Tutorial (vignette) for the eurostat R package

2018-03-28

R Tools for Eurostat Open Data

This rOpenGov R package provides tools to access Eurostat database, which you can also browse on-line for the data sets and documentation. For contact information and source code, see the package website.

Installation

Release version (CRAN):

```
install.packages("eurostat")
```

Development version (Github):

```
library(devtools)
install_github("ropengov/eurostat")
```

Overall, the eurostat package includes the following functions:

```
clean_eurostat_cache    Clean Eurostat Cache
cut_to_classes          Cuts the Values Column into Classes and
                        Polishes the Labels
dic_order               Order of Variable Levels from Eurostat
                        Dictionary.
eu_countries            Countries and Country Codes
eurostat-package        R Tools for Eurostat open data
eurotime2date           Date Conversion from Eurostat Time Format
eurotime2num            Conversion of Eurostat Time Format to Numeric
get_eurostat            Read Eurostat Data
get_eurostat_dic        Download Eurostat Dictionary
get_eurostat_geospatial
                        Download Geospatial Data from GISCO
get_eurostat_json       Get Data from Eurostat API in JSON
get_eurostat_raw        Download Data from Eurostat Database
get_eurostat_toc        Download Table of Contents of Eurostat Data
                        Sets
harmonize_country_code
                        Harmonize Country Code
label_eurostat          Get Eurostat Codes
merge_eurostat_geodata
                        Merge Geospatial GISCO Data with Eurostat
                        data_frame
search_eurostat         Grep Datasets Titles from Eurostat
tgs00026                Auxiliary Data
```

Finding data

Function `get_eurostat_toc()` downloads a table of contents of eurostat datasets. The values in column 'code' should be used to download a selected dataset.

```
# Load the package
library(eurostat)
library(rvest)

# Get Eurostat data listing
toc <- get_eurostat_toc()

# Check the first items
library(knitr)
kable(head(toc))
```

title	code	type	last update of data	last table structure
Database by themes	data	folder	NA	NA
General and regional statistics	general	folder	NA	NA
European and national indicators for short-term analysis	euroind	folder	NA	NA
Business and consumer surveys (source: DG ECFIN)	ei_bcs	folder	NA	NA
Consumer surveys (source: DG ECFIN)	ei_bcs_cs	folder	NA	NA
Consumers - monthly data	ei_bsco_m	dataset	27.03.2018	27.03.2018

With `search_eurostat()` you can search the table of contents for particular patterns, e.g. all datasets related to *passenger transport*. The kable function to produces nice markdown output. Note that with the `type` argument of this function you could restrict the search to for instance datasets or tables.

```
# info about passengers
kable(head(search_eurostat("passenger transport")))
```

title
Volume of passenger transport relative to GDP
Modal split of passenger transport
Railway transport - total annual passenger transport (1 000 pass., million pkm)
Railway transport - passenger transport by type of transport (detailed reporting only) (1 000 pass.)
Railway transport - passenger transport by type of transport (detailed reporting only) (million pkm)
International railway passenger transport from the reporting country to the country of disembarkation (1 000 passengers)

Codes for the dataset can be searched also from the Eurostat database. The Eurostat database gives codes in the Data Navigation Tree after every dataset in parenthesis.

Downloading data

The package supports two of the Eurostats download methods: the bulk download facility and the Web Services' JSON API. The bulk download facility is the fastest method to download whole datasets. It is also often the only way as the JSON API has limitation of maximum 50 sub-indicators at a time and whole datasets usually exceeds that. To download only a small section of the dataset the JSON API is faster, as it allows to make a data selection before downloading.

A user does not usually have to bother with methods, as both are used via main function `get_eurostat()`. If only the table id is given, the whole table is downloaded from the bulk download facility. If also filters are defined the JSON API is used.

Here an example of indicator 'Modal split of passenger transport'. This is the percentage share of each mode of transport in total inland transport, expressed in passenger-kilometres (pkm) based on transport by passenger cars, buses and coaches, and trains. All data should be based on movements on national territory, regardless of the nationality of the vehicle. However, the data collection is not harmonized at the EU level.

