been revised multiple times, and different benchmark years' correspondence tables are keyed to different JSIC revisions (e.g. the 2011 table uses the 2007 revision, while 2015/2020 use the 2013 revision); 'jsic_revision_year' records which revision 'jsic_code'/'jsic_name' come from so results from different 'year's aren't compared as if they used the same JSIC codes.

Unlike [io_sector_get()]/[io_sector_conversion_get()], 'axis' only ever accepts "output": the source only ever publishes this correspondence keyed to the output/industry axis's

basic-classification codes (its own header calls the key a "column code"), and the input axis uses a differently-grained code for basic industry sectors that this correspondence has no published equivalent for. Passing 'axis = "input"' errors rather than silently ignoring the argument. 'sector_name' therefore only ever matches the output axis's [io_sector_get()]/[io_sector_conversion_get()] output; joining it against the input axis's returns zero matches (confirmed empirically: the two axes' basic-level sector codes – and therefore 'sector_name', which embeds the code – differ).

Value

A tibble with one row per IO-sector/JSIC-code pair, with columns 'sector_name', 'jsic_code', 'jsic_name', 'note' ('NA' except for sectors split across multiple JSIC codes or vice versa), and 'jsic_revision_year' (the JSIC revision 'jsic_code'/'jsic_name' are keyed to).

io_sector_jsic_target

Declare a target that reads the correspondence between IO sectors and JSIC

Description

For use inside a 'targets.R' pipeline, e.g. 'tar_plan(io_sector_jsic_target(...))'. 'io_sector_jsic_target()' uses non-standard evaluation to capture 'name', mirroring [tarchives::tar_target_archive()]; 'io_sector_jsic_target_raw()' takes 'name' as a string instead, for programmatic use such as building several targets in a loop, mirroring [tarchives::tar_target_archive_raw()].

Usage

```
io_sector_jsic_target(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis = c("output"),
  language = c("ja", "en"),
  ...
)

io_sector_jsic_target_raw(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis = c("output"),
  language = c("ja", "en"),
  ...
)
```

Arguments

name	Symbol (<code>'io_sector_jsic_target()'</code>) or string (<code>'io_sector_jsic_target_raw()'</code>), name of the target.
region_type, region_class	Currently only <code>'region_type = "regional"'</code> , <code>'region_class = "nation"'</code> tables have JSIC correspondence data; see <code>[io_sector_available]</code> to confirm which <code>'year'</code> s do.
year	Year of the data.
region	Which single region to select within <code>'region_class'</code> , matching <code>[io_table_get()]</code> 's <code>'region'</code> . Defaults to <code>'NULL'</code> , which resolves to <code>"nation"</code> – as of writing, sector data only exists for <code>'region_class = "nation"'</code> pipelines, so there's no other value to pass yet; the argument exists so a future <code>'region_class = "pref"'</code> / <code>"block"</code> sector archive (keyed per-prefecture/per-block the same way <code>[io_table_get()]</code> 's tables are) wouldn't need a signature change to support it.
axis	Always <code>"output"</code> – see Details.
language	Language of the sector names, <code>"ja"</code> (the default) or <code>"en"</code> , matching <code>[io_table_get()]</code> 's <code>'language'</code> .
...	Arguments passed to <code>[tarchives::tar_target_archive_raw()]</code> .

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

<code>io_sector_target</code>	<i>Declare a target that reads a sector classification</i>
-------------------------------	--

Description

For use inside a `'_targets.R'` pipeline, e.g. `'tar_plan(io_sector_target(...))'`. `'io_sector_target()'` uses non-standard evaluation to capture `'name'`, mirroring `[tarchives::tar_target_archive()]`; `'io_sector_target_raw()'` takes `'name'` as a string instead, for programmatic use such as building several targets in a loop, mirroring `[tarchives::tar_target_archive_raw()]`.

