Supplementary material for “Likelihood-based inference in temporal hierarchies”

December 16, 2022

Abstract

We present a simulation study that validates the results presented in Section 3 and Appendix A of ller et al. (2022) and further illustrates the effect of small training sets. Everything is implemented as R-functions so that anyone can try the algorithms for their own particular choice of parameters.

1 Introduction

This document is meant as a supplement to ller et al. (2022), which will be referred to as “the article”. It compares the theoretical derivations given in Appendix A of the article and illustrate the effect of small sample estimation. This document will refer directly to equations in the article and, hence, should not be seen as a standalone document that can be read independently from the article. The idea is that anyone is free to download and modify the R-code for different choices of parameters and sizes of the training set (with proper reference to the work in ller et al. (2022)).

All functions are listed in Appendix A and can be sourced by

```r
files <- list.files("funs/")
for(i in 1:length(files)){source(paste0("funs/",files[i]))}
```

2 Review of one example

In this section, we will go through the calculation for one particular choice of parameters in the data-generating AR(2) process (eq. (1) of the article). A number of small and simple R-functions are used throughout this section. They can all be found in Appendix A.1.

First, parameters for the data-generating AR(2) process are chosen as

```
phi1 <- 1.25; phi2 <- -0.5
sigma <- 1
```

The half-year process is simulated by

$$y_t = \texttt{phi1} \cdot y_{t-1} + \texttt{phi2} \cdot y_{t-2} + \epsilon_t; \quad \epsilon_t \sim N(0, \texttt{sigma}^2). \tag{1}$$

and all half-year observations are collected in the vector $\boldsymbol{y} = [y_1, y_2, \dots, y_{2\mathrm{T}}]^T$.

Next, the base process is simulated with these parameters, the annual process is created, and all measurements are collected in the matrix $\boldsymbol{Y}$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}^{\mathrm{A}} & \boldsymbol{y}^{\mathrm{H}_1} & \boldsymbol{y}^{\mathrm{H}_2} \end{bmatrix}, \tag{2}$$

with $\boldsymbol{y}^{\mathrm{H}_1} = [y_1, y_3, ..., y_{2\mathrm{T}-1}]^T$, $\boldsymbol{y}^{\mathrm{H}_2} = [y_2, y_4, ..., y_{2\mathrm{T}}]^T$ and $\boldsymbol{y}^{\mathrm{A}} = \boldsymbol{y}^{\mathrm{H}_1} + \boldsymbol{y}^{\mathrm{H}_2} = [y_2^{\mathrm{A}}, y_4^{\mathrm{A}}, \dots, y_{2\mathrm{T}}^{\mathrm{A}}]^T$.

Initially, a large T is chosen in order to mimic the perfect information setup treated in the article.

```
## Simulate
set.seed(1234)
T <- 1e6
n.burnin <- 1e4
y <- arima.sim(n = 2 * T + n.burnin, list(ar = c(phi1, phi2)),
               sd = sigma)

## remove burnin
y <- y[-(1:n.burnin)]

## set up matrix
y1 <- y[seq(1,2 * T,by=2)]
y2 <- y[seq(2,2 * T,by=2)]
yA <- y1 + y2 ## anual process
Y.train <- cbind(yA, y1, y2)
```

Both the annual and the half-year process are modelled by AR(1) process, and the estimation of the two processes is based on $\boldsymbol{y}^{\mathrm{A}}$ and $\boldsymbol{y}$, respectively, as shown in the R-code below (`phiH.hat` and `phiA.hat` are estimates of the AR(1) parameters)

```
## Estimate parameters in AR(1) models
phiH.hat <- coef(arima(y, order=c(1, 0, 0), include.mean=FALSE))
phiA.hat <- coef(arima(yA, order=c(1, 0, 0), include.mean=FALSE))
```

In order to compare with the results presented in the article we need the auto-correlation up to lag three (eqs. (A.10)–(A.12) in the article).

```
## Empirical results
rho.hat <- acf(y, plot = FALSE)$acf[2:4]

## From article
rho.true <- rho.true.fun(phi1,phi2)

## Compare with estimated
cbind(rho.hat, rho.true)


##      rho.hat rho.true
## [1,]   0.833    0.833
## [2,]   0.542    0.542
## [3,]   0.261    0.260
```

as expected the difference is small. The auto-covariance (eq. (A.13) in the article), is now calculated and compared with the empirical result by

```
## Empirical result
gamma.hat <- c(1,rho.hat) * var(y)

## From article
gamma.true <- gamma.true.fun(phi1,phi2,sigma)
cbind(gamma.hat,gamma.true)


##      gamma.hat gamma.true
## [1,]      4.37       4.36
## [2,]      3.64       3.64
## [3,]      2.37       2.36
## [4,]      1.14       1.14
```

Again the difference is small. For the anual process (eq. (A.16) in the article) we have

```
rho1A.hat <- acf(yA,plot=FALSE)$acf[2]
rho1A.true <- (gamma.true[2] + 2 * gamma.true[3] + gamma.true[4]) /
    (2 * (gamma.true[1]+gamma.true[2]))
c(rho1A.hat,rho1A.true)


## [1] 0.594 0.594
```

The difference between the true and estimated correlation is again very small.

We now set up the forecast matrix ($\hat{\boldsymbol{Y}}$)

$$\hat{\boldsymbol{Y}} = \begin{bmatrix} \hat{\boldsymbol{y}}^{\mathrm{A}} & \hat{\boldsymbol{y}}^{\mathrm{H}_1} & \hat{\boldsymbol{y}}^{\mathrm{H}_2} \end{bmatrix}, \tag{3}$$

where $\hat{\boldsymbol{y}}^{\mathrm{A}} = [\hat{y}^{\mathrm{A}}_4, \hat{y}^{\mathrm{A}}_6, \dots, \hat{y}^{\mathrm{A}}_{2\mathrm{T}}]^T$, with

$$\hat{y}^{\mathrm{A}}_{2t} = \texttt{phiA.hat} \cdot y^{\mathrm{A}}_{2t-2}; \quad t = \{2, 6, \dots, T\}, \tag{4}$$

and $\hat{\boldsymbol{y}}^{\mathrm{H}_1} = [\hat{y}^{\mathrm{H}_1}_3, \hat{y}^{\mathrm{H}_1}_5, \dots, \hat{y}^{\mathrm{H}_1}_{2\mathrm{T}-1}]^T$, and $\hat{\boldsymbol{y}}^{\mathrm{H}_2} = [\hat{y}^{\mathrm{H}_2}_4, \hat{y}^{\mathrm{H}_2}_6, \dots, \hat{y}^{\mathrm{H}_2}_{2\mathrm{T}}]^T$, with

$$\begin{aligned} \hat{y}^{\mathrm{H}_1}_{2t-1} &= \texttt{phiH.hat} \cdot \hat{y}_{2t-2} = \texttt{phiH.hat} \cdot \hat{y}^{\mathrm{H}_2}_{2t-2}; \quad t = \{2, 4, \dots, T\} && (5) \\ \hat{y}^{\mathrm{H}_2}_{2t} &= \texttt{phiH.hat}^2 \cdot \hat{y}_{2t-2} = \texttt{phiH.hat}^2 \cdot \hat{y}^{\mathrm{H}_2}_{2t-2}; \quad t = \{2, 4, \dots, T\}, && (6) \end{aligned}$$

