From published tables to rtauargus input: an (updated) automated approach

C. Baudry, J. Jamme and N. Ferrer-Pradines

02-24-25

Reminder from previous presentation

During the March 2024 meeting, we presented the first developments of this tool. It allows, from a simple metadata table, to:

- Handle hierarchies
- Split the list of tables in linked table clusters
- Detect and regroup tables that are included in others (allowing to produce inclusion graphs)

Changes since previous presentation

- the analysis functions are now part of `rtauargus` package as a pre-release: https://github.com/InseeFrLab/rtauargus/releases/tag/v-1.2.999-dev
- the input metadata structure was simplified and column names of this dataframe are now expected in English
- in order to keep `rtauargus` dependencies scarce, inclusion graphs are no longer part of the function but can still be produced autonomously

Changes since previous presentation

- intermediatary functions are now hidden for the user but their outputs can be accessed through the option `verbose=TRUE`
- the package includes an example of metadata dataframe
- vignettes in French and English are available to guide users step by step, including examples for inclusion graphs: https://inseefrlab.github.io/rtauargus/articles/auto_metadata.html
- a new function allows the metadata to be a Eurostat template

Formal description of a table

table_name : indicator ⊗ {grouping_var_1 x grouping_var_2}

Example :

T1: turnover_pizzas ⊗ {nuts2 x size}

	BE10	BE21	...	Total
wf1	10	8	…	50
wf2	10	12	…	50
Total	20	20	60	100

List of published tables

- T1: to_pizzas ⊗ {nuts2 x size}
- T2: to_pizzas ⊗ {nuts3 x size}
- T3: to_pizzas ⊗ {a10 x nuts2}
- T4: to_pizzas ⊗ {a10 x nuts3}
- T5: to_pizzas ⊗ {a21 x nuts2}
- T6: to_pizzas ⊗ {a21 x nuts3}
- T7: to_batavia ⊗ {a10 x size}
- T8: to_batavia ⊗ {a10 x cj}
- T9: to_arugula ⊗ {a10 x size}
- T10: to_arugula ⊗ {a10 x cj}
- T11: to_lettuce ⊗ {a10 x size}
- T12: to_lettuce ⊗ {a10 x cj}

List of published tables

- T1: to_pizzas ⊗ {nuts2 x size}
- T2: to_pizzas ⊗ {nuts3 x size}
- T3: to_pizzas ⊗ {a10 x nuts2}
- T4: to_pizzas ⊗ {a10 x nuts3}
- T5: to_pizzas ⊗ {a21 x nuts2}
- T6: to_pizzas ⊗ {a21 x nuts3}
- T7: to_batavia ⊗ {a10 x size}
- T8: to_batavia ⊗ {a10 x cj}
- T9: to_arugula ⊗ {a10 x size}
- T10: to_arugula ⊗ {a10 x cj}
- T11: to_lettuce ⊗ {a10 x size}
- T12: to_lettuce ⊗ {a10 x cj}

List of published tables

- T1: to_pizzas ⊗ { nuts2 x size}
- T2: to_pizzas ⊗ { nuts3 x size}
- T3: to_pizzas ⊗ { a10 x nuts2}
- T4: to_pizzas ⊗ { a10 x nuts3}
- T5: to_pizzas ⊗ { a21 x nuts2}
- T6: to_pizzas ⊗ { a21 x nuts3}
- T7: to_batavia ⊗ { a10 x size}
- T8: to_batavia ⊗ { a10 x cj}
- T9: to_arugula ⊗ { a10 x size}
- T10: to_arugula ⊗ { a10 x cj}
- T11: to_lettuce ⊗ { a10 x size}
- T12: to_lettuce ⊗ { a10 x cj}

List of tables to protect

- T1_T2: to_pizzas $\otimes$ { HRC_NUTS x size}
- T3_T4_T5_T6: to_pizzas $\otimes$ { HRC_NAF x HRC_NUTS}
- T8_T10_T12: to_lettuce $\otimes$ { HRC_NAF x diversity x HRC_lettuce^{h}}
- T7_T9_T11: to_lettuce $\otimes$ { HRC_NAF x size x HRC_lettuce^{h} }

With *HRC_lettuce*^{h} a holding variable.

