Package: tiltIndicator (via r-universe)

November 1, 2024

Title Indicators for the 'TILT' Project

Version 0.0.0.9201

Description Indicators for the 'TILT' project.

License GPL (>= 3)

URL https://github.com/2DegreesInvesting/tiltIndicator

BugReports https://github.com/2DegreesInvesting/tiltIndicator/issues

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, glue, lifecycle, magrittr, memoise, purrr, readr, rlang (>= 0.4.11), stats, stringr, tibble, tidyr, tidyselect, tiltToyData (>= 0.0.0.9003), utils, vctrs, withr

Suggests covr, fs, furrr, future, here, progress, rappdirs, rmarkdown, roxygen2, testthat (>= 3.0.0), usethis

Remotes 2degreesinvesting/tiltToyData

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

Repository https://2degreesinvesting.r-universe.dev

RemoteUrl https://github.com/2DegreesInvesting/tiltIndicator

RemoteRef v0.0.0.9201

RemoteSha b1317db448feefbd4e2b1836362a1ed73bc2bb8b

Contents

`emissions_profile` *Calculate the indicator "emissions profile"*

Description

The "emissions profile" measures the absolute GHG emissions of a product in comparison to a chosen benchmark of products. The assessment is first performed on a product-level and can then be aggregated to the company-level. The profile is expressed as the share of a company's products that are in a "high", "medium", or "low" emission profile category, based on the comparison to the benchmark. A higher emission profile indicates a larger impact on climate change compared to the benchmark and can therefore also be interpreted as one climate risk component to assess products, companies, or loan portfolios.

The "emission profile" is calculated in these steps:

- The relative GHG emissions per product are collected from a Life-Cycle-Analysis (LCA) database by matching the products from our company dataset to the products from the LCA dataset.
- All products are ranked according to their GHG emissions.
- The products are grouped by the following benchmarks:
  1. `all`: All products.
  2. `isic_4digit`: All products within the same ISIC 4 digit code (example: 0112 Growing of rice).
  3. `tilt_sector`: All products within the same tilt sector (example: agriculture).
  4. `unit`: All products with the same unit (example: kg).
  5. `unit_isic_4digit`: All products with the same unit within the same ISIC 4 digit section (example: kg + 0112 Growing of rice).
  6. `unit_tilt_sector`: All products with the same unit within the same tilt sector (example: kg + agriculture).
- For each benchmark, products are assigned to the emission profile category "low", "medium" or "high", depending on the GHG emissions arising from their production process in comparison to all other products within the same benchmark. For the assignment of the three categories, thresholds are used. Please find more information about the thresholds in the Thresholds section.

For the company-level results, we aggregate all products from the same category and benchmark and set them in relation to all products that the company produces. The company-level results are expressed as the company's share of products per category "low", "medium", and "high" in comparison to each benchmark.

The output of this indicator contains the following:

- A column indicating the benchmark to which a product is compared.
- A column indicating whether the product has "low", "medium" or "high" relative GHG emissions.
- A column indicating the share of the products per category and benchmark".

Usage

```
emissions_profile(companies, co2, low_threshold = 1/3, high_threshold = 2/3)
```

Arguments

`companies`, `co2`	A dataframe like the dataset with a matching name in tiltToyData (see Reference).
`low_threshold`	A numeric value to segment low and medium emission profile products.
`high_threshold`	A numeric value to segment medium and high emission profile products.

Value

A data frame with the column `companies_id`, and the nested columns`product` and `company` holding the outputs at product and company level. Unnesting `product` yields a data frame with at least columns `companies_id`, `grouped_by`, `risk_category`. Unnesting `company` yields a data frame with at least columns `companies_id`, `grouped_by`, `risk_category`, `value`. Any column in the input datasets ending with `*rowid` is also passed as is to the output at product level. The exception is any column named exactly `rowid`– which is a reserved name and throws an error. Note this feature makes no sense at company level because potentially multiple rows in the input datasets are summarized into a single row in the output at company level.