Usage

```
io_sector_target(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en"),
```

```

    ...
)

io_sector_target_raw(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  axis,
  language = c("ja", "en"),
  ...
)

```

Arguments

name	Symbol (<code>'io_sector_target()'</code>) or string (<code>'io_sector_target_raw()'</code>), name of the target.
region_type	Table "shape" whose sector classification to fetch, <code>"regional"</code> (the default) or <code>"multiregional"</code> – see <code>[io_table_get()]</code> .
region_class	Region granularity whose sector classification to fetch, <code>"nation"</code> (the default), <code>"pref"</code> , or <code>"block"</code> – see <code>[io_table_get()]</code> . Not every <code>'region_type'/'region_class'/'year'</code> combination has sector data; see <code>[io_sector_available]</code> to list which do.
year	Year of the data.
region	Which single region to select within <code>'region_class'</code> , matching <code>[io_table_get()]</code> 's <code>'region'</code> . Defaults to <code>'NULL'</code> , which resolves to <code>"nation"</code> – as of writing, sector data only exists for <code>'region_class = "nation"'</code> pipelines, so there's no other value to pass yet; the argument exists so a future <code>'region_class = "pref"/'"block"'</code> sector archive (keyed per-prefecture/per-block the same way <code>[io_table_get()]</code> 's tables are) wouldn't need a signature change to support it.
axis	Which axis of the table's classification to fetch, <code>"input"</code> (rows) or <code>"output"</code> (columns). No default – must be specified: the two return different sector sets (endogenous plus gross-value-added sectors on the input axis, endogenous plus final-demand sectors on the output axis), and even the shared endogenous sectors use different codes at the <code>"basic"</code> tier.
language	Language of the sector names, <code>"ja"</code> (the default) or <code>"en"</code> , matching <code>[io_table_get()]</code> 's <code>'language'</code> .
...	Arguments passed to <code>[tarchives::tar_target_archive_raw()]</code> .

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

io_table_available	<i>List available input-output tables</i>
--------------------	---

Description

One row per table available via `[io_table_get()]/[io_table_target()]`. Useful for discovering which arguments they accept for a given ‘region_class’/‘year’ – in particular ‘sector_class’, whose choices for ‘region_class = "block"' vary by ‘year’ (FY2005 has three; every other year has one) and so aren’t listed in `[io_table_get()]`’s documentation. Pre-computed and shipped as package data (rather than a function) since introspecting every tarchive pipeline is too slow to redo on every use – including inside a ‘_targets.R’ pipeline via ‘tarchetypes::tar_eval(..., values = io_table_available)’, which re-sources on every run and would otherwise pay that cost every time.

Usage

```
io_table_available
```

Format

A tibble with one row per available table, with columns ‘region_type’, ‘region_class’, ‘year’, ‘region’ (“nation” for every table except a ‘region_class = "pref"’, ‘region_type = "regional"’ table, where it’s a real per-prefecture code_name fragment, e.g. “01_hokkaido” – see `[io_table_get()]`’s ‘region’; never ‘NA’), ‘sector_class’, ‘price_type’, ‘import_type’, ‘language’, and ‘name’ (the archive target name, for `[io_table_target()]`’s ‘...’/ ‘tarchives::tar_read_archive_raw()’)

Source

Computed from the tarchive pipelines bundled with this package by ‘data-row/io_table_available.R’; re-run that script and reinstall whenever ‘inst/tarchives/’ changes.

io_table_get	<i>Get an input-output table</i>
--------------	----------------------------------

Description

Get an input-output table

Usage

```
io_table_get(
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  sector_class,
```

```

price_type = c("producer_price", "purchaser_price"),
import_type = c("competitive_import", "noncompetitive_import"),
language = c("ja", "en")
)