This implies that when setting up the forecast matrix, we condition on the first row of $\boldsymbol{Y}$, with $\boldsymbol{Y}_{1,:} = [y^{\mathrm{A}}_2, y^{\mathrm{H}_1}_1, y^{\mathrm{H}_2}_2] = [y^{\mathrm{A}}_2, y_1, y_2]$. Hence, the first row of $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ are disregarded in the R-code below

```r
## calculations based on simulated process values
Y.hat <- Y.train * NA
Y.hat[-1, ] <- cbind(phiA.hat * yA[-T], phiH.hat * y2[-T],
                     phiH.hat^2 * y2[-T])
Sigma.hat <- var(Y.train[-1, ]- Y.hat[-1, ])
```

The matrices in eq. (A.4) of the article ($\boldsymbol{\Phi}$, $\boldsymbol{\Phi}_\epsilon$ and $\boldsymbol{\Gamma}$) are set up as functions (see Appendix A.1). The variance–covariance of the base forecast error (eq. (A.6) in the article) is calculated and compared to the simulation answer above by

```r
## True value of matrices (in eq. (A.4))
Gamma <- Gamma.fun(rho.true.fun(phi1, phi2)[1], rho1A.true)
Phi <- Phi.fun(phi1, phi2)
Phi.epsilon <- Phi.epsilon.fun(phi1, phi2)

## eq (A.7)
VyH <- matrix(gamma.true[2], ncol = 2, nrow = 2)
diag(VyH) <- gamma.true[1]

## eq (A.6)
Sigma.true <- (Phi - Gamma) %*% VyH %*% t(Phi - Gamma) +
    sigma^2 * Phi.epsilon %*% t(Phi.epsilon)

## Compare
Sigma.hat - Sigma.true


##          yA      y1      y2
## yA 0.00929 0.00359 0.00500
## y1 0.00359 0.00163 0.00176
## y2 0.00500 0.00176 0.00273
```

Again we see a small difference indicating that the method calculates the correct variance–covariance matrix.

Here, we take a small detour and investigate the dimension of the observed errors. In the article we estimate the parameters in the model (it is implicit that we consider one row of the forecast matrix ($\hat{\boldsymbol{Y}}$))

$$\hat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{b} + \boldsymbol{\epsilon}, \tag{7}$$

where $\boldsymbol{\epsilon}$ follows a normal distribution with variance–covariance matrix $\boldsymbol{\Sigma}_\epsilon$. The optimal estimate of $\boldsymbol{b}$ is

$$\begin{aligned}\hat{\boldsymbol{b}} = \tilde{\boldsymbol{y}}_B =& (\boldsymbol{S}^T\boldsymbol{\Sigma}_\epsilon^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{\Sigma}_\epsilon^{-1}\hat{\boldsymbol{y}} \qquad (8)\\ =& \boldsymbol{P}\hat{\boldsymbol{y}}, \qquad (9)\end{aligned}$$

and we can write

$$\tilde{\boldsymbol{y}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{\Sigma}_\epsilon^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{\Sigma}_\epsilon^{-1}\hat{\boldsymbol{y}} = \boldsymbol{H}\hat{\boldsymbol{y}}. \tag{10}$$

The observed residuals that would serve as a candidate for estimating $\boldsymbol{\Sigma}_\epsilon$ would be

$$\boldsymbol{r} = \hat{\boldsymbol{y}} - \tilde{\boldsymbol{y}} = (\boldsymbol{I} - \boldsymbol{H})\hat{\boldsymbol{y}}. \tag{11}$$

The observed residuals are in $\mathbb{R}^n$, but as a consequence or Cohran's Theorem (see, e.g., Madsen and Thyregod, 2011, p. 51), the dimension of $\boldsymbol{r}$ is

$$\text{trace}(\boldsymbol{I} - \boldsymbol{H}) = n - m, \tag{12}$$

which in our case is 3-2=1 and, hence, $\boldsymbol{\Sigma}_\epsilon$ cannot be identified from the observed differences. This is exemplified below

```
## Projection matrix
S <- matrix(0,ncol=2, nrow=3)
S[1, ] <- 1
S[2:3,1:2] <- diag(2)
Pfull.true <- solve(t(S) %*% solve(Sigma.true) %*% S) %*%
    t(S) %*% solve(Sigma.true)
Pfull.hat <- solve(t(S) %*% solve(Sigma.hat) %*% S) %*%
    t(S) %*% solve(Sigma.hat)

## trace
sum(diag(diag(3)-S%*%Pfull.true))


## [1] 1


sum(diag(diag(3)-S%*%Pfull.hat))


## [1] 1
```

Another approach would be to use the observed difference between $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$, but as $\tilde{\boldsymbol{y}} = \boldsymbol{S}\tilde{\boldsymbol{y}}_B$ and $\boldsymbol{y} = \boldsymbol{S}\boldsymbol{y}_B$ the dimension of $\boldsymbol{y} - \tilde{\boldsymbol{y}}$ is (at most) m. Hence, the variance–covariance of the observed base forecast error $(\boldsymbol{y} - \hat{\boldsymbol{y}})$ is often used as a proxy for $\boldsymbol{\Sigma}_\epsilon$ (and that is also the main theme in the article). I.e. we use

$$\tilde{\boldsymbol{\Sigma}} = V[\boldsymbol{y} - \hat{\boldsymbol{y}}]. \tag{13}$$

as an etimate of $\boldsymbol{\Sigma}_\epsilon$ (of course we use the empirical version (13) in actual calculation on empirical data).

We now go back to the main track of this presentation and investigte eq. (A.8) of the article. The observed estimate of $\tilde{\boldsymbol{\Sigma}}$ is

```
H <- S %*% Pfull.hat
Y.tilde <- cbind(Y.hat %*% H[1,], Y.hat %*% H[2,], Y.hat %*% H[3,])

Sigma.tilde.sim <- var(Y.train[-1, ] - Y.tilde[-1, ])
```

Finally the variance in eq. (A.8) of the article is calculated, and compared to the empirical version, by

```
Sigma.tilde <- (Phi - S %*% Pfull.true %*% Gamma) %*% VyH %*%
      t(Phi - S %*% Pfull.true %*% Gamma) +
      sigma^2 * Phi.epsilon %*% t(Phi.epsilon)

Sigma.tilde - Sigma.tilde.sim


##           yA       y1       y2
## yA -0.00724 -0.00309 -0.00415
## y1 -0.00309 -0.00151 -0.00158
## y2 -0.00415 -0.00158 -0.00257
```

We see that the difference is small, which illustrates that the method indeed calculates the correct variance–covarince $\tilde{\boldsymbol{\Sigma}}$.

3 Comparison on an independent test set

For completeness, and because performance measures is usually calculated using independent data, we now compare eq. (A.8) of the article with the estimated variance using an independent data-set (i.e., errors are calculated using the parameters estimated from the training set).