Analysis automation steps

1. The user enters the metadata for the tables to be published in the required format.

Then the program:

1. Identifies hierarchies and renames variables accordingly
2. Breaks down the request into independent sub-requests (clusters)
3. Detects overlapping tables
4. Groups tables included in each other into a single table
5. Creates a summary of the tables needing protection

Metadata set included in package

```
data(metadata_pizza_lettuce)

md_pizza_lettuce <- metadata_pizza_lettuce %>%
  mutate(field="FR_ent_23")

str(md_pizza_lettuce)
```

```
## 'data.frame':    12 obs. of  9 variables:
##  $ table_name    : chr  "T1" "T2" "T3" "T4" ...
##  $ field         : chr  "FR_ent_23" "FR_ent_23" "FR_ent_23" "FR_ent_23" ...
##  $ hrc_field     : logi  NA NA NA NA NA NA ...
##  $ indicator     : chr  "to_pizza" "to_pizza" "to_pizza" "to_pizza" ...
##  $ hrc_indicator : chr  NA NA NA NA ...
##  $ spanning_1    : chr  "nuts2" "nuts3" "a10" "a10" ...
##  $ hrc_spanning_1: chr  "hrc_nuts" "hrc_nuts" "hrc_naf" "hrc_naf" ...
##  $ spanning_2    : chr  "size" "size" "nuts2" "nuts3" ...
##  $ hrc_spanning_2: chr  NA NA "hrc_nuts" "hrc_nuts" ...
```

Metadata set included in package

table_name	field	indicator	hrc_indicator	spanning_1	hrc_spanning_1	spanning_2	hrc_spanning_2
T1	FR_ent_23	to_pizza	NA	nuts2	hrc_nuts	size	NA
T2	FR_ent_23	to_pizza	NA	nuts3	hrc_nuts	size	NA
T3	FR_ent_23	to_pizza	NA	a10	hrc_naf	nuts2	hrc_nuts
T4	FR_ent_23	to_pizza	NA	a10	hrc_naf	nuts3	hrc_nuts
T5	FR_ent_23	to_pizza	NA	a21	hrc_naf	nuts2	hrc_nuts
T6	FR_ent_23	to_pizza	NA	a21	hrc_naf	nuts3	hrc_nuts
T7	FR_ent_23	to_batavia	hrc_lettuce	a10	hrc_naf	size	NA
T8	FR_ent_23	to_batavia	hrc_lettuce	a10	hrc_naf	cj	NA
T9	FR_ent_23	to_arugula	hrc_lettuce	a10	hrc_naf	size	NA
T10	FR_ent_23	to_arugula	hrc_lettuce	a10	hrc_naf	cj	NA
T11	FR_ent_23	to_lettuce	hrc_lettuce	a10	hrc_naf	size	NA
T12	FR_ent_23	to_lettuce	hrc_lettuce	a10	hrc_naf	cj	NA

All-in-one function

```
cluster_id_dataframe <- analyse_metadata(md_pizza_lettuce,
                                         verbose = FALSE)
```

```
kable(cluster_id_dataframe %>% select(-starts_with("hrc"))) %>%
  style_pres()
```

cluster	table_name	field	indicator	spanning_1	spanning_2	spanning_3
FR_ent_23.hrc_lettuce	T10.T12.T8	FR_ent_23	LETTUCE	HRC_NAF	cj	HRC_LETTUCE^h
FR_ent_23.hrc_lettuce	T11.T7.T9	FR_ent_23	LETTUCE	HRC_NAF	size	HRC_LETTUCE^h
FR_ent_23.to_pizza	T1.T2	FR_ent_23	to_pizza	HRC_NUTS	size	NA
FR_ent_23.to_pizza	T3.T4.T5.T6	FR_ent_23	to_pizza	HRC_NAF	HRC_NUTS	NA

