Vignette for `c.tveci` and `r.tveci`

This example demonstrates how to run code for fitting a threshold vector error correction (TVEC) model which includes data imputation for modeling minimum gasoline prices in a city based on crude oil prices.

The data set used to illustrate the code includes these prices for the city of Perth each day in 2015. The data set `PerthMin2015.dat` has four columns `date`, `MinPrice`, `CP`, and `TodayCPAvailable`. The column `date` gives the date in *yyyymmdd* format, `MinPrice` gives the lowest gas price in Perth on that date, `CP` is the crude oil price on that date, and `TodayCPAvailable` is an indicator variable which is 1 if the crude oil price was available on the date and 0 if not. The following commands can be used to read the data set into R and to view the last few lines.

```
> perthData=as.matrix(read.table("PerthMin2015.dat", sep="\t",
+ header=TRUE))
> tail(perthData,15)
           date MinPrice    CP TodayCPAvailable
[351,] 20151217    124.9 34.95                1
[352,] 20151218    121.9 34.73                1
[353,] 20151219    118.9  0.00                0
[354,] 20151220    115.9  0.00                0
[355,] 20151221    112.9 34.74                1
[356,] 20151222    134.9 36.14                1
[357,] 20151223    127.9 37.50                1
[358,] 20151224    125.9 38.10                1
[359,] 20151225    121.9  0.00                0
[360,] 20151226    118.9  0.00                0
[361,] 20151227    114.9  0.00                0
[362,] 20151228    111.9 36.81                1
[363,] 20151229    129.9 37.87                1
[364,] 20151230    127.9 36.60                1
[365,] 20151231    124.9 37.04                1
```

The source code for five R functions are freely available in the file `Rcode.R`. It can be loaded into R with the following command.

```
> source("Rcode.R")
```

The function `r.tvec` fits a TVEC model that allows for additional explantory variables if available in the matrix `otherX`. The function `c.tvec` is a wrapper function for C code that is equivalent to `r.tvec`, but faster since

it is written in C rather than R. The function `predict.tvec` accepts the fitted TVEC model and input variables needed to predict the minimum gasoline prices on the next day. The function `r.tveci` implements the novel imputation procedure for estimating minimum gasoline prices based on the TVEC model by imputing missing crude oil prices. The function `c.tveci` is equivalent but uses the `c.tvec` function internally instead of the `r.tvec` function for fitting intermediate TVEC models so again it is faster because the computationally intense parts use C code instead of R code.

Some of the computationally intense parts are written in C to improve the speed of the code. The C source code is freely available in the file `tvec.c`. On a 64-bit Windows machine, the C library can be loaded using the following command.

```
> dyn.load("tvec.dll")
```

For users who prefer not to use pre-complied code, information on using and creating shared objects and DLL files from source code is described in section 5.3 of the manual *Writing R Extensions* available at
`https://cran.r-project.org/doc/manuals/r-release/R-exts.html`
and the links therein.

The following command shows how to use `c.tveci` to fit the TVEC model with data imputation.

```
> c.tveci.model=c.tveci(perthData[,2:4],otherX=NULL,lag=7)
```

Estimates of the parameters β, γ, and A can be extracted as follows.

```
> c.tveci.model$beta
[1] 2.485031
> c.tveci.model$gamma
[1] -23.49134
> c.tveci.model$A
              [,1]          [,2]
 [1,]  0.657464776 -0.0640037550
 [2,] 26.964071997 -1.3834669567
 [3,]  0.045929340  0.0883909500
 [4,] -1.598539043 -0.5158844018
 [5,]  0.250834801  0.0364460943
 [6,] -0.162630564 -1.0535092885
 [7,] -0.157445711  0.0354086570
 [8,] -2.381661579 -0.3275854288
 [9,] -0.174165647  0.0006437405
```

```
[10,]  0.701938881 -0.4317512551
[11,] -0.555751888 -0.0300257600
[12,]  0.280600450 -0.5417526773
[13,] -0.309200769  0.0075029184
[14,] -0.051480041 -0.3783098821
[15,]  0.150411011 -0.0445297302
[16,] -0.574459435 -0.5333856631
[17,]  0.007588942  0.0008712315
[18,] -0.300923104 -0.0069614959
[19,] -0.475463000 -0.0122684091
[20,]  0.172415463 -0.1499350877
[21,] -0.430621787 -0.0034398839
[22,] -0.156951525 -0.0618227198
[23,] -0.385253952 -0.0185141341
[24,] -0.109217591  0.1807023351
[25,] -0.359273852  0.0092258659
[26,] -0.097996782  0.0572361988
[27,] -0.319527890 -0.0065896080
[28,]  0.088564853 -0.0455894004
[29,] -0.282338380 -0.0176420312
[30,] -0.096759720 -0.0098054168
[31,]  0.483957744 -0.0063457632
[32,]  0.004916119 -0.0534337235
```

As part of the output, the data set with the imputed missing values are also stored in a matrix `imputed`. The last fifteen lines for this matrix in the output with this data set are shown below.

```
> tail(c.tveci.model$imputed,15)
       MinPrice       CP
[351,]    124.9 34.95000
[352,]    121.9 34.73000
[353,]    118.9 34.87144
[354,]    115.9 34.67435
[355,]    112.9 34.74000
[356,]    134.9 36.14000
[357,]    127.9 37.50000
[358,]    125.9 38.10000
[359,]    121.9 37.92839
[360,]    118.9 38.62688
[361,]    114.9 38.54207
```

```
[362,]    111.9 36.81000
[363,]    129.9 37.87000
[364,]    127.9 36.60000
[365,]    124.9 37.04000
```

The function `r.tveci` gives identical results. However, the computation time on a Dell 3.4 GHz workstation running Windows 10 is 23 seconds for `r.tveci` but only 5 seconds for `c.tveci`.

We can use the results for the fitted model stored in `c.tveci.model` to predict the change in minimum gas price and the change in crude oil price on January 1, 2016 using the function `predict.tveci`.

```
> predict.tveci(c.tveci.model)
             [,1]
[1,] -3.17819595
[2,] -0.07977937
```

So, the minimum gas price is predicted to decrease by 3.18 cents; the actual price decrease from December 31, 2015 to January 1, 2016 turned out to be 5.5 cents. (No crude oil price was reported on January 1, 2016 since it was a holiday.)

The `c.tveci` function also gives the option of including additional covariates in the argument `otherX`. For example, consider adding the explanatory variable

```
> oX=matrix(c(0,0,0,0,0,1,rep(c(rep(0,6),1),51),0,0),365,1)
```

which is an indicator for the day of the week being a Tuesday. Then the following commands show how to fit the TVEC model with data imputation and predict the next day.

```
> c.tveci.model=c.tveci(perthData[,2:4],otherX=oX,lag=7)
> predict.tveci(c.tveci.model)
          [,1]
[1,] -1.223218
[2,]  1.245802
```