We start by simulating the test set

```
###################################################
## Generating a test set
## Simulate
set.seed(1234)
T.test <- 1e6
n.burnin <- 1e4
y.test <- arima.sim(n = 2 * T.test + n.burnin,
                    list(ar = c(phi1, phi2)),
                    sd = sigma)

y.test <- y.test[-(1:n.burnin)]

## set up matrix
y1 <- y.test[seq(1,2 * T.test,by=2)]
y2 <- y.test[seq(2,2 * T.test,by=2)]
yA <- y1 + y2
Y.test <- cbind(yA, y1, y2)
```

do predictions using the estimated AR(1) models

```
Y.hat <- Y.test * NA
Y.hat[-1, ] <- cbind(phiA.hat * yA[-T.test], phiH.hat * y2[-T.test],
                     phiH.hat^2 * y2[-T.test])
```

and compare the observed variance with the result of eq. (A.6) in the article

```
Sigma.hat <- var(Y.test[-1, ]- Y.hat[-1, ])
Sigma.hat - Sigma.true

##         yA      y1      y2
## yA 0.00929 0.00359 0.00500
## y1 0.00359 0.00163 0.00176
## y2 0.00500 0.00176 0.00273
```

Finally, we compare with eq. (A.8) of the article

```
Y.tilde <- cbind(Y.hat %*% H[1, ], Y.hat %*% H[2, ], Y.hat %*% H[3, ])
Sigma.tilde.hat <- var(Y.tilde[-1, ] - Y.test[-1, ])
Sigma.tilde - Sigma.tilde.hat

##          yA       y1       y2
## yA -0.00724 -0.00309 -0.00415
## y1 -0.00309 -0.00151 -0.00158
## y2 -0.00415 -0.00158 -0.00257
```

Once again we see small differences between the methods presented in the article and the simlated results.

4 Putting it all together

The functions and calculations for the theoretical results presented above are put together in R functions that do the calculation for a specific choice or collection of ϕ_1 and ϕ_2. All R-function are in the library `funs` and in Appendix A.2. It should be noted that there are no checks for reasonable values in the R-functions, i.e., no warning is given if, e.g., a non-stationary AR(2) process is chosen.

For one specific choice of parameters the result is calculated by

```
Vars <- VarFunTrue(phi1=1.5, phi2=-0.6, sigma = 1)
rbind(names(Vars)[1:3],names(Vars)[4:6])


##      [,1]         [,2]          [,3]
## [1,] "Sigma"      "Sigma.tilde" "Sigma.tilde.re"
## [2,] "Sigma.perf" "Pfull"       "Pre"
```

The outputs are:

`Sigma`	Variance of the base forecast error
`Sigma.tilde`	Variance of the reconciled forecast error using the full base forecast error variance–covariance matrix
`Sigma.tilde.re`	Variance of the reconciled forecast error ignoring cross-covariances between levels in base forecast error variance–covariance matrix
`Sigma.tilde.perf`	Variance of the perfect forecast, i.e., using the true process parameters
`Pfull`	Matrix $\boldsymbol{P}$ (see eq. (9)) with the full base forecast error variance–covariance matrix.
`Pre`	Matrix $\boldsymbol{P}$ (see eq. (9)) ignoring cross-covariances between levels in base forecast error variance–covariance matrix.

In order to create Figure 1 in the article, a function that takes vectors of ϕ_1 and ϕ_2 as input and returns most of the key numbers discussed above (using all combinations of ϕ_1 and ϕ_2) is implemented. An example of its use is given below

```
phi1 <- 0.75
phi2 <- seq(-0.75, 0.24, by = 0.01)
Summ <- VarFunRap(phi1, phi2)
```

`Summ` is a dataframe containing the square root of the diagonal elements of the variance–covariance matrices reported from `VarFunTrue` as well as all

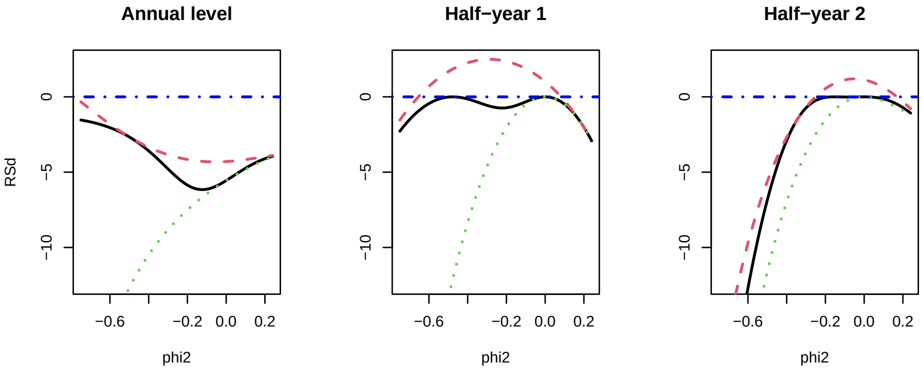

Figure 1: Reproducing the results in Figure 1 in the article

elements of the matrices `Pfull` and `Pre` and RSd (see eq. (9) of the article) for the three forecasts (the values plotted in Figure 1 in the article).

From the dataframe we can reproduce the results in Figure 1 of the article (resulting in Figure 1) by (see Appendix A.3 for the plot function)

```
ylim = c(-12.5, 2.5)
plotFig1(Summ)
```

In addition, we plot the elements of the matrices `Pfull` and `Pre` (Figure 2)

```
ylim = c(-0.6, 1.2)
plotProj(Summ)
```

It is particularly interesting to note that when $\phi_2 = 0$ (i.e., the half-year model is correctly specified), then the projection matrix, using the full version of the variance–covariance matrix, is

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

Hence, only the correctly specified model is used, while the misspecified yearly model is ignored. We also note that the projection matrix when ignoring cross-correlation between aggregation levels is almost constant, indepedently of the choice of ϕ_2.

5 Simulation-based results

In this section, we look at simulations of the process in order to verify the results given in the previous sections, and illustrate the effect estimating

