Time Series Database Interface: R SQLite (TSSQLite)

November 7, 2009

1 Introduction

The code from the vignette that generates this guide can be loaded into an editor with *edit(vignette("TSSQLite"))*. This uses the default editor, which can be changed using *options()*. It should be possible to view the pdf version of the guide for this package with *print(vignette("TSSQLite"))*.

WARNING: running these example will overwrite tables in the SQLite "test" database on the server.

In SQLite there does not seem to be any need to set user or password information, and examples here all use the localhost.

Once R is started, the functions in this package are made available with

```
> library("TSSQLite")
```

This will also load required packages *TSdbi*, *DBI*, *RSQLite*, *methods*, and *tframe*. Some examples below also require *zoo*, and *tseries*.

The next small section of code is necessary to setup database tables that are used in the examples below. It needs to be done only once for a database and might typically be done by an administrator setting up the database, rather than by an end user.

```
> m <- dbDriver("SQLite")
> con <- dbConnect(m, dbname = "test")
> source(system.file("TSsql/CreateTables.TSsql", package = "TSdbi"))
> dbDisconnect(con)
```

More detailed description of the instructions for building the database tables is given in the vignette for the *TSdbi* package. Those instruction show how to build the database using database utilites rather than R, which might be the way a system administrator would build the database.

2 Using the Database - TSdbi Functions

This section gives several simple examples of putting series on and reading them from the database. (If a large number of series are to be loaded into a database, one would typically do this with a batch process using the database program's utilities for loading data.) The first thing to do is to establish a connection to the database:

```
> m <- dbDriver("SQLite")
> con <- TSconnect(m, dbname = "test")
```

TSconnect uses *dbConnect* from the *DBI* package, but checks that the database has expected tables, and checks for additional features. (It cannot be used before the tables are created, as done in the previous section.)

This puts a series called *vec* on the database and then reads is back

```
> z <- ts(rnorm(10), start = c(1990, 1), frequency = 1)
> seriesNames(z) <- "vec"
> if (TSexists("vec", con)) TSdelete("vec", con)
> TSput(z, con)
> z <- TSget("vec", con)
```

If the series is printed it is seen to be a "ts" time series with some extra attributes.

TSput fails if the series already exists on the *con*, so the above example checks and deletes the series if it already exists. *TSreplace* does not fail if the series does not yet exist, so examples below use it instead. Several plots below show original data and the data retrieved after it is written to the database. One is added to the original data so that both lines are visible.

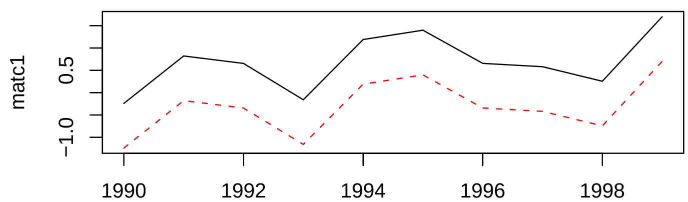
And now more examples:

