Package: hdflex (via r-universe)

August 29, 2024

Type Package

Title High-Dimensional Aggregate Density Forecasts

Version 0.2.1

Maintainer Sven Lehmann <sven.lehmann@uni-rostock.de>

Description Provides a forecasting method that maps vast numbers of (scalar-valued) signals of any type into an aggregate density forecast in a time-varying and computationally fast manner. The method proceeds in two steps: First, it transforms a predictive signal into a density forecast. Second, it combines the generated candidate density forecasts into an ultimate density forecast. The methods are explained in detail in Adaemmer et al. (2023) <doi:10.2139/ssrn.4342487>.

License GPL (>= 2)

Depends R (>= 4.3.0)

Imports checkmate (>= 2.3.1), dplyr (>= 1.1.4), parallel (>= 4.3.0), Rcpp, roll (>= 1.1.6), stats (>= 4.3.0), stringr (>= 1.5.1)

Suggests testthat (>= 3.2.1)

LinkingTo Rcpp, RcppArmadillo

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation Yes

RoxygenNote 7.3.1

URL https://github.com/lehmasve/hdflex

BugReports https://github.com/lehmasve/hdflex/issues

Repository https://lehmasve.r-universe.dev

RemoteUrl https://github.com/lehmasve/hdflex

RemoteRef HEAD

RemoteSha c5fdd4bd217b4fbb1486cd95a1933a661c85bff3

Contents

`benchmark_ar2` *AR(2) benchmark forecasts for quarterly U.S. inflation*

Description

Out-of-sample one-step-ahead AR(2) benchmark forecasts for the period from 1991-Q2 to 2021-Q4. The AR(2) models are estimated with OLS and intercept.

Usage

```
benchmark_ar2
```

Format

A matrix with 123 quarterly observations (rows) and 4 benchmarks (columns):

GDPCTPI OOS-AR2-benchmark forecast for quarterly GDP deflator (GDPCTPI).

PCECTPI OOS-AR2-benchmark forecast for quarterly PCE deflator (PCECTPI).

CPIAUCSL OOS-AR2-benchmark forecast for quarterly Total CPI (CPIAUCSL).

CPILFESL OOS-AR2-benchmark forecast for quarterly Core CPI (CPILFESL).

Source

<https://doi.org/10.1111/iere.12623>

References

Koop, G. and Korobilis, D. (2023) "Bayesian dynamic variable selection in high dimensions." *International Economic Review*.

dsc — *Generate dynamic subset forecast combinations*

Description

‘dsc()‘ can be used to generate forecast combinations from a set of candidate density forecasts. For each period, ‘dsc()‘ selects a subset of predictive densities with highest ranks regarding (local) predictive accuracy. Both the identities of the candidate forecasts that are used for building the combined forecast and the subset sizes may vary over time based on the data. If only one candidate forecast is picked, the approach (temporarily) collapses to pure model selection.

Usage

```
dsc(gamma_grid, psi_grid, y, mu_mat, var_mat, delta, n_cores)
```

Arguments

`gamma_grid`	A numerical vector that contains discount factors to exponentially down-weight the past predictive performance of the candidate forecasts.
`psi_grid`	An integer vector that controls the (possible) sizes of the active subsets.
`y`	A matrix of dimension ‘T * 1‘ or numeric vector of length ‘T‘ containing the observations of the target variable.
`mu_mat`	A matrix with ‘T‘ rows containing the first moment of each predictive density in each column.
`var_mat`	A matrix with ‘T‘ rows containing the second moment of each predictive density in each column.
`delta`	A numeric value denoting the discount factor used to down-weight the past predictive performance of the subset combinations.
`n_cores`	An integer that denotes the number of CPU-cores used for the computational estimation.

Value

A list that contains: * (1) a vector with the first moments (point forecasts) of the STSC-Model, * (2) a vector with the the second moments (variance) of the STSC-Model, * (3) a vector that contains the selected values for gamma, * (4) a vector that contains the selected values for psi and * (5) a matrix that indicates the selected signals for every point in time.

Author(s)

Philipp Adämmer, Sven Lehmann, Rainer Schüssler

References

Beckmann, J., Koop, G., Korobilis, D., and Schüssler, R. A. (2020) "Exchange rate predictability and dynamic bayesian learning." *Journal of Applied Econometrics*, 35 (4): 410–421.