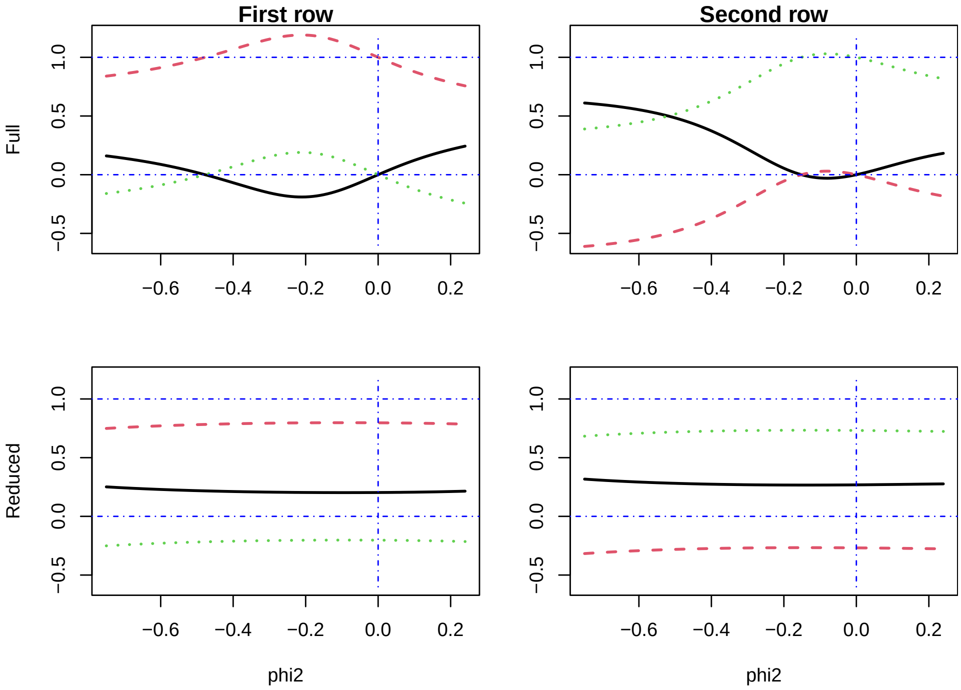

Figure 2: Elements of `Pfull` (top row) and `Pre` (bottom row). Black: $\boldsymbol{P}_{i,1}$, red: $\boldsymbol{P}_{i,2}$, green: $\boldsymbol{P}_{i,3}$.

parameters (i.e., the distribution of RSd and $\boldsymbol{P}$). The R-functions are given in Appendix A.2.

For a specific choice of ϕ_1 and ϕ_2, the values that were discussed in the previous section can be calculated based on one realisation of the training and the test set, using

```r
set.seed(1234)
phi1 <- 0.75
phi2 <- -0.2
T.train <- 1e3
T.test <- 1e3
sums.sim <- VarFunSim(phi1, phi2, T.train, T.test, n.burnin = 1e4,
                      sigma = 1)
```

The output is similar to the output of `VarFunTrue` (see page 8 for a presentation).

The reported values of the variance–covariance matrices are calculated on the test set, while values for the matrices `Pfull` and `Pre` are calculated on the training set.

Since we are simulating values, we would usually be interested in more than one realisation. This is obtained by

```r
set.seed(1234)
k <- 1e3
sums.sim <- VarFunSim.k(phi1, phi2, T.train, T.test, k,
                        n.burnin=1e4, sigma=1)
```

In this case the values are calculated `k` $= 1000$ times, and the reported values are in a dataframe with `k` rows. Finally, the values can be calculated for a range of AR(2) parameters, but only a summary (in terms of a defined set of quantiles) of the simulations is reported for each combination of the AR(2) parameters. This is exemplified in two cases below (one with a small training set and one with a large training set)

```r
set.seed(1234)
phi1 <- 0.75
phi2 <- seq(-0.75,0.24,by=0.01)
q <- c(0.05, 0.5, 0.95)

## Small training set
T.train <- 1e2
T.test <- 1e3
k <- 1e3
Summ.sim1 <- VarFunSimRap.k(phi1, phi2, T.train, T.test, k, q,
                            n.burnin = 1e4, sigma = 1)

## Large training set
T.train <- 1e4
T.test <- 1e4
k <- 100
Summ.sim2 <- VarFunSimRap.k(phi1, phi2, T.train, T.test, k, q,
                            n.burnin = 1e4, sigma = 1)
```

In Figure 3, the simulation results are compared with the theoretical results using (see Appendix A.3 for plot functions)

```r
par(mfrow=c(2,3))
PlotSimRSd(Summ,Summ.sim1,q,lwd=2)
PlotSimRSd(Summ,Summ.sim2,q,lwd=2)
```

In both cases we see that the theoretical values of RSd are in the center of the plotted quantiles and further that the variation becomes very small when the training set is large.

Finally, the matrices `Pfull` and `Pre` can be plotted for the small sample case by (the result is shown in Figure 4)

```r
par(mfrow=c(2,2))
PlotSimP(Summ,Summ.sim1,q,lwd=2)
```

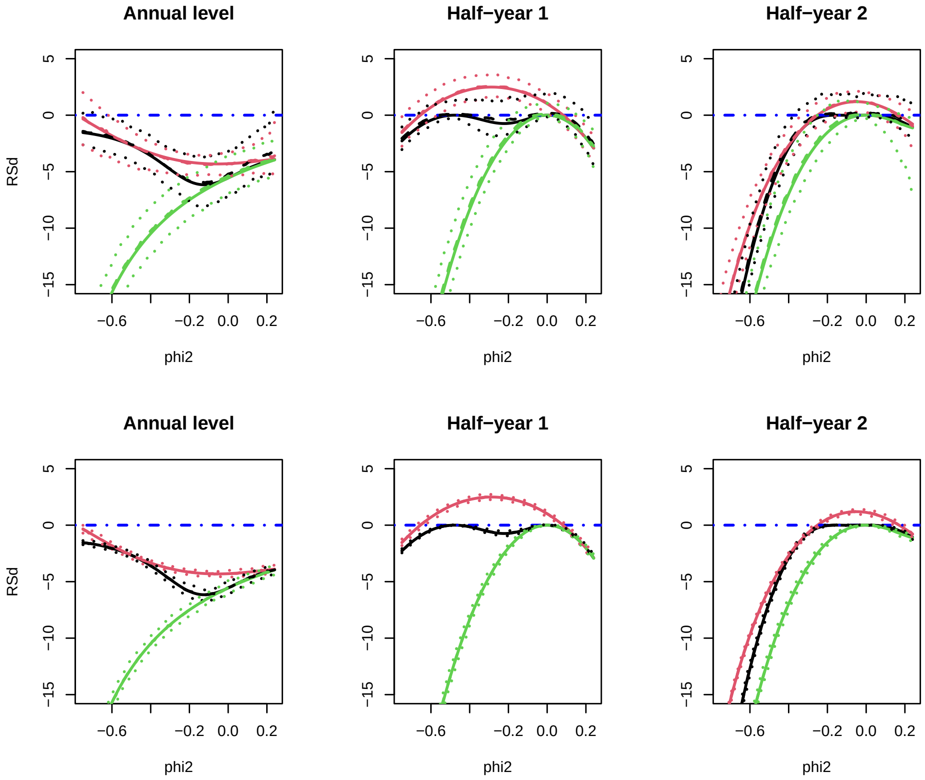

Figure 3: Comparing simulated and theoretical results using a small training set (top row) and a large training set (bottom row). Colors are as in the article, dotted lines represent 0.05 and 0.95 quantiles from simulations, and dashed line represent the median of the simulated values.