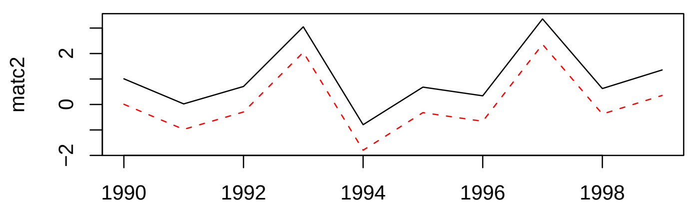
```
> z <- ts(matrix(rnorm(20), 10, 2), start = c(1990, 1), frequency = 1)
> seriesNames(z) <- c("matc1", "matc2")
> TSreplace(z, con)

[1] TRUE

> TSget("matc1", con)

Time Series:
Start = 1990
End = 1999
Frequency = 1
         1          2          3          4          5          6          7
-0.7060459  1.2433445 -1.6558812 -1.0956140 -1.9227243  1.6755923 -0.9495925
         8          9         10
-0.2891391  0.6889406  1.2805746
attr(,"seriesNames")
[1] matc1
```

```
attr(,"TSrefperiod")
[1] NA
attr(,"TSmeta")
An object of class "TSmeta"
Slot "TSdescription":
[1] NA

Slot "TSdoc":
[1] NA

Slot "TSlabel":
[1] NA

Slot "serIDs":
[1] "matc1"

Slot "conType":
[1] "TSSQLiteConnection"
attr(,"package")
[1] "TSSQLite"

Slot "DateStamp":
[1] NA

Slot "dbname":
[1] "test"

Slot "hasVintages":
[1] FALSE

Slot "hasPanels":
[1] FALSE

> TSget("matc2", con)

Time Series:
Start = 1990
End = 1999
Frequency = 1
         1          2          3          4          5          6          7
 0.3345641 -0.1325715 -1.2097930 -1.4234570 -0.3734598  0.6508256 -0.8703147
         8          9         10
-0.6242307 -0.4000986 -0.5907594
attr(,"seriesNames")
[1] matc2
attr(,"TSrefperiod")
```

```
[1] NA
attr(,"TSmeta")
An object of class "TSmeta"
Slot "TSdescription":
[1] NA

Slot "TSdoc":
[1] NA

Slot "TSlabel":
[1] NA

Slot "serIDs":
[1] "matc2"

Slot "conType":
[1] "TSSQLiteConnection"
attr(,"package")
[1] "TSSQLite"

Slot "DateStamp":
[1] NA

Slot "dbname":
[1] "test"

Slot "hasVintages":
[1] FALSE

Slot "hasPanels":
[1] FALSE

> TSget(c("matc1", "matc2"), con)

Time Series:
Start = 1990
End = 1999
Frequency = 1
          matc1      matc2
1990 -0.7060459  0.3345641
1991  1.2433445 -0.1325715
1992 -1.6558812 -1.2097930
1993 -1.0956140 -1.4234570
1994 -1.9227243 -0.3734598
1995  1.6755923  0.6508256
1996 -0.9495925 -0.8703147
```

```
1997 -0.2891391 -0.6242307
1998  0.6889406 -0.4000986
1999  1.2805746 -0.5907594
attr(,"TSrefperiod")
[1] NA NA
attr(,"TSmeta")
An object of class "TSmeta"
Slot "TSdescription":
[1] NA

Slot "TSdoc":
[1] NA

Slot "TSlabel":
[1] NA

Slot "serIDs":
[1] "matc1" "matc2"

Slot "conType":
[1] "TSSQLiteConnection"
attr(,"package")
[1] "TSSQLite"

Slot "DateStamp":
[1] NA

Slot "dbname":
[1] "test"

Slot "hasVintages":
[1] FALSE

Slot "hasPanels":
[1] FALSE

> tfplot(z + 1, TSget(c("matc1", "matc2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```





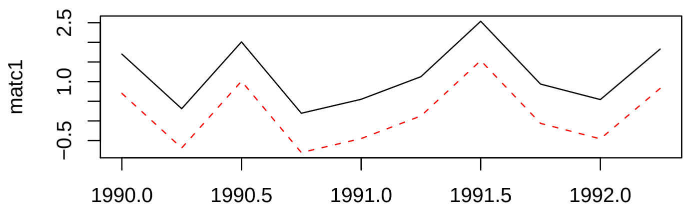

```
> z <- ts(matrix(rnorm(20), 10, 2), start = c(1990, 1), frequency = 4)
> seriesNames(z) <- c("matc1", "matc2")
> TSreplace(z, con)
```

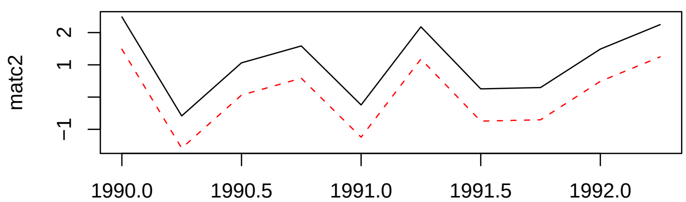
```
[1] TRUE
```