Koop, G. and Korobilis, D. (2012) "Forecasting inflation using dynamic model averaging." *International Economic Review*, 53 (3): 867–886.

Koop, G. and Korobilis, D. (2023) "Bayesian dynamic variable selection in high dimensions." *International Economic Review*.

Raftery, A. E., Kárn‘y, M., and Ettler, P. (2010) "Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill." *Technometrics*, 52 (1): 52–66.

Del Negro, M., Hasegawa, R. B., and Schorfheide, F. (2016) "Dynamic prediction pools: An investigation of financial frictions and forecasting performance." *Journal of Econometrics*, 192 (2): 391–405.

West, M. and Harrison, J. (1997) "Bayesian forecasting and dynamic models" *Springer*, 2nd edn.

See Also

`https://github.com/lehmasve/hdflex#readme`

Examples

```
# See example for tvc().
```

`hdflex` *hdflex: High-Dimensional Density Forecasts*

Description

hdflex contains the forecasting algorithm STSC developed by Adämmer, Lehmann and Schüssler (2023) <doi:10.2139/ssrn.4342487>. STSC is a novel time series forecasting method designed to handle very large sets of predictive signals, many of which are irrelevant or have only short-lived predictive power. Please cite the paper when using the package.

Author(s)

Philipp Adämmer, Sven Lehmann, Rainer Schüssler

`inflation_data` *Dataset to estimate quarterly U.S. inflation*

Description

A novel, high-dimensional dataset built by Koop and Korobilis (2023) that merges predictive signals from several mainstream aggregate macroeconomic and financial datasets. The dataset includes the FRED-QD dataset of McCracken and Ng (2020), augment with portfolio data used in Jurado et al. (2015), stock market predictors from Welch and Goyal (2008), survey data from University of Michigan consumer surveys, commodity prices from the World Bank's Pink Sheet database, and key macroeconomic indicators from the Federal Reserve Economic Data for four economies (Canada, Germany, Japan, United Kingdom). The data is already pre-processed to perform one-step-ahead forecasts and augmented with (external) point forecasts from Koop & Korobilis (2023). The dataset spans the period 1960-Q3 to 2021-Q4.

Usage

```
inflation_data
```

Format

A matrix with 245 quarterly observations (rows) and 516 variables (columns).

Column 1:4 Transformed target variables: GDP deflator (GDPCTPI), PCE deflator (PCECTPI), Total CPI (CPIAUCSL), Core CPI (CPILFESL)

Column 5:8 First lag of the target variables

Column 9:12 Second lag of the target variables

Column 13:16 All four (lagged) price series transformed with second differences of logarithms

Column 17:452 All remaining (lagged and transformed) signals from the FRED-QD dataset of McCracken and Ng (2020), portfolio data used in Jurado et al. (2015), stock market predictors from Welch and Goyal (2008), survey data from University of Michigan consumer surveys, commodity prices from the World Bank's Pink Sheet database, and key macroeconomic indicators from the Federal Reserve Economic Data for Canada, Germany, Japan & United Kingdom.

Column 453:468 External point forecasts for quarterly GDP deflator (GDPCTPI) generated by the MatLab Code from Koop and Korobilis (2023). The forecasts were generated out-of-sample from 1976-Q1 to 2021-Q4.

Column 469:484 External point forecasts for quarterly PCE deflator (PCECTPI) generated by the MatLab Code from Koop and Korobilis (2023). The forecasts were generated out-of-sample from 1976-Q1 to 2021-Q4.

Column 485:500 External point forecasts for quarterly Total CPI (CPIAUCSL) generated by the MatLab Code from Koop and Korobilis (2023). The forecasts were generated out-of-sample from 1976-Q1 to 2021-Q4.

Column 501:516 External point forecasts for quarterly Core CPI (CPILFESL) generated by the MatLab Code from Koop and Korobilis (2023). The forecasts were generated out-of-sample from 1976-Q1 to 2021-Q4.

Source

<https://doi.org/10.1111/iere.12623>

References

Jurado, K., Ludvigson, S. C., and Ng, S. (2015) "Measuring uncertainty." *American Economic Review*, 105 (3): 1177–1216.

Koop, G. and Korobilis, D. (2023) "Bayesian dynamic variable selection in high dimensions." *International Economic Review*.

