Back testing for model validation

There are two approaches for model validation and selection: in-sample comparison and out-of-sample comparison.

In-sample comparison: When applying techniques such as AIC/BIC and EACF, the entire dataset is used for both the estimation of model coefficients and for model comparison. Thus those criteria select the best model that describes the dynamic structure of the observed time series

Out-sample comparison: If objective is forecasting, models should be compared based on their forecasting performance. One approach is called back-testing, and described below.

Back-testing: A time series $\{x_t\}$ of length T is divided into two subsamples: an estimation (training) subset $E=\{x_1,\ldots,x_K\}$ and a forecasting (validation) subset $F=\{x_{K+1},\ldots,x_T\}$. As a rule of thumb, 60-80% of the data are used for estimation and the remaining 20-40% observations are used for model validation. The estimation subsample should be large enough so that model parameters can be estimated.

Step 0: The model *m* is estimated using the first K observations (Estimation dataset E). Then one-step ahead forecast $\hat{x}_K(1)$ and its forecast error $e_{K+1} = x_{K+1} - \hat{x}_K(1)$ are computed.

Step 1: *Rolling Window:* Advance the estimation by **one data point**, re-estimate the model using data up to K+j , i.e. $E_j=\{x_{1+j},\ldots,x_{K+j}\}$ for j=1 and compute the next one-step ahead forecast and error. Note that the data value to be forecasted by the model is never used for estimation.

Step 2: Repeat Step 1 for j=2,…,T-K until we have all the forecast errors $\{e_{K+1},\ldots,e_T\}$ where $e_{K+j} = x_{K+j} - \hat{x}_{K+j-1}(1)$ for the data in the validation set $F=\{x_{K+1},\ldots,x_T\}$.

A summary metric for the forecast errors is computed using the Root Mean Square Forecasting Error as

$$RMSFE\ (m) = \sqrt{\sum_{j=K+1}^{T} [e_j(1)]^2 / (T-K)}$$ where m is the model to be tested.

An alternative metric is the Absolute Mean Forecast Error of model m computed as

$$MAFE\ (m) = \sum_{j=K+1}^{T} |e_j(1)| / (T-K)$$

We can use either the RMSFE or the MAFE metrics to compare different models $\{m_1, m_2, ..m_s\}$. For each model we apply the backtesting procedure and compute RMSFE(m_i) or MAFE(m_i). The final model is selected as the one that minimizes RMSFE(m_i) or MAFE(m_i) depending on which metric is selected. Thus the selected model is the one that with the smallest prediction errors on average.

How to compute back-testing in R and SAS

R CODE

We can use the backtest() function written by Tsay that is defined in the backtest.R file. Download the file from Week 4 documents and save it in your work folder.

The backtest() function has the following arguments:
backtest(m1,rt,orig,h,xre=NULL,fixed=NULL,inc.mean=TRUE)

- m1: is a time-series model object created using the arima() function
- rt: the time series to be analyzed (numeric vector)
- orig: is the starting forecast origin (i.e. K = size of the estimation dataset)
- h: forecast horizon - it should be **h=1** for one-step ahead forecasts
- xre: the independent variable (if covariates are added to the model)
- fixed: parameter constraint if some model parameters are set to zero in the arima() model. Same definition of fixed in the arima() function.
- inc.mean = TRUE if model contains intercept. inc.mean = FALSE if model has no intercept.

Example in R (see unemprate_BT.R under week 4)
The following code assumes that the models m1 and m2 are defined using the arima() function, drate is the time series data to be analyzed, orig=700 (so 700 values are used for estimation), h=1 (1 step-ahead forecast is computed)

```
source("backtest.R")
pm1 = backtest(m1, drate, orig=700, h=1)
pm2 = backtest(m2, drate, orig=700, h=1, fixed=c(0,NA,NA,NA,NA,NA))
```

SAS CODE (by Bill Qualls)

The following macro computes the back-testing procedure in SAS. The macro was written by Bill Qualls, a student in the 2013 Winter section of CSC425. For simplicity, just save the macro on top of your SAS file. You can recall the macro in your code using the following statement:

```
%backtest(trainpct=perc_value, dataset=dataname, var=varname, date=date_name,
                    p=p_value, q=q_value, interval=int_value);
```

where

TRAINPCT = Percent of dataset to be used for training. Example: trainpct = 80 for 80% data for training and 20% testing.
DATASET = SAS dataset containing time series data. Example: DATASET=Unemp
VAR = Name of time series variable. Example: VAR=ratechg

P = order of specified AR component of ARMA model– same as in ARIMA PROC. Omit otherwise. Example: p=3 for AR(3) model, or p=(1 3 6) for AR(6) with only lag 1, 3 and 6 coefficients.

Q = order of specified MA component of ARMA model – same as in ARIMA PROC. Omit otherwise. Example: p=3, q=3 for ARMA(3,3) model; or Q=(1 3 6) for MA(6) with only lag 1,3,6 coefficients

DATE = Name of date variable. Defaults to date. Example: DATE=date (where date is defined in SAS dataset defined in Dataset attribute.

INTERVAL = Date interval. Defaults to month. (same definition as in FORECAST statement of PROC ARIMA). Example: INTERVAL=day

Just insert the %backtest() statement in your SAS code. See example of application of macro in unemprate_BT_macro.sas under week 4. Macro is saved in backtesting.sas file for your convenience.