```
> TSget(c("matc1", "matc2"), con)
```

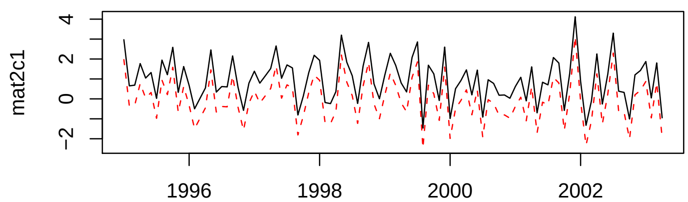
```
             matc1       matc2
1990 Q1  1.0109385 -0.06033777
1990 Q2 -0.2996076  0.61005070
1990 Q3 -0.1437853 -0.28847469
1990 Q4 -0.1171355  0.50617075
1991 Q1  1.1763224 -0.12007934
1991 Q2  0.7401393  0.11175086
1991 Q3  1.5285295  1.97731712
1991 Q4 -1.1647018  0.86622174
1992 Q1 -0.1204523 -0.82924619
1992 Q2  0.5873416  0.88111401
attr(,"TSrefperiod")
[1] NA NA
attr(,"TSmeta")
An object of class "TSmeta"
```

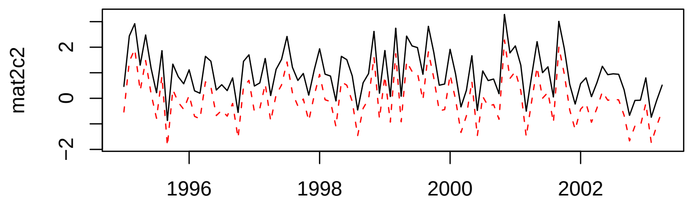
```
Slot "TSdescription":
[1] NA

Slot "TSdoc":
[1] NA

Slot "TSlabel":
[1] NA

Slot "serIDs":
[1] "matc1" "matc2"

Slot "conType":
[1] "TSSQLiteConnection"
attr(,"package")
[1] "TSSQLite"

Slot "DateStamp":
[1] NA

Slot "dbname":
[1] "test"

Slot "hasVintages":
[1] FALSE

Slot "hasPanels":
[1] FALSE

> tfplot(z + 1, TSget(c("matc1", "matc2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```






```
> z <- ts(matrix(rnorm(200), 100, 2), start = c(1995, 1), frequency = 12)
> seriesNames(z) <- c("mat2c1", "mat2c2")
> TSreplace(z, con)

[1] TRUE

> tfplot(z + 1, TSget(c("mat2c1", "mat2c2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```





The following extract information about the series from the database, although not much information has been added for these examples.

```
> TSmeta("mat2c1", con)
> TSmeta("vec", con)
> TSdates("vec", con)
> TSdescription("vec", con)
> TSdoc("vec", con)
```

Below are exampoles that make more use of *TSdescription* and codeTSdoc.

Often it is convenient to set the default connection:

```
> options(TSconnection = con)
```

and then the *con* specification can be omitted from the function calls unless another connection is needed. The *con* can still be specified, and some examples below do specify it, just to illustrate the alternative syntax.

```
> z <- TSget("mat2c1")
> TSmeta("mat2c1")
```

```
An object of class "TSmeta"
Slot "TSdescription":
[1] "NA"
```

```
Slot "TSdoc":
[1] "NA"

Slot "TSlabel":
[1] NA

Slot "serIDs":
[1] "mat2c1"

Slot "conType":
[1] "TSSQLiteConnection"
attr(,"package")
[1] "TSSQLite"

Slot "DateStamp":
[1] NA

Slot "dbname":
[1] "test"

Slot "hasVintages":
[1] FALSE

Slot "hasPanels":
[1] FALSE
```

Data documentation can be in two forms, a description specified by *TSdescription* or longer documentation specified by *TSdoc*. These can be added to the time series object, in which case they will be written to the database when *TSput* or *TSreplace* is used to put the series on the database. Alternatively, they can be specified as arguments to *TSput* or *TSreplace*. The description or documentation will be retrieved as part of the series object with *TSget* only if this is specified with the logical arguments *TSdescription* and *TSdoc*. They can also be retrieved directly from the database with the functions *TSdescription* and *TSdoc*.

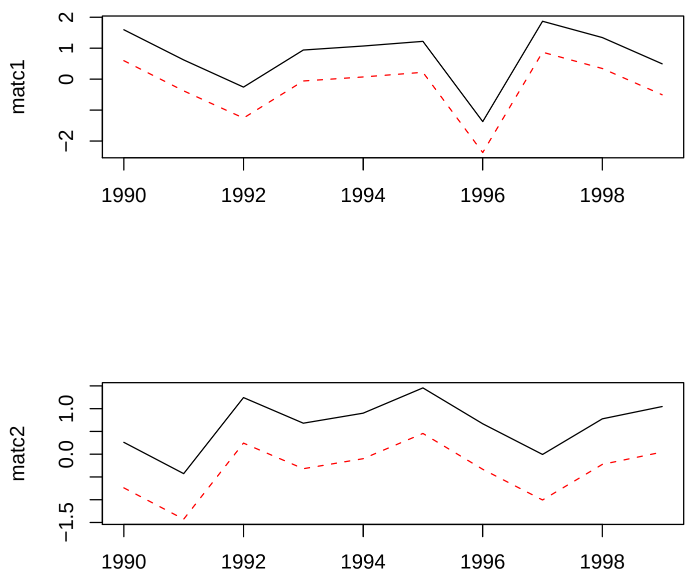
```
> z <- ts(matrix(rnorm(10), 10, 1), start = c(1990, 1), frequency = 1)
> TSreplace(z, serIDs = "Series1", con)
```