McCracken, M., and S. Ng (2020) “FRED-QD: A Quarterly Database for Macroeconomic Research” *National Bureau of Economic Research*, Working Paper 26872.

Welch, I. and Goyal, A. (2008) "A comprehensive look at the empirical performance of equity premium prediction." *The Review of Financial Studies*, 21 (4): 1455–1508.

`stsc` *Signal-Transform-Subset-Combination (STSC)*

Description

‘stsc()‘ is a time series forecasting method designed to handle vast sets of predictive signals, many of which are irrelevant or short-lived. The method transforms heterogeneous scalar-valued signals into candidate density forecasts via time-varying coefficient models (TV-C), and subsequently, combines them into a final density forecast via dynamic subset combination (DSC).

Usage

```
stsc(
  y,
  X,
  Ext_F,
  sample_length,
  lambda_grid,
  kappa_grid,
  burn_in_tvc,
  gamma_grid,
  psi_grid,
  delta,
  burn_in_dsc,
  method,
  equal_weight,
  risk_aversion = NULL,
  min_weight = NULL,
  max_weight = NULL
)
```

Arguments

`y`	A matrix of dimension ‘T * 1‘ or numeric vector of length ‘T‘ containing the observations of the target variable.
`X`	A matrix with ‘T‘ rows containing the lagged ’simple’ signals in each column. Use NULL if no ’simple’ signal shall be included.
`Ext_F`	A matrix with ‘T‘ rows containing point forecasts of y in each column. Use NULL if no point forecasts shall be included.
`sample_length`	An integer that denotes the number of observations used to initialize the observational variance and the coefficients’ variance in the TV-C models.
`lambda_grid`	A numeric vector with values between 0 and 1 denoting the discount factor(s) that control the dynamics of the time-varying coefficients. Each signal in combination with each value of lambda provides a separate candidate forecast. Constant coefficients are nested for the case ‘lambda = 1‘.
`kappa_grid`	A numeric vector between 0 and 1 to accommodate time-varying volatility in the TV-C models. The observational variance is estimated via Exponentially Weighted Moving Average. Constant variance is nested for the case ‘kappa = 1‘. Each signal in combination with each value of kappa provides a separate forecast.
`burn_in_tvc`	An integer value ‘>= 1‘ that denotes the number of observations used to ’initialize’ the TV-C models. After ’burn_in_tvc’ observations the generated sum of discounted predictive log-likelihoods (DPLLs) of each Candidate Model (TV-C model) and Subset Combination (combination of gamma and psi) is resetted. ‘burn_in_tvc = 1‘ means no burn-in period is applied.
`gamma_grid`	A numerical vector that contains discount factors between 0 and 1 to exponentially down-weight the past predictive performance of the candidate forecasts.
`psi_grid`	An integer vector that controls the (possible) sizes of the active subsets.
`delta`	A numeric value between 0 and 1 denoting the discount factor used to down-weight the past predictive performance of the subset combinations.
`burn_in_dsc`	An integer value ‘>= 1‘ that denotes the number of observations used to ’initialize’ the Dynamic Subset Combinations. After ’burn_in_dsc’ observations the generated sum of discounted predictive log-likelihoods (DPLLs) of each Subset Combination (combination of gamma and psi) is resetted. ‘burn_in_dsc = 1‘ means no burn-in period is applied.
`method`	An integer of the set ‘1, 2, 3, 4‘ that denotes the method used to rank the Candidate Models (TV-C models) and Subset Combinations according to their performance. Default is ‘method = 1‘ which ranks according to their generated sum of discounted predictive log-likelihoods (DPLLs). ‘method = 2‘ uses Squared-Error (SE) instead of DPLLs. ‘method = 3‘ uses Absolute-Error (AE) and ‘method = 4‘ uses Compounded-Returns (in this case the target variable y has to be a time series of financial returns).
`equal_weight`	A boolean that denotes whether equal weights are used to combine the candidate forecasts within a subset. If ‘FALSE‘, the weights are calculated using the softmax-function on the predictive log-scores of the candidate models. The method proposed in Adaemmer et al (2023) uses equal weights to combine the candidate forecasts.

