

# User manual for R functions used in Post-Shock Persistence Testing

November 15, 2021

This document describes the functions that are used in the post-shock persistence testing.

## Contents

1	<code>scm</code> function	1
2	<code>sel</code> function	2
3	<code>taSPA.mhfc</code> function	3
4	<code>vote</code> function	4
5	<code>ols.est.alphahat</code> function	4
6	<code>ps.indic.W.permanent</code>	6
7	<code>ps.indic.W.complicate</code>	9
8	<code>ps.indic.W.decay</code>	9
9	<code>ps.indic.W.decay.nls</code>	10
	References	10

# 1 scm function

`scm(X, Tstar)` is a core function in post-shock prediction methodology that is used to compute the synthetic weights  $\mathbf{W}^*$  in [Lin and Eck \(2021\)](#). It computes the weights to minimize the Euclidean distance between the convex combination of the covariates in the donor pool and the covariates of the time series of interest. The optimization is based on nonlinear optimization using the `solnp` in the R package `Rsolnp` ([Ghalanos, Theussl, & Ghalanos, 2012](#)).

Suppose the donor pool size is  $n$ . `scm(X, Tstar)` takes  $X$  and  $Tstar$  as inputs and outputs a vector of weights  $\mathbf{W}^*$  and other convergence details related to `solnp`, where

- $X$  is a list of covariates with length  $n + 1$ , where the first element of the list should be the covariates of the time series of interest and the remaining elements of the list are the covariates of the donors. The covariates should be in the class of `matrix` with size  $T_i \times p$  for  $i = 1, \dots, n + 1$ .
- $Tstar$  is a vector of shock-effect time points when the shock occurs at  $T_i^* + 1$  for  $i = 1, \dots, n + 1$ . The size of  $Tstar$  should be  $n + 1$ .

*Examples:*

```
p <- 2; n <- 10;
T <- round(rgamma(n = n + 1, shape = 15, scale = 10)) # Time Length
Tstar <- c() # Shock Time Points
for (t in T) {
  Tstar <- c(Tstar, sample((p + 3 + 1):(t - 1), size = 1))
}
phi <- round(runif(n + 1, 0, 1), 3) # autoregressive parameters
# construction of design matrix
X <- c()
for (i in 1:(n + 1)) {
  Ti <- T[i]
  Tstari <- Tstar[i]
  X[[i]] <- matrix(rgamma(n = p * (Ti + 1), shape = 1, scale = 1),
                    ncol = p, byrow = T)
}
scm(X, Tstar)
```

# 2 sel function

`sel(yhat, y)` function is a function that computes the squared error loss between  $\hat{y}$  and  $y$ , where

- $yhat$  is either a vector or a scalar

- $y$  is either a vector or a scalar

$y_{\text{hat}}$  and  $y$  must have equal length. The output of `sel(yhat, y)` is the squared error loss. It may be a vector or scalar.

*Examples:*

```
sel(rnorm(10), rep(0,10))
```

### 3 taSPA.mhfc function

`taSPA.mhfc(ell, d, B, bw)` is a core function that implements the hypothesis test of [Quaedvlieg \(2021\)](#) that compares two forecasts in terms of average superior predictive ability (aSPA) for multiple horizons. See the theoretical and algorithmic details of the hypotheses in [Quaedvlieg \(2021\)](#). Let  $H$  be the maximum horizon number for the forecasts and  $T$  be the sample size. The test statistic is

$$t_{aSPA,ij} = \frac{\sqrt{T}\bar{d}_{ij}}{\hat{\xi}_{ij}},$$

where  $\bar{d}_{ij,t}$  ( $H \times 1$ ) is the loss differential between forecasts  $i$  and  $j$  at time  $t$ ,  $\bar{d}_{ij}$  is the average across  $t = 1, \dots, T$ ,  $\bar{d}_{ij} = \mathbf{w}'\bar{\mathbf{d}}_{ij}$ ,  $\xi_{ij} = \sqrt{\mathbf{w}'\mathbf{\Omega}_{ij}\mathbf{w}}$  is estimated by HAC estimator,  $\mathbf{\Omega}_{ij}$  is the asymptotic variance of  $\bar{\mathbf{d}}_{ij}$ .  $\mathbf{w} = \mathbf{1}_H/H$  is taken as default. The computation of HAC estimator requires specification of a bandwidth parameter `bw`. Provided with a loss differential matrix `d`, this test uses moving-block bootstrap to approximate the distribution of the test statistic. Note that