```
[1] TRUE
```

```
> zz <- TSget("Series1", con)
> TSreplace(z, serIDs = "Series1", con, TSdescription = "short rnorm series",
      TSdoc = "Series created as an example in the vignette.")
```

```
[1] TRUE
```

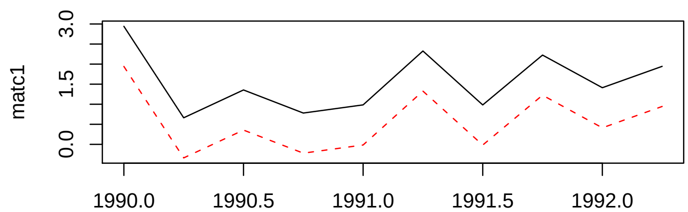
```
> zz <- TSget("Series1", con, TSdescription = TRUE, TSdoc = TRUE)
> start(zz)

[1] 1990    1

> end(zz)

[1] 1999    1

> TSdescription(zz)

[1] "short rnorm series"

> TSdoc(zz)

[1] "Series created as an example in the vignette."

> TSdescription("Series1", con)

[1] "short rnorm series"

> TSdoc("Series1", con)

[1] "Series created as an example in the vignette."

> z <- ts(rnorm(10), start = c(1990, 1), frequency = 1)
> seriesNames(z) <- "vec"
> TSreplace(z, con)

[1] TRUE

> zz <- TSget("vec", con)
> z <- ts(matrix(rnorm(20), 10, 2), start = c(1990, 1), frequency = 1)
> seriesNames(z) <- c("matc1", "matc2")
> TSreplace(z, con)

[1] TRUE

> tfplot(z + 1, TSget(c("matc1", "matc2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```

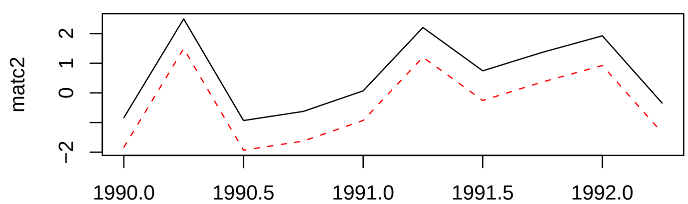



```
> z <- ts(matrix(rnorm(20), 10, 2), start = c(1990, 1), frequency = 4)
> seriesNames(z) <- c("matc1", "matc2")
> TSreplace(z, con)
```

```
[1] TRUE
```

```
> tfplot(z + 1, TSget(c("matc1", "matc2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```

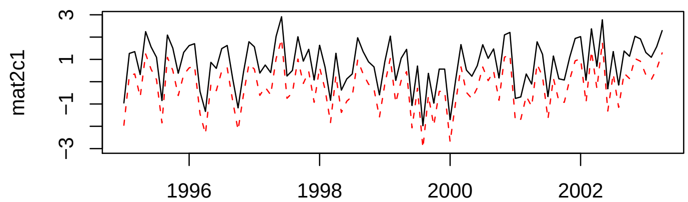


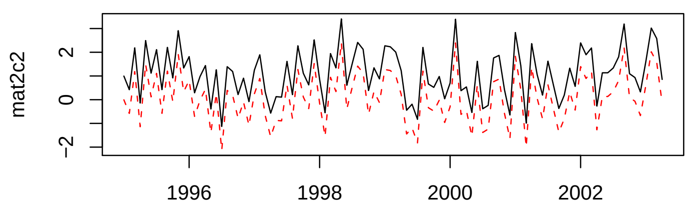