See Also

Other main functions: `emissions_profile_upstream()`, `sector_profile_upstream()`, `sector_profile()`

Examples

```
library(tiltIndicator)
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_emissions_profile_any_companies())
products <- read_csv(toy_emissions_profile_products_ecoinvent())

both <- emissions_profile(companies, products)
both
```

```
both |> unnest_product()

both |> unnest_company()
```

`emissions_profile_upstream` *Calculate the indicator "emissions profile upstream"*

Description

The indicator "emissions profile upstream" assesses the transition risk of the upstream products due to their relative carbon footprint to other upstream products. As a default option, each upstream product is compared to the carbon footprint of every other upstream product. Upstream products with a higher carbon footprint face a higher risk. On a company-level, the indicator proxies for the supply chain risk of a company - based on its inputs.

The indicator "emissions profile upstream" is therefore similar to the Product Carbon Transition Risk Indicator, but it focuses on the upstream products and not the product of the company. Upstream products are, for example, resources, packaging materials, energy and enabling services (such as tractor use on farm) to produce the product.

After identifying each carbon footprint for one upstream product, the input products are ranked according to their footprint. The ranking method is explained in the Thresholds section.

After assessing the upstream products' transition risk based on the carbon footprint of each product, they are aggregated at the company-level. We derive what percentage of the upstream products are high, medium and low transition risk.

This indicator consists of 2 broad steps:

1. Score upstream products: Identifying the upstream products for each product, and calculating the relative carbon footprint per upstream product.
2. Score companies: Aggregating on the company-level.

The sample data set includes inputs and co2 footprints for each product from Ecoinvent and sectors from Europages. NOTE: the following columns are a completely random selection and do not reflect the true information:

- co2 footprints (not allowed to share licensed data right now)
- sectors (as the matching with ecoinvent is not done yet, we do not have one sector per product yet)

Usage

```
emissions_profile_upstream(
  companies,
  co2,
  low_threshold = 1/3,
  high_threshold = 2/3
)
```

Arguments

companies, co2	A dataframe like the dataset with a matching name in tiltToyData (see Reference).
low_threshold	A numeric value to segment low and medium emission profile products.
high_threshold	A numeric value to segment medium and high emission profile products.

Value

A data frame with the column companies_id, and the nested columnsproduct and company holding the outputs at product and company level. Unnesting product yields a data frame with at least columns companies_id, grouped_by, risk_category. Unnesting company yields a data frame with at least columns companies_id, grouped_by, risk_category, value. Any column in the input datasets ending with *rowid is also passed as is to the output at product level. The exception is any column named exactly rowid– which is a reserved name and throws an error. Note this feature makes no sense at company level because potentially multiple rows in the input datasets are summarized into a single row in the output at company level.

See Also

Other main functions: emissions_profile(), sector_profile_upstream(), sector_profile()

Examples

```
library(tiltIndicator)
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_emissions_profile_any_companies())
inputs <- read_csv(toy_emissions_profile_upstream_products_ecoinvent())

both <- emissions_profile_upstream(companies, inputs)

both |> unnest_product()

both |> unnest_company()
```

---

jitter_range *Expand a range adding some random noise*

Description

This function expands a range by adding noise to the left of the minimum values and to the right of the maximum values.

Usage

```
jitter_range(data, factor = 1, amount = NULL)
```

Arguments

data	A dataframe with columns min and max.
factor	numeric.
amount	numeric; if positive, used as *amount* (see below), otherwise, if = 0 the default is factor * z/50. Default (NULL): factor * d/5 where d is about the smallest difference between x values.

Value

The input dataframe with the additional columns min_jitter and max_jitter.

See Also

jitter()

Other helpers: summarize_range(), unnest_product()

Examples

```
library(tibble)
set.seed(123)

data <- tibble(min = -2:2, max = -1:3)

data |> jitter_range(amount = 0.1)

data |> jitter_range(amount = 2)
```

---

sector_profile *Calculate the indicator "sector profile"*

Description

The indicator "sector profile" measures the transition risk of products based on the sector's emissions targets the product belongs to. Those sector emission reduction targets vary across scenarios (e.g., net zero scenario or 1.5° scenario) and the time horizon (e.g., reduction needed in 2030, 2040, 2050 to achieve the targets).

After assessing each product, all the products with the same category are aggregated and set in relation to all products of the company. We, therefore, derive company-level information.

Usage

```
sector_profile(
  companies,
  scenarios,
  low_threshold = ifelse(scenarios$year == 2030, 1/9, 1/3),
  high_threshold = ifelse(scenarios$year == 2030, 2/9, 2/3)
)
```

Arguments

companies, scenarios	A dataframe like the dataset with a matching name in tiltToyData (see Reference).
low_threshold	A numeric value to segment low and medium reduction targets.
high_threshold	A numeric value to segment medium and high reduction targets.