- `ell` is the block length of the moving block bootstrap. Theory of [Quaedvlieg \(2021\)](#) requires `ell` =  $o(T^{1/2})$ .
- `d` is the loss differential matrix of size  $T \times H$ . It must be of `matrix` class. The element in  $t$ th row and  $h$ th column should be the loss differential between two  $h$ -step forecasts for  $t$ th observation.
- `B` is the bootstrap sample size.
- `bw` is the bandwidth of computation for HAC estimator. HAC estimation is based on HAC from [Schlittgen \(2021\)](#) using quadratic spectral kernel.

The output of `taSPA.mhfc(ell, d, B, bw)` is the  $p$ -value of the test, which is a scalar. Importantly, `taSPA.mhfc(ell, d, B, bw)` tests a *one-sided* hypothesis. Thus it is necessary to check whether the way about how the loss differential `d` is computed matches the hypothesis that is tested. See more details of the hypothesis that is tested in [Quaedvlieg \(2021\)](#).

*Examples:*

```

y <- arima.sim(list(2,0,0), n = 100)

T <- 50; H <- 10; K <- 40

m1.L.i <- matrix(NA, nrow = T, ncol = H)
m2.L.i <- matrix(NA, nrow = T, ncol = H)
for (h in 1:H) {
  for (t in 1:T) {
    y.t <- y[(t + H - h + 1):(t + K + H - h)]

    m1 <- arima(y.t, order = c(1, 0, 0))
    m2 <- arima(y.t, order = c(2, 0, 0))

    y.t.hat.1 <- predict(m1, h)$pred[h]
    y.t.hat.2 <- predict(m1, h)$pred[h]

    m1.L.i[t, h] <- sel(y = y[t + K + H], yhat = y.t.hat.1)
    m2.L.i[t, h] <- sel(y = y[t + K + H], yhat = y.t.hat.2)
  }
}
d <- m2.L.i - m1.L.i

taSPA.mhfc(ell = 2, d = d, B = 200, bw = 4)

```

## 4 vote function

`vote(ps, sig)` is a function that yields the significance voting from a vector of  $p$ -values given a significance level. To be specific, if over 50% of the tests, each of which corresponds to a  $p$ -value in the donor pool, are significant according to the given significance level, the output of `vote(ps, sig)` would be 1. It is 0 otherwise. Note that

- `ps` is a vector of  $p$ -values with values in  $[0, 1]$ .
- `sig` is the significance level. It is 0.05 in default. It should be a scalar.

*Examples:*

```
vote(runif(10), .05)
```

## 5 ols.est.alphahat function

`ols.est.alphahat(Tstar, Y, X)` is the core function in [Lin and Eck \(2021\)](#) that is used to compute the weighted adjusted shock effect estimate  $\hat{\alpha}_{\text{wadj}}$  that aggregates information

in the donor pool. Suppose the donor pool size is  $n$ . Given the shock time points, it performs  $n$  OLS regressions across the donor pool to estimate the shock effect of each donor. The regression is based on the **AR**(1) model with covariates as specified in Section 2.1 of [Lin and Eck \(2021\)](#). The function internally uses `scm` function to compute the weights  $\mathbf{W}^*$  and weight the donors' shock effects to compute  $\hat{\alpha}_{\text{wadj}}$ . It also computes the  $\hat{\alpha}_{\text{IVW}}$  estimate and  $\hat{\alpha}_{\text{adj}}$  estimate. See more details of these two estimates in [Lin and Eck \(2021\)](#). Note that

- $\mathbf{X}$  is a list of covariates with length  $n + 1$ , where the first element of the list should be the covariates of the time series of interest and the remaining elements of the list are the covariates of the donors. The covariates should be in the class of `matrix` with size  $T_i \times p$  for  $i = 1, \dots, n + 1$ .
- $\mathbf{Tstar}$  is a vector of shock-effect time points when the shock occurs at  $T_i^* + 1$  for  $i = 1, \dots, n + 1$ . The size of  $\mathbf{Tstar}$  should be  $n + 1$ .
- $\mathbf{Y}$  is a list of responses with length  $n + 1$ , where the first element of the list should be the response of the time series of interest and the remaining elements of the list are the responses of the donors. It should be a vector.

