# Read and Prepare Data for VAR Estimation

`read_data.m`

*by* Jaromir Benes

15 March 2014

Summary

Read two CSV data files with some basic U.S. time series (some monthly, some quarterly), filter the data using an HP filter with tunes, and prepare a database that will be later used to estimate a VAR model.

## Contents

## 1 Clear Workspace

```
clear;
close all;
clc;
%#ok<*NOPTS>
```

## 2 Load CSV Data Files

There are two CSV data files included in this tutorial: `USQuarterly.csv` and `USMonthly.csv`. These files have been downloaded from the St Louis FRB FRED database. Run the function `dbload` to read the data files and create databases. Because the date format used in FRED files is different from the default IRIS format, use the option `'dateFormat='`. In addition, because quarterly dates are represented by a monthly calendar date in FRED files, use the option `'freq='` to tell IRIS that those are quarterly and not monthly data.

Note CSVs are plain text files with time series arranged columnwise and separated by commas. The files can can be opened, viewed and edited in any spreadsheet programs (e.g. MS Excel or Mac Numbers) or plain text editors. IRIS uses CSV as the main format for exporting and imporing data from and to other software packages (EViews, Troll, etc.).

The resulting databases, `Q` 1 and `M` 2, returned from the function `dbload` contain every variable from the respective CSV file as a tseries object.

```
Q = dbload('USQuarterly.csv', ...
    'dateformat=','YYYY-MM-01','freq=',4,'leadingRow=','DATE');

Q 1

M = dbload('USMonthly.csv', ...
    'dateformat=','YYYY-MM-01','leadingRow=','DATE');

M 2
```

```
Q =
    BOPBCA: [87x1 tseries]
       GDP: [88x1 tseries]
    GDPC96: [88x1 tseries]
M =
      AHETPI: [265x1 tseries]
        AWHI: [265x1 tseries]
    CPIAUCSL: [265x1 tseries]
    CPILEGNS: [265x1 tseries]
    CPILEGSL: [265x1 tseries]
    CPILFENS: [265x1 tseries]
    CPILFESL: [265x1 tseries]
       LOANS: [265x1 tseries]
        M1SL: [265x1 tseries]
      PCEC96: [204x1 tseries]
      PPICEM: [265x1 tseries]
       TB3MS: [265x1 tseries]
```

## 3 Convert Monthly Series to Quarterly

The VAR model will be estimated on quarterly data. Convert therefore monthly series to quarterly. The function `convert` [3] does averaging by default; there are though other options available .

Convert the series one by one in a loop. Note that we could replace the following five lines with a single command, making use of the function `dbfun`, which applies a function to every field of a database:

```
fn = @(x) convert(x,'Q');
M = dbfun(fn,M);
```

This line would do exactly the same job.

```
list = fieldnames(M);
for i = 1 : length(list)
    name = list{i};
    M.(name) = convert(M.(name),4); [3]
end
```

## 4 Transform Data

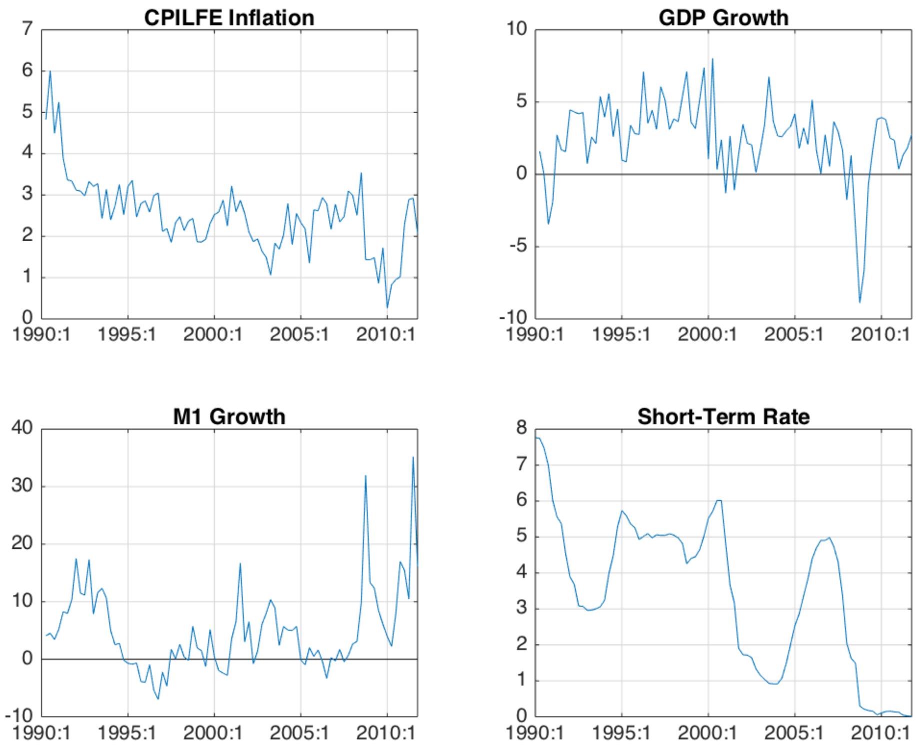
Create four series used later in estimating a VAR model:

- CPI inflation;
- GDP growth;
- M1 growth;
- 3-month interest rate.