Value

A data frame with the column companies_id, and the nested columnsproduct and company holding the outputs at product and company level. Unnesting product yields a data frame with at least columns companies_id, grouped_by, risk_category. Unnesting company yields a data frame with at least columns companies_id, grouped_by, risk_category, value. Any column in the input datasets ending with *rowid is also passed as is to the output at product level. The exception is any column named exactly rowid– which is a reserved name and throws an error. Note this feature makes no sense at company level because potentially multiple rows in the input datasets are summarized into a single row in the output at company level.

See Also

Other main functions: emissions_profile_upstream(), emissions_profile(), sector_profile_upstream()

Examples

```
library(tiltIndicator)
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_sector_profile_companies())
scenarios <- read_csv(toy_sector_profile_any_scenarios())

both <- sector_profile(companies, scenarios)
both

both |> unnest_product()

both |> unnest_company()
```

---

sector_profile_any_pivot_type_sector_subsector

Restructure "sector profile" companies

Description

Restructure "sector profile" companies

Usage

```
sector_profile_any_pivot_type_sector_subsector(data)
```

Arguments

data	A dataframe with these columns: • ipr_sector • ipr_subsector • weo_product • weo_flow

Value

A companies dataset as required by sector functions.

See Also

Other pre-processing helpers: sector_profile_any_prepare_scenario(), sector_profile_any_prune_companies()

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(readr, warn.conflicts = FALSE)

raw_companies <- example_raw_companies()
glimpse(raw_companies)

companies <- sector_profile_any_pivot_type_sector_subsector(raw_companies)
companies
```

---

sector_profile_any_polish_output_at_company_level

Polish output at company level

Description

Polish output at company level

Usage

```
sector_profile_any_polish_output_at_company_level(data)
```

Arguments

data	The output of sector_profile*() functions.

Value

A dataframe.

Examples

```
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_sector_profile_companies())
scenarios <- read_csv(toy_sector_profile_any_scenarios())

sector_profile(companies, scenarios) |>
  unnest_company() |>
  sector_profile_any_polish_output_at_company_level()

companies_upstream <- read_csv(toy_sector_profile_upstream_companies())
inputs <- read_csv(toy_sector_profile_upstream_products())

sector_profile_upstream(companies_upstream, scenarios, inputs) |>
  unnest_company() |>
  sector_profile_any_polish_output_at_company_level()
```

---

sector_profile_any_prepare_scenario

Given a named list of scenarios returns a cleaner `scenarios` *dataframe*

Description

Given a named list of scenarios returns a cleaner `scenarios` dataframe

Usage

```
sector_profile_any_prepare_scenario(scenarios)
```

Arguments

`scenarios`	A named list of identically structured scenarios.

Value

A single, cleaner dataframe with an additional column to identify which rows come from which scenario.

See Also

Other pre-processing helpers: `sector_profile_any_pivot_type_sector_subsector()`, `sector_profile_any_prune_c`

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(readr, warn.conflicts = FALSE)

raw_weo <- example_raw_weo()
raw_ipr <- example_raw_ipr()
raw_scenarios <- list(weo = raw_weo, ipr = raw_ipr)

sector_profile_any_prepare_scenario(raw_scenarios)
```

`sector_profile_any_prune_companies`

Drop rows where the product info is `NA` *& sector info is duplicated*

Description

For each company, this function drops rows where the product information is missing and the sector information is duplicated.

Usage

```
sector_profile_any_prune_companies(data)
```

Arguments

`data`	Typically a "sector profile" `*companies` dataframe.

Value

A dataframe with maybe fewer rows than the input `data`.

See Also

Other pre-processing helpers: `sector_profile_any_pivot_type_sector_subsector()`, `sector_profile_any_prepare_`

Examples

```
library(dplyr)
# styler: off
companies <- tribble(
  ~row, ~companies_id, ~clustered, ~activity_uuid_product_uuid, ~tilt_sector,
    1L,           "a",       "b1",                         "c1",          "x",
    2L,           "a",         NA,                           NA,          "x",
    3L,           "a",         NA,                           NA,          "y",
    4L,           "a",         NA,                           NA,          "y"
  )
# styler: off

# Keep row 1: Has product info
```

```
# Drop row 2: Lacks product info and sector info is duplicated
# Keep row 3: Lacks product info but sector info is unique
# Drop row 4: Lacks product info and sector info is duplicated
companies

sector_profile_any_prune_companies(companies)
```