```
> z <- ts(matrix(rnorm(200), 100, 2), start = c(1995, 1), frequency = 12)
> seriesNames(z) <- c("mat2c1", "mat2c2")
> TSreplace(z, con)
```

```
[1] TRUE
```

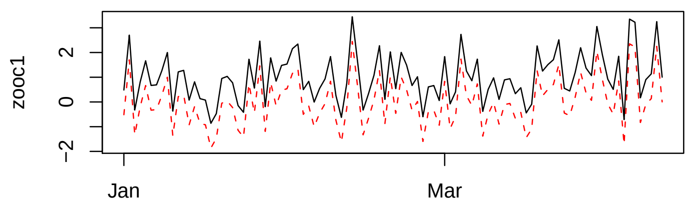
```
> tfplot(z + 1, TSget(c("mat2c1", "mat2c2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```

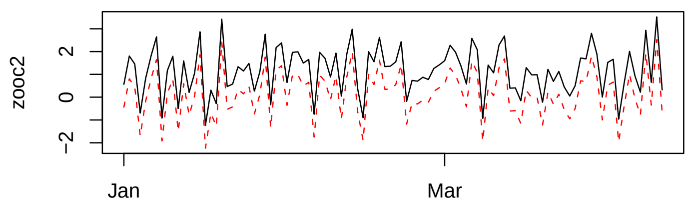




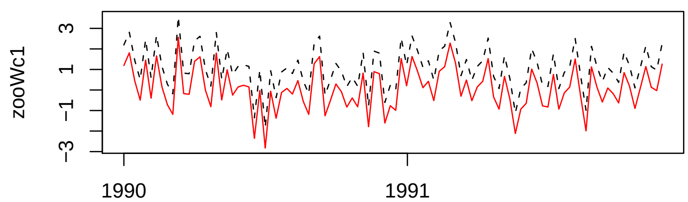
The following examples use dates and times which are not handled by *ts*, so the *zoo* time representation is used.

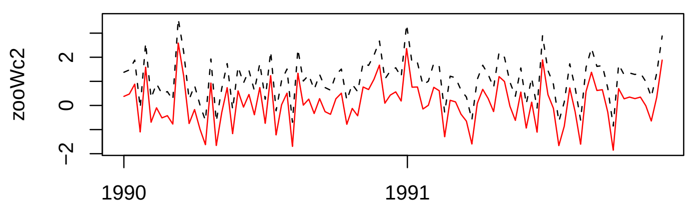
```
> require("zoo")
> z <- zoo(matrix(rnorm(200), 100, 2), as.Date("1990-01-01") +
      0:99)
> seriesNames(z) <- c("zooc1", "zooc2")
> TSreplace(z, con, Table = "D")

[1] TRUE

> tfplot(z + 1, TSget(c("zooc1", "zooc2"), con), lty = c("solid",
      "dashed"), col = c("black", "red"))
```






```
> z <- zoo(matrix(rnorm(200), 100, 2), as.Date("1990-01-01") +
      0:99 * 7)
> seriesNames(z) <- c("zooWc1", "zooWc2")
> TSreplace(z, con, Table = "W")

[1] TRUE

> tfplot(z + 1, TSget(c("zooWc1", "zooWc2"), con), col = c("black",
      "red"), lty = c("dashed", "solid"))
```






```
> dbDisconnect(con)
```

3 Examples Using Web Data

This section illustrates fetching data from a web server and loading it into the database. This would be a very slow way to load a database, but provides examples of different kinds of time series data. The fetching is done with *TShistQuote* which provides a wrapper for *get.hist.quote* from package *tseries* to give syntax consistent with the *TSdbi*.

Fetching data may fail due to lack of an Interenet connection or delays.

