Supplemental material

Toy example – Additional Tables

1. Toy example, reported in the manuscript

Table S1

The loadings and the VAR(1) coefficients of the PC-VAR(1) approach with three components

		Comp 1	Comp 2	Comp 3
Loadings	V1	0.93	0.26	0.21
	V2	0.94	0.25	0.2
	V3	0.93	0.27	0.21
	V4	0.29	0.27	0.92
	V5	0.28	0.93	0.19
	V6	0.27	0.93	0.21
VAR(1) coefficients	Comp 1.lag 1	0.47	0.18	0.2
	Comp 2.lag 1	0.2	0.33	0.16
	Comp 3.lag 1	0.21	0.23	0.33

Table S2

The loadings and the VAR(1) coefficients of the EFA-VAR(1) approach with three factors

		Fact 1	Fact 2	Fact 3
Loadings	V1	0.92	0.27	0.21
	V2	0.93	0.25	0.2
	V3	0.92	0.28	0.21
	V4	0.29	0.27	0.91
	V5	0.28	0.94	0.18
	V6	0.28	0.91	0.22
VAR(1) coefficients	Fact 1.lag 1	0.47	0.18	0.2
	Fact 2.lag 1	0.2	0.34	0.15
	Fact 3.lag 1	0.21	0.23	0.33

2. Toy example – Identically generated as the reported one, apart from 50% expected noise variance (instead of 5%)

Table S3

The loadings and the VAR(1) coefficients of the PC-VAR(1) approach with three components

		Comp 1	Comp 2	Comp 3
Loadings	V1	0.85	0.23	0.12
	V2	0.85	0.16	0.17
	V3	0.84	0.16	0.18
	V4	0.26	0.26	0.93
	V5	0.21	0.88	0.18
	V6	0.19	0.89	0.16
VAR(1) coefficients	Comp 1.lag 1	0.42	0.19	0.25
	Comp 2.lag 1	0.13	0.34	0.17
	Comp 3.lag 1	0.22	0.23	0.23

Table S4

The loadings and the VAR(1) coefficients of the EFA-VAR(1) approach with three factors

		Fact 1	Fact 2	Fact 3
Loadings	V1	0.8	0.22	0.14
	V2	0.79	0.17	0.17
	V3	0.76	0.19	0.18
	V4	0.27	0.27	0.92
	V5	0.26	0.65	0.22
	V6	0.18	0.97	0.13
VAR(1) coefficients	Fact 1.lag 1	0.43	0.17	0.23
	Fact 2.lag 1	0.15	0.39	0.13
	Fact 3.lag 1	0.24	0.16	0.23

R code to generate the toy example data set and to apply the methods under study

```
### Generate toy example according to a dynamic factor model

# y(t) = B*f(t) + e(t)
# f(t) = Phi*f(t-1) + u(t)

# T -> Toy example data
# B -> loading matrix
# F -> Component scores
# E -> Error scores

# Phi -> Lagged relations
# U -> Innovations


# 1. Load packages + define input variables

library('MASS')
library('graphicalVAR')

nVar <- 6
nComp <- 3
nTime <- 500
error <- .05


# 2. Create stationary Phi matrix

Phi <- matrix(0,nComp,nComp)
diag(Phi) <- .4 # The diagonal elements
Phi[diag(1,3)==0] <- .2 # The off-diagonal elements

# To check for stationarity: Modulus of the eigenvalues < 1

spectDecomp <- eigen(Phi)
eigenvalues <- spectDecomp$values
eigenvalues


# 3. Generate component scores F, following a VAR(1) process

Sigma <- matrix(1,nComp,nComp)
Sigma[diag(1,3)==0] <- .2

nIntro <- 1000
U <- mvrnorm(nTime + nIntro,matrix(0,nComp,1),Sigma)
F <- matrix(0,nTime + nIntro,nComp)
F[1,] <- U[1,]

for (t in 2: (nTime + nIntro)){
  F[t,] <- F[t-1,] %*% Phi + U[t,]
}

F <- F[-(1:nIntro),]
```

```
# 4. Combine component scores F, with loading matrix B and error values E to obtain latent structure

eye <- diag(1,3)
B <- eye[c(1,1,1,2,3,3),] # Loading matrix

randomErrorValues <- rnorm(nTime * nVar,0,1)
randomError <- matrix(randomErrorValues,nTime,nVar)
E <- sqrt(error) * randomError

data <- F %*% t(B) + E


### Standardization

# Function to standardize based on the population standard deviation

pop.var <- function(x) var(x) * (length(x)-1) / length(x)
pop.sd <- function(x) sqrt(pop.var(x))

standardize.popsd <- function(dataset){
  means <- apply(dataset,2,mean)
  stdevs <- apply(dataset,2,pop.sd)
  dataCent <- sweep(dataset,2,means)
  dataSt <- sweep(dataCent,2,stdevs,"/")
  return(dataSt)
}

# Standardize data

dataSt <- standardize.popsd(data)

# First split data into criterion and predictor (i.e., lagged) variables, and then standardize

X <- head(data,-1)
Y <- tail(data,-1)
XSt <- standardize.popsd(X)
YSt <- standardize.popsd(Y)


### Analyses


library('graphicalVAR')
library('qgraph')
library('glmnet')
library('psych')


# 1. VAR(1)

VARfit <- lsfit(XSt,YSt,intercept = FALSE)
Phi_VAR <- VARfit$coefficients
```

```
# 2. Univariate lasso VAR(1)


XSt <- as.matrix(XSt)
YSt <- as.matrix(YSt)

K <- 10 # The number of folds
nTime <- nTime – 1
BlockedCVInd <- as.numeric(factor(sort(rank(1:nTime)%%K)))

Phi_lassoVAR <- matrix(0,nVar,nVar