This function outputs

- `alphahat` is a vector of  $n$  length and contains the OLS shock effect estimates for the donors.
- `est` is a vector of three shock effect estimates,  $\hat{\alpha}_{\text{adj}}$ ,  $\hat{\alpha}_{\text{wadj}}$ , and  $\hat{\alpha}_{\text{IVW}}$ .
- `Tstar` is the same as the one in the input.
- `X` is the same as the one in the input.
- `Y` is the same as the one in the input.
- `lmod` is a list of  $n$  `lm` objects that were fitted by OLS for each donor. The ordering is the same as how the donor is ordered in `X`, `Y`, and `Tstar`.
- `res` is a list of  $n$  residual objects corresponding to each element of `lmod`.
- `Wstar` is the synthetic weights, as defined in the output of `scm`.
- `se` is a vector of length  $n$  and each element is the OLS standard error for the OLS shock effect estimate for each donor.

*Examples:*

```
n <- 10; p <- 2
Ts <- ceiling(rgamma(n + 1, scale = 10, shape = 10)) # Time Length
Tstar <- c() # Shock Time Points
for (t in T) {
  Tstar <- c(Tstar, sample((p + 3 + 1):(t - 1), size = 1))
}
```

```

phi <- round(runif(n + 1, 0, 1), 3) # autoregressive parameters

X <- c()
alpha <- c()
# construction of design matrix and shock effects
gamma <- c()
for (i in 1:(n + 1)) {
  Ti <- Ts[i]
  Tstari <- Tstar[i]
  Ki <- K[i]
  X[[i]] <- matrix(rgamma(n = p * (Ti + Ki + H + 1),
                        shape = 1, scale = 1), ncol = p, byrow = TRUE)
  gamma[[i]] <- matrix(rnorm(p, mean = 1, sd = 1), nrow = 1)
  epsilonildei <- rnorm(n = 1, sd = 1)
  alpha <- c(alpha, 1 + gamma[[i]] %*% X[[i]][Tstari + 1, ] +
             epsilonildei)
}

# generation of yit
Y <- c()
for (i in 1:(n + 1)) {
  yi0 <- rnorm(1)
  Tstari <- Tstar[i]
  alphai <- alpha[i]
  phii <- phi[i]
  xi <- X[[i]]
  yi <- yi0
  thetai <- matrix(rnorm(p), nrow = 1)
  etai <- rnorm(1)
  for (t in 2:(Ts[i] + 1)) {
    epsilonit <- rnorm(n = 1, sd = 1)
    yi <- c(yi, etai + alphai * ifelse(t >= Tstari + 2,
                                       yes = 1, no = 0) +
           phii * yi[t - 1] + thetai %*% xi[t, ] + epsilonit)
  }
  Y[[i]] <- yi
}
ols.est.alphahat(Tstar, Y, X)

```

## 6 ps.indic.W.permanent

`ps.indic.W.permanent(Tstar, Y, X, K, H, Ts, ell, B, bw, sig.level)` is a core function that produces the synthetic weights from `scm`, a sequence of  $p$ -values and correspond-

ing rejection/acceptance decisions for each donor in the donor pool based on a give significance level and the test function `taSPA.mhfc`. The assumed model for this function is

$$y_{i,t} = \eta_i + \phi_i y_{i,t-1} + x_{i,t} \theta + \alpha_i I(t > T_i^* + 1) + \varepsilon_{i,t},$$

$$\alpha_i = \mu_\alpha + x_{i,T_i^*+1} + \varepsilon_{\alpha,i},$$

where  $T_i^* + 1$  is the time point when the shock occurs for  $i = 2, \dots, n + 1$ ,  $x_{i,t}$  is the covariate, and  $\eta_i$  is the intercept. Note that only  $\theta$ ,  $\eta_i$ , and  $\alpha_i$  are estimable. The above model is called permanent model due to the presence of a persistent shock term. For each donor, this function computes the adjusted forecast and unadjusted forecast, construct the corresponding loss differential matrix, and perform the forecast comparison test. Suppose the donor pool size is  $n$ . The input for this function is