`risk_aversion`	A double '>= 0' that denotes the risk aversion of an investor. A higher value indicates a risk avoiding behaviour.
`min_weight`	A double that denotes the lower bound for the weight placed on the market. A non-negative value rules out short sales.
`max_weight`	A double that denotes the upper bound for the weight placed on the market. A value of e.g. 2 allows for a maximum leverage ratio of two.

Value

A list that contains: * (1) a vector with the first moments (point forecasts) of the STSC-Model, * (2) a vector with the second moments (variance) of the STSC-Model, * (3) a vector that contains the selected values for gamma, * (4) a vector that contains the selected values for psi and * (5) a matrix that indicates the selected signals for every point in time.

Author(s)

Philipp Adämmer, Sven Lehmann, Rainer Schüssler

References

Beckmann, J., Koop, G., Korobilis, D., and Schüssler, R. A. (2020) "Exchange rate predictability and dynamic bayesian learning." *Journal of Applied Econometrics*, 35 (4): 410–421.

Dangl, T. and Halling, M. (2012) "Predictive regressions with time-varying coefficients." *Journal of Financial Economics*, 106 (1): 157–181.

Del Negro, M., Hasegawa, R. B., and Schorfheide, F. (2016) "Dynamic prediction pools: An investigation of financial frictions and forecasting performance." *Journal of Econometrics*, 192 (2): 391–405.

Koop, G. and Korobilis, D. (2012) "Forecasting inflation using dynamic model averaging." *International Economic Review*, 53 (3): 867–886.

Koop, G. and Korobilis, D. (2023) "Bayesian dynamic variable selection in high dimensions." *International Economic Review*.

Raftery, A. E., Kárn'y, M., and Ettler, P. (2010) "Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill." *Technometrics*, 52 (1): 52–66.

West, M. and Harrison, J. (1997) "Bayesian forecasting and dynamic models" *Springer*, 2nd edn.

