Package 'timeseriesdb'

August 29, 2016

Type Package

Version 0.2.1

Title Manage Time Series with R and PostgreSQL

Description Store and organize a large amount of low frequency time series data.
The package was designed to manage a large catalog of official statistics which are typically published on monthly, quarterly or yearly basis. Thus timeseriesdb is optimized to handle a large number of lower frequency time series as opposed to a smaller amount of high frequency time series such as real time data from measuring devices. Hence timeseriesdb provides the opportunity to store extensive multi-lingual meta information. The package also provides a web GUI to explore the underlying PostgreSQL database interactively.

Author ``Matthias Bannert <bannert@kof.ethz.ch> [aut, cre]"

Depends R (>= 3.0.0), RPostgreSQL, methods

Imports RJSONIO, xts, zoo, reshape2, shiny, DBI

Date 2015-04-02

License GPL-2

LazyData true

Maintainer 'Matthias Bannert' <bannert@kof.ethz.ch>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-04-04 07:46:09

R topics documented:

`activateTsSet` *Activate a Set of Time Series*

Description

Activate a set of time series to get in the user's sight. Deactivated sets are not deleted though.

Usage

```
activateTsSet(con, set_name, user_name = Sys.info()["user"],
  tbl = "timeseries_sets", schema = "timeseries")
```

Arguments

`con`	PostgreSQL connection object
`set_name`	character name of the set to be activated.
`user_name`	character name of the user. Defaults to system user.
`tbl`	character name of set tqble. Defaults to timeseries_sets.
`schema`	character name of the database schema. Defaults to timeseries.

Author(s)

Matthias Bannert, Ioan Gabriel Bucur

addMetaInformation *Add Meta Information to R Environments*

Description

This function adds meta information to environments that are explicitly meant to store Meta Information. This function can be used separately in interactive R Session or to facilitate mapping database information to R.

Usage

```
addMetaInformation(series, map_list, meta_env = NULL, overwrite_objects = F,
  overwrite_elements = T)
```

Arguments

series	character name key of
map_list	list to represent key value mapping. Could also be of class miro.
meta_env	an environment that already holds meta information and should be extended. Defaults to NULL in which case it creates and returns a new environment.
overwrite_objects	logical should the entire existing meta information be overwritten inside the environment? Defaults to FALSE
overwrite_elements	logical should single matching elements of a meta information objectes be overwritten. Defaults to TRUE.

beginTransaction *Convenience Wrapper to SQL classics for BEGIN,COMMIT, ROLLBACK*

Description

this set of function can speed up loops by starting a transaction, performing several queries and ending them with either commit or rollback.

Usage

```
beginTransaction(con)

commitTransaction(con)

rollbackTransaction(con)
```

Arguments

con	PostgreSQL connection object.

createChoices *Create Choices Boxes Dynamically*

Description

Dynamically create choices boxes depending on interactive choices by the user. The UI reacts to user input. We use a somewhat polymorphic approach here to avoid too much if/else and improve readabitity.

Usage

```
createChoices(x, input = NULL, keys = NULL, con, ...)

## S3 method for class 'key'
createChoices(x, input = NULL, keys = NULL, ...)

## S3 method for class 'md'
createChoices(x, input = NULL, keys = NULL, con, ...)

## S3 method for class 'set'
createChoices(x, input = NULL, keys = NULL, con, ...)
```

Arguments

x	query type as character.
input	input that triggers action. Defaults to NULL.
keys	results to rendered to the choice box. Defaults to NULL.
con	PostgreSQL Connection object.
...	addtional arguments to be passed to the methods.

createConObj *Conveniently Create Connection Object to PostgreSQL based timeseriesdb*

Description

Create a conection object while getting user information from the R session. Also standard db parameters like port and driver are set. Yet flexible information like host or dbname should be added to Sys.setenv environments.

Usage

```
createConObj(dbuser = Sys.info()["user"],
  dbname = Sys.getenv("TIMESERIESDB_NAME"),
  dbhost = Sys.getenv("TIMESERIESDB_HOST"), passwd, dbport = 5432)
```

Arguments

dbuser	character username. Defaults to reading username from Sys.info()
dbname	character name of the database, assumes dbname is stored in TIMESERIESDB_NAME.
dbhost	character host address, asssumes dbhost ist stored in TIMESERIESDB_HOST.
passwd	character password is used. No defaults, best way to pass a password is to .rs.askForPassword to hide password entries when using R Studio.
dbport	integer port number defaults to 5432 for postgres

`createHstore` *Create Hstore*

Description

Function to Create Hstore Key Value Pair Mapping

Usage

```
createHstore(x, ...)

## S3 method for class 'ts'
createHstore(x, ...)

## S3 method for class 'data.frame'
createHstore(x, ...)

## S3 method for class 'list'
createHstore(x, ...)
```

Arguments

x	a time series object, a two column data frame or object of S3 class miro (meta information for R objects).
...	optional arguments, fct = TRUE create text expressions of hstore function calls. also for data.frames key_pos and value_pos could be given if they are different from 1 and 2. e.g. position of the key col and pasition of the value col in a data.frame.