Pick and print the id of the data set to download:

```
# For the original data, see
# http://ec.europa.eu/eurostat/tgm/table.do?tab=table&init=1&plugin=1&language=en&pcode=tsdtr210
id <- search_eurostat("Modal split of passenger transport",
                      type = "table")$code[1]
print(id)
```

[1] "t2020_rk310"

Get the whole corresponding table. As the table is annual data, it is more convient to use a numeric time variable than use the default date format:

```
dat <- get_eurostat(id, time_format = "num")
```

Investigate the structure of the downloaded data set:

```
str(dat)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2431 obs. of  5 variables:
##  $ unit   : Factor w/ 1 level "PC": 1 1 1 1 1 1 1 1 1 1 ...
##  $ vehicle: Factor w/ 3 levels "BUS_TOT","CAR",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ geo    : Factor w/ 35 levels "AT","BE","CH",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ time   : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ values : num  11 10.6 3.7 9.1 11.3 32.4 14.9 13.5 6 24.8 ...
```

```
kable(head(dat))
```

unit	vehicle	geo	time	values
PC	BUS_TOT	AT	1990	11.0
PC	BUS_TOT	BE	1990	10.6
PC	BUS_TOT	CH	1990	3.7
PC	BUS_TOT	DE	1990	9.1
PC	BUS_TOT	DK	1990	11.3
PC	BUS_TOT	EL	1990	32.4

Or you can get only a part of the dataset by defining `filters` argument. It should be named list, where names corresponds to variable names (lower case) and values are vectors of codes corresponding desidered series (upper case). For time variable, in addition to a `time`, also a `sinceTimePeriod` and a `lastTimePeriod` can be used.

```
dat2 <- get_eurostat(id, filters = list(geo = c("EU28", "FI"), lastTimePeriod=1), time_format = "num")
kable(dat2)
```

Replacing codes with labels

By default variables are returned as Eurostat codes, but to get human-readable labels instead, use a `type = "label"` argument.

```
dat12 <- get_eurostat(id, filters = list(geo = c("EU28", "FI"),
                                         lastTimePeriod = 1),
                      type = "label", time_format = "num")
kable(head(dat12))
```

Eurostat codes in the downloaded data set can be replaced with human-readable labels from the Eurostat dictionaries with the `label_eurostat()` function.

```
dat1 <- label_eurostat(dat)
kable(head(dat1))
```

unit	vehicle	geo	time	values
Percentage	Motor coaches, buses and trolley buses	Austria	1990	11.0
Percentage	Motor coaches, buses and trolley buses	Belgium	1990	10.6
Percentage	Motor coaches, buses and trolley buses	Switzerland	1990	3.7
Percentage	Motor coaches, buses and trolley buses	Germany (until 1990 former territory of the FRG)	1990	9.1
Percentage	Motor coaches, buses and trolley buses	Denmark	1990	11.3
Percentage	Motor coaches, buses and trolley buses	Greece	1990	32.4

The `label_eurostat()` allows conversion of individual variable vectors or variable names as well.

```
label_eurostat_vars(names(dat1))
```

Vehicle information has 3 levels. You can check them now with:

```
levels(dat1$vehicle)
```

Selecting and modifying data

EFTA, Eurozone, EU and EU candidate countries

To facilitate smooth visualization of standard European geographic areas, the package provides ready-made lists of the country codes used in the eurostat database for EFTA (efta_countries), Euro area (ea_countries), EU (eu_countries) and EU candidate countries (eu_candidate_countries). These can be used to select specific groups of countries for closer investigation. For conversions with other standard country coding systems, see the countrycode R package. To retrieve the country code list for EFTA, for instance, use:

```
data(efta_countries)
kable(efta_countries)
```

code	name
IS	Iceland
LI	Liechtenstein
NO	Norway
CH	Switzerland

EU data from 2012 in all vehicles:

```
dat_eu12 <- subset(dat1, geo == "European Union (28 countries)" & time == 2012)
kable(dat_eu12, row.names = FALSE)
```

unit vehicle geo time values —— ——— —- —— ——-

EU data from 2000 - 2012 with vehicle types as variables:

Reshaping the data is best done with `spread()` in `tidyr`.

```
library("tidyr")
dat_eu_0012 <- subset(dat, geo == "EU28" & time %in% 2000:2012)
dat_eu_0012_wide <- spread(dat_eu_0012, vehicle, values)
kable(subset(dat_eu_0012_wide, select = -geo), row.names = FALSE)
```

unit	time	BUS_TOT	CAR	TRN
PC	2000	10.4	82.4	7.2
PC	2001	10.2	82.7	7.1
PC	2002	9.9	83.3	6.8
PC	2003	9.9	83.5	6.7
PC	2004	9.8	83.4	6.8
PC	2005	9.8	83.3	6.9
PC	2006	9.7	83.2	7.1
PC	2007	9.8	83.1	7.1
PC	2008	9.9	82.8	7.4
PC	2009	9.3	83.6	7.1
PC	2010	9.4	83.5	7.2
PC	2011	9.4	83.2	7.3
PC	2012	9.5	82.8	7.7