```

Arguments

region_type	Table "shape": "regional" for a single-region table (see [econio::io_table_regional()]) or "multiregional" for a table with an explicit region dimension (see [econio::io_table_multiregional()]).
region_class	Region granularity: "nation" (the default) for the single national table; "pref" for prefecture-level, either the 47-prefecture interregional table ('region_type = "multiregional"') or one specific prefecture's own table ('region_type = "regional"', see 'region'); or "block" for the official 9-region block interregional table ('region_type = "multiregional"' only).
year	Year of the data.
region	Which single region to select within 'region_class'. Only applicable, and required, when 'region_class = "pref"' and 'region_type = "regional"', where it selects one prefecture by the code_name fragment used in the archive (e.g. "01_hokkaido", "13_tokyo" – see [io_table_available()] to list them). Defaults to 'NULL' in every other case, where it resolves to "nation": 'region_class = "nation"' always covers the whole country as a single scope, and 'region_class = "pref"/"block"' under 'region_type = "multiregional"' also cover the whole country as their scope (they're broken into prefecture/block detail internally – that's 'region_class', not a different 'region'). If 'region' is omitted where it's actually required ('region_class = "pref"', 'region_type = "regional"'), it likewise resolves to "nation", which doesn't match any real per-prefecture archive name, so the call fails with the same generic "can't find it, here's what's available" error as an invalid 'region' rather than a dedicated missing-argument error.
sector_class	Sector classification granularity. No default – must be specified. For 'region_class = "nation"', the full set ("basic", "small", "medium", "large", "template") – each a direct translation of a real official Japanese classification tier name. No official documentation gives a common tier name to the 'region_class = "pref"' or "block" tables' sector granularities, only their sector counts, so 'sector_class' there is that count as a string instead: for 'region_class = "pref"' with 'region_type = "regional"' (one prefecture's own table), each prefecture publishes its own count (e.g. "105" for Hokkaido, "187" for Osaka); for 'region_class = "pref"' with 'region_type = "multiregional"' (the interregional table), the count varies by 'year' ("26" for 'year = 2005', "31" for 'year = 2011'); for 'region_class = "block"', the count varies by 'year' too ("12"/"29"/"53" for 'year = 2005'; a single year-specific count otherwise) – see [io_table_available()] to list them all.
price_type	Price basis of the table, "producer_price" (the default) or "purchaser_price".
import_type	Import convention used in the table, "competitive_import" (the default) or "noncompetitive_import" (matches 'competitive_import = TRUE'/'FALSE').

in `[econioread::io_table_read_sector_types()]`. “basic” and “small” sector classes are only available for “competitive_import”. Only “competitive_import” is supported when ‘region_class’ isn’t “nation”.

language Language of the table’s sector names, “ja” (the default) or “en”. Japanese is the authoritative source (sector *type* classification is derived from the Japanese sector classification workbook); English names are pre-translated from it when each tarchive is built (see ‘translate_iotable_sector()’ in ‘inst/tarchives/R/translate.R’), and a sector with no English source name keeps its Japanese name. Only supported when ‘region_class = “nation”’; a non-default value errors otherwise.

Value

An input-output table object for the specified year.

<code>io_table_target</code>	<i>Declare a target that reads an input-output table</i>
------------------------------	--

Description

For use inside a ‘_targets.R’ pipeline, e.g. ‘tar_plan(io_table_target(...))’. ‘io_table_target()’ uses non-standard evaluation to capture ‘name’, mirroring `[tarchives::tar_target_archive()]`; ‘io_table_target_raw()’ takes ‘name’ as a string instead, for programmatic use such as building several targets in a loop, mirroring `[tarchives::tar_target_archive_raw()]`.

Usage

```
io_table_target(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  sector_class,
  price_type = c("producer_price", "purchaser_price"),
  import_type = c("competitive_import", "noncompetitive_import"),
  language = c("ja", "en"),
  ...
)

io_table_target_raw(
  name,
  region_type = c("regional", "multiregional"),
  region_class = c("nation", "pref", "block"),
  year,
  region = NULL,
  sector_class,
```

```

price_type = c("producer_price", "purchaser_price"),
import_type = c("competitive_import", "noncompetitive_import"),
language = c("ja", "en"),
...
)