See Also

https://github.com/lehmasve/hdflex#readme

Examples

```
#########################################################
######### Forecasting quarterly U.S. inflation ##########
#### Please see Koop & Korobilis (2023) for further  ####
#### details regarding the data & external forecasts ####
#########################################################
```

```
# Packages
library("hdflex")

########## Get Data ##########
# Load Data
inflation_data   <-  inflation_data
benchmark_ar2    <-  benchmark_ar2

# Set Index for Target Variable
i  <-  1   # (1 -> GDPCTPI; 2 -> PCECTPI; 3 -> CPIAUCSL; 4 -> CPILFESL)

# Subset Data (keep only data relevant for target variable i)
dataset  <-  inflation_data[, c(1+(i-1),                          # Target Variable
                                5+(i-1),                          # Lag 1
                                9+(i-1),                          # Lag 2
                             (13:16)[-i],                    # Remaining Price Series
                          17:452,                      # Exogenous Predictor Variables
                             seq(453+(i-1)*16,468+(i-1)*16))]  # Ext. Point Forecasts

########## STSC ##########
# Set Target Variable
y  <-  dataset[,  1, drop = FALSE]

# Set 'Simple' Signals
X  <-  dataset[, 2:442, drop = FALSE]

# Set External Point Forecasts (Koop & Korobilis 2023)
Ext_F  <-  dataset[, 443:458, drop = FALSE]

# Set Dates
dates  <-  rownames(dataset)

# Set TV-C-Parameter
sample_length  <-  4 * 5
lambda_grid    <-  c(0.90, 0.95, 1)
kappa_grid     <-  0.98

# Set DSC-Parameter
gamma_grid  <-  c(0.40, 0.50, 0.60, 0.70, 0.80, 0.90,
                  0.91, 0.92, 0.93, 0.94, 0.95, 0.96,
                  0.97, 0.98, 0.99, 1.00)
psi_grid    <-  c(1:100)
delta       <-  0.95

# Apply STSC-Function
results  <-  hdflex::stsc(y,
                          X,
                          Ext_F,
                          sample_length,
                          lambda_grid,
                          kappa_grid,
                          burn_in_tvc = 79,
                          gamma_grid,
                          psi_grid,
                          delta,
                          burn_in_dsc = 1,
                          method = 1,
                          equal_weight = TRUE,
                          risk_aversion = NULL,
                          min_weight = NULL,
                          max_weight = NULL)

# Assign DSC-Results
forecast_stsc    <-  results[[1]]
variance_stsc    <-  results[[2]]
chosen_gamma     <-  results[[3]]
chosen_psi       <-  results[[4]]
chosen_signals   <-  results[[5]]

# Define Evaluation Period
eval_date_start      <-  "1991-01-01"
eval_date_end        <-  "2021-12-31"
eval_period_idx      <-  which(dates > eval_date_start & dates <= eval_date_end)

# Trim Objects
oos_y                <-  y[eval_period_idx, ]
oos_forecast_stsc    <-  forecast_stsc[eval_period_idx]
oos_variance_stsc    <-  variance_stsc[eval_period_idx]
oos_chosen_gamma     <-  chosen_gamma[eval_period_idx]
oos_chosen_psi       <-  chosen_psi[eval_period_idx]
oos_chosen_signals   <-  chosen_signals[eval_period_idx, , drop = FALSE]
oos_dates            <-  dates[eval_period_idx]

# Add Dates
names(oos_forecast_stsc)     <-  oos_dates
names(oos_variance_stsc)     <-  oos_dates
names(oos_chosen_gamma)      <-  oos_dates
names(oos_chosen_psi)        <-  oos_dates
rownames(oos_chosen_signals) <-  oos_dates

### Part 2: Evaluation ###
# Apply Summary-Function
summary_results  <-  summary_stsc(oos_y,
                                  benchmark_ar2[, i],
                                  oos_forecast_stsc)

# Assign Summary-Results
cssed  <-  summary_results[[3]]
mse    <-  summary_results[[4]]

########## Results ##########
# Relative MSE
print(paste("Relative MSE:", round(mse[[1]] / mse[[2]], 4)))

# Plot CSSED
plot(x    = as.Date(oos_dates),
     y    = cssed,
     ylim = c(-0.0008, 0.0008),
     main = "Cumulated squared error differences",
     type = "l",
     lwd  = 1.5,
     xlab = "Date",
     ylab = "CSSED") + abline(h = 0, lty = 2, col = "darkgray")

# Plot Predictive Signals
vec  <-  seq_len(dim(oos_chosen_signals)[2])
mat  <-  oos_chosen_signals %*% diag(vec)
mat[mat == 0]  <- NA
matplot(x    = as.Date(oos_dates),
        y    = mat,
        cex  = 0.4,
        pch  = 20,
        type = "p",
        main = "Evolution of selected signal(s)",
        xlab = "Date",
        ylab = "Predictive Signal")

# Plot Psi
plot(x    = as.Date(oos_dates),
     y    = oos_chosen_psi,
     ylim = c(1, 100),
     main = "Evolution of the subset size",
     type = "p",
     cex  = 0.75,
     pch  = 20,
     xlab = "Date",
     ylab = "Psi")
```

`summary_stsc` *Statistical summary of the STSC-results*

Description

'summary_stsc()' returns a statistical summary of the results from dsc(). It provides statistical measures such as Clark-West-Statistic, OOS-R2, Mean-Squared-Error and Cumulated Sum of Squared-Error-Differences.

Usage

```
summary_stsc(oos_y, oos_benchmark, oos_forecast_stsc)
```

Arguments

`oos_y`	A matrix of dimension 'T * 1' or numeric vector of length 'T' containing the out-of-sample observations of the target variable.

`oos_benchmark`	A matrix of dimension ‘T * 1‘ or numeric vector of length ‘T‘ containing the out-of-sample forecasts of an arbitrary benchmark (i.e. prevailing historical mean).
`oos_forecast_stsc`	A matrix of dimension ‘T * 1‘ or numeric vector of length ‘T‘ containing the out-of-sample forecasts of dsc().

Value

List that contains: * (1) the Clark-West-Statistic, * (2) the Out-of-Sample R2, * (3) a vector with the CSSED between the STSC-Forecast and the benchmark and * (4) a list with the MSE of the STSC-Model and the benchmark.

Author(s)

Philipp Adämmer, Sven Lehmann, Rainer Schüssler

References

Clark, T. E. and West, K. D. (2007) "Approximately normal tests for equal predictive accuracy in nested models." *Journal of Econometrics*, 138 (1): 291–311.