First establish a connection to the database where data will be saved:

```
> con <- TSconnect("SQLite", dbname = "test")
```

Now connect to the web server and fetch data:

```
> require("TShistQuote")
> Yahoo <- TSconnect("histQuote", dbname = "yahoo")
> x <- TSget("^gspc", quote = "Close", con = Yahoo)
> plot(x)
> tfplot(x)
> TSrefperiod(x)
```

```
[1] "Close"
```

```
> TSdescription(x)
```

```
[1] "^gspc Close  from  yahoo"
```

```
> TSdoc(x)
```

```
[1] "^gspc Close  from  yahoo retrieved  2009-11-07 15:11:16"
```

```
> TSlabel(x)
```

```
[1] "^gspc Close"
```

Then write the data to the local server, specifying table B for business day data (using TSreplace in case the series is already there from running this example previously):

```
> TSreplace(x, serIDs = "gspc", Table = "B", con = con)
```

```
[1] TRUE
```

and check the saved version:

```
> TSrefperiod(TSget(serIDs = "gspc", con = con))
```

```
[1] "Close"
```

```
> TSdescription("gspc", con = con)
```

```
[1] "^gspc Close  from  yahoo"
```

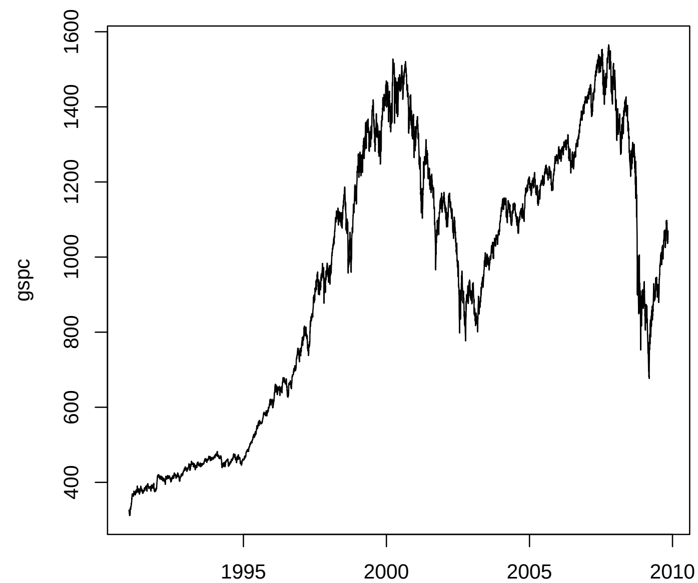
```
> TSdoc("gspc", con = con)
```

```
[1] "^gspc Close  from  yahoo retrieved  2009-11-07 15:11:16"
```

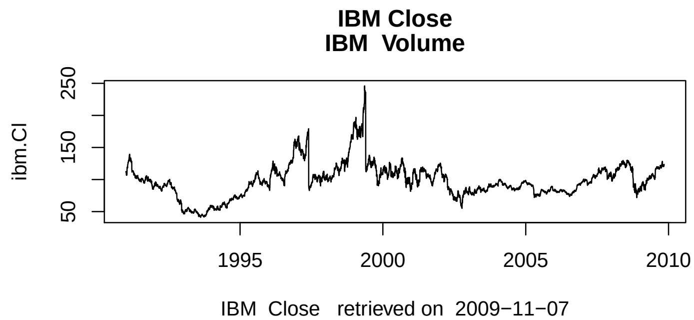
```
> TSlabel("gspc", con = con)
```

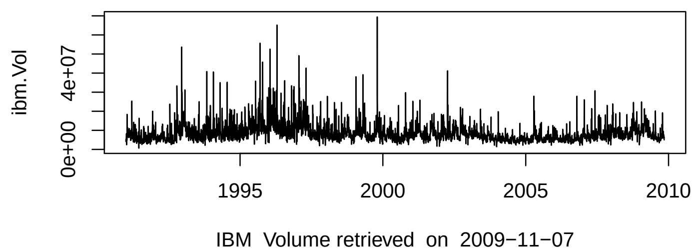
```
[1] NA
```

```
> tfplot(TSget(serIDs = "gspc", con = con))
```




```
> x <- TSget("ibm", quote = c("Close", "Vol"), con = Yahoo)
> TSreplace(x, serIDs = c("ibm.Cl", "ibm.Vol"), con = con, Table = "B",
      TSdescription. = c("IBM Close", "IBM  Volume"), TSdoc. = paste(c("IBM  Close   retriev
          "IBM  Volume retrieved  on "), Sys.Date()))

[1] TRUE

> z <- TSget(serIDs = c("ibm.Cl", "ibm.Vol"), TSdescription = TRUE,
      TSdoc = TRUE, con = con)
> TSdescription(z)

[1] "IBM Close"   "IBM  Volume"

> TSdoc(z)

[1] "IBM  Close   retrieved on  2009-11-07"
[2] "IBM  Volume retrieved  on  2009-11-07"

> tfplot(z, xlab = TSdoc(z), Title = TSdescription(z))
> tfplot(z, Title = "IBM", start = "2007-01-01")
```