Train passengers for selected EU countries in 2000 - 2012

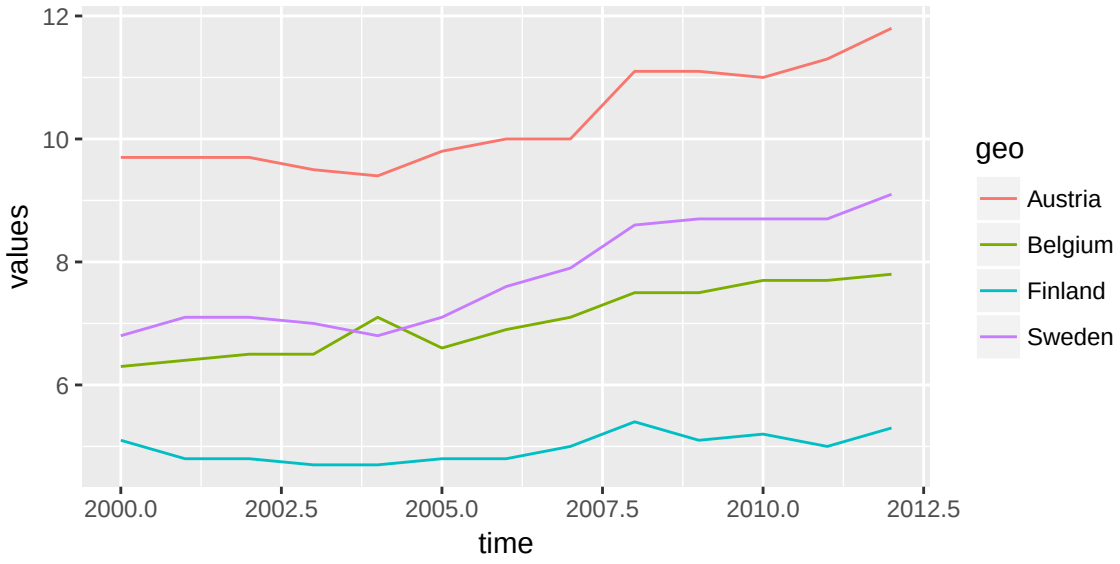
```
dat_trains <- subset(datl, geo %in% c("Austria", "Belgium", "Finland", "Sweden")
                     & time %in% 2000:2012
                     & vehicle == "Trains")

dat_trains_wide <- spread(dat_trains, geo, values)
kable(subset(dat_trains_wide, select = -vehicle), row.names = FALSE)
```

unit	time	Austria	Belgium	Finland	Sweden
Percentage	2000	9.7	6.3	5.1	6.8
Percentage	2001	9.7	6.4	4.8	7.1
Percentage	2002	9.7	6.5	4.8	7.1
Percentage	2003	9.5	6.5	4.7	7.0
Percentage	2004	9.4	7.1	4.7	6.8
Percentage	2005	9.8	6.6	4.8	7.1
Percentage	2006	10.0	6.9	4.8	7.6
Percentage	2007	10.0	7.1	5.0	7.9
Percentage	2008	11.1	7.5	5.4	8.6
Percentage	2009	11.1	7.5	5.1	8.7
Percentage	2010	11.0	7.7	5.2	8.7
Percentage	2011	11.3	7.7	5.0	8.7
Percentage	2012	11.8	7.8	5.3	9.1

Visualization

Visualizing train passenger data with `ggplot2`:

```
library(ggplot2)
p <- ggplot(dat_trains, aes(x = time, y = values, colour = geo))
p <- p + geom_line()
print(p)
```