The function `apct` [4] computes quarter-on-quarter (or period-by-period in general) percent growth rates annualised. The function `dbplot` [6] creates a figure window and plots graphs for each series specified in the list (notice how titles are entered in double quotes [5]); the `Inf` stands for the entire date range available.

```
d = struct();
d.pp = apct(M.CPILEGSL); 4
d.yy = apct(Q.GDPC96);
d.mm = apct(M.M1SL);
d.r = M.TB3MS;

d

dbplot(d,Inf, ...
    {'"CPILFE Inflation" pp', ... 5
    '"GDP Growth" yy', ...
    '"M1 Growth" mm', ...
    '"Short-Term Rate" r'}, ...
    'zeroLine=',true); 6
```

```
d =
    pp: [87x1 tseries]
    yy: [87x1 tseries]
    mm: [87x1 tseries]
     r: [88x1 tseries]
```



## 5 Define Dates

Use the GDP growth series to define the start date and end date of the historical sample. This is because GDP data come usually with the longest publication lag.

```
startHist = get(d.yy,'start')
endHist = get(d.yy,'end')
```

```
startHist =
   7.9610e+03
endHist =
   8.0470e+03
```

## 6 Run Plain HP Filter

Both the growth rates and interest rates display considerable trend, or low-frequency, movements.

To go the easiest possible way to estimate a stationary VAR, detrend all series by an HP filter. First, run the common HP, with a default smoothing parameter (lambda) of 1,600 for quarterly series. The HP lambda parameter can be changed by setting the option `'lambda='` (the option takes a default value depending on the periodicity of the input series). Capture the low-frequency trend components in the database `t0` [7], and the cyclical (gap) components in the database `g0` [8].

```
t1 = struct();
g1 = struct();

[t1.pp,g1.pp] = hpf(d.pp); [7] [8]
[t1.yy,g1.yy] = hpf(d.yy);
[t1.mm,g1.mm] = hpf(d.mm);
[t1.r,g1.r] = hpf(d.r);

t1
g1
```

```
t1 =
    pp: [87x1 tseries]
    yy: [87x1 tseries]
    mm: [87x1 tseries]
     r: [88x1 tseries]
g1 =
    pp: [87x1 tseries]
    yy: [87x1 tseries]
    mm: [87x1 tseries]
     r: [88x1 tseries]
```

## 7 Run HP Filter with Tunes

Run another HP filter with two modifications:

- Increase the smoothing parameter to 10,000 by using the option `'lambda='`.
- Use tunes (or judgmental adjustments) on the rate of change in trends. Specifically, a restriction that at the beginning and at the end of the sample, the change in the trend must be zero. If projected into the future or into the past, the trends would remain flat lines at both ends.

Create two new databases to capture the results, `t2` and `g2`; these databases will be later used for estimating a VAR.

```
lmb = 10000;

```

```
x = tseries();
x(startHist) = 0;
x(endHist) = 0;

t2 = struct();
g2 = struct();

[t2.pp,g2.pp] = hpf(d.pp,startHist:endHist, ...
    'lambda=',lmb,'change=',x); [9]

[t2.yy,g2.yy] = hpf(d.yy,startHist:endHist, ...
    'lambda=',lmb,'change=',x);

[t2.mm,g2.mm] = hpf(d.mm,startHist:endHist, ...
    'lambda=',lmb,'change=',x);

[t2.r,g2.r] = hpf(d.r,startHist:endHist, ...
    'lambda=',lmb,'change=',x);

t2
g2
```