See Also

https://github.com/lehmasve/hdflex#readme

Examples

```
# See example for tvc().
```

`tvc` *Compute density forecasts based on univariate time-varying coefficient (TV-C) models in state-space form*

Description

‘tvc()‘ can be used to generate density forecasts based on univariate time-varying coefficient models. In each forecasting model, we include an intercept and one predictive signal. The predictive signal either represents the value of a ’simple’ signal or the the value of an external point forecast. All models are estimated independently from each other and estimation and forecasting are carried out recursively.

Usage

```
tvc(y, X, Ext_F, lambda_grid, kappa_grid, init_length, n_cores)
```

Arguments

`y`	A matrix of dimension ‘T * 1‘ or numeric vector of length ‘T‘ containing the observations of the target variable.
`X`	A matrix with ‘T‘ rows containing the lagged ’simple’ signals in each column. Use NULL if no ’simple’ signal shall be included.
`Ext_F`	A matrix with ‘T‘ rows containing point forecasts of y in each column. Use NULL if no point forecasts shall be included.
`lambda_grid`	A numeric vector denoting the discount factor(s) that control the dynamics of the coefficients. Each signal in combination with each value of lambda provides a separate candidate forecast. Constant coefficients are nested for the case ‘lambda = 1‘.
`kappa_grid`	A numeric vector to accommodate time-varying volatility. The observational variance is estimated via Exponentially Weighted Moving Average. Constant variance is nested for the case ‘kappa = 1‘. Each signal in combination with each value of kappa provides a separate forecast.
`init_length`	An integer that denotes the number of observations used to initialize the observational variance and the coefficients’ variance.
`n_cores`	An integer that denotes the number of CPU-cores used for the computation.

Value

A list that contains:

* (1) a matrix with the first moments (point forecasts) of the conditionally normal predictive distributions and

* (2) a matrix with the second moments (variance) of the conditionally normal predictive distributions.

Author(s)

Philipp Adämmer, Sven Lehmann, Rainer Schüssler

References

Beckmann, J., Koop, G., Korobilis, D., and Schüssler, R. A. (2020) "Exchange rate predictability and dynamic bayesian learning." *Journal of Applied Econometrics*, 35 (4): 410–421.

Dangl, T. and Halling, M. (2012) "Predictive regressions with time-varying coefficients." *Journal of Financial Economics*, 106 (1): 157–181.

Koop, G. and Korobilis, D. (2012) "Forecasting inflation using dynamic model averaging." *International Economic Review*, 53 (3): 867–886.

Koop, G. and Korobilis, D. (2023) "Bayesian dynamic variable selection in high dimensions." *International Economic Review*.

Raftery, A. E., Kárn‘y, M., and Ettler, P. (2010) "Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill." *Technometrics*, 52 (1): 52–66.

West, M. and Harrison, J. (1997) "Bayesian forecasting and dynamic models" *Springer*, 2nd edn.

