Package 'ugatsdb'

July 22, 2025

Title Uganda Time Series Database API

Version 0.2.3

Date 2022-11-20

Description An R API providing easy access to a relational database with macroeconomic, financial and development related time series data for Uganda. Overall more than 5000 series at varying frequency (daily, monthly, quarterly, annual in fiscal or calendar years) can be accessed through the API. The data is provided by the Bank of Uganda, the Ugandan Ministry of Finance, Planning and Economic Development, the IMF and the World Bank. The database is being updated once a month.

URL https://mepd.finance.go.ug/apps.html

License GPL-3

Encoding UTF-8

Imports DBI, RMySQL, data.table, collapse, writexl

Suggests magrittr, xts, dygraphs

RoxygenNote 7.1.2

NeedsCompilation no

Author Sebastian Krantz [aut, cre]

Maintainer Sebastian Krantz <sebastian.krantz@graduateinstitute.ch>

Repository CRAN

Date/Publication 2022-11-24 00:20:02 UTC

Contents

`ugatsdb-package` *Uganda Time Series Database API*

Description

An R API providing easy access to a relational database with macroeconomic, financial and development related time series data for Uganda. Overall more than 5000 series at varying frequency (daily, monthly, quarterly, annual in fiscal or calendar years) can be accessed through the API. The data is provided by the Bank of Uganda, the Ugandan Ministry of Finance, Planning and Economic Development, the IMF and the World Bank. The database is being updated once a month.

Functions

Functions to retrieve tables identifying the data

`datasources()`
`datasets()`
`series()`

Function to retrieve the data from the database

`get_data()`

Functions to reshape data and add temporal identifiers

`long2wide()`
`wide2long()`
`expand_date()`

Function to export wide format data to Excel

`wide2excel()`

Helper functions (useful esp. for common Excel formats)

`make_date()`
`transpose_wide()`

Global Macros with core ID variables in the database

`.IDvars`
`.Tvars`

Function to renew database connection without reloading the package

`ugatsdb_reconnect()`

Examples

```
library(ugatsdb)
library(magrittr) # Pipe %>% operators
library(xts)      # Time series class and pretty plots

# Plotting daily IFEM Buying and Selling Rates from the Bank of Uganda
get_data("BOU_E", c("E_IFEM_B", "E_IFEM_S"), from = 2020) %>%
  as.xts %>% plot

library(dygraphs)

# Same generating a dynamic chart
get_data("BOU_E", c("E_IFEM_B", "E_IFEM_S"), from = 2020) %>%
  as.xts %>% dygraph

# Static plot but with legend showing variable labels
get_data("BOU_E", c("E_IFEM_B", "E_IFEM_S"), from = 2020, wide = FALSE) %>%
  long2wide(names_from = "Label") %>% as.xts %>%
  plot(legend.loc = "topleft")
```

`.IDvars` *Global Identifier Macros*

Description

The macro `.IDvars` contains the string `c("DSID", "Series")` denoting variables that uniquely identify series in the database. *Note* that the series code contained in "Series" alone is not sufficient to uniquely identify a series as some series are recorded with the same code in multiple datasets (mostly either the same data aggregated at a different frequency, or a different collection of indicators). For example goods exports with the series code "EX_G" are recorded in the datasets "BOU_MMI", "BOU_MMI_A" (annual data), and "BOU_MMI_FY" (fiscal year data).

The macro `.Tvars` contains the string `c("Date", "Year", "Quarter", "FY", "QFY", "Month", "Day")` denoting temporal identifiers generated by `expand_date`. The "Date" variable is sufficient to uniquely identify a point in time in the database.

Usage

```
.IDvars
.Tvars
```

See Also

`ugatsdb`

Examples

```
.IDvars
.Tvars
```

`datasets` *Retrieve Datasets Table*

Description

This function pulls and return a table called 'DATASET' from the database.

