

The *eurostat* package

R tools to access open data from Eurostat database

Search and download

Data in the Eurostat database is stored in tables. Each table has an identifier, a short table_code, and a description (e.g. *tsdtr420* - People killed in road accidents).

Key eurostat functions allow to find the table_code, download the eurostat table and polish labels in the table.

Find the table code

The `search_eurostat(pattern,...)` function scans the directory of Eurostat tables and returns codes and descriptions of tables that match pattern.

```
library("eurostat")
query <- search_eurostat("road", type = "table")
query[1:3,1:2]
##           title      code
## 1 Goods transport by road ttr00005
## 2 People killed in road accidents tsdtr420
## 3 Enterprises with broadband access tin00090
```

Download the table

The `get_eurostat(id, time_format = "date", filters = "none", type = "code", cache = TRUE, ...)` function downloads the requested table from the Eurostat bulk download facility or from The Eurostat Web Services JSON API (if `filters` are defined). Downloaded data is cached (if `cache=TRUE`). Additional arguments define how to read the time column (`time_format`) and if table dimensions shall be kept as codes or converted to labels (`type`).

```
dat <- get_eurostat(id="tsdtr420", time_format="num")
head(dat)
##   unit  sex  geo  time values
## 1  NR    T   AT  1999   1079
## 2  NR    T   BE  1999   1397
## 3  NR    T   CZ  1999   1455
## 4  NR    T   DK  1999    514
## 5  NR    T   EL  1999   2116
## 6  NR    T   ES  1999   5738
```

Add labels

The `label_eurostat(x, lang = "en", ...)` gets definitions for Eurostat codes and replace them with labels in given language ("en", "fr" or "de").

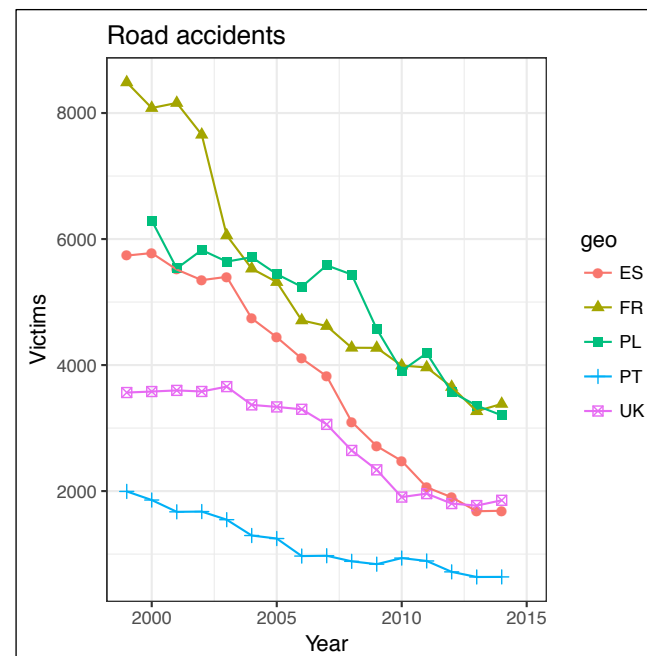
```
dat <- label_eurostat(dat)
head(dat)
##   unit  sex  geo  time values
## 1 Number Total  Austria 1999   1079
## 2 Number Total  Belgium 1999   1397
## 3 Number Total Czech Republic 1999   1455
## 4 Number Total  Denmark 1999    514
## 5 Number Total  Greece 1999   2116
## 6 Number Total  Spain 1999   5738
```