- `Tstar` is a vector of shock-effect time points when the shock occurs at  $T_i^* + 1$  for  $i = 1, \dots, n + 1$ . The size of `Tstar` should be  $n + 1$ .
- `Y` is a list of responses with length  $n + 1$ , where the first element of the list should be the response of the time series of interest and the remaining elements of the list are the responses of the donors. It should be a vector.
- `X` is a list of covariates with length  $n + 1$ , where the first element of the list should be the covariates of the time series of interest and the remaining elements of the list are the covariates of the donors. The covariates should be in the class of `matrix` with size  $T_i \times p$  for  $i = 1, \dots, n + 1$ . Note that the sample size of each element in `X` should be  $T_i + K_i + H$ , where  $K_i$  is the training sample size and  $T_i$  is the length of the donor time series that are wished to be considered as the data used in forecast comparison.  $H$  is the maximum horizon.
- `K` is a vector of training sample sizes that are used to compute  $h$ -step forecast for each data point. `K` has a length of  $n + 1$ . The first element of `K` can be `NA` as forecast comparison is not available for the time series of interest.
- `H` is the maximum  $h$  that an  $h$ -step forecast that is wished to be considered in the multi-horizon forecast comparison testing procedure in [Quaedvlieg \(2021\)](#).
- `Ts` is a vector of length  $n + 1$ , and it specifies the length of the time series (ordered from the time series of interest to donors).
- `ell` is the block length of the moving block bootstrap. See details in `taSPA.mhfc`.
- `B` is the bootstrap sample size. See details in `taSPA.mhfc`.
- `bw` is the bandwidth of computation for HAC estimator. See details in `taSPA.mhfc`.
- `sig.level` is the significance level for the test. It is 0.05 in default.

The output of this function is

- $ps$  is a sequence of  $p$ -values from forecast comparison test of [Quaedvlieg \(2021\)](#) for the donors. It is of length  $n$ .
- $W$  is a vector of synthetic weights and is of length  $n$ .
- $Is$  is a sequence of rejection/acceptance decisions from forecast comparison for the donors, where 1 stands for rejection and 0 for acceptance. It is of length  $n$ .

*Examples:*

```
n <- 10; p <- 2
K <- ceiling(rgamma(n + 1, scale = 5, shape = 10)) # training sample size
Ts <- ceiling(rgamma(n + 1, scale = 10, shape = 10)) # Time Length
Tstar <- c()
for (i in 1:(n + 1)) {
  Tstar <- c(Tstar, max(Ts[i] + 1, ceiling(0.5 * (Ts[i] + K[i] + H))))
}
phi <- round(runif(n + 1, 0, 1), 3) # autoregressive parameters

X <- c()
alpha <- c()
# construction of design matrix and shock effects
gamma <- c()
for (i in 1:(n + 1)) {
  Ti <- Ts[i]
  Tstari <- Tstar[i]
  Ki <- K[i]
  X[[i]] <- matrix(rgamma(n = p * (Ti + Ki + H + 1),
                        shape = 1, scale = 1), ncol = p, byrow = TRUE)
  gamma[[i]] <- matrix(rnorm(p, mean = 1, sd = 1), nrow = 1)
  epsilonildei <- rnorm(n = 1, sd = 1)
  alpha <- c(alpha, 1 + gamma[[i]] %*% X[[i]][Tstari + 1, ] +
              epsilonildei)
}

# generation of yit
Y <- c()
for (i in 1:(n + 1)) {
  yi0 <- rnorm(1)
  Tstari <- Tstar[i]
  alphai <- alpha[i]
  phii <- phi[i]
  xi <- X[[i]]
  yi <- yi0
  thetai <- matrix(rnorm(p), nrow = 1)
  etai <- rnorm(1)
  for (t in 2:(K[i] + Ts[i] + H + 1)) {
```



```

epsilonit <- rnorm(n = 1, sd = 1)
yi <- c(yi, etai + alhai * ifelse(t >= Tstari + 2,
                                yes = 1, no = 0) +
        phii * yi[t - 1] + thetai %*% xi[t, ] + epsilonit)
}
Y[[i]] <- yi
}

est <- ps.indic.W.permanent(Tstar = Tstar, Y = Y, X = X, K = K,
                           H = 10, Ts = Ts, ell = 4, B = 200, bw = 4)