Usage

```
datasets(ordered = TRUE)
```

Arguments

`ordered`	logical. `TRUE` orders the result in the order data was entered into the database, while `FALSE` returns the result in a random order, to the benefit of faster query execution.

Details

The 'DATASET' table gives information about the different datasets read into the database from various sources. It provides a unique id for each dataset, the frequency of data, the minimum and maximum time coverage, when the dataset was last updated, a description, the source (matching the 'Source' column in the 'DATASOURCE' table), and an (optional) url providing direct access to the raw data.

Value

A `data.table` with information about the available datasets in the database.

See Also

`datasources`, `series`, `ugatsdb`

Examples

```
datasets()
```

datasources *Retrieve Data Sources Table*

Description

This function pulls and returns a table called 'DATASOURCE' from the database.

Usage

```
datasources(ordered = TRUE)
```

Arguments

ordered	logical. TRUE orders the result in the order data was entered into the database, while FALSE returns the result in a random order, to the benefit of faster query execution.

Details

The 'DATASOURCE' table gives information about the various sources / providers of data in this database, including the source website, the number of datasets available from the source, a description of the source and the way data is accessed from the source.

Value

A data.table with information about the sources of data in the database.

See Also

datasets, ugatsdb

Examples

```
datasources()
```

expand_date *Generate Temporal Identifiers from a Date Column*

Description

This function expands a date column and generates additional temporal identifiers from it (such as the year, month, quarter, fiscal year etc.).

Usage

```
expand_date(
  x,
  gen = c("Year", "Quarter", "FY", "QFY", "Month"),
  origin = "1899-12-30",
  keep.date = TRUE,
  remove.missing.date = TRUE,
  sort = TRUE,
  as.factor = TRUE,
  ...
)
```

Arguments

x	either a vector of class 'Date', or coercible to date using as.Date, or a data frame / list with a date variable in the first column.
gen	character. A vector of identifiers to generate from x. The possible identifiers are found in .Tvars. The default setting is to generate all identifiers apart from "Day".
origin	character / Date. Passed to as.Date: for converting numeric x to date.
keep.date	logical. TRUE will keep the date variable in the first column of the resulting dataset, FALSE will remove the date variable in favor of the generated identifiers.
remove.missing.date	logical. TRUE will remove missing values in x. If x is a dataset, the corresponding rows will be removed.
sort	logical. TRUE will sort the data by the date column.
as.factor	TRUE will generate quarters, fiscal years and months ('Quarter', 'FY', 'QFY', 'Month') as factor variables. It is also possible to use as.factor = "ordered" to generate ordered factors. FALSE will generate fiscal years as character and quarters and months as integer variables.
...	not used.

Value

A data.table containing the computed identifiers as columns. See Examples.

See Also

make_date, ugatsdb

Examples

```
# First a basic example
x <- seq.Date(as.Date("1999-01-01"), as.Date("2000-01-01"), by = "month")
expand_date(x)
expand_date(x, gen = c("Year", "Month"), keep.date = FALSE)
```

```
# Now using the API
expand_date(get_data("BOU_CPI")) # Getting Monthly CPI data from the Bank of Uganda

# Same thing
get_data("BOU_CPI", expand.date = TRUE)
```

`get_data` *Retrieve Data from the Database*

Description

This is the main function of the package to retrieve data from the database. It constructs an SQL query which is sent to the database and returns the data as a `data.table` in R.

