Package 'fame'

November 5, 2021

Version 2.21.1

Title Interface for FAME Time Series Database

Author Jeff Hallman <jeffrey.j.hallman@frb.gov>

Maintainer Jeff Hallman <jeffrey.j.hallman@frb.gov>

Depends R (>= 2.3)

Imports tis

Description Read and write FAME databases.

License Unlimited

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-11-05 18:43:06 UTC

R topics documented:

fameAddAttribute *Add a FAME Attribute*

Description

Adds a user-defined attribute to data items in a FAME database. Data items in a FAME database can have up to 20 user-defined attributes of each of 6 types. Before you can set an item's `blah` attribute, the database itself must know about the attribute. This function defines the attribute to the database, so that `fameSetAttribute` can set it.

Usage

```
fameAddAttribute(name, type = c("string", "date", "boolean",
                 "precision", "numeric", "namelist"),
db)
```

Arguments

`name`	name of the attribute
`type`	one of the six predefined types. Note that FAME "precision" series are 64-bit doubles, while FAME "numeric" items are 32-bit floats.
`db`	string giving the relative or full path to the Fame database to read or write from. If the `fameLocalPath()` function is defined, it will be called on this argument to obtain a path.

Value

invisibly returns a character vector of whatever FAME wrote to standard output or standard error while attempting the assignment.

See Also

`fameSetAttribute` to set attributes once they've been defined.

fameAttributes *Fame Attributes*

Description

FAME has some predefined attributes (such as creation and update dates) for databases and data objects, and also allows user defined attributes for objects in a database. Currently, this package deals only with data object attributes.

Among the more useful predefined attributes you can access are an objects creation and update dates, via the `fameCreated` and `fameUpdated` convenience functions. However, you can also use

the more general fameAttribute function to access a string representation of any attribute, including user-defined ones.

Setting attributes

Before user-defined attributes can be set for a data object, the database itself must know the names and types of the attributes. Each data object can have up to twenty user-defined attributes of each of six data types: boolean, date, namelist, numeric, precision, and string. The names of the user-defined attributes are stored in the database in six namelist scalars named boolean_attribute_names, date_attribute_names, and so on, one for each data type. Functions will be provided shortly to do this.

Once the database knows what attributes are allowed, they can be set via fameSetAttribute.

fameAttribute reads one attribute of one object in a Fame database and returns it as a string. fameSetAttribute sets the value of one attribute of one object in a Fame database. getFameCreated and getFameUpdated return an object's created and updated attributes, respectively, as POSIXlt dates.

Usage

```
fameAttribute(attribute, fname, db)
fameSetAttribute(attribute, value, fname, db)
fameCreated(fname, db)
fameUpdated(fname, db)
```

Arguments

attribute	string naming the attribute to be returned
fname	name of an object in a FAME database
db	string giving the relative or full path to the Fame database to read or write from. If the fameLocalPath() function is defined, it will be called on this argument to obtain a path.
value	value to set the attribute to

Value

fameAttribute returns a named string.

fameSetAttribute returns whatever strings FAME wrote to standard output while setting the attribute.

fameCreated and fameUpdated each return a POSIXlt object.

Examples

```
## Not run:
fameUpdated("gdp.q", db = "somedb")

## End(Not run)
```

`fameConnection` *Fame 'master/dbback' server connection*

Description

Creates a connection to a Fame service running on a remote host.

Usage

```
fameConnection(service = "", host = "", user = "", password = "", stopOnFail = TRUE)
## S3 method for class 'fameConnection'
print(x, ...)
## S3 method for class 'fameConnection'
close(con, ...)
```

Arguments

`service`	The name of the service on the server, e.g., "someService"
`host`	The server name
`user`	User name on the server
`password`	A string
`stopOnFail`	if `TRUE` (the default), failing to connect generates an R error. Otherwise, the function completes but the result is kind of useless.
`x`	A fameConnection
`con`	A fameConnection
`...`	For `print`, these are arguments passed on to `print.matrix`. For `close`, they're ignored.

Details

`fameConnection` uses the C HLI function cfmopen to create a connection to a Fame service. The connection it returns can be used in subsequent calls to `fameWildlist` or `getfame`.