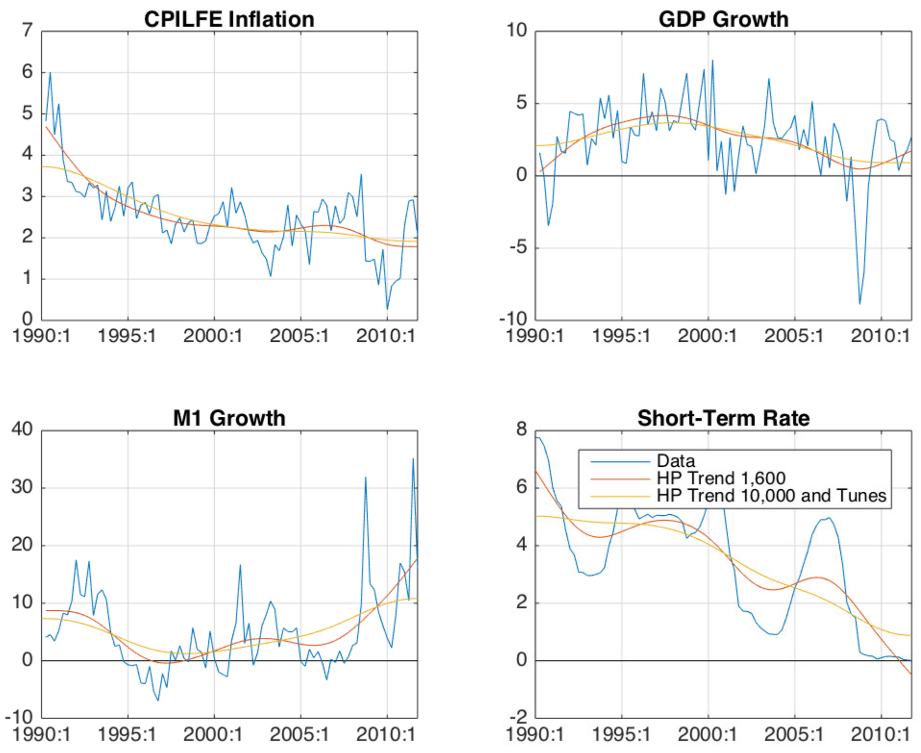
```
t2 =
    pp: [88x1 tseries]
    yy: [88x1 tseries]
    mm: [88x1 tseries]
     r: [88x1 tseries]
g2 =
    pp: [87x1 tseries]
    yy: [87x1 tseries]
    mm: [87x1 tseries]
     r: [87x1 tseries]
```

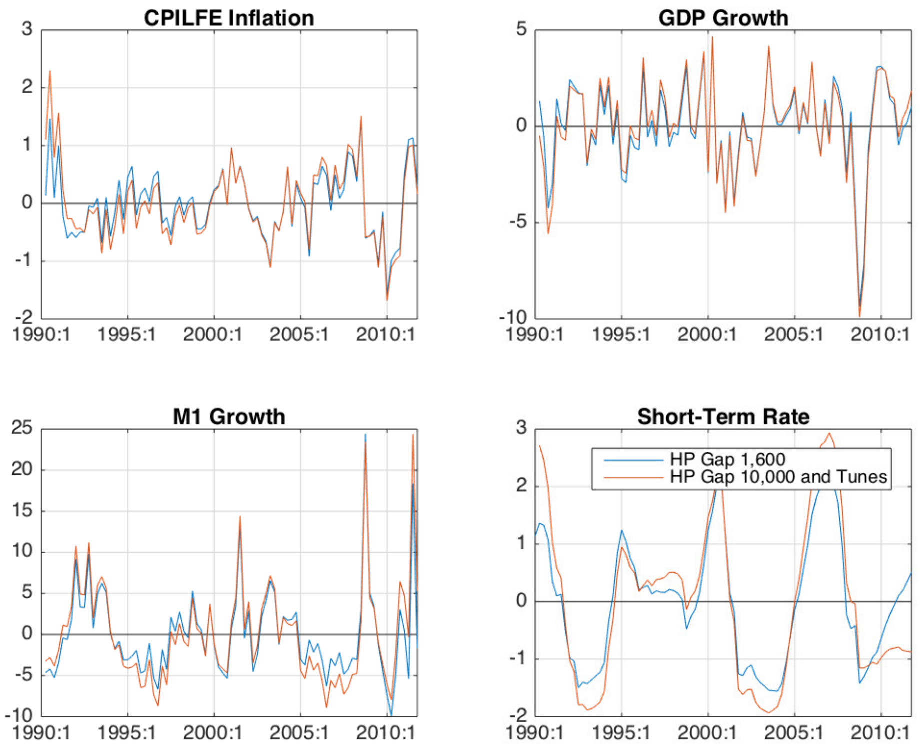
## 8 Plot Filtered Data

Plot the trend [10] and the gap [11] databases against the historical data.

```
dbplot(d & t1 & t2,Inf, ... [10]
    {'"CPILFE Inflation" pp', ...
    '"GDP Growth" yy', ...
    '"M1 Growth" mm', ...
    '"Short-Term Rate" r'}, ...
    'zeroLine=',true);

legend('Data','HP Trend 1,600','HP Trend 10,000 and Tunes');
```

```

dbplot(g1 & g2,Inf, ... 11
    {'"CPILFE Inflation" pp', ...
    '"GDP Growth" yy', ...
    '"M1 Growth" mm', ...
    '"Short-Term Rate" r'}, ...
    'zeroLine=',true);

legend('HP Gap 1,600','HP Gap 10,000 and Tunes');
```





## 9 Save Databases and Dates for Further Use

Save all data to a mat file for further use.

```
save read_data.mat d g2 t1 t2 startHist endHist;
```

## 10 Help on IRIS Functions Used in This File

Use either `help` to display help in the command window, or `idoc` to display help in an HTML browser window.

```
help dbase/dbload
help dbase/dbfun
help dbase/dbplot
help tseries/convert
help tseries/get
help tseries/hpf
```