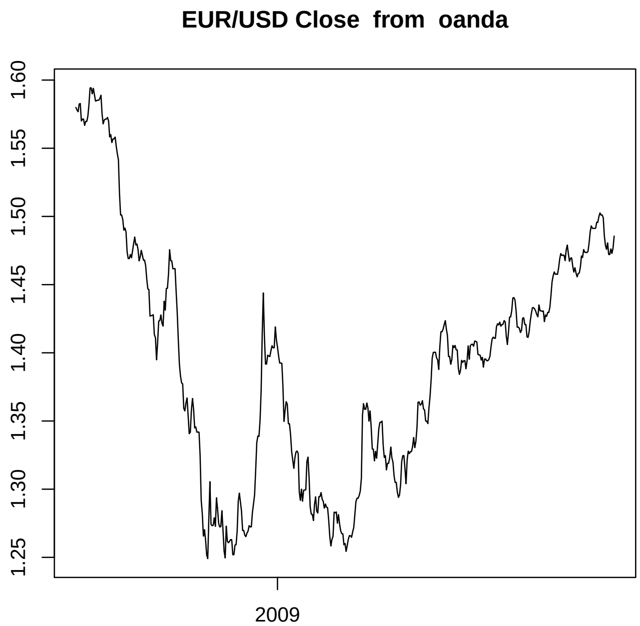
Oanda has maximum of 500 days, so the start date is specified here so as to not exceed that.

```
> Oanda <- TSconnect("histQuote", dbname = "oanda")
> x <- TSget("EUR/USD", start = Sys.Date() - 495, con = Oanda)
> TSreplace(x, serIDs = "EUR/USD", Table = "D", con = con)
```

```
[1] TRUE
```

Then check the saved version:

```
> z <- TSget(serIDs = "EUR/USD", TSlabel = TRUE, TSdescription = TRUE,
      con = con)
> tfplot(z, Title = TSdescription(z), ylab = TSlabel(z))
> tfplot(z, Title = "EUR/USD", start = "2007-01-01")
> tfplot(z, Title = "EUR/USD", start = "2007-03-01")
> tfplot(z, Title = "EUR/USD", start = Sys.Date() - 14, end = Sys.Date(),
      xlab = format(Sys.Date(), "%Y"))
```




```
> dbDisconnect(con)
> dbDisconnect(Yahoo)
> dbDisconnect(Oanda)
```

3.1 Examples Using TSdbi with ets

The database called "ets" is available at the Bank of Canada. These examples are illustrated in the *TSMySQL* and *TSpadi* packages, but ets is not yet implemented under *TSSQLite*.

4 Examples Using DBI and direct SQL Queries

The following examples are queries using the underlying "DBI" functions. They should not often be needed to access time series, but may be useful to get at more detailed information, or formulate special queries.

```
> m <- dbDriver("SQLite")
> con <- TSconnect(m, dbname = "test")
> options(TSconnection = con)

> dbListTables(con)
```

```
 [1] "A"    "B"    "D"    "I"    "M"    "Meta" "Q"    "S"    "T"    "U"
[11] "W"
```

If schema queries are supported then table information can be obtained in a (almost) generic SQL way. On some systems this will fail because users do not have read priveleges on the INFORMATION_SCHEMA table. This does not seem to be an issue in SQLite, but I have not figured out the SQLite implementation so the following are wrapped in *try()*.

```
> try(dbGetQuery(con, paste("SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.Columns ",
      " WHERE TABLE_SCHEMA='test' AND table_name='A' ;")))
> try(dbGetQuery(con, paste("SELECT COLUMN_NAME, COLUMN_DEFAULT, COLLATION_NAME, DATA_TYPE,'
      "CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, NUMERIC_PRECISION",
      "FROM INFORMATION_SCHEMA.Columns WHERE TABLE_SCHEMA='test' AND table_name='A' ;")))
> try(dbGetQuery(con, paste("SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, NUMERI
      "FROM INFORMATION_SCHEMA.Columns WHERE TABLE_SCHEMA='test' AND table_name='M';")))
```

Finally, to disconnect gracefully, one should

```
> dbDisconnect(options()$TSconnection)
> options(TSconnection = NULL)
```