The `print` and `close` functions, when called on a fameConnection, do what they say.

Value

A fameConnection is an integer with attributes "service", "host", "user" and "password".

Note

These functions have not really been tested at all, as the package maintainer does not have ready access to a Fame server.

See Also

`getfame`, `fameWildlist`

fameCustomization *Local Customization of the Fame Interface*

Description

You can define two local functions, `fameLocalInit` and `fameLocalPath`, to customize some aspects of the FAME interface.

`fameLocalInit`: The first time one of the functions that interfaces with a Fame database is called, the internal function `fameStart` is called to initialize the HLI (Host Language Interface) and open a work database. After accomplishing that, `fameStart` checks to see if a function named `fameLocalInit` exists. If so, it is called with no arguments immediately after opening the work database.

`fameLocalPath`: Used to customize `getFamePath`.

fameIsScalar *Check for (Fame or Numeric) Scalar or Time Index Series*

Description

`fameIsScalar` checks whether or not its argument could be a 'scalar' in FAME parlance, i.e., if it is a character vector or an atomic object of length one and not a time series or time indexed series. `ti` objects of length one are what FAME calls a 'date', other scalars are strings, numbers, and logicals of length one. A vector of strings may be represented in FAME as a 'namelist'.

Usage

```
fameIsScalar(x)
```

Arguments

x	object to be checked.

Value

`TRUE` or `FALSE`.

Examples

```
fameIsScalar("this is a scalar, since it has length one")
fameIsScalar(c("returns FALSE", "since it has length two"))
fameIsScalar(tis::today())
```

getfame *Fame Interface*

Description

getfame and putfame read and write scalars (ti, numeric, character or logical objects of length 1) and numeric tis (Time Indexed Series) from and to Fame databases. FAME case series and series with string or FAME "date" types can be read by getfame, but not written with putfame.

fameWhats returns information about an object in a database, including its name, class, type, basis and observed attributes, as well as start (a ti Time Index) and length. If getDoc is TRUE, it will also include description and documentation components.

fameWhats is a wrapper around the function fameWhat, which provides the same information in a lower-level form.

fameWildlist returns a list giving the name, class, type and frequency of the objects in the database with names that match wildString.

Usage

```
getfame(sernames, db, connection = NULL, save = FALSE, envir = parent.frame(),
        start = NULL, end = NULL, getDoc = TRUE)
putfame(serlist, db, access = "shared", update = TRUE,
        checkBasisAndObserved = FALSE, envir = parent.frame())
fameWhats(db, fname, connection = NULL, getDoc = TRUE)
fameWildlist(db, wildString = "?", connection = NULL, nMax = 1000, charMode = TRUE)
```

Arguments

sernames	character vector of Fame names of series and/or scalars to retrieve.
db	string giving the relative or full path to the Fame database to read or write from. If the fameLocalPath() function is defined, it will be called on this argument to obtain a path. To open a remote database, use a string like "servername /path/to/a/database.db".
connection	created by calling fameConnection. Leave this at the default NULL setting for local databases.
save	if TRUE the retrieved series are individually saved in the environment specified by envir.
envir	for getfame, the environment used by assign to save the retrieved series if save is TRUE. For putfame, if serlist is a character vector, the environment in which to find the series that will be stored in the database. The default environment for both functions is the frame of the caller.
start	a ti object, or something that the ti function can turn into a ti object. The time index for the first observation in the returned series. The default is the series start in the database.

end	a ti object, or something that the ti function can turn into a ti object. The time index for the last observation in the returned series. The default is the series end in the database.
getDoc	if TRUE (the default), also get the series description and documentation attributes, accessible via functions of the same names.
serlist	the objects to be written to the database. This can be a character vector giving the names of the objects in the environment specified by envir, a list containing the objects, or an object itself. Objects that can be written to the database include scalars (single numbers, strings, and ti dates), and tis series.
access	string specifying the access mode to open the database in. Should be one of c("create","overwrite","read","shared","update"). **Warning:** Opening an existing database in "overwrite" mode wipes out the database. You almost never want to do this.
update	if TRUE (the default), existing series in the database will be updated. If FALSE, existing series in the database with the same names will be replaced by the series in serlist. Fame scalars (single strings, numbers or ti dates) always overwrite their previous incarnations.
checkBasisAndObserved	if TRUE and update == TRUE, the basis and observed attributes of any existing series with the same name will be checked for consistency with the updating series from serlist. If the basis or observed attributes differ, the update will not happen. This argument has no effect with respect to non-series objects.
fname	name of an object in a FAME database
wildString	string containing FAME wildcards
nMax	maximum number of matches to return
charMode	if TRUE (the default) return class, type and freq components as strings, rather than integer codes.