eurostat and plots

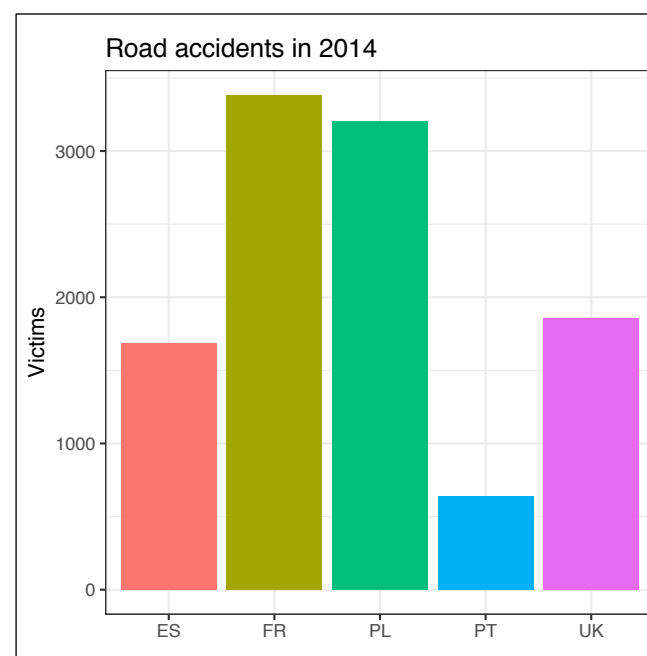
The `get_eurostat()` function returns tibbles in the long format. Packages `dplyr` and `tidyr` are well suited to transform these objects. The `ggplot2` package is well suited to plot these objects.

```
t1 <- get_eurostat("tsdtr420", filters =
  list(geo = c("UK", "FR", "PL", "ES", "PT")))
```

```
library("ggplot2")
ggplot(t1, aes(x = time, y = values, color = geo,
  group = geo, shape = geo)) +
  geom_point(size = 2) +
  geom_line() + theme_bw() +
  labs(title="Road accidents", x = "Year", y = "Victims")
```



```
library("dplyr")
t2 <- t1 %>% filter(time == "2014-01-01")
ggplot(t2, aes(geo, values, fill=geo)) +
  geom_bar(stat = "identity") + theme_bw() +
  theme(legend.position = "none") +
  labs(title="Road accidents in 2014", x="", y="Victims")
```



eurostat and maps

Fetch and process data

There are three functions to work with geospatial data from GISCO. The `get_eurostat_geospatial()` returns preprocessed spatial data as sp-objects or as data frames. The `merge_eurostat_geospatial()` both downloads and merges the geospatial data with a preloaded tabular data. The `cut_to_classes()` is a wrapper for `cut()` - function and is used for categorizing data for maps with tidy labels.

```
library("eurostat")
library("dplyr")
```

```
fertility <- get_eurostat("demo_r_frate3") %>%
  filter(time == "2014-01-01") %>%
  mutate(cat = cut_to_classes(values, n=7, decimals=1))
```

```
mapdata <- merge_eurostat_geodata(fertility,
  resolution = "20")
```

```
head(select(mapdata, geo, values, cat, long, lat, order, id))
##   geo values  cat  long  lat order id
## 1 AT124  1.39 1.3 ~< 1.5 15.54245 48.90770 214 10
## 2 AT124  1.39 1.3 ~< 1.5 15.75363 48.85218 215 10
## 3 AT124  1.39 1.3 ~< 1.5 15.88763 48.78511 216 10
## 4 AT124  1.39 1.3 ~< 1.5 15.81535 48.69270 217 10
## 5 AT124  1.39 1.3 ~< 1.5 15.94094 48.67173 218 10
## 6 AT124  1.39 1.3 ~< 1.5 15.90833 48.59815 219 10
```

Draw a cartogram

The object returned by `merge_eurostat_geospatial()` are ready to be plotted with ggplot2 package. The `coord_map()` function is useful to set the projection while `labs()` adds annotations to the plot.

```
library("ggplot2")
ggplot(mapdata, aes(x = long, y = lat, group = group)) +
  geom_polygon(aes(fill=cat), color="grey", size = .1) +
  scale_fill_brewer(palette = "RdYlGn") +
  labs(title="Fertility rate, by NUTS-3 regions, 2014",
    subtitle="Avg. number of live births per woman",
    fill="Total fertility rate(%)") + theme_light() +
  coord_map(xlim=c(-12,44), ylim=c(35,67))
```