Details

This function creates a key value pair mapping from a time series object. It returns an hstore object that can be inserted to a PostgreSQL database relation field of type hstore.

Author(s)

Matthias Bannert

Examples

```
ts1 <- ts(rnorm(100),start = c(1990,1),frequency = 4)
createHstore(ts1)
```

`createUI` *Polymorphic Helpers for the Shiny Based GUI*

Description

Though R is not strictly OO it is sometimes useful to add some object orientation. In the case of a shiny app with an dynamically created user interface a bit of pseudo polymorphism helps to avoid too many but,ifs and elses. Depending on the query type that is selected by the user a UI creation method is called.

Usage

```
createUI(x, con, ...)

## S3 method for class 'key'
createUI(x, con, ...)

## S3 method for class 'set'
createUI(x, con, ...)

## S3 method for class 'md'
createUI(x, con, ...)
```

Arguments

`x`	character query type.
`con`	PostgreSQL Connection object.
`...`	additional arguments to be passed to the respective methods.

`dbIsValid,PostgreSQLConnection-method`
Check Validity of a PostgreSQL connection

Description

Is the PostgreSQL connection expired?

Usage

```
## S4 method for signature 'PostgreSQLConnection'
dbIsValid(dbObj)
```

Arguments

dbObj	PostgreSQL connection object.

---

deactivateTsSet *Deactivate a Set of Time Series*

Description

This deactivates a set of time series to get out of the user's sight, but it's not the deleted because users may not delete sets.

Usage

```
deactivateTsSet(con, set_name, user_name = Sys.info()["user"],
  tbl = "timeseries_sets", schema = "timeseries")
```

Arguments

con	PostgreSQL connection object
set_name	character name of the set to be deactivated.
user_name	character name of the user. Defaults to system user.
tbl	character name of set tqble. Defaults to timeseries_sets.
schema	character name of the database schema. Defaults to timeseries.

Author(s)

Matthias Bannert, Ioan Gabriel Bucur

---

deleteTimeSeries *Delete Time Series from the database*

Description

This function deletes time series AND their metainformation from the database. All meta information in all series will be deleted. To only edit the original time series use storeTimeSeries to overwrite existing series.

Usage

```
deleteTimeSeries(series, con, tbl_main = "timeseries_main",
  schema = "timeseries")
```

Arguments

series	character name of the timeseries
con	a PostgreSQL connection object
tbl_main	character name of the table that contains the main time series catalog. Defaults to 'timeseries_main'.
schema	SQL schema name. Defaults to 'timeseries'.

---

exploreDb *Start a GUI to Explore Data*

Description

Start a graphical user interface in the user's standard web browser to search and explore time series data. Data can be searched using regular expressions for keys. Hits can subsequently selected from a select box and are plotted in a joint time series plot.

Usage

```
exploreDb(con, browser = T)
```

Arguments

con	PostgreSQL Connection object
browser	logical should app be fired up in the web browser? Defaults to TRUE.

---

exportTsList *Export A List of Time Series to CSV*

Description

Export a List of time series to semi-colon separated csv file. Typically multiple time series are exported as long format (melted format) files.

Usage

```
exportTsList(tl, fname = NULL, cast = T)
```

Arguments

tl	list of time series
fname	character file name. If set to NULL a standard file name chunk + Sys.Date is used.
cast	logical. Should the resulting data.frame be cast to wide format? Defaults to TRUE

getListDepth *Determine depth of a list*

Description

This function recursively checks the depth of a list and returns an integer value of depth

Usage

```
getListDepth(this)
```

Arguments

this	an object of class list

Details

Hat tip to flodel at stackoverflow for suggesting this light weight way analyze depth of a nested list. Further complexity needs to be added to cover the fact that data.frame are lists, too. A more sophisticated recursive function can be found in the gatveys2 package.

References

http://stackoverflow.com/questions/13432863/determine-level-of-nesting-in-r

getMeta *Quickly Handle Meta Information*

Description

Sometimes reading the entire meta description for all language or multiple time series might not be necessary. Quick handle operators help users to access the information quickly as a non-nested list for only one language is returned. These functions are alpha status, more will follow.