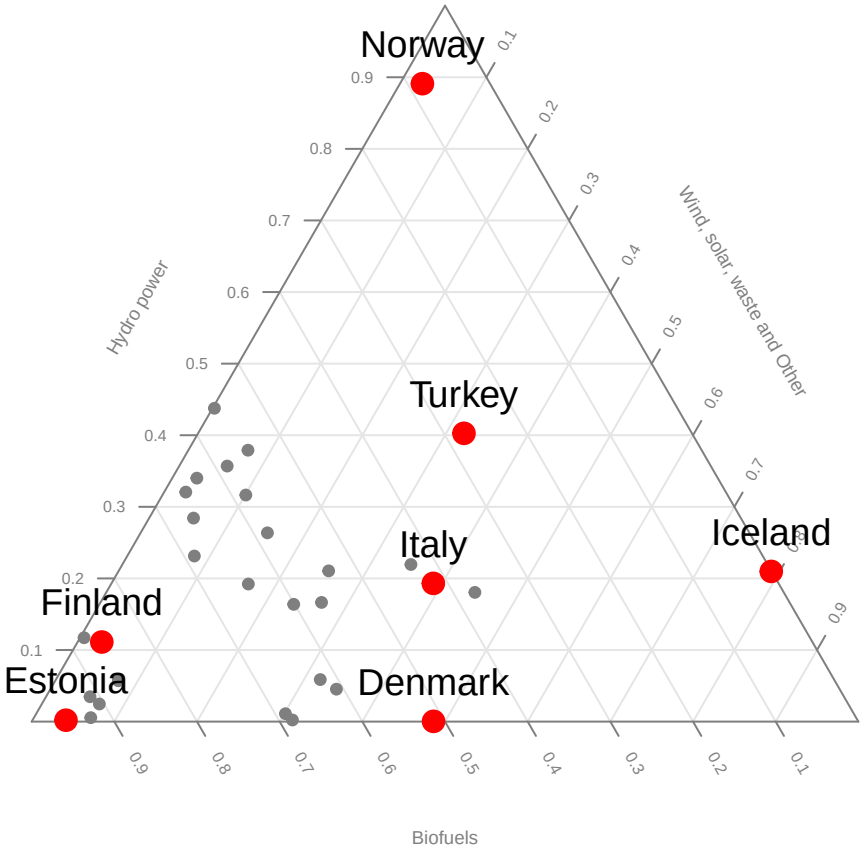


Triangle plot

Triangle plot is handy for visualizing data sets with three variables.

```
library(tidyr)
library(plotrix)
library(eurostat)
library(dplyr)
library(tidyr)

# All sources of renewable energy are to be grouped into three sets
 dict <- c("Solid biofuels (excluding charcoal)" = "Biofuels",
 "Biogasoline" = "Biofuels",
 "Other liquid biofuels" = "Biofuels",
 "Biodiesels" = "Biofuels",
 "Biogas" = "Biofuels",
 "Hydro power" = "Hydro power",
 "Tide, Wave and Ocean" = "Hydro power",
 "Solar thermal" = "Wind, solar, waste and Other",
 "Geothermal Energy" = "Wind, solar, waste and Other",
 "Solar photovoltaic" = "Wind, solar, waste and Other",
 "Municipal waste (renewable)" = "Wind, solar, waste and Other",
 "Wind power" = "Wind, solar, waste and Other",
 "Bio jet kerosene" = "Wind, solar, waste and Other")
# Some cleaning of the data is required
 energy3 <- get_eurostat("ten00081") %>%
 label_eurostat(dat) %>%
 filter(time == "2013-01-01",
```

```
 product != "Renewable energies") %>%
 mutate(nproduct = dict[as.character(product)], # just three categories
 geo = gsub(geo, pattern=" \\(.*", replacement="")) %>%
 select(nproduct, geo, values) %>%
 group_by(nproduct, geo) %>%
 summarise(svalue = sum(values)) %>%
 group_by(geo) %>%
 mutate(tvalue = sum(svalue),
 svalue = svalue/sum(svalue)) %>%
 filter(tvalue > 1000) %>% # only large countries
 spread(nproduct, svalue)

# Triangle plot
 par(cex=0.75, mar=c(0,0,0,0))
 positions <- plotrix::triax.plot(as.matrix(energy3[, c(3,5,4)]),
                     show.grid = TRUE,
                     label.points= FALSE, point.labels = energy3$geo,
                     col.axis="gray50", col.grid="gray90",
                     pch = 19, cex.axis=0.8, cex.ticks=0.7, col="grey50")

 # Larger labels
 ind <- which(energy3$geo %in%  c("Norway", "Iceland","Denmark","Estonia", "Turkey", "Italy", "Finland"
 df <- data.frame(positions$xypos, geo = energy3$geo)
 points(df$x[ind], df$y[ind], cex=2, col="red", pch=19)
 text(df$x[ind], df$y[ind], df$geo[ind], adj = c(0.5,-1), cex=1.5)
```



Maps

Disposable income of private households by NUTS 2 regions at 1:60mln resolution using tmap

The mapping examples below use `tmap` package.