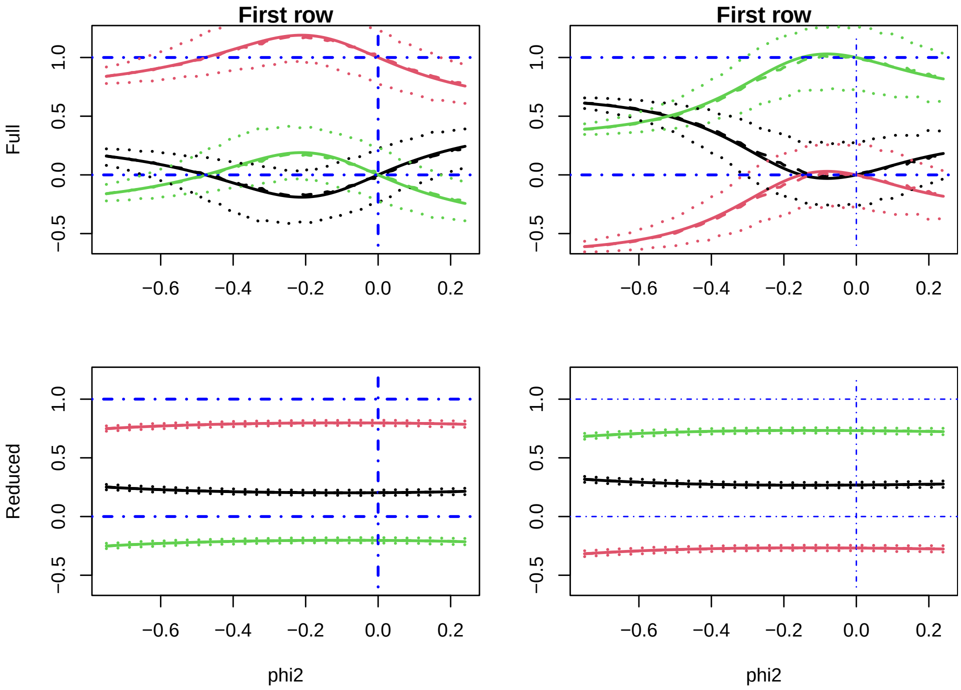

Figure 4: Elements of `Pfull` (top row) and `Pre` (bottom row), for the small training sets. Black: $\boldsymbol{P}_{i,1}$, red: $\boldsymbol{P}_{i,2}$, green: $\boldsymbol{P}_{i,3}$. dotted lines represent 0.05 and 0.95 quantiles from simulations, and dashed line represent the median of the simulated values.

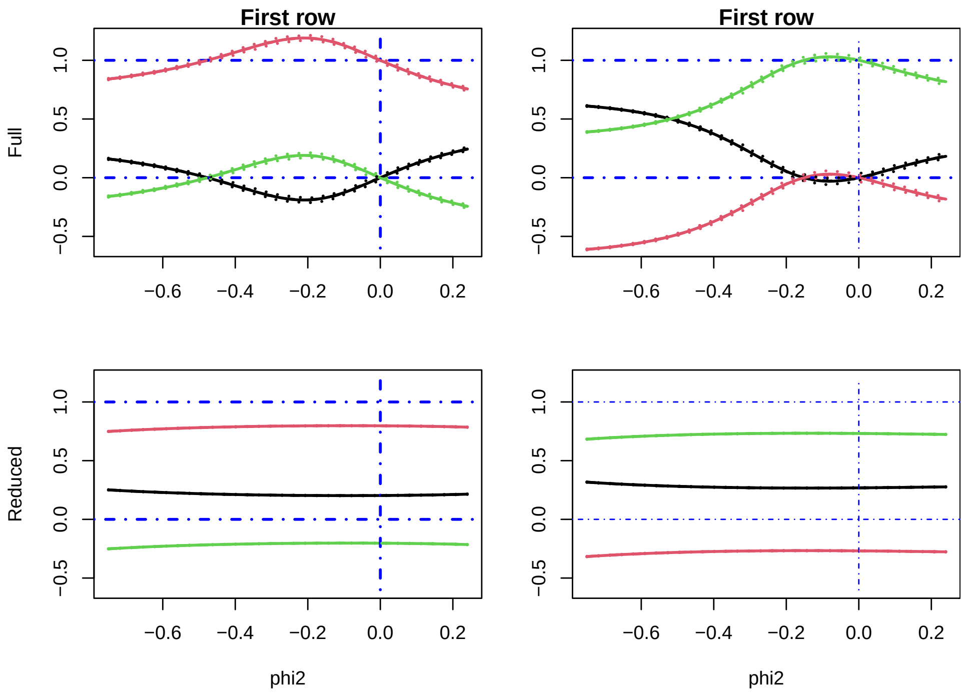

Figure 5: Elements of `Pfull` (top row) and `Pre` (bottom row), for the large training sets. Black: $\boldsymbol{P}_{i,1}$, red: $\boldsymbol{P}_{i,2}$, green: $\boldsymbol{P}_{i,3}$. dotted lines represent 0.05 and 0.95 quantiles from simulations, and dashed line represent the median of the simulated values.

Notice that there is a considerable variation when the full variance–covariance matrix is used, while the variation is much smaller when ignoring cross-covariances.

The same plot can be produced for the large-sample case (resulting in Figure 5) by

```
par(mfrow=c(2,2))
PlotSimP(Summ,Summ.sim2,q,lwd=2)
```

Here we see a very small variation of the elements of $\boldsymbol{P}$ in both cases.

The plots in this section illustrate that the theoretical values from the article can be reproduced in simulation experiments. Further, the example illustrates the effect of estimation errors.

6 Conclusion and discussions

We have verified the results presented in Section 3 and Appendix A of ller et al. (2022). It serves as an illustration of how we work with data and the uncertainties that arise in that process. In particular, the role of parameter uncertainty is illustrated in the small-sample simulation presented in Section 5. The contributions to that uncertainty are: estimation of AR-parameters, uncertainty in the estimation of the variance–covariance matrix ($\boldsymbol{\Sigma}$), plus a contribution from the realisation of the test set. We have aimed at minimising the last contribution by considering large test sets.

References

ller, J. K. M., P. Nystrup, and H. Madsen. “Likelihood-based inference in temporal hierarchies.” *In press* (2022).

Madsen, H. and P. Thyregod. *Introduction to general and generalized linear models*. Texts in statistical science. CRC Press (2011).

A R-functions in library `funs`

A.1 Simple R-functions

```
rho.true.fun <- function(phi1,phi2){
    rho.true <- numeric(3)
    rho.true[1:2] <- c(phi1/(1-phi2), (phi1^2 + phi2 * (1 - phi2)) / (1 - phi2))
    rho.true[3] <-  phi1 * rho.true[2] + phi2 * rho.true[1]
    rho.true
}
```

```
gamma.true.fun <- function(phi1,phi2,sigma){
    rho.true <- rho.true.fun(phi1,phi2)
    gamma0.true <- sigma^2 / (1-phi1 * rho.true[1] - phi2 * rho.true[2])
    c(1, rho.true) * gamma0.true

}
```

```
Phi.fun <- function(phi1,phi2){
    Phi <- matrix(0, ncol = 2, nrow = 3)
    Phi <- matrix(0,ncol=2,nrow=3)
    Phi[1,1] <- phi1 * phi2 + phi2
    Phi[1,2] <- phi1 + phi1^2 + phi2
    Phi[2,1] <- phi2
    Phi[2,2] <- phi1
    Phi[3,1] <- phi1 * phi2
    Phi[3,2] <- phi1^2 + phi2
    Phi
}
```

```
Phi.epsilon.fun <- function(phi1,phi2){
    Phi.epsilon <- matrix(0,ncol=2,nrow=3)
    Phi.epsilon[1,1] <- phi1 + 1
    Phi.epsilon[1,2] <- 1
    Phi.epsilon[2,1] <- 1
    Phi.epsilon[3,1] <- phi1
    Phi.epsilon[3,2] <- 1
    Phi.epsilon
}
```

```
Phi.epsilon.fun <- function(phi1,phi2){
    Phi.epsilon <- matrix(0,ncol=2,nrow=3)
    Phi.epsilon[1,1] <- phi1 + 1
    Phi.epsilon[1,2] <- 1
    Phi.epsilon[2,1] <- 1
    Phi.epsilon[3,1] <- phi1
    Phi.epsilon[3,2] <- 1
    Phi.epsilon
}
```