All-in-one function

```
kable(cluster_id_dataframe %>% select(-starts_with("span"))) %>%
  style_pres()
```

cluster	table_name	field	indicator	hrc_spanning_1	hrc_spanning_2	hrc_spanning_3
FR_ent_23.hrc_lettuce	T10.T12.T8	FR_ent_23	LETTUCE	hrc_naf	NA	hrc_lettuce
FR_ent_23.hrc_lettuce	T11.T7.T9	FR_ent_23	LETTUCE	hrc_naf	NA	hrc_lettuce
FR_ent_23.to_pizza	T1.T2	FR_ent_23	to_pizza	hrc_nuts	NA	NA
FR_ent_23.to_pizza	T3.T4.T5.T6	FR_ent_23	to_pizza	hrc_naf	hrc_nuts	NA

Handle hierarchies

```
detailed_analysis <- analyse_metadata(md_pizza_lettuce, verbose = TRUE)
```

```
detailed_analysis$identify_hrc %>% head(14) %>%  kable() %>% style_pres()
```

table_name	field	hrc_field	indicator	hrc_indicator	spanning	hrc_spanning
T1	FR_ent_23	NA	to_pizza	NA	HRC_NUTS	hrc_nuts
T1	FR_ent_23	NA	to_pizza	NA	size	NA
T10	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_NAF	hrc_naf
T10	FR_ent_23	NA	LETTUCE	hrc_lettuce	cj	NA
T10	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_LETTUCE^h	hrc_lettuce
T11	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_NAF	hrc_naf
T11	FR_ent_23	NA	LETTUCE	hrc_lettuce	size	NA
T11	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_LETTUCE^h	hrc_lettuce
T12	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_NAF	hrc_naf
T12	FR_ent_23	NA	LETTUCE	hrc_lettuce	cj	NA
T12	FR_ent_23	NA	LETTUCE	hrc_lettuce	HRC_LETTUCE^h	hrc_lettuce
T2	FR_ent_23	NA	to_pizza	NA	HRC_NUTS	hrc_nuts
T2	FR_ent_23	NA	to_pizza	NA	size	NA
T3	FR_ent_23	NA	to_pizza	NA	HRC_NAF	hrc_naf

Split in clusters

Split the list of tables in independant clusters, i.e. linked tables clusters.

Independant tables do not need to be treated together. Call Tau-Argus multiple times independently.