```

## 7 ps.indic.W.complicate

`ps.indic.W.complicate(Tstar, Y, X, K, H, Ts, ell, B, bw, sig.levl)` performs the same functionality as `ps.indic.W.permanent` but with a different model as

$$\begin{aligned}
y_{i,t} &= \left( \eta_i + \sum_{j=1}^{q_2} \phi_{i,j} y_{i,t-j} + \sum_{j=0}^{q_2-1} x_{i,t-j} \theta_{i,j+1} \right) (1 - D_{i,t}) + f(\mathcal{F}_{i,t} \alpha_i) D_{i,t} + \varepsilon_{i,t} \\
f(\mathcal{F}_{i,t}, \alpha_i) &= \alpha_i + \sum_{j=1}^{q_1} \tilde{\phi}_{i,j} y_{i,t-j} + \sum_{j=0}^{q_2-1} x_{i,t-j} \tilde{\theta}_{i,j+1} \\
\alpha_i &= \mu_\alpha + x_{i,T_i^*+1} + \varepsilon_{\alpha,i}
\end{aligned}$$

where  $D_{i,t} = I(t > T_i^* + 1)$ . Note that only  $\eta_i, \alpha_i - \eta_i, \phi_{i,j}, \theta_{i,j+1}, \tilde{\phi}_{i,j} - \phi_{i,j}, \tilde{\theta}_{i,j+1} - \theta_{i,j+1}$  are estimable. The role of the shock in this model is that the dynamics of  $y_{i,t}$  change after the shock.

Examples are similar to those for `ps.indic.W.permanent`.

## 8 ps.indic.W.decay

`ps.indic.W.decay(Tstar, Y, X, K, H, Ts, ell, B, bw, sig.levl)` performs the same functionality as `ps.indic.W.permanent` but with a different model as

$$\begin{aligned}
y_{i,t} &= \eta_i + \phi_i y_{i,t-1} + x_{i,t} \theta + \xi_i D_{i,t} + \varepsilon_{i,t}, \\
\xi_i &= \alpha_i \cdot e^{-(t-T_i^*-1)} \\
\alpha_i &= \mu_\alpha + x_{i,T_i^*+1} + \varepsilon_{\alpha,i},
\end{aligned}$$

where  $D_{i,t} = I(t > T_i^* + 1)$ . Note that only  $\eta_i, \phi_i, \alpha_i, \theta$  are estimable. This model captures the type of the shock that has exponential decay.

Examples are similar to those for `ps.indic.W.permanent`.

## 9 `ps.indic.W.decay.nls`

`ps.indic.W.decay.nls(Tstar, Y, X, K, H, Ts, ell, B, bw, sig.levl)` performs the same functionality as `ps.indic.W.permanent` but with a different model as

$$\begin{aligned}y_{i,t} &= \eta_i + \phi_i y_{i,t-1} + x_{i,t} \boldsymbol{\theta} + \xi_i D_{i,t} + \varepsilon_{i,t}, \\ \xi_i &= \alpha_i \cdot e^{-c(t-T_i^*-1)} \\ \alpha_i &= \mu_\alpha + x_{i,T_i^*+1} + \varepsilon_{\alpha,i},\end{aligned}$$

where  $D_{i,t} = I(t > T_i^* + 1)$ . Note that only  $\eta_i, \phi_i, \alpha_i, \boldsymbol{\theta}, c$  are estimable. This model is more general than `ps.indic.W.decay` with enabling estimability of the speed of the exponential decay,  $c$ . The estimation is based on `nls` function. Note that singularity problems may occur for this model when applied to complicate or large dataset.

Examples are similar to those for `ps.indic.W.permanent`.

## References

- Ghalanos, A., Theussl, S., & Ghalanos, M. A. (2012). Package ‘`rsolnp`’.
- Lin, J., & Eck, D. J. (2021). Minimizing post-shock forecasting error through aggregation of outside information. *International Journal of Forecasting*.
- Quaedvlieg, R. (2021). Multi-horizon forecast comparison. *Journal of Business & Economic Statistics*, 39(1), 40–53.
- Schlittgen, R. (2021). R package ‘`tsapp`’.