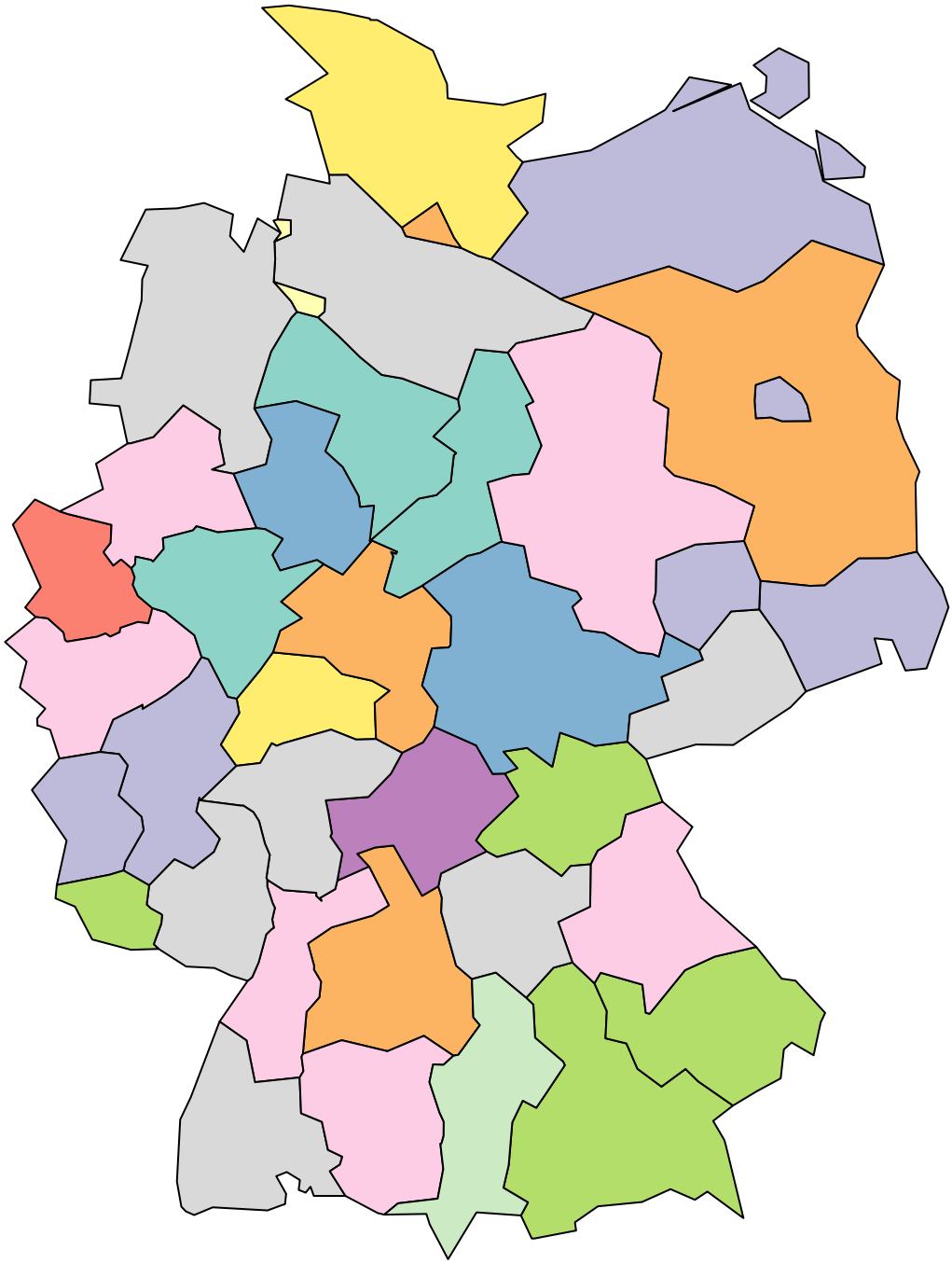
```
library(dplyr)
library(eurostat)
library(sf)
library(tmap)

# Use sf object downloaded from GISCO
gisco <- get_eurostat_geospatial(resolution = 60)
```

```
##
##       COPYRIGHT NOTICE
##
##       When data downloaded from this page
##       <http://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical
##       is used in any printed or electronic publication,
##       in addition to any other provisions
##       applicable to the whole Eurostat website,
##       data source will have to be acknowledged
##       in the legend of the map and
##       in the introductory page of the publication
##       with the following copyright notice:
##
##       - EN: (C) EuroGeographics for the administrative boundaries
##       - FR: (C) EuroGeographics pour les limites administratives
##       - DE: (C) EuroGeographics bezuglich der Verwaltungsgrenzen
##
##       For publications in languages other than
##       English, French or German,
##       the translation of the copyright notice
##       in the language of the publication shall be used.
##
##       If you intend to use the data commercially,
##       please contact EuroGeographics for
##       information regarding their licence agreements.
##
```

```
## Reading cache file /tmp/RtmpGkEYMz/eurostat/cache_geg60.RData
```

```
## Map with resolution 1: 60  read from cache file:  /tmp/RtmpGkEYMz/eurostat/cache_geg60.RData
```

```
# Since sf objects inherit directly from data.frame and sf has implemented
# the necessary dplyr verbs, this (plotting NUTS2-level units) just works
map0 <- gisco %>%
  dplyr::filter(STAT_LEVL_ == 2 & grepl("DE", NUTS_ID)) %>%
  dplyr::select(NUTS_ID) %>%
  plot()
```