Usage

```
get_data(
  dsid = NULL,
  series = NULL,
  from = NULL,
  to = NULL,
  labels = TRUE,
  wide = TRUE,
  expand.date = FALSE,
  ordered = TRUE,
  return.query = FALSE,
  ...
)
```

Arguments

`dsid`	character. (Optional) id's of datasets matching the 'DSID' column of the 'DATASET' table (retrieved using `datasets()`). If none of the following arguments are used, all series from those datasets will be returned.
`series`	character. (Optional) codes of series matching the 'Series' column of the 'Series' table (retrieved using `series()`).
`from`	set the start time of the data retrieved by either supplying a start date, a date-string of the form `"YYYY-MM-DD"` or `"YYYY-MM"`, year-quarters of the form `"YYYYQN"` or `"YYYY-QN"`, a numeric year `YYYY` (numeric or character), or a fiscal year of the form `"YYYY/YY"`. These expressions are converted to a regular date by `make_date`.
`to`	same as `from`: to set the time period until which data is retrieved. For expressions that are not full "YYYY-MM-DD" dates, the last day of the period is chosen.

labels	logical. TRUE will also return labels (series descriptions) along with the series codes.
wide	logical. TRUE calls long2wide on the result. FALSE returns the data in a long format without missing values (suitable for ggplot2).
expand.date	logical. TRUE will call expand_date on the result.
ordered	logical. TRUE orders the result by 'Date' and, if labels = TRUE, by series, maintaining the column-order of series in the dataset(s). FALSE returns the result in a random order, to the benefit of faster query execution.
return.query	logical. TRUE will not query the database but instead return the constructed SQL query as a character string.
...	further arguments passed to long2wide (if wide = TRUE) or expand_date (if expand.date = TRUE), no conflicts between these two.

Details

If labels = FALSE, the 'SERIES' table is not joined to the 'DATA' table, and ordered = TRUE will order datasets and series retrieved in alphabetic order. If labels = TRUE data is ordered by series and date within each dataset, preserving the order of columns in the dataset. If multiple datasets are received they are ordered alphabetically according to the 'DSID' column.

It is possible query multiple series from multiple datasets e.g. get_data(c("DSID1", "DSID2"), c("SERFROM1", "SERFROM2")) etc., but care needs to be taken that the series queried do not occur in both datasets (see .IDvars, and check using series(c("DSID1", "DSID2"))). Series from datasets at different frequencies can be queried, but, if wide = TRUE, this will result in missing values for all but the first observations per period in the lower frequency series.

Value

A data.table with the result of the query.

See Also

long2wide, expand_date, ugatsdb

Examples

```
# Return monthly macroeconomic indicators from the year 2000 onwards
get_data("BOU_MMI", from = 2000, wide = FALSE)

# Return wide format with date expanded
get_data("BOU_MMI", from = 2000, expand.date = TRUE)

# Same thing in multiple steps (with additional customization options):
library(magrittr) # Pipe %>% operators
get_data("BOU_MMI", from = 2000, wide = FALSE) %>% long2wide %>% expand_date

# Getting a single series
get_data("BOU_MMI", "M2", 2000)

# Getting High-Frequency activity indicators from BoU and Revenue & Expense from MoFPED
```

```
get_data(c("BOU_MMI", "MOF_TOT", "WB_WDI"), c("CIEA", "BTI", "REV_GRA", "EXP_LEN"))

# Getting daily interest rates and plotting
library(xts)   # Time series class
get_data("BOU_I", from = 2018, wide = FALSE) %>%
   long2wide(names_from = "Label") %>%
   as.xts %>%
   plot(legend.loc = "topleft")
```

`long2wide` *Reshape Long API Data to Column-Based Format*

Description

This function automatically reshapes long (stacked) raw data from the API (`get_data(..., wide = FALSE)`) to a wide format where each variable has its own column.

Usage

```
long2wide(
  data,
  id_cols = intersect(.Tvars, names(data)),
  names_from = "Series",
  values_from = "Value",
  labels_from = if (any(names(data) == "Label")) "Label" else NULL,
  expand.date = FALSE,
  ...
)
```

Arguments

`data`	raw data from the API: A long format data frame where all values are stacked in a value column.
`id_cols`	character. Temporal identifiers of the data. By default all variables in `.Tvars` are selected.
`names_from`	character. The column containing the series codes. These will become the names of the new columns in the wider data format.
`values_from`	character. The column containing the data values.
`labels_from`	character. The column containing the labels describing the series.
`expand.date`	logical. `TRUE` will call `expand_date` on the data after reshaping.
`...`	further arguments passed to `dcast` or `expand_date`, no conflicts between these two.