Details

Fame names vs. R names:

The R names of series may differ from their Fame names. For getfame, names(sernames) holds the R names of the retrieved series. If sernames does not have a names attributes, the R names will be the same as the Fame names.

Naming for putfame is more complicated, because the series specified by serlist for putfame may be univariate or multivariate. For a multivariate series, the column names of the matrix become the Fame names. Not having a name for each column is thus an error.

A univariate series may be a single-column matrix. If it is, and it has a column name, that becomes the Fame name of the series. Otherwise, the Fame name of a univariate series is the corresponding element of names(serlist). If serlist is an actual list of series, names(serlist) must be of the same length. For character vector serlist a names attribute is optional. If there isn't one, the Fame names will be the same as the R names.

Consistency checking when update == TRUE:

If there is already an existing series in the database with the same name as one in serlist, the Fame class, type, and frequency are checked for consistency between the updating series and the

existing series in the database. In addition, if checkBasisAndObserved is TRUE, those attributes are also checked. Inconsistencies for any of the checked attributes between the updating existing series will abort the update. The default value for checkBasisAndObserved is set to FALSE because this inconsistency is very common in MRA code.

Value

getfame returns a list of the retrieved objects. If save is T, the list is returned invisibly. The names of the list are the R names described in the details. Fame scalars are returned as strings, ti dates, or numbers. Case series are returned as vectors of the appropriate type, with the names attribute of each vector set to the case numbers. If getDoc is TRUE (the default), retrieved objects will also have attributes named description and documentation.

putfame invisibly returns an empty string.

fameWhats returns NULL if it can't find fname in db. Otherwise, it return a list with components

name	name of the object in the database
class	'series', 'scalar', or 'formula'
type	'undefined', 'numeric', 'namelist', 'boolean', 'string', 'precision', or 'date'

If the object is a date, there will also be a component

tif	tifName for the date's frequency

If the object is a series, there will also be components

basis	'undefined', 'daily' or 'business'
observed	'undefined', 'beginning', 'ending', 'averaged', 'summed', 'annualized', 'formula', 'high', or 'low'
length	number of observations in the series
start	a ti (TimeIndex) start date for the series

Note

These functions use the Fame HLI and a child server process. The function fameRunning is called to see if the server process is already running. If not, fameStart starts it and the HLI. Your .Last function should call fameStop to shut them down gracefully. In any given R session, once the Fame HLI has died for any reason, it cannot be restarted. This is a Fame limitation. On exit, getfame always closes whatever databases it opened, so there's no reason not to just leave the server running as long as the R session is alive. Death of the R process kills the server process as well.

Please note that the fameConnection functionality is not well-tested.

See Also

ti, tis, fameConnection

Examples

```
## Not run:
mydb <- "pathToMyDatabase"
boink <- getfame("gdp.q", db = mydb)     ## returns a list
gpd.q <- boink[[1]]                      ## or boink$gdp.q
getfame("gdp.q", db = mydb, save = TRUE) ## saves gdp.q in the current frame

## saves the series as "nominalIncome"
getfame(c(nominalIncome = "gdp.q"), db = mydb, save = TRUE)
seriesA <- tis(1:24, start = c(2002, 1), freq = 12)
seriesB <- tis(1:104, start = c(2002, 1), tif = "wmonday")
documentation(seriesB) <- paste("Line", 1:4, "of seriesB documentation")
## store them as "mser" and "wser"
putfame(c(mser = "seriesA", wser = "seriesB"), db = "myfame.db")

matrixSeries <- cbind(a = seriesA, b = seriesA + 3)
putfame(matrixSeries, db = "myfame.db") ## stores as "a" and "b" in Fame

## storing a scalar as "myscalar"
putfame(c(myscalar = 42), db = "myfame.db")

fameWildlist("myfame.db")
fameWhats("myfame.db", fname = "wser", getDoc = TRUE)

## End(Not run)
```

`getFamePath` *Path to a Fame Database*

Description

Attempts to resolve its argument to a database path.