Usage

```
getMeta(series, lang, con, tbl = "meta_data_localized",
  schema = "timeseries")
```

Arguments

series	an R time series object
lang	character name of the language of the meta information. Typically 'de', 'it', 'fr' or 'en'.
con	connection object
tbl	character name of the table that contains the meta information.
schema	SQL schema name. Defaults to 'timeseries'.

importTsList *Import A Long Format CSV File Into a List of Time Series*

Description

Generates a List of time series of class tslist from a flat file that contains multiple time series.

Usage

```
importTsList(f, base_ts = T)
```

Arguments

f	character file name
base_ts	logical should the export be casted to R basic time series or left as xts? Defaults to TRUE.

listTsSets *List All Time Series Sets for a Particular User*

Description

Show the names of all sets that are available to a particular user.

Usage

```
listTsSets(con, user_name = Sys.info()["user"], tbl = "timeseries_sets",
  schema = "timeseries")
```

Arguments

con	PostgreSQL connection object
user_name	character name of the user. Defaults to system user.
tbl	character name of set tqble. Defaults to timeseries_sets.
schema	character name of the database schema. Defaults to timeseries.

Author(s)

Matthias Bannert, Gabriel Bucur

`loadTsSet` *Load a Time Series Set*

Description

Loads a Time Series Set.

Usage

```
loadTsSet(con, set_name, user_name = Sys.info()["user"],
  tbl = "timeseries_sets", schema = "timeseries")
```

Arguments

`con`	PostgreSQL connection object
`set_name`	character name of the set to be loaded.
`user_name`	character name of the user. Defaults to system user.
`tbl`	character name of set tqble. Defaults to timeseries_sets.
`schema`	character name of the database schema. Defaults to timeseries.

Author(s)

Matthias Bannert, Ioan Gabriel Bucur

`plot.tslist` *Plot a list of Time Series Directly plot time series of mixed frequency into one plot.* `readTimeSeries` *returns a list of time series. The returned list is also of class tslist which makes it very convenient to plot timeseries, directly out of the database. This function gets automatically the right ranges for the axes to plot all values and dates in contained in the list.*

Description

Plot a list of Time Series Directly plot time series of mixed frequency into one plot. `readTimeSeries` returns a list of time series. The returned list is also of class tslist which makes it very convenient to plot timeseries, directly out of the database. This function gets automatically the right ranges for the axes to plot all values and dates in contained in the list.

Usage

```
## S3 method for class 'tslist'
plot(x, ..., use_legend = T, shiny_legend = F, lwd = 3)
```

Arguments

x	a list of time series. Object should be of class tslist.
...	parameters than can simply by passed on tot the plot function
use_legend	logical. Should legend be used. Defaults to TRUE. Useful to switch of if so many time series are drawn that they are hard to distinguish anyway.
shiny_legend	logical, is plot used in context of a shiny app? Defaults to FALSE.
lwd	line width argument passed to other internal plotting functions. Defaults to 3.

---

readMetaInformation *Read Meta Information from a Time Series Database*

Description

This function reads meta information from a timeseriesdb package PostgreSQL database and puts into a meta information environment.

Usage

```
readMetaInformation(series, con, locale = "de", tbl = "meta_data_localized",
  overwrite_objects = F, overwrite_elements = T, meta_env = NULL,
  schema = "timeseries")
```

Arguments

series	character name of a time series object.
con	PostgreSQL connection object
locale	character denoting the locale of the meta information that is queried. defaults to 'de' for German. At the KOF Swiss Economic Institute meta information should be available als in English 'en', French 'fr' and Italian 'it'. Set the locale to NULL to query unlocalized meta information.
tbl	character name of the table that contains meta information. Defaults to 'meta_data_localized'. Choose meta 'meta_data_unlocalized' when locale is set to NULL.
overwrite_objects	logical should the entire object for a key be overwritten. Defaults to FALSE.
overwrite_elements	logical should single elements inside the environment be overwritten. Defaults to TRUE.
meta_env	environment to which the meta information should be added. Defaults to NULL. In this case an environment will be returned. If you run this function in a loop best create an empty environment before the loop or apply call and pass the environment to this function. By doing so new elements will be added to the environment.
schema	SQL schema name. Defaults to timeseries.

readTimeSeries *Read Time Series From PostgreSQL database*

Description

This function reads a time series from a postgres database, that uses key value pair storage (hstore), to archive time series. After reading the information from the database a standard R time series object of class 'ts' is built and returned.