See Also

https://github.com/lehmasve/hdflex#readme

Examples

```
#########################################################
######### Forecasting quarterly U.S. inflation ##########
#### Please see Koop & Korobilis (2023) for further  ####
#### details regarding the data & external forecasts ####
#########################################################

# Packages
library("hdflex")

########## Get Data ##########
# Load Data
inflation_data   <-  inflation_data
benchmark_ar2    <-  benchmark_ar2

# Set Index for Target Variable
i  <-  1   # (1 -> GDPCTPI; 2 -> PCECTPI; 3 -> CPIAUCSL; 4 -> CPILFESL)

# Subset Data (keep only data relevant for target variable i)
dataset  <-  inflation_data[, c(1+(i-1),                          # Target Variable
                                5+(i-1),                          # Lag 1
                                9+(i-1),                          # Lag 2
                             (13:16)[-i],                     # Remaining Price Series
                         17:452,                          # Exogenous Predictor Variables
                             seq(453+(i-1)*16,468+(i-1)*16))]  # Ext. Point Forecasts

########## STSC ##########
### Part 1: TV-C Model ###
# Set Target Variable
y  <-  dataset[,  1, drop = FALSE]

# Set 'Simple' Signals
X  <-  dataset[, 2:442, drop = FALSE]

# Set External Point Forecasts (Koop & Korobilis 2023)
Ext_F  <-  dataset[, 443:458, drop = FALSE]

# Set TV-C-Parameter
sample_length  <-  4 * 5
lambda_grid    <-  c(0.90, 0.95, 1)
kappa_grid     <-  0.98
n_cores        <-  1

# Apply TV-C-Function
results  <-  hdflex::tvc(y,
                         X,
                         Ext_F,
                         lambda_grid,
                         kappa_grid,
                         sample_length,
                         n_cores)

# Assign TV-C-Results
forecast_tvc      <-  results[[1]]
variance_tvc      <-  results[[2]]

# Define Burn-In Period
sample_period_idx  <-  80:nrow(dataset)
sub_forecast_tvc   <-  forecast_tvc[sample_period_idx, , drop = FALSE]
sub_variance_tvc   <-  variance_tvc[sample_period_idx, , drop = FALSE]
sub_y              <-  y[sample_period_idx, , drop = FALSE]
sub_dates          <-  rownames(dataset)[sample_period_idx]

### Part 2: Dynamic Subset Combination ###
# Set DSC-Parameter
nr_mods     <-  ncol(sub_forecast_tvc)
gamma_grid  <-  c(0.40, 0.50, 0.60, 0.70, 0.80, 0.90,
                  0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.00)
psi_grid    <-  c(1:100)
delta       <-  0.95
n_cores     <-  1

# Apply DSC-Function
results  <-  hdflex::dsc(gamma_grid,
                         psi_grid,
                         sub_y,
                         sub_forecast_tvc,
                         sub_variance_tvc,
                         delta,
                         n_cores)

# Assign DSC-Results
sub_forecast_stsc    <-  results[[1]]
sub_variance_stsc    <-  results[[2]]
sub_chosen_gamma     <-  results[[3]]
sub_chosen_psi       <-  results[[4]]
sub_chosen_signals   <-  results[[5]]

# Define Evaluation Period
eval_date_start      <-  "1991-01-01"
eval_date_end        <-  "2021-12-31"
eval_period_idx     <-  which(sub_dates > eval_date_start & sub_dates <= eval_date_end)

# Trim Objects
oos_y                <-  sub_y[eval_period_idx, ]
oos_forecast_stsc    <-  sub_forecast_stsc[eval_period_idx]
oos_variance_stsc    <-  sub_variance_stsc[eval_period_idx]
oos_chosen_gamma     <-  sub_chosen_gamma[eval_period_idx]
oos_chosen_psi       <-  sub_chosen_psi[eval_period_idx]
oos_chosen_signals   <-  sub_chosen_signals[eval_period_idx, , drop = FALSE]
```

```
oos_dates            <-  sub_dates[eval_period_idx]

# Add Dates
names(oos_forecast_stsc)     <-  oos_dates
names(oos_variance_stsc)     <-  oos_dates
names(oos_chosen_gamma)      <-  oos_dates
names(oos_chosen_psi)        <-  oos_dates
rownames(oos_chosen_signals) <-  oos_dates

### Part 3: Evaluation ###
# Apply Summary-Function
summary_results  <-  summary_stsc(oos_y,
                                  benchmark_ar2[, i],
                                  oos_forecast_stsc)
# Assign Summary-Results
cssed  <-  summary_results[[3]]
mse    <-  summary_results[[4]]

########## Results ##########
# Relative MSE
print(paste("Relative MSE:", round(mse[[1]] / mse[[2]], 4)))

# Plot CSSED
plot(x    = as.Date(oos_dates),
     y    = cssed,
     ylim = c(-0.0008, 0.0008),
     main = "Cumulated squared error differences",
     type = "l",
     lwd  = 1.5,
     xlab = "Date",
     ylab = "CSSED") + abline(h = 0, lty = 2, col = "darkgray")

# Plot Predictive Signals
vec  <-  seq_len(dim(oos_chosen_signals)[2])
mat  <-  oos_chosen_signals %*% diag(vec)
mat[mat == 0]  <- NA
matplot(x    = as.Date(oos_dates),
        y    = mat,
        cex  = 0.4,
        pch  = 20,
        type = "p",
        main = "Evolution of selected signal(s)",
        xlab = "Date",
        ylab = "Predictive Signal")

# Plot Psi
plot(x    = as.Date(oos_dates),
     y    = oos_chosen_psi,
     ylim = c(1, 100),
     main = "Evolution of the subset size",
     type = "p",
     cex  = 0.75,
     pch  = 20,
           xlab = "Date",
           ylab = "Psi")
```

Index