Usage

```
getFamePath(dbString, stopOnFail = TRUE)
```

Arguments

`dbString`	A database name understood by the local customizing function `fameLocalPath` (if it exists), the name of a file in the current directory, or the full path name of a database.
`stopOnFail`	If `TRUE`, stop with an error message if a readable database cannot be found.

Details

If a local function called fameLocalPath exists, it is called first on dbString to get a pathname. Otherwise, dbString is presumed to be the pathname. The pathname is checked for existence and readability. If there is a white space in the pathname, it is presumed valid, as that might specifying a Fame Server. Otherwise, if there is a failure and stopOnFail is TRUE, the function fails with an error message.

putfame calls this function with stopOnFail = FALSE, because it normally creates a new database if there isn't one there already.

Value

A path to a Fame database or Fame Server specification.

See Also

fameCustomization

---

lowLevelFame *Low Level Fame Interface Functions*

Description

These are most of the lower level functions used in the FAME interface. Most users will never need any of these functions, as the higher level function getfame and putfame do almost everything they want to do. The functions documented here were written in the course of implementing getfame and putfame, and some of them may prove useful on their own.

fameRunning answers TRUE if there is a process called "FAME SERVER" already running under the user's id and with the current R process as its parent process.

fameStart initializes the FAME HLI and may open a work database. Since the work database is always the first one opened, its key is always 0.

fameStop kills the HLI session and the FAME SERVER process started by fameStart. In any given R session, you cannot restart the HLI once it has died for any reason. (This is a FAME limitation, not an R one.) Death of the R process also kills the child FAME SERVER process. So it rarely makes sense to call fameStop explicitly, as it makes any subsequent FAME interaction in the current R session impossible.

fameState returns the state of the FAME HLI interface, which is one of c("none","starting","running","dead"). "none" means the HLI has not been started, "starting" means the HLI has not finished starting yet (this usually indicates an error), "running" means the HLI is up and running, while "dead" means the HLI has been killed, either by an error or by fameStop. In either case, once the HLI is "dead", it cannot be revived.

fameCommand sends its string argument to the child FAME SERVER process to be executed. If silent is TRUE, it invisibly returns a status code that can be sent to fameStatusMessage to get an error message. If silent is FALSE, the status message is echoed to standard output. If capture is TRUE, output from the FAME output channel is returned, with a "status" attribute holding the FAME status code.

fameStatusMessage looks up and returns the error message associated with its argument.

fameConnect opens a connection to FAME Database Server (Master or MCADBS), returning an integer connKey which is used by fameDbOpen to open a database on that connection.

fameDbOpen opens the named database in the given access mode. It returns an integer dbKey, which is a required argument for some of the other functions documented here.

fameDbClose closes the database associated with the given dbKey.

fameDeleteObject deletes a named object from a database.

fameWriteSeries writes the tis (Time Indexed Series) object ser as fname in the database associated with dbKey. If an object by that name already exists in the database and update is TRUE, the frequency and type of ser are checked for consistency with the existing object, and if checkBasisAndObserved is TRUE (not the default), those items are also checked. Any inconsistencies cause the update to fail. If all checks are OK, then the range covered by ser is written to the database. If update if FALSE, any existing series called fname in the database will be replaced by ser. This function should probably not be called directly, as putfame provides a nicer interface.

fameWriteScalar writes the tis (Time Indexed Series) object ser as fname in the database associated with dbKey. If an object by that name already exists in the database and update is TRUE, the frequency and type of ser are checked for consistency with the existing object, and if checkBasisAndObserved is TRUE (not the default), those items are also checked. Any inconsistencies cause the update to fail. If all checks are OK, then the range covered by ser is written to the database. If update if FALSE, any existing series called fname in the database will be replaced by ser. This function should probably not be called directly, as putfame provides a nicer interface.

fameWhat returns a list of low level information about an object in a database, including components named status, dbKey, name, class, type, freq, basis, observ, fyear, fprd, lyear, lprd, obs, and range. If getDoc is TRUE, it will also include description and documentation components. See the FAME documentation for the CHLI functions cfmwhat and cfmsrng for details.