A.2 R-functions for collecting results

```
VarFunTrue <- function(phi1, phi2, sigma = 1){
    S <- matrix(0,ncol=2, nrow=3)
    S[1, ] <- 1
    S[2:3,1:2] <- diag(2)

    gamma.true <- gamma.true.fun(phi1,phi2,sigma)
    rho1A.true <- (gamma.true[2] + 2 * gamma.true[3] + gamma.true[4]) / (2 * (gamma.true[1]+gamma.true[2]))

    Phi <- Phi.fun(phi1,phi2)
    Phi.epsilon <- Phi.epsilon.fun(phi1,phi2)


    Gamma <- Gamma.fun(rho.true.fun(phi1,phi2)[1],rho1A.true)
    ## eq (A.7)
    VyH <- matrix(gamma.true[2],ncol=2,nrow=2)
    diag(VyH) <- gamma.true[1]

    ## eq (A.6)
    Sigma <- (Phi-Gamma) %*% VyH %*% t(Phi-Gamma) + sigma^2 * Phi.epsilon %*% t(Phi.epsilon)

    Sigma.re <- Sigma
    Sigma.re[1,-1] <- Sigma.re[-1,1] <- 0

    ## Projection matrices
    Pfull <- solve(t(S) %*% solve(Sigma) %*% S) %*% t(S) %*% solve(Sigma)
    Pre <- solve(t(S) %*% solve(Sigma.re) %*% S) %*% t(S) %*% solve(Sigma.re)

    ## eq (A.8)
    Sigma.tilde <- (Phi-S%*%Pfull%*%Gamma) %*% VyH %*% t(Phi-S%*%Pfull%*%Gamma) +
        sigma^2 * Phi.epsilon %*% t(Phi.epsilon)
    Sigma.tilde.re <- (Phi - S %*% Pre %*% Gamma) %*% VyH %*% t(Phi-S %*% Pre %*% Gamma) +
        sigma^2 * Phi.epsilon %*% t(Phi.epsilon)

    ## Perfect forecast
    Sigma.perf <- sigma^2 * Phi.epsilon %*% t(Phi.epsilon)

    return(list(Sigma = Sigma, Sigma.tilde = Sigma.tilde, Sigma.tilde.re = Sigma.tilde.re,
                Sigma.perf = Sigma.perf, Pfull = Pfull, Pre = Pre))
}
```

```
VarFunRap <- function(phi1,phi2){
    Summ <- matrix(nrow=length(phi1)*length(phi2),ncol=35)
    Summ <- data.frame(Summ)
    colnames(Summ)[1:2] <- c("phi1","phi2")
    colnames(Summ)[3:5] <- c("sig1","sig2","sig3")
    colnames(Summ)[6:8] <- c("sigTilde1","sigTilde2","sigTilde3")
    colnames(Summ)[9:11] <- c("sigTildeRe1","sigTildeRe2","sigTildeRe3")
    colnames(Summ)[12:14] <- c("sigPerf1","sigPerf2","sigPerf3")
    colnames(Summ)[15:20] <- c("Pfull11","Pfull12","Pfull13","Pfull21","Pfull22","Pfull23")
    colnames(Summ)[21:26] <- c("Pre11","Pre12","Pre13","Pre21","Pre22","Pre23")
    colnames(Summ)[27:29] <- c("RSd1","RSd2","RSd3")
    colnames(Summ)[30:32] <- c("RSd.re1","RSd.re2","RSd.re3")
    colnames(Summ)[33:35] <- c("RSd.perf1","RSd.perf2","RSd.perf3")
    count <- 1
    i<-j<-1
    for(i in 1:length(phi1)){
        for(j in 1:length(phi2)){
            Summ[count, c("phi1", "phi2")] <- c(phi1[i], phi2[j])
            if(phi1[i]==0 & phi2[j]==0){Summ[count,-(1:2)] <- NA
                count <- count + 1
            }
            if(phi1[i]!=0 | phi2[j]!=0){
                tmp <- VarFunTrue(phi1[i],phi2[j])
                Summ[count, c("sig1","sig2","sig3")] <- sqrt(diag(tmp$Sigma))
                Summ[count, c("sigTilde1","sigTilde2","sigTilde3")] <- sqrt(diag(tmp$Sigma.tilde))
                Summ[count, c("sigTildeRe1","sigTildeRe2","sigTildeRe3")] <- sqrt(diag(tmp$Sigma.tilde.re))
                Summ[count, c("sigPerf1","sigPerf2","sigPerf3")] <-  sqrt(diag(tmp$Sigma.perf))
                Summ[count, c("Pfull11","Pfull12","Pfull13","Pfull21","Pfull22","Pfull23")] <-
                    as.vector(t(tmp$Pfull))
                Summ[count, c("Pre11","Pre12","Pre13","Pre21","Pre22","Pre23")] <-  as.vector(t(tmp$Pre))
                Summ[count, c("RSd1","RSd2","RSd3")] <- (sqrt(diag(tmp$Sigma.tilde)) - sqrt(diag(tmp$Sigma))) /
                    sqrt(diag(tmp$Sigma)) * 100
                Summ[count, c("RSd.re1","RSd.re2","RSd.re3")] <-
                    (sqrt(diag(tmp$Sigma.tilde.re)) - sqrt(diag(tmp$Sigma))) / sqrt(diag(tmp$Sigma)) * 100
                Summ[count, c("RSd.perf1","RSd.perf2","RSd.perf3")] <-
                    (sqrt(diag(tmp$Sigma.perf)) - sqrt(diag(tmp$Sigma))) / sqrt(diag(tmp$Sigma)) *100
                count <- count + 1
            }
        }
    }
    return(data.frame(Summ))
}
```

```
VarFunSim <- function(phi1,phi2,T.train,T.test,n.burnin=1e4,sigma=1){
    y.train <- arima.sim(n = 2 * T.train + n.burnin, list(ar = c(phi1, phi2)), sd = sigma)
    y.train <- y.train[-(1:n.burnin)]
    ## set up matrix
    y1.train <- y.train[seq(1, 2 * T.train,by=2)]
    y2.train <- y.train[seq(2, 2 * T.train,by=2)]
    yA.train <- y1.train + y2.train
    Y.train <- cbind(yA.train, y1.train, y2.train)
    phi1.hat <- coef(arima(y.train,order = c(1,0,0),include.mean=FALSE,method="CSS"))
    phi1A.hat <- coef(arima(yA.train,order = c(1,0,0),include.mean=FALSE,method="CSS"))