Usage

```
readTimeSeries(series, con, tbl = "v_timeseries_json",
  schema = "timeseries")
```

Arguments

series	character vector of series names.
con	a PostgreSQL connection object
tbl	character string denoting the name of the view containing the json records.
schema	SQL schema name. Defaults to timeseries.

Author(s)

Matthias Bannert, Gabriel Bucur

resolveOverlap *Concatenate Time Series and Resolve Overlap Automatically*

Description

Append time series to each other. Resolve overlap determines which of two ts class time series is reaching further and arranges the two series into first and second series accordingly. Both time series are concatenated to one if both series had the same frequency. Typically this function is used concatenate two series that have a certain overlap, but one series clearly starts earlier while the other lasts longer. If one series starts earlier and stops later, all elements of the shorter series will be inserted into the larger series, i.e. elements of the smaller series will replace the elements of the longer series. Usually ts2 is kept.

Usage

```
resolveOverlap(ts1, ts2, keep_ts2 = T)
```

Arguments

ts1	ts time series, typically the older series
ts2	ts time series, typically the younger series
keep_ts2	logical should ts2 be kept? Defaults to TRUE.

Examples

```
ts1 <- ts(rnorm(100),start = c(1990,1),frequency = 4)
ts2 <- ts(1:18,start = c(2000,1),frequency = 4)
resolveOverlap(ts1,ts2)

# automatical detection of correction sequence!
ts1 <- ts(rnorm(90),start = c(1990,1),frequency = 4)
ts2 <- ts(1:60,start = c(2000,1),frequency = 4)
resolveOverlap(ts1,ts2)

# both series are of the same length use sequence of arguments.
ts1 <- ts(rnorm(100),start = c(1990,1),frequency = 4)
ts2 <- ts(1:48,start = c(2003,1),frequency = 4)
resolveOverlap(ts1,ts2)
ts1 <- ts(rnorm(101),start = c(1990,1),frequency = 4)
ts2 <- ts(1:61,start = c(2000,1),frequency = 4)
resolveOverlap(ts1,ts2)
#' clearly dominatn ts2 series
ts1 <- ts(rnorm(50),start = c(1990,1),frequency = 4)
ts2 <- ts(1:100,start = c(1990,1),frequency = 4)
resolveOverlap(ts1,ts2)
```

---

searchKeys *Search Keys From Pre-Defined Or Key Based Queries*

Description

Search Keys From Pre-Defined Or Key Based Queries

Usage

```
searchKeys(x, input = NULL, con, ...)

## S3 method for class 'key'
searchKeys(x, input = NULL, con, ...)

## S3 method for class 'md'
searchKeys(x, input = NULL, con, ...)

## S3 method for class 'set'
searchKeys(x, input = NULL, con, ...)
```

Arguments

x	query type as character.
input	input that triggers action. Defaults to NULL.
con	PostgreSQL Connection object.
...	addtional arguments to be passed to the methods.

---

searchKVP *Search Key-Value Pairs, look for existing keys in an Hstore*

Description

Search hstore key value in PostgreSQL. Very handsome when crawling the database by meta informaiton. Currently works for non translated meta information.

Usage

```
searchKVP(key, value, con = get(Sys.getenv("TIMESERIESDB_CON")),
  hstore = "meta_data", tbl = "meta_data_unlocalized", where = NULL,
  schema = "timeseries")

lookForKey(key, con = get(Sys.getenv("TIMESERIESDB_CON")),
  hstore = "meta_data", tbl = "meta_data_unlocalized", where = NULL,
  schema = "timeseries")
```

Arguments

key	character
value	in the hstore
con	PostgreSQL connection object
hstore	name of the hstore column
tbl	name of the table to be queried. defaults to 'meta_data_localized'
where	character restrict the SQL query by an additional where clause. Defaults to NULL.
schema	SQL schema name. defaults to timeseries. E.g.: ts_key LIKE ...

setAttrListWise *Set Attributes to Each Element of List According to a Given Vector*

Description

An attribute is set to all elements of a list given a vector of possible instances of the the attribute. Note that this function fails to excecute if the vector is not of the same length list.

Usage

```
setAttrListWise(li, attrib, vec)
```

Arguments

li	a list
attrib	character name of the attribute
vec	vector containing all instances of the attribute

storeMetaInformation *Store Meta Information to the Database*

Description

This function stores meta information to the database for a given time series. Make sure that corresponding time series had been inserted to the main table before.