Value

A `data.table` with the reshaped data.

See Also

wide2long, wide2excel, ugatsdb

Examples

```
# Return monthly macroeconomic indicators from the year 2000 onwards
long2wide(get_data("BOU_MMI", from = 2000, wide = FALSE))
```

---

make_date *Coerce Vectors to Dates*

Description

This function coerces date strings i.e. "YYYY-MM-DD" or "YYYY-MM", years e.g. 2015 (numeric or character), year-quarters e.g. "2015Q1" or "2015-Q1", year-months e.g. "2015M01" or "2015-M01", fiscal years e.g. "1997/98" or numeric values representing dates (e.g. previously imported Excel date) to a regular R date.

Usage

```
make_date(x, end = FALSE, origin = "1899-12-30")
```

Arguments

x	a character date string "YYYY-MM-DD" or "YYYY-MM", year-quarter "YYYYQN" or "YYYY-QN", , year-month "YYYYMNN" or "YYYY-MNN", fiscal year "YYYY/YY" or calendar year YYYY (numeric or character), or a numeric value corresponding to a date that can be passed to as.Date.numeric.
end	logical. TRUE replaces missing time information with a period end-date which is the last day of the period. FALSE replaces missing month and day information with "-01", so the year date is the 1st of January, the fiscal year date the 1st of July, and for months / quarters the 1st day of the month / quarter.
origin	a date or date-string that can be used as reference for converting numeric values to dates. The default corresponds to dates generated in Excel for Windows. See as.Date.numeric.

Value

A Date vector.

See Also

expand_date, ugatsdb

Examples

```
make_date("2011-05")
make_date(2011)
make_date("2011/12")
make_date("2011/12", end = TRUE)
make_date("2011Q1")
make_date("2011Q1", end = TRUE)
```

`series` *Retrieve Series Table*

Description

This function pulls and returns a table called 'SERIES' from the database.

Usage

```
series(dsid = NULL, dataset.info = TRUE, ordered = TRUE, return.query = FALSE)
```

Arguments

`dsid`	character. (Optional) id's of datasets matching the 'DSID' column of the 'DATASET' table (retrieved using `datasets()`) for which series information is to be returned.
`dataset.info`	logical. `TRUE` returns additional information from the 'DATASET' table about the datasets in which the series are recorded. `FALSE` only returns the raw 'SERIES' table.
`ordered`	logical. `TRUE` orders the result in the order data was entered into the database, while `FALSE` returns the result in a random order, to the benefit of faster query execution.
`return.query`	logical. `TRUE` will not query the database but instead return the constructed SQL query as a character string.

Details

The 'SERIES' table gives information about all of the time series in the database. Each series is given a code which is however not unique across datasets (see `.IDvars`). Each series also has a label describing the series. Further information recorded are the minimum and maximum time coverage, and (optionally) a separate series source and url. By default `dataset.info = TRUE` and the frequency of the data, the date when the dataset containing the series was last updated, the dataset and data source are added to the 'SERIES' table from the 'DATASET' table.

If `dataset.info = FALSE`, the 'DATASET' table is not joined to the 'SERIES' table, and `ordered = TRUE` only orders the series within each dataset to maintain the column order of series in the source data. In that case the datasets are returned in alphabetic order of 'DSID', not the order in which they were entered into the 'DATASET' table.

Value

A data.table with information about the available time series in the database.