    Y.hat <- cbind(phi1A.hat * yA.train[-T.train], phi1.hat * y2.train[-T.train],
                   phi1.hat^2 * y2.train[-T.train])
    Sigma <- var(Y.train[-1, ]-Y.hat)

    Sigma.re <- Sigma
    Sigma.re[1,-1] <- Sigma.re[-1,1] <- 0

    ## Projection matrices
    Pfull <- solve(t(S) %*% solve(Sigma) %*% S) %*%
        t(S) %*% solve(Sigma)
    Pre <- solve(t(S) %*% solve(Sigma.re) %*% S) %*%
        t(S) %*% solve(Sigma.re)


    ## perfect model
    fit <- arima(y.train,order=c(2,0,0),include.mean = FALSE,method="CSS")
    phi1.perf <- coef(fit)[1]
    phi2.perf <- coef(fit)[2]

    ##################################################
    ## Test set
    ##################################################

    ## simulate
    y.test <- arima.sim(n = 2 * T.test + n.burnin, list(ar = c(phi1, phi2)), sd = sigma)
    y.test <- y.test[-(1:n.burnin)]
    ## set up matrix
    y1.test <- y.test[seq(1,2 * T.test, by=2)]
    y2.test <- y.test[seq(2,2 * T.test, by=2)]
    yA.test <- y1.test + y2.test
    Y.test <- cbind(yA.test, y1.test, y2.test)

    Y.hat <- cbind(phi1A.hat * yA.test[-T.test], phi1.hat * y2.test[-T.test],
                   phi1.hat^2 * y2.test[-T.test])
    y1.hat.perf <- phi1.perf * y2.test[-T.test] + phi2.perf * y1.test[-T.test]
    y2.hat.perf <- phi1.perf * y1.hat.perf[-T.test] + phi2.perf * y2.test[-T.test]

    Y.hat.perf <- cbind(y1.hat.perf + y2.hat.perf, y1.hat.perf, y2.hat.perf)


    ## Observed variances
    Sigma <- var(Y.test[-1, ]-Y.hat)
    Sigma.perf <- var(Y.test[-1, ]-Y.hat.perf)

    ## Reconciled

    Hfull <- S %*% Pfull
    Hre <- S %*% Pre
    Y.tilde1 <- cbind(Y.hat%*%Hfull[1,],Y.hat%*%Hfull[2,],Y.hat%*%Hfull[3,])
    Y.tilde2 <- cbind(Y.hat%*%Hre[1,],Y.hat%*%Hre[2,],Y.hat%*%Hre[3,])
    Sigma.tilde <- var(Y.test[-1, ]-Y.tilde1)
    Sigma.tilde.re <- var(Y.test[-1, ]-Y.tilde2)

    return(list(Sigma = Sigma, Sigma.tilde = Sigma.tilde, Sigma.tilde.re = Sigma.tilde.re,
                Sigma.perf = Sigma.perf, Pfull = Pfull, Pre = Pre))
}
```

```
VarFunSim.k <- function(phi1, phi2, T.train, T.test, k, n.burnin = 1e4, sigma = 1){
    Summ <- matrix(nrow=k,ncol=33)
    Summ <- data.frame(Summ)
    colnames(Summ)[1:3] <- c("sig1","sig2","sig3")
    colnames(Summ)[4:6] <- c("sigTilde1","sigTilde2","sigTilde3")
    colnames(Summ)[7:9] <- c("sigTildeRe1","sigTildeRe2","sigTildeRe3")
    colnames(Summ)[10:12] <- c("sigPerf1","sigPerf2","sigPerf3")
    colnames(Summ)[13:18] <- c("Pfull11","Pfull12","Pfull13","Pfull21","Pfull22","Pfull23")
    colnames(Summ)[19:24] <- c("Pre11","Pre12","Pre13","Pre21","Pre22","Pre23")
    colnames(Summ)[25:27] <- c("RSd1","RSd2","RSd3")
    colnames(Summ)[28:30] <- c("RSd.re1","RSd.re2","RSd.re3")
    colnames(Summ)[31:33] <- c("RSd.perf1","RSd.perf2","RSd.perf3")
    for(i in 1:k){
        tmp <- VarFunSim(phi1,phi2,T.train,T.test,n.burnin=1e4,sigma=1)
        Summ[i, c("sig1","sig2","sig3")] <- sqrt(diag(tmp$Sigma))
        Summ[i, c("sigTilde1","sigTilde2","sigTilde3")] <- sqrt(diag(tmp$Sigma.tilde))
        Summ[i, c("sigTildeRe1","sigTildeRe2","sigTildeRe3")] <-
                sqrt(diag(tmp$Sigma.tilde.re))
        Summ[i, c("sigPerf1","sigPerf2","sigPerf3")] <-  sqrt(diag(tmp$Sigma.perf))
        Summ[i, c("Pfull11","Pfull12","Pfull13","Pfull21","Pfull22","Pfull23")] <-
            as.vector(t(tmp$Pfull))
        Summ[i, c("Pre11","Pre12","Pre13","Pre21","Pre22","Pre23")] <-
            as.vector(t(tmp$Pre))
        Summ[i, c("RSd1","RSd2","RSd3")] <- (sqrt(diag(tmp$Sigma.tilde)) -
                                             sqrt(diag(tmp$Sigma))) /
            sqrt(diag(tmp$Sigma)) * 100
        Summ[i, c("RSd.re1","RSd.re2","RSd.re3")] <-
            (sqrt(diag(tmp$Sigma.tilde.re)) - sqrt(diag(tmp$Sigma))) / sqrt(diag(tmp$Sigma)) *
            100
        Summ[i, c("RSd.perf1","RSd.perf2","RSd.perf3")] <-
            (sqrt(diag(tmp$Sigma.perf)) - sqrt(diag(tmp$Sigma))) / sqrt(diag(tmp$Sigma)) *100
    }
    return(data.frame(Summ))
}
```

```
VarFunSimRap.k <- function(phi1, phi2, T.train, T.test, k, q, n.burnin = 1e4, sigma = 1){
    Summ <- c()
    for(i in 1:length(phi1)){
        for(j in 1:length(phi2)){
            tmp <- VarFunSim.k(phi1[i], phi2[j], T.train, T.test, k, n.burnin = 1e4, sigma = 1)
            tmp <- apply(tmp, 2,quantile,probs=q)
            Summ <- rbind(Summ, cbind(phi1=phi1[i], phi2=phi2[j], k, q, tmp))
        }
    }
    data.frame(Summ)
}
```