`sector_profile_upstream` *Calculate the indicator "sector profile upstream"*

Description

The indicator "sector profile upstream" assesses the transition risk of the input products based on the sector's emissions targets the input product belongs to. This indicator can be aggregated on company level and as such inform about the supply chain risk of an SME, based on its inputs' transition risk. The sector emission reduction targets vary across scenarios (e.g., net zero scenario or 1.5° scenario) and the time horizon (e.g., reduction needed in 2030, 2040, 2050 to achieve the targets). It, therefore, is similar to the Product Sector Risk Indicator but focuses on the input products that the company needs to produce its products.The input products are, for example, resources, packaging materials, energy and enabling services (such as tractor use on farm) to produce the product.

After identifying each carbon footprint for one input product, the input products are ranked according to their footprint. The ranking method is explained in the Thresholds section.

After assessing the input products for each product, they are aggregated at company-level to derive what percentage of the input products required by the company to produce its products have high, medium and low sector transition risk. We, therefore, derive the company-level information.

Please note that carbon emissions or emissions always mean CO2e.

Usage

```
sector_profile_upstream(
  companies,
  scenarios,
  inputs,
  low_threshold = ifelse(scenarios$year == 2030, 1/9, 1/3),
  high_threshold = ifelse(scenarios$year == 2030, 2/9, 2/3)
)
```

Arguments

`companies`, `scenarios`, `inputs`	A dataframe like the dataset with a matching name in tiltToyData (see Reference).
`low_threshold`	A numeric value to segment low and medium emission profile products.
`high_threshold`	A numeric value to segment medium and high emission profile products.

Value

A data frame with the column `companies_id`, and the nested columns`product` and `company` holding the outputs at product and company level. Unnesting `product` yields a data frame with at least columns `companies_id`, `grouped_by`, `risk_category`. Unnesting `company` yields a data frame with at least columns `companies_id`, `grouped_by`, `risk_category`, `value`. Any column in the input datasets ending with `*rowid` is also passed as is to the output at product level. The exception is any column named exactly `rowid`– which is a reserved name and throws an error. Note this feature makes no sense at company level because potentially multiple rows in the input datasets are summarized into a single row in the output at company level.

See Also

Other main functions: `emissions_profile_upstream()`, `emissions_profile()`, `sector_profile()`

Examples

```
library(tiltIndicator)
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_sector_profile_upstream_companies())
scenarios <- read_csv(toy_sector_profile_any_scenarios())
inputs <- read_csv(toy_sector_profile_upstream_products())

both <- sector_profile_upstream(companies, scenarios, inputs)
both

both |> unnest_product()

both |> unnest_company()
```

`summarize_range` *Summarize the range of a column by groups*

Description

This function is a shortcut to `dplyr::summarize(data, min = min(x), max = max(x))`.

Usage

```
summarize_range(data, col, .by = NULL)
```

Arguments

`data`	A dataframe.
`col`	Unquoted expression giving the name of a column in `data`.

`.by` **[Experimental]**
<tidy-select> Optionally, a selection of columns to group by for just this operation, functioning as an alternative to `group_by()`. For details and examples, see ?dplyr_by.

Value

A dataframe:

- The rows come from the underlying groups.
- The columns come from the grouping keys plus the new columns `min` and `max`.
- The groups are dropped.

See Also

`dplyr::summarize()`

Other helpers: `jitter_range()`, `unnest_product()`

Examples

```
library(tibble)

data <- tibble(x = 1:4, group = c(1, 1, 2, 2))
data

summarize_range(data, x, .by = group)
```

---

unnest_product *Unnest product- and company-level results*

Description

Unnest product- and company-level results

Usage

```
unnest_product(data)

unnest_company(data)
```

Arguments

`data`	A nested data frame, e.g. the output of `sector_profile()`.

Value

A data frame.

See Also

Other helpers: jitter_range(), summarize_range()

Examples

```
library(tiltToyData)
library(readr)
options(readr.show_col_types = FALSE)

companies <- read_csv(toy_sector_profile_companies())
scenarios <- read_csv(toy_sector_profile_any_scenarios())

both <- sector_profile(companies, scenarios)
both

both |> unnest_product()

both |> unnest_company()
```

Index