Usage

```
fameRunning()
fameStart(workingDB = TRUE)
fameStop()
fameState()
fameCommand(string, silent = TRUE, capture = FALSE)
fameStatusMessage(code)
fameDbOpen(dbName, accessMode = "read", connection = NULL, stopOnFail = TRUE)
fameDbClose(dbKey, closeConnection = FALSE)
fameDeleteObject(db, fname)
fameWriteScalar(dbName, fname, scalar, update = TRUE,
                type = c("date", "precision", "boolean", "string", "namelist"))
fameWriteSeries(dbKey, fname, ser, update = FALSE, checkBasisAndObserved = FALSE)
fameWhat(dbKey, fname, getDoc = FALSE)
```

Arguments

workingDB	if TRUE (the default) open a temporary working database whose dbKey will be 0
string	a FAME command to be executed

silent	run the command quietly if TRUE
capture	capture and return strings from the FAME output channel if TRUE
code	an integer status code from FAME
dbName	name of or path to the database to open
accessMode	a string specifying the access model to open the database in: one of "read", "create", "overwrite", "update", or "shared".
connection	created by calling fameConnection. Leave this at the default NULL setting for local databases.
stopOnFail	should the function stop() if the database cannot be opened? Also, see the return value below.
dbKey	integer returned by dbOpen
closeConnection	if TRUE, close the fameConnection associated with the database as well. The default is to leave the connection open. This implies that whoever opened the connection in the first place should explicitly close it. This only applies to databases that were opened on a connection, not to local databases.
fname	name of an object in a FAME database
scalar	object to be written to the database
type	kind of FAME scalar to create in the database. If this argument is missing, it is inferred from characteristics of scalar. In fact, the only time you should ever really need to use this argument is if scalar is a character object of length one but you really want it to be written as a 'namelist' of length one, rather than a 'string' scalar.
update	if TRUE update any existing series by the same name in place. If FALSE, replace existing series.
db	can take dbKey or dbName form; that is, it can be an integer returned by dbOpen, or it can be the name of a database or path to a database.
ser	a tis time series
checkBasisAndObserved	see description above for fameWriteSeries
getDoc	if TRUE, also return the description and documentation attributes.

Value

fameRunning return a Boolean.

fameStart and fameStop return nothing.

fameStatus returns a string as documented above.

If capture is FALSE, fameCommand invisibly returns a status code. If capture is TRUE, strings sent to the FAME output channel are returned as a character vector, and the status code is returned as the "status" attribute of that vector.

fameStatusMessage returns a message string.

fameDbOpen returns an integer dbKey. If the parameter stopOnFail is FALSE and the open does not succeed, the returned integer is -1 and it will have attributes "status" and "statusMessage" indicating the nature of the failure.

fameDbClose returns a status code.

fameDeleteObject returns a status code.

The return value from fameWriteScalar is useless and should be ignored.

fameWriteSeries returns a status code.

fameWhat returns a list.

See Also

getfame, putfame, fameCustomization

---

osUtilities *Operating System Utilities*

Description

user returns the user id of the current user.

runningLinux is shorthand for Sys.info()["sysname"] == "Linux".

runningWindows is shorthand for .Platform$OS.type == "windows".

Usage

```
user()
runningLinux()
runningWindows()
```

---

tifToFameName *FAME Names for Time Index Frequencies*

Description

Returns the FAME names for the given time index frequencies.

Usage

```
tifToFameName(tif)
```

Arguments

tif	character vector of tifNames or a numeric vector of tif codes.

Value

A character vector as long as the input giving the FAME names of the input.

See Also

tif, tifName

Examples

```
tifToFameName(tis::tif(tis::today()))
tifToFameName(tis::tif(tis::latestMonth()))
tifToFameName(tis::tifName(tis::today()))
tifToFameName(tis::tifName(tis::latestMonth()))
```

Index