See Also

datasets, ugatsdb

Examples

```
# By default returns all series with additional information
series()

# Raw series table
series(dataset.info = FALSE)

# Only series in the Monthly Macroeconomic Indicators of the BoU
series("BOU_MMI")
```

---

transpose_wide *Transpose a Wide Dataset to a Row-Based Format*

Description

This function is called by wide2excel with option transpose = TRUE to generate a row-based tabular data format from a wide data frame in R that is suitable for exporting to Excel.

Usage

```
transpose_wide(data, date.format = "%d/%m/%Y")
```

Arguments

data	a wide format data frame where each column is a variable and the first variable uniquely identifies the data.
date.format	a format for date columns which is passed to format.Date. When transposing wide, dates are converted to character. The default R YYYY-MM-DD format for dates is often not recognized by Excel. By default dates are transformed to DD/MM/YYYY format which Excel (UK English) recognizes. Putting FALSE here does not transform dates into another format.

Value

A transposed data frame or data.table (the class of the input is preserved).

See Also

transpose, long2wide, wide2excel, ugatsdb

Examples

```
transpose_wide(get_data("BOU_CPI"))
```

---

ugatsdb_reconnect *Reconnect to Database*

Description

This function terminates an existing connection to the database server and attempts to reconnect to it. It is now somewhat redundant by the safe query mechanism introduced in v0.2.1 of the package, where each query is evaluated inside `tryCatch` and the database connection is renewed if the query fails. This function can still be used to manually renew the database connection.

Usage

```
ugatsdb_reconnect()
```

See Also

`ugatsdb`

Examples

```
ugatsdb_reconnect()
```

---

wide2excel *Export Wide Data to Excel*

Description

This function exports a wide format dataset to a column- (default) or row-oriented Excel format.

Usage

```
wide2excel(data, ..., transpose = FALSE, transpose.date.format = "%d/%m/%Y")
```

Arguments

`data`	a wide dataset from `get_data` or reshaped to a wide format with `long2wide`.
`...`	further arguments to `write_xlsx`. As a minimum a path needs to be supplied that ends with the name of the Excel file. See Examples.
`transpose`	logical. If `TRUE`, the result is returned in a row-oriented Excel format. The default is column oriented (same as the dataset in R).
`transpose.date.format`	argument passed to `transpose_wide`, setting the format of date columns when data is transposed.

See Also

transpose_wide, write_xlsx, ugatsdb

Examples

```
## Not run:
# Getting macroeconomic indicators from Bank of Uganda in fiscal years
data <- get_data("BOU_MMI_FY", from = "2000/01")

# Saving to different Excel formats
wide2excel(data, "BOU_MMI_FY.xlsx")
wide2excel(data, "BOU_MMI_FY.xlsx", transpose = TRUE)

# Saving to alternative path
wide2excel(data, "C:/Users/.../BOU_MMI_FY.xlsx")

## End(Not run)
```

---

wide2long *Reshape Column-Based Data to Long Format*

Description

This function automatically reshapes wide (column-based) data into a long format akin to the format of the raw data coming from the database (get_data(..., wide = FALSE)).

Usage

```
wide2long(
  data,
  id_cols = intersect(.Tvars, names(data)),
  to_value = setdiff(names(data), id_cols),
  variable_name = "Series",
  value_name = "Value",
  label_name = "Label",
  na.rm = TRUE,
  ...
)
```

Arguments

data	a wide format data frame where all series have their own column.
id_cols	character. Temporal identifiers of the data. By default all variables in .Tvars are selected.
to_value	character. The names of all series to be stacked into the long format data frame.
variable_name	character. The name of the variable to store the names of the series.

value_name	character. The name of the variable to store the data values.
label_name	character. The name of the variable to store the series labels.
na.rm	logical. TRUE will remove all missing values from the long data frame.
...	further arguments passed to melt.

Value

A data.table with the reshaped data.

See Also

long2wide, ugatsdb

Examples

```
# Return monthly macroeconomic indicators from the year 2000 onwards
data <- get_data("BOU_MMI", from = 2000)
wide2long(data)
```

Index