Another example on map data manipulation

```
# Map example 1
# Load example data set
data("tgs00026")
# Can be retrieved from the eurostat service with:
# tgs00026 <- get_eurostat("tgs00026", time_format = "raw")

# Convert
```

```
euro_sf <- tgs00026 %>%
  # subset to have only a single row per geo
  dplyr::filter(time == 2010, nchar(as.character(geo)) == 4) %>%
  # categorise
  dplyr::mutate(cat = cut_to_classes(values, n = 5)) %>%
  # merge with the spatial data
  # NOTE! geo becomes character
  dplyr::inner_join(gisco, ., by = c("NUTS_ID" = "geo")) %>%
  # use a proper coordinate reference syste (CRS):
  # epsg projection 3035 - etrs89 / etrs-laea
  sf::st_transform("+init=epsg:3035")
```

```
## Warning: Column `NUTS_ID`/`geo` joining factors with different levels,
## coercing to character vector
```

Load example data (map)

```
data(Europe)
```

Construct the map

```
map1 <- tmap::tm_shape(Europe) +
  tmap::tm_fill(col = "lightgrey") +
  tmap::tm_shape(euro_sf) +
  tmap::tm_grid(labels.inside.frame = FALSE) +
  tmap::tm_polygons("income", title = "Disposable household\nincomes in 2010",
                    palette = "Oranges") +
  tmap::tm_format_Europe()
print(map1)
```

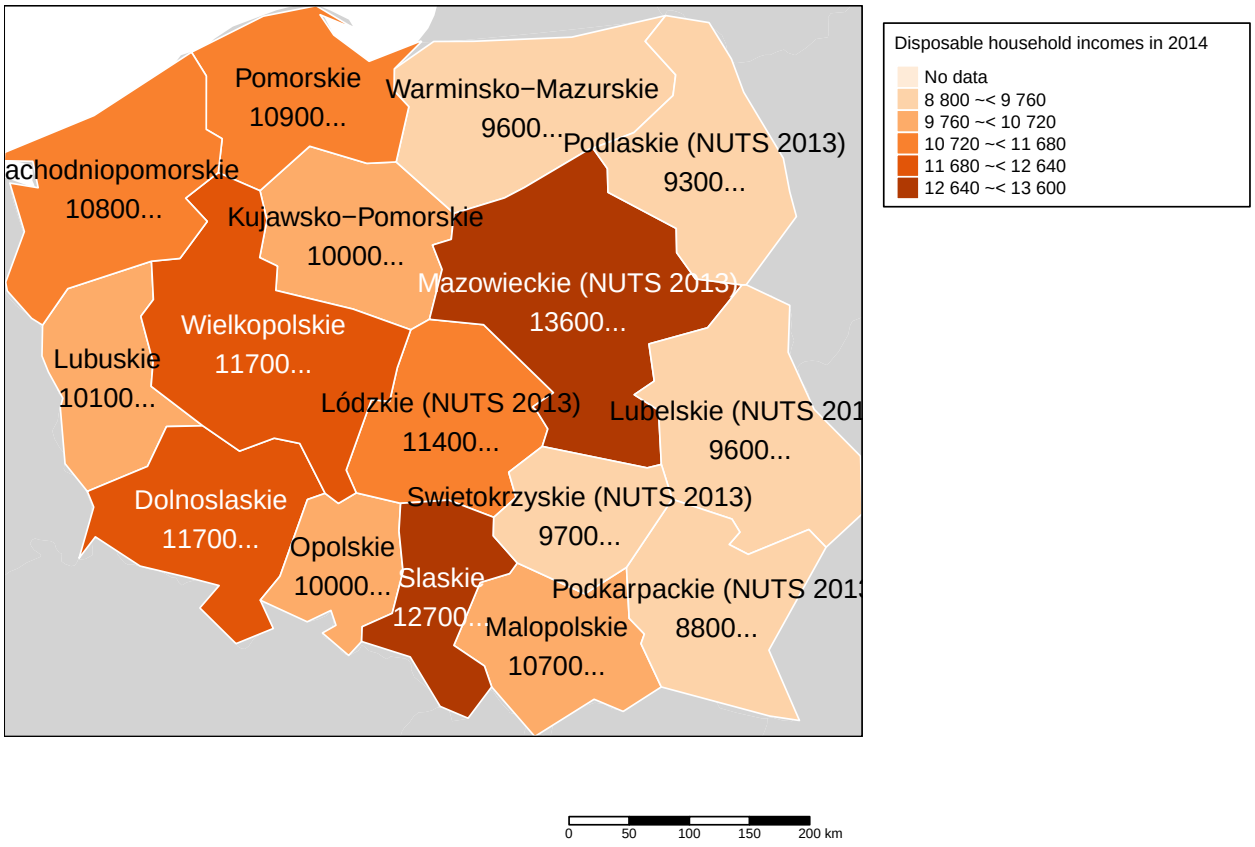
Interactive maps can be generated as well

```
# Interactive
tmap_mode("view")
map1

# Set the mode back to normal plotting
tmap_mode("plot")
print(map1)
```