Example in SAS:

The following code assumes that the SAS data set containing the time series variable is *mydata* , *drate* contains the daily data to be analyzed, training set contains 80% of data, and the chosen ARMA model is an ARMA(2,3).

```
%backtest(trainpct=80, dataset=mydata, var=drate, date=date, p=2, q=3, interval=day);
```

```
/*MACRO FOR BACKTESTING */

%macro backtest(TRAINPCT=80, DATASET=, VAR=, P=, Q=, DATE=date, INTERVAL=month);

* ------------------------------------------------------------------------ ;
*   T I M E   S E R I E S   B A C K T E S T   M A C R O
* ------------------------------------------------------------------------ ;
*   DePaul CSC425, Winter 2013, Dr. Raffaella Settimi
*   Macro written by Bill Qualls, First Analytics
* ------------------------------------------------------------------------ ;
*  E X P L A N A T I O N   O F   P A R A M E T E R S
*  (Order of variables is insignificant)
* ------------------------------------------------------------------------ ;
*   TRAINPCT .. Percent of dataset to be used for training.
*               So, (100 - TRAINPCT)% will be used for evaluation.
*               Example: TRAINPCT=80
*   DATASET  .. Time series dataset. Libname optional, defaults to Work.
*               Example: DATASET=Work.Unemp
*   VAR      .. Name of time series variable.
*               Example: VAR=ratechg
```

```
*   P        .. Specified for AR models. Omit otherwise.
*               Example: P=(1 3 6)
*   Q        .. Specified for MA models. Omit otherwise.
*               Example: Q=(1 3 6)
*   DATE     .. Name of date variable. Defaults to date.
*               Example: DATE=date
*   INTERVAL .. Date interval. Defaults to month.
*               Example: INTERVAL=day
* ------------------------------------------------------------------------ ;
*  S A M P L E   U S A G E
*  %backtest(trainpct=80, dataset=work.unemp, var=ratedif, p=5, interval=month);
* ------------------------------------------------------------------------ ;

* How many records are in the dataset? ;
data _null_;
call symput('NRECS', trim(left(nrecs)));
set &DATASET nobs=nrecs;
stop;
run;

* Determine which ones are exclusively for training based on TRAINPCT ;
%let SIZE_OF_ROLLING_WINDOW = %sysfunc(round(&NRECS * &TRAINPCT / 100));

* create a working copy of dataset with observation number  ;
* as a variable for use in a where clause with proc arima.  ;
* also add a placeholder for the predicted value.           ;

data Work._MY_COPY_ (keep = &VAR &DATE _OBS_ _PRED_);
set &DATASET;
    _OBS_ = _N_;
    _PRED_ = .;
run;

* turn off log -- too lengthy ;
filename junk dummy;
proc printto log=junk print=junk;
run;

* Will build the model once for each record used in evaluation. ;
* Each time I will predict one record forward.                  ;

%let MODELS_TO_BE_BUILT = %sysevalf(&NRECS - &SIZE_OF_ROLLING_WINDOW);

%do i = 1 %to &MODELS_TO_BE_BUILT;
```

```
* Model using SIZE_OF_ROLLING_WINDOW records, and make one prediction ;
proc arima data=Work._MY_COPY_ plots=none;
where _OBS_ ge &i
     and _OBS_ le (&i + &SIZE_OF_ROLLING_WINDOW - 1);
identify var=&VAR noprint;
estimate
    %if (&P ne ) %then %do; p=&P %end;
    %if (&Q ne ) %then %do; q=&Q %end;
    method=ml noprint;
forecast lead=1 id=&DATE interval=&INTERVAL out=Work._MY_RESULTS_ noprint;
run;

* get the predicted value (in the last record) as a macro variable ;
data _null_;
p = nrecs;
set Work._MY_RESULTS_ point=p nobs=nrecs;
call symput("FORECAST", forecast);
stop;
run;

* move that prediction to its place in the output file ;
proc sql noprint;
update Work._MY_COPY_
    set _PRED_ = &FORECAST
    where _OBS_ = &i + &SIZE_OF_ROLLING_WINDOW;
quit;
run;

* show progress so far ;
%if (%sysfunc(mod(&i, 5)) = 0) %then %do;

         * print on;
    proc printto log=log print=print;
    run;

    %put Finished &i iterations;

    * print off again;
    proc printto log=junk print=junk;
    run;

%end;
```

```
%end;

* turn print back on ;
proc printto log=log print=print;
run;

* calculate prediction error;
data Work._MY_COPY_;
set Work._MY_COPY_;
Predicted_Error = (&VAR - _PRED_) ;
Predicted_Error_Squared = (&VAR - _PRED_)**2;
absresidual = abs(Predicted_Error);
run;

* compute and report the mean square forecast error;
%if (&P eq ) %then %let PP = ; %else %let PP = p=&P;
%if (&Q eq ) %then %let QQ = ; %else %let QQ = q=&Q;
title "Backtest results for &DATASET";
title2 "Model: &PP &QQ";

proc summary data=Work._MY_COPY_;
where _OBS_ > &SIZE_OF_ROLLING_WINDOW;
var Predicted_Error absresidual;
output out=outm mean(absresidual)=mafe mean(Predicted_Error_Squared)=msfe;
run;

data outm;
set outm;
rmsfe=sqrt(msfe);
run;

proc print data=outm;
run;

/*
proc means data=Work._MY_COPY_ N MEAN STDDEV CLM T PROBT;
where _OBS_ > &SIZE_OF_ROLLING_WINDOW;
var Predicted_Error_Squared;
run;
title;
*/
%mend backtest;
```