```

Arguments

name	Symbol (<code>'io_table_target()'</code>) or string (<code>'io_table_target_raw()'</code>), name of the target.
region_type	Table "shape": <code>"regional"</code> for a single-region table (see <code>[econio::io_table_regional()]</code>) or <code>"multiregional"</code> for a table with an explicit region dimension (see <code>[econio::io_table_multiregional()]</code>).
region_class	Region granularity: <code>"nation"</code> (the default) for the single national table; <code>"pref"</code> for prefecture-level, either the 47-prefecture interregional table (<code>'region_type = "multiregional"'</code>) or one specific prefecture's own table (<code>'region_type = "regional"'</code> , see <code>'region'</code>); or <code>"block"</code> for the official 9-region block interregional table (<code>'region_type = "multiregional"'</code> only).
year	Year of the data.
region	Which single region to select within <code>'region_class'</code> . Only applicable, and required, when <code>'region_class = "pref"'</code> and <code>'region_type = "regional"'</code> , where it selects one prefecture by the <code>code_name</code> fragment used in the archive (e.g. <code>"01_hokkaido"</code> , <code>"13_tokyo"</code> – see <code>[io_table_available()]</code> to list them). Defaults to <code>'NULL'</code> in every other case, where it resolves to <code>"nation"</code> : <code>'region_class = "nation"'</code> always covers the whole country as a single scope, and <code>'region_class = "pref"'</code> / <code>"block"</code> under <code>'region_type = "multiregional"'</code> also cover the whole country as their scope (they're broken into prefecture/block detail internally – that's <code>'region_class'</code> , not a different <code>'region'</code>). If <code>'region'</code> is omitted where it's actually required (<code>'region_class = "pref"'</code> , <code>'region_type = "regional"'</code>), it likewise resolves to <code>"nation"</code> , which doesn't match any real per-prefecture archive name, so the call fails with the same generic <code>"can't find it, here's what's available"</code> error as an invalid <code>'region'</code> rather than a dedicated missing-argument error.
sector_class	Sector classification granularity. No default – must be specified. For <code>'region_class = "nation"'</code> , the full set (<code>"basic"</code> , <code>"small"</code> , <code>"medium"</code> , <code>"large"</code> , <code>"template"</code>) – each a direct translation of a real official Japanese classification tier name. No official documentation gives a common tier name to the <code>'region_class = "pref"'</code> or <code>"block"</code> tables' sector granularities, only their sector counts, so <code>'sector_class'</code> there is that count as a string instead: for <code>'region_class = "pref"'</code> with <code>'region_type = "regional"'</code> (one prefecture's own table), each prefecture publishes its own count (e.g. <code>"105"</code> for Hokkaido, <code>"187"</code> for Osaka); for <code>'region_class = "pref"'</code> with <code>'region_type = "multiregional"'</code> (the interregional table), the count varies by <code>'year'</code> (<code>"26"</code> for <code>'year = 2005'</code> , <code>"31"</code> for <code>'year = 2011'</code>); for <code>'region_class = "block"</code> , the count varies by <code>'year'</code> too (<code>"12"/"29"/"53"</code> for <code>'year = 2005'</code> ; a single year-specific count otherwise) – see <code>[io_table_available()]</code> to list them all.

price_type	Price basis of the table, "producer_price" (the default) or "purchaser_price".
import_type	Import convention used in the table, "competitive_import" (the default) or "noncompetitive_import" (matches 'competitive_import = TRUE'/'FALSE' in [econio::io_table_read_sector_types()]). "basic" and "small" sector classes are only available for "competitive_import". Only "competitive_import" is supported when 'region_class' isn't "nation".
language	Language of the table's sector names, "ja" (the default) or "en". Japanese is the authoritative source (sector *type* classification is derived from the Japanese sector classification workbook); English names are pre-translated from it when each tarchive is built (see 'translate_iotable_sector()' in 'inst/tarchives/R/translate.R'), and a sector with no English source name keeps its Japanese name. Only supported when 'region_class = "nation"'; a non-default value errors otherwise.
...	Arguments passed to [tarchives::tar_target_archive_raw()].

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

Index

- * datasets
 - io_margin_available, 2
 - io_sector_available, 6
 - io_table_available, 15
- io_margin_available, 2
- io_margin_get, 3
- io_margin_target, 4
- io_margin_target_raw
 - (*io_margin_target*), 4
- io_sector_available, 6
- io_sector_conversion_get, 7
- io_sector_conversion_target, 8
- io_sector_conversion_target_raw
 - (*io_sector_conversion_target*), 8
- io_sector_get, 9
- io_sector_jsic_get, 10
- io_sector_jsic_target, 12
- io_sector_jsic_target_raw
 - (*io_sector_jsic_target*), 12
- io_sector_target, 13
- io_sector_target_raw
 - (*io_sector_target*), 13
- io_table_available, 15
- io_table_get, 15
- io_table_target, 17
- io_table_target_raw
 - (*io_table_target*), 17
- list(), 6, 9, 13, 14, 19