A.3 R-functions for plotting

```
plotFig1 <- function(Summ){
    par(mfrow=c(1,3))
    matplot(Summ$phi2,cbind(Summ$RSd1,Summ$RSd.re1,Summ$RSd.perf1),type="l",ylim=ylim,
                    lwd=2,main="Annual level",ylab="RSd",xlab="phi2")
    abline(a=0,b=0,lty=4,col="blue",lwd=2)

    matplot(Summ$phi2,cbind(Summ$RSd2,Summ$RSd.re2,Summ$RSd.perf2),type="l",ylim=ylim,
            lwd=2,main="Half-year 1",ylab="",xlab="phi2")
    abline(a=0,b=0,lty=4,col="blue",lwd=2)

    matplot(Summ$phi2,cbind(Summ$RSd3,Summ$RSd.re3,Summ$RSd.perf3),type="l",ylim=ylim,
            lwd=2,main="Half-year 2",ylab="",xlab="phi2")
    abline(a=0,b=0,lty=4,col="blue",lwd=2,main="Half-year 1")
}
```

```r
plotProj <- function(Summ){
    par(mfrow=c(2,2),mar=c(4,4,1,0))
    matplot(Summ$phi2,cbind(Summ$Pfull11,Summ$Pfull12,Summ$Pfull13),type="l",lwd=2,ylim=ylim,
            ylab="Full",main="First row",xlab="")
    abline(a=0,b=0,col="blue",lty=4)
    abline(a=1,b=0,col="blue",lty=4)
    lines(c(0,0),ylim,col="blue",lty=4)

    matplot(Summ$phi2,cbind(Summ$Pfull21,Summ$Pfull22,Summ$Pfull23),type="l",lwd=2,ylim=ylim,
            main="Second row",ylab="",xlab="")
    abline(a=0,b=0,col="blue",lty=4)
    abline(a=1,b=0,col="blue",lty=4)
    lines(c(0,0),ylim,col="blue",lty=4)

    matplot(Summ$phi2,cbind(Summ$Pre11,Summ$Pre12,Summ$Pre13),type="l",lwd=2,ylim=ylim,
            ylab="Reduced",xlab="phi2")
    abline(a=0,b=0,col="blue",lty=4)
    abline(a=1,b=0,col="blue",lty=4)
    lines(c(0,0),ylim,col="blue",lty=4)

    matplot(Summ$phi2,cbind(Summ$Pre21,Summ$Pre22,Summ$Pre23),type="l",lwd=2,ylim=ylim,
            ylab="",xlab="phi2")
    abline(a=0,b=0,col="blue",lty=4)
    abline(a=1,b=0,col="blue",lty=4)
    lines(c(0,0),ylim,col="blue",lty=4)
}
```

```r
PlotSimRSd <- function(Summ,Summ.sim,q,lwd=2){
    ylim <- c(-15,5)
    matplot(Summ$phi2,cbind(Summ$RSd1,Summ$RSd.re1,Summ$RSd.perf1),type="l",ylim=ylim,
            main="Annual level",ylab="RSd",xlab="phi2",lty=1,lwd=lwd)
    abline(a=0,b=0,lty=4,col="blue",lwd=lwd)
    I05 <- Summ.sim$q==q[1]
    I50 <- Summ.sim$q==q[2]
    I95 <- Summ.sim$q==q[3]
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("RSd1","RSd.re1","RSd.perf1")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("RSd1","RSd.re1","RSd.perf1")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("RSd1","RSd.re1","RSd.perf1")],lty=3,lwd=lwd)

    matplot(Summ$phi2,cbind(Summ$RSd2,Summ$RSd.re2,Summ$RSd.perf2),type="l",ylim=ylim,
            main="Half-year 1",ylab="",xlab="phi2",lwd=lwd,lty=1)
    abline(a=0,b=0,lty=4,col="blue",lwd=2)
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("RSd2","RSd.re2","RSd.perf2")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("RSd2","RSd.re2","RSd.perf2")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("RSd2","RSd.re2","RSd.perf2")],lty=3,lwd=lwd)

    matplot(Summ$phi2,cbind(Summ$RSd3,Summ$RSd.re3,Summ$RSd.perf3),type="l",ylim=ylim,
            main="Half-year 2",ylab="",xlab="phi2",lty=1,lwd=lwd)
    abline(a=0,b=0,lty=4,col="blue",main="Half-year 1",lwd=lwd)
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("RSd3","RSd.re3","RSd.perf3")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("RSd3","RSd.re3","RSd.perf3")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("RSd3","RSd.re3","RSd.perf3")],lty=3,lwd=lwd)
}
```

```r
PlotSimP <- function(Summ,Summ.sim,q,lwd=2){
    par(mfrow=c(2,2),mar=c(4,4,1,0))
    ylim=c(-0.6,1.2)
    matplot(Summ$phi2,cbind(Summ$Pfull11,Summ$Pfull12,Summ$Pfull13),type="l",ylim=ylim,
            ylab="Full",main="First row",xlab="",lwd=lwd,lty=1)
    abline(a=0,b=0,col="blue",lty=4,,lwd=lwd)
    abline(a=1,b=0,col="blue",lty=4,,lwd=lwd)
    lines(c(0,0),ylim,col="blue",lty=4,,lwd=lwd)

    I05 <- Summ.sim$q==q[1]
    I50 <- Summ.sim$q==q[2]
    I95 <- Summ.sim$q==q[3]
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("Pfull11","Pfull12","Pfull13")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("Pfull11","Pfull12","Pfull13")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("Pfull11","Pfull12","Pfull13")],lty=3,lwd=lwd)

    matplot(Summ$phi2,cbind(Summ$Pfull21,Summ$Pfull22,Summ$Pfull23),type="l",ylim=ylim,
            ylab="",main="First row",xlab="",lty=1,lwd=lwd)
    abline(a=0,b=0,col="blue",lty=4,lwd=lwd)
    abline(a=1,b=0,col="blue",lty=4,lwd=lwd)
    lines(c(0,0),ylim,col="blue",lty=4)
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("Pfull21","Pfull22","Pfull23")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("Pfull21","Pfull22","Pfull23")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("Pfull21","Pfull22","Pfull23")],lty=3,lwd=lwd)

    matplot(Summ$phi2,cbind(Summ$Pre11,Summ$Pre12,Summ$Pre13),type="l",ylim=ylim,
            ylab="Reduced",main="",xlab="phi2",lty=1,lwd=lwd)
    abline(a=0,b=0,col="blue",lty=4,lwd=lwd)
    abline(a=1,b=0,col="blue",lty=4,lwd=lwd)
    lines(c(0,0),ylim,col="blue",lty=4,lwd=lwd)
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("Pre11","Pre12","Pre13")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("Pre11","Pre12","Pre13")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("Pre11","Pre12","Pre13")],lty=3,lwd=lwd)


    matplot(Summ$phi2,cbind(Summ$Pre21,Summ$Pre22,Summ$Pre23),type="l",ylim=ylim,
            ylab="",main="",xlab="phi2",lty=1,lwd=lwd)
    abline(a=0,b=0,col="blue",lty=4)
    abline(a=1,b=0,col="blue",lty=4)
    lines(c(0,0),ylim,col="blue",lty=4)
    matlines(Summ.sim$phi2[I05],Summ.sim[I05,c("Pre21","Pre22","Pre23")],lty=3,lwd=lwd)
    matlines(Summ.sim$phi2[I50],Summ.sim[I50,c("Pre21","Pre22","Pre23")],lty=2,lwd=lwd)
    matlines(Summ.sim$phi2[I95],Summ.sim[I95,c("Pre21","Pre22","Pre23")],lty=3,lwd=lwd)
}
```