Time Series Database Interface: R Padi to Fame (TSpadi)

October 26, 2011

1 Introduction

The code from the vignette that generates this guide can be loaded into an editor with *edit(vignette("TSpadi"))*. 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("TSpadi"))*.

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

```
> library("TSpadi")
```

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

TSpadi is a *TSdbi* style interface to a server which extracts data from a backend database. In these examples, Fame is the backend database engine. The R client communicates with the server using an RPC based protocol which has been called "padi", so only the server requires the Fame programs and licences.

TSpadi does not support writing data to a database, so examples below are restricted to an already available database. Alternate approaches for interfacing to Fame are the R packages *fame* and *TSfame*. The later is just a wrapper for the former, to give it a *TSdbi* style interface. Both allow writing to the Fame database, but both also require the Fame HLI programs and licences for the client R machine.

While *TSpadi* does not require Fame on the R clients, there is additional work to set up the padi server and configure it to load databases. Please check additional comments in the last section "TS PADI Server" before deciding about using the *TSpadi* interface (and you may want to contact me, Paul Gilbert). I do actively use this interface, but one of the main purposes of *TSpadi* is to standardize on the *TSdbi* API so that it is easier to move among different database engines.

1.1 Examples Using TSdbi with ets

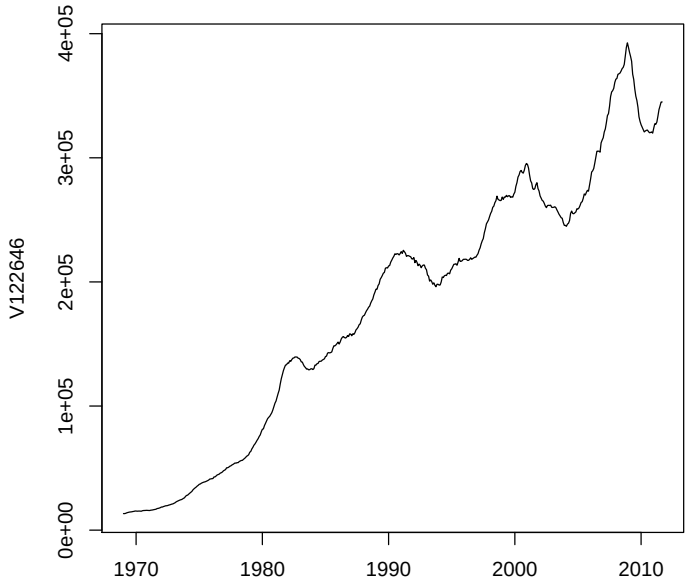
These examples use a database called "ets" which is available at the Bank of Canada. This set of examples illustrates how the programs might be used if a

larger database is available. Typically a large database would be installed using database scripts directly rather than from R with *TSput* or *TSreplace*.

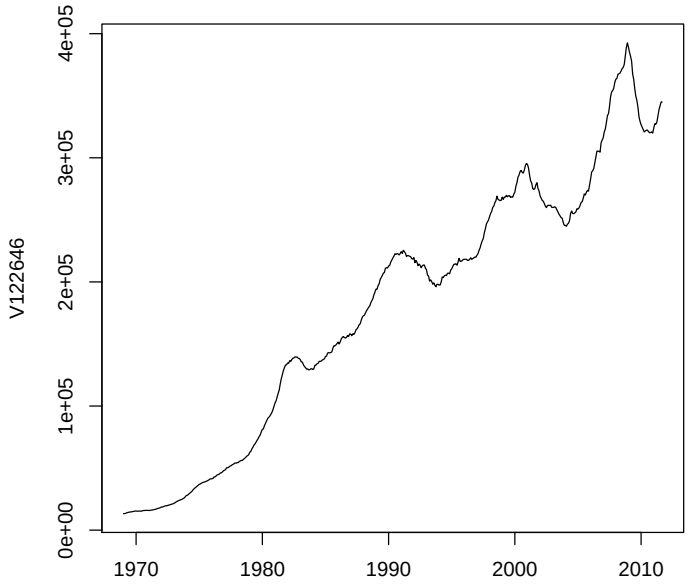
The following are wrapped in *if (!inherits(conets, "try-error"))* so that the vignette will build even when the database is not available. This seems to require an explicit call to *print()*, but that is not usually needed to display results below. Another artifact of this is that results printed in the if block do not display until the end of the block. Also, graphics are not displayed in the vignette because, when they are not generated, the missing file causes an error building the vignette.

```
> m <- dbDriver("padi")
> conets <- try(TSconnect(m, dbname = "ets"))
> if (!inherits(conets, "try-error")) {
      options(TSconnection = conets)
      print(TSmeta("M.SDR.CCUSMA02.ST"))
      z <- TSget("M.SDR.CCUSMA02.ST")
      EXCH.IDs <- t(matrix(c("M.SDR.CCUSMA02.ST", "SDR/USD exchange rate",
          "M.CAN.CCUSMA02.ST", "CAN/USD exchange rate", "M.MEX.CCUSMA02.ST",
          "MEX/USD exchange rate", "M.JPN.CCUSMA02.ST", "JPN/USD exchange rate",
          "M.EMU.CCUSMA02.ST", "Euro/USD exchange rate", "M.OTO.CCUSMA02.ST",
          "OECD /USD exchange rate", "M.G7M.CCUSMA02.ST", "G7   /USD exchange rate",
          "M.E15.CCUSMA02.ST", "Euro 15. /USD exchange rate"),
          2, 8))
      print(TSdates(EXCH.IDs[, 1]))
      z <- TSdates(EXCH.IDs[, 1])
      print(start(z))
      print(end(z))
      tfplot(TSget(serIDs = "V122646", conets))
  }

serIDs:  M.SDR.CCUSMA02.ST
 from dbname  ets using  TSpadiConnection
     [,1]
[1,] "M.SDR.CCUSMA02.ST from 1960 1 to 2011 9       12"
[2,] "M.CAN.CCUSMA02.ST from 1960 1 to 2011 9       12"
[3,] "M.MEX.CCUSMA02.ST from 1963 1 to 2011 9       12"
[4,] "M.JPN.CCUSMA02.ST from 1960 1 to 2011 9       12"
[5,] "M.EMU.CCUSMA02.ST from 1979 1 to 2011 9       12"
[6,] "M.OTO.CCUSMA02.ST  not available"
[7,] "M.G7M.CCUSMA02.ST  not available"
[8,] "M.E15.CCUSMA02.ST  not available"
[[1]]
[1] 1960    1

[[2]]
[1] 1960    1
```