Disposable income of private households by NUTS 2 regions in Poland with labels at 1:1mln resolution using tmap

```
library(eurostat)
library(dplyr)
library(ggplot2)
library(RColorBrewer)

# Downloading and manipulating the tabular data
euro_sf2 <- tgs00026 %>%
  # subsetting to year 2014 and NUTS-3 level
  dplyr::filter(time == 2014, nchar(as.character(geo)) == 4, grepl("PL",geo)) %>%
  # label the single geo column
  mutate(label = paste0(label_eurostat(.)[["geo"]], "\n", values, "€"),
         income = cut_to_classes(values)) %>%
```

```
  # merge with geodata
  dplyr::inner_join(gisco, ., by = c("NUTS_ID" = "geo")) %>%
  # use a proper coordinate reference syste (CRS):
  # epsg projection 3035 - etrs89 / etrs-laea
  sf::st_transform("+init=epsg:3035")


# plot map
map2 <- tm_shape(Europe) +
  tm_fill("lightgrey") +
  tm_shape(euro_sf2, is.master = TRUE) +
  tm_polygons("income", title = "Disposable household incomes in 2014",
              palette = "Oranges", border.col = "white") +
  tm_text("label", just = "center") +
  tm_scale_bar() +
  tm_format_Europe(legend.outside = TRUE, attr.outside = TRUE)
map2
```



SDMX

Eurostat data is available also in the SDMX format. The eurostat R package does not provide custom tools for this but the generic rsdmx R package can be used to access data in that format when necessary:

```
library(rsdmx)

# Data set URL
```

```
url <- "http://ec.europa.eu/eurostat/SDMX/diss-web/rest/data/cdh_e_fos/..PC.FOS1.BE/?startperiod=2005&e

# Read the data from eurostat
d <- readSDMX(url)

# Convert to data frame and show the first entries
df <- as.data.frame(d)

kable(head(df))
```

UNIT	Y_GRAD	FOS07	GEO	FREQ	obsTime	obsValue	OBS_STATUS
PC	TOTAL	FOS1	BE	A	2009	NA	na
PC	TOTAL	FOS1	BE	A	2006	NA	na
PC	Y_GE1990	FOS1	BE	A	2009	43.75	NA
PC	Y_GE1990	FOS1	BE	A	2006	NA	na

Further examples

For further examples, see the package homepage.

Citations and related work

Citing the data sources

Eurostat data: cite Eurostat.

Administrative boundaries: cite EuroGeographics

Citing the eurostat R package

For main developers and contributors, see the package homepage.