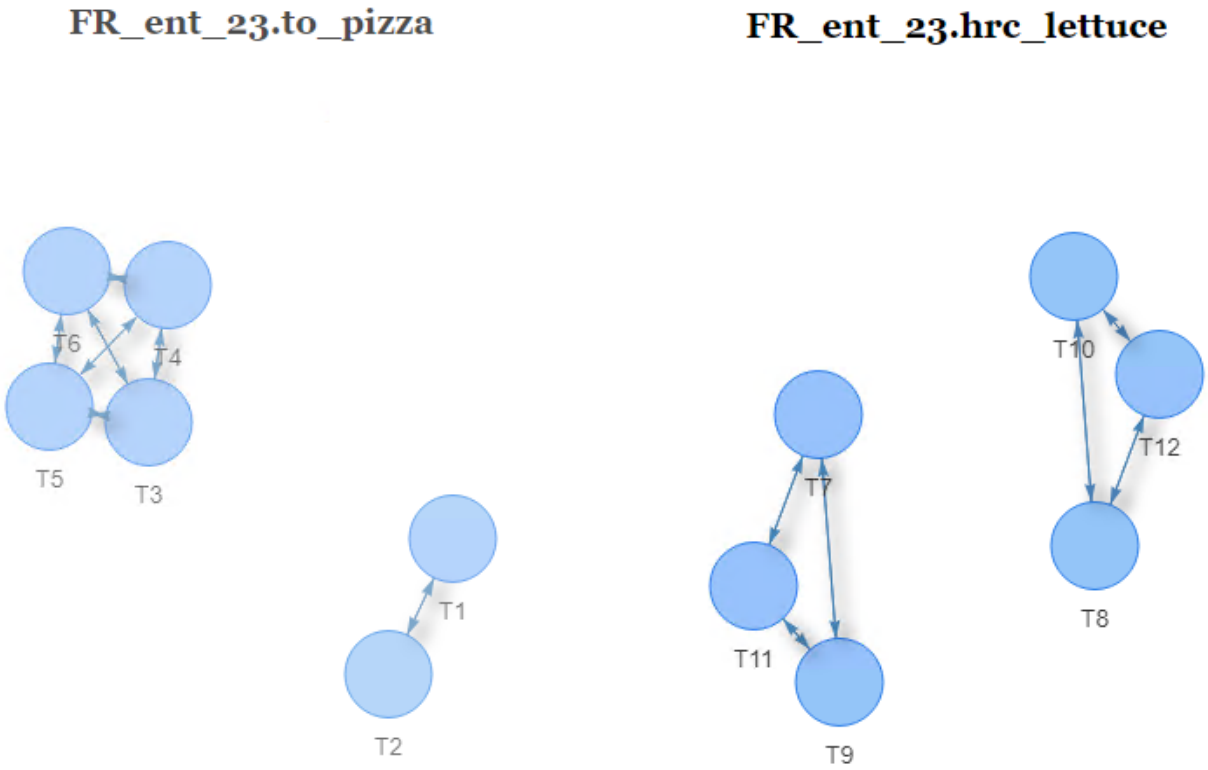
```
lSplitlettuce <- detailed_analysis$split_in_clusters
```

```
names(lSplitlettuce)
```

```
## [1] "FR_ent_23.hrc_lettuce" "FR_ent_23.to_pizza"
```

Inclusion graphs

Tables included in others are documented in data frames `detailed_analysis$create_edges` which can be represented with inclusion graphs:



Summary

From the **12 published tables** defined in the metadata file, the program suggests to **protect 4 rearranged tables**.

For this particular example `rtauargus::tab_multi_manager()` would be called twice:

- Once for the salad turnover tables
- Once for the pizza turnover tables, for example:

```
safe_tables <- tab_multi_manager(
  list_tables = list(T1T2 = pizza_nuts_size,
                     T3T4T5T6 = pizza_nace_nuts),
  list_explanatory_vars = list(T1T2 = c("NUTS", "size"),
                               T3T4T5T6 = c("act", "NUTS")),
  hrc = c(NUTS = "hrc_nuts.hrc", act = "hrc_nace.hrc"),
  freq = "N_OBS", value = "to", totcode = "Total",
  secret_var = "is_secret_prim")
```

Analyse from a Eurostat template

- The template is a .csv file giving the structure of the XML files sent by the NSIs to Eurostat.
- It is usually available on CIRCABC
- Some columns are pre-filled, describing all the cells expected by Eurostat. The analysis only focuses on these columns.
- The idea is to deduct from the cells description the tables to protect.
- It works with most Eurostat templates but not all (to-date).

Analyse from a Eurostat template

- New function `template_formatted()` automatically analyses the cells of the file and returns the list of tables to protect
- The user needs to classify the variables: indicator, spanning and field variables.
- For the spanning variables, it is necessary to specify the modality corresponding to the total.

```
format_template <- template_formatted(
  data = enterprise_template,
  indicator_column = "INDICATOR",
  spanning_var_tot = list(
    ACTIVITY = "BTSXO_S94",
    NUMBER_EMPL = "_T",
    LEGAL_FORM = "_T"),
  field_columns = c("TIME_PERIOD")
  )
```

Further work

- Keep on testing the program on different lists of tables, especially for the non-nested hierarchies option recently added
- Check the hierarchies provided (nested, non-nested)
- Automatically generate the input tables for `rtauargus` functions `tab_rtauargus()` and `tab_multi_manager()` using the output of this analysis program
- Release `rtauargus` with the addition of the two analysis functions