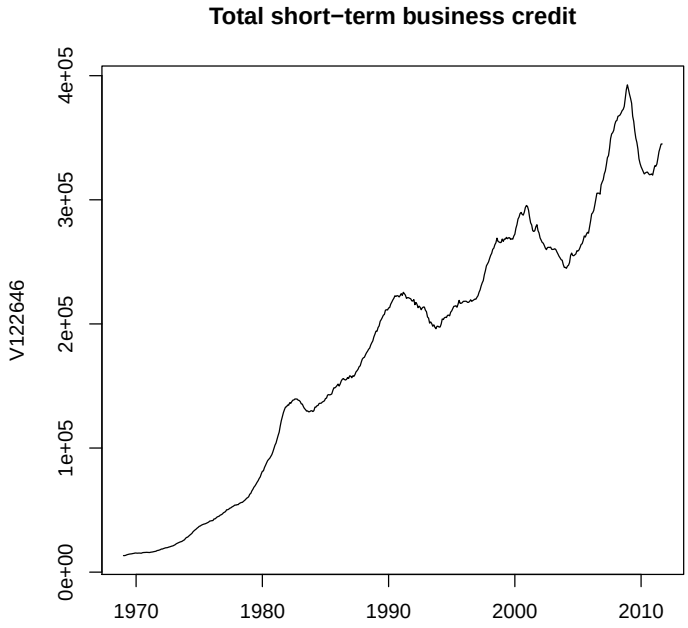
```
[[3]]
[1] 1963    1

[[4]]
[1] 1960    1

[[5]]
[1] 1979    1

[[6]]
[1] NA

[[7]]
[1] NA

[[8]]
[1] NA

[[1]]
[1] 2011    9

[[2]]
[1] 2011    9

[[3]]
[1] 2011    9

[[4]]
[1] 2011    9

[[5]]
[1] 2011    9

[[6]]
[1] NA

[[7]]
[1] NA

[[8]]
[1] NA
```



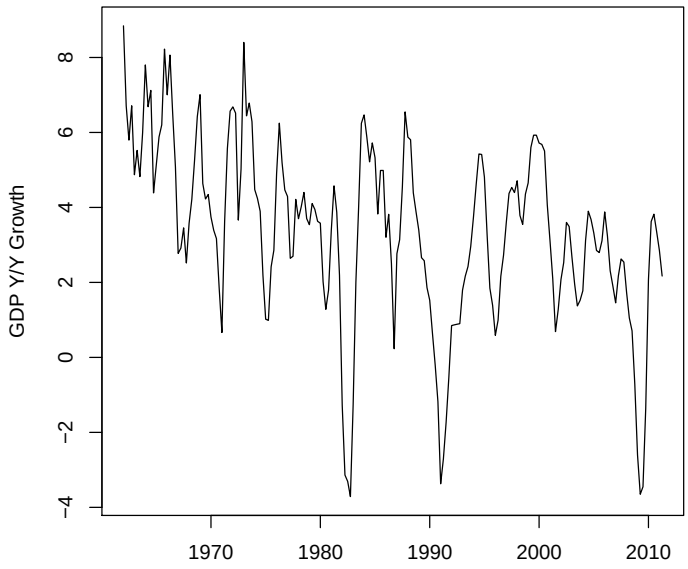

```
> if (!inherits(conets, "try-error")) {
      print(TSdescription(TSget("V122646", TSdescription = TRUE)))
      print(TSdoc(TSget("V122646", TSdoc = TRUE)))
      tfplot(TSget("V122646", names = "V122646", conets))
  }

[1] NA
[1] NA
```




```
> if (!inherits(conets, "try-error")) {
      z <- TSget("V122646", TSdescription = TRUE)
      tfplot(z, Title = "Total short-term business credit")
  }
```




```
> if (!inherits(conets, "try-error")) {
      ETSgdp <- annualizedGrowth(aggregate(TSget("V1992067"), nfrequency = 4,
          FUN = mean), lag = 4, names = "GDP Y/Y Growth")
      tfplot(ETSgdp)
  }
```



2 TS PADI Server

Building a padi server will require some programming effort. The padi interface is getting to be fairly old and, although it still works, some of the underlying code should probably be replaced with a newer approach, perhaps based on something like SOAP.

Code and a description of a prototype of a standard for a Time Series Protocol for Application - Database Interface (TS PADI) may still be available at http://www.bank-banque-canada.ca/pgilbert. The code includes a working interface to a Fame database.