This work can be freely used, modified and distributed under the BSD-2-clause (modified FreeBSD) license:

```
citation("eurostat")
```

```
##
## Kindly cite the eurostat R package as follows:
##
##   (C) Leo Lahti, Janne Huovari, Markus Kainu, Przemyslaw Biecek.
##   Retrieval and analysis of Eurostat open data with the eurostat
##   package. R Journal 9(1):385-392, 2017. Version 3.1.6001 Package
##   URL: http://ropengov.github.io/eurostat Manuscript URL:
##   https://journal.r-project.org/archive/2017/RJ-2017-019/index.html
##
## A BibTeX entry for LaTeX users is
##
##   @Misc{,
##     title = {eurostat R package},
##     author = {Leo Lahti and Janne Huovari and Markus Kainu and Przemyslaw Biecek},
```

```
##     journal = {R Journal},
##     volume = {9},
##     number = {1},
##     pages = {385-392},
##     year = {2017},
##     url = {https://journal.r-project.org/archive/2017/RJ-2017-019/index.html},
##     note = {Version 3.1.6001},
##   }
```

Related work

This rOpenGov R package is based on the earlier CRAN packages statfi and smarterpoland.

The independent reurostat package develops related Eurostat tools but seems to be in an experimental stage at the time of writing this tutorial.

The more generic quandl, datamart, rsdmx, and pdfetch packages may provide access to some versions of eurostat data but these packages are more generic and hence, in contrast to the eurostat R package, lack tools that are specifically customized to facilitate eurostat analysis.

Contact

For contact information, see the package homepage.

Version info

This tutorial was created with

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 17.10
##
## Matrix products: default
## BLAS: /home/lei/bin/R-3.4.3/lib/libRblas.so
## LAPACK: /home/lei/bin/R-3.4.3/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rsdmx_0.5-10       RColorBrewer_1.1-2 tmap_1.11-1
##  [4] sf_0.6-0           dplyr_0.7.4        plotrix_3.7
##  [7] ggplot2_2.2.1.9000 tidyr_0.8.0        bindrcpp_0.2
```

```
## [10] rvest_0.3.2        xml2_1.2.0         eurostat_3.1.6001
## [13] rmarkdown_1.8      knitr_1.19
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-131       bitops_1.0-6       satellite_1.0.1
##  [4] webshot_0.5.0.9000 gmodels_2.16.2     httr_1.3.1
##  [7] rprojroot_1.3-2    mapview_2.3.0      tools_3.4.3
## [10] backports_1.1.2    rgdal_1.2-16       R6_2.2.2
## [13] KernSmooth_2.23-15 spData_0.2.7.0     rgeos_0.3-26
## [16] DBI_0.7            lazyeval_0.2.1     colorspace_1.3-2
## [19] raster_2.6-7       withr_2.1.1.9000   sp_1.2-7
## [22] tidyselect_0.2.3   leaflet_1.1.0      curl_3.1
## [25] compiler_3.4.3     expm_0.999-2       labeling_0.3
## [28] scales_0.5.0.9000  rmapshaper_0.3.0   classInt_0.1-24
## [31] readr_1.1.1        stringr_1.2.0      digest_0.6.15
## [34] R.utils_2.6.0      base64enc_0.1-3    dichromat_2.0-0
## [37] pkgconfig_2.0.1    htmltools_0.3.6    highr_0.6
## [40] jsonvalidate_1.0.0 htmlwidgets_1.0    rlang_0.1.6.9003
## [43] shiny_1.0.5        bindr_0.1          jsonlite_1.5
## [46] crosstalk_1.0.0    gtools_3.5.0       R.oo_1.21.0
## [49] spdep_0.7-4        RCurl_1.95-4.10    magrittr_1.5
## [52] geosphere_1.5-7    Matrix_1.2-12      Rcpp_0.12.15
## [55] munsell_0.4.3      R.methodsS3_1.7.1  stringi_1.1.6
## [58] yaml_2.1.16        MASS_7.3-48        tmaptools_1.2-3
## [61] plyr_1.8.4         grid_3.4.3         gdata_2.18.0
## [64] udunits2_0.13      deldir_0.1-14      lattice_0.20-35
## [67] splines_3.4.3      hms_0.4.1          pillar_1.1.0
## [70] boot_1.3-20        gdalUtils_2.0.1.7  geojsonlint_0.2.0
## [73] stats4_3.4.3       codetools_0.2-15   LearnBayes_2.15
## [76] osmar_1.1-7        XML_3.98-1.9       glue_1.2.0
## [79] evaluate_0.10.1    V8_1.5             png_0.1-7
## [82] foreach_1.4.4      httpuv_1.3.5       gtable_0.2.0
## [85] purrr_0.2.4        assertthat_0.2.0   mime_0.5
## [88] xtable_1.8-2       e1071_1.6-8        coda_0.19-1
## [91] viridisLite_0.3.0  class_7.3-14       tibble_1.4.2
## [94] iterators_1.0.9    units_0.5-1
```