Usage

```
storeMetaInformation(series, con, tbl = "meta_data_localized",
  lookup_env = "meta_data_localized", locale = "de", overwrite = F,
  quiet = F, schema = "timeseries")
```

Arguments

series	a character name of an time series object
con	a PostgreSQL connection object
tbl	name of the meta information table, defaults to localized meta data: meta_data_localized. Alternatively choose meta_data_unlocalized if you are not translating meta information.
lookup_env	name of the R environment in which to look for meta information objects
locale	character locale fo the metainformation. Defaults to Germen 'de'. See also readMetaInformation. If locale is set to NULL unlocalized meta is updated. Make sure to change tbl to 'meta_data_unlocalized'.

overwrite	logical, defaults to FALSE.
quiet	logical, should there be console output for every query result ? Defaults to FALSE.
schema	SQL schema name, defaults to 'timeseries'.

`storeTimeSeries` *Write an R Time Series to a PostgreSQL database*

Description

This function writes time series object into a relational PostgreSQL database make use of PostgreSQL own 'key'=>'value' storage called hstore. The schema and database needs to created first. The parent R Package of this functions suggests a database structure designed to store a larger amount of time series. This function uses INSERT INTO instead of the more convenient dbWritetable for performance reasons. DO NOT USE THIS FUNCTIONS IN LOOPS OR LAPPLY! This function can handle a set of time series on it's own and is much faster than looping over a list. Non-unique primary keys are overwritten !

Usage

```
storeTimeSeries(series, con, li = NULL, tbl = "timeseries_main",
  md_unlocal = "meta_data_unlocalized", lookup_env = .GlobalEnv,
  overwrite = T, schema = "timeseries")
```

Arguments

series	character name of a time series, S3 class ts. When used with lists it is convenient to set series to names(li). Note that the series name needs to be unique in the database!
con	a PostgreSQL connection object.
li	list of time series. Defaults to NULL to no break legacy calls that use lookup environments.
tbl	character string denoting the name of the main time series table in the PostgreSQL database.
md_unlocal	character string denoting the name of the table that holds unlocalized meta information.
lookup_env	environment to look in for timeseries. Defaults to .GobalEnv.
overwrite	logical should existing records (same primary key) be overwritten? Defaults to TRUE.
schema	SQL schema name. Defaults to timeseries.

Author(s)

Matthias Bannert, Gabriel Bucur

storeTsSet *Store a New Set of Time Series*

Description

Store a new set of Time Series to the database. Users can select the time series keys that should be grouped inside a set.

Usage

```
storeTsSet(con, set_name, set_keys, user_name = Sys.info()["user"],
  description = "", active = TRUE, tbl = "timeseries_sets",
  schema = "timeseries")
```

Arguments

con	PostgreSQL connection object
set_name	character name of a set time series in the database.
set_keys	list of keys contained in the set and their type of key.
user_name	character name of the user. Defaults to system user.
description	character description of the set to be stored in the db.
active	logical should a set be active? Defaults to TRUE. If set to FALSE a set is not seen directly in the GUI directly after being stored and needs to be activated first.
tbl	character name of set tqble. Defaults to timeseries_sets.
schema	character name of the database schema. Defaults to timeseries.

Author(s)

Ioan Gabriel Bucur, Matthias Bannert

zooLikeDateconvert *Zoo like Date Conversion*

Description

This function is taken from the zoo package. It is basically the S3 method as.Date.numeric of the package zoo. It is used to turn 2005.75 (3rd quarter of 2005) like date formats into dates like 2005-07-01.

Usage

```
zooLikeDateConvert(x, offset = 0, ...)
```

Arguments

x	object of class ts
offset	numeric defaults to 0. See the zoo package for more information.
...	optional arguments.

Author(s)

Achim Zeileis, Gabor Grothendieck, Jeffrey A. Ryan, Felix Andrews

---

%k% *Search the Database by Keys*

Description

Quick handle operator to search the database by keys. All time series whose key fit the regular expression which was handed to the operator are returned in a list.

Create ' This function creates a new function operator for a particular key. Name the function operator style to get the most out of it.

Usage

```
conObj %k% regexp

createMetaDataHandle(key, keep_keys = FALSE, tbl = "meta_data_unlocalized",
  schema = "timeseries")
```

Arguments

conObj	PostgreSQL Connection object.
regexp	character regular expression pattern. Do not set this manually, because the quick handle operator only takes two arguments. use Sys.setenv to change the schema.
key	character name of the key inside the hstore.
keep_keys	logical should primary time series keys be kept? Defaults to FALSE. If set to TRUE dynamically created meta information handlers always use ts_key as key type no matter the key type used for the query. This comes handy when storing sets of time series.
tbl	character name of the table that holds meta data. Defaults to meta_data_unlocalized. Also supports meta_data_localized
schema	character database schema name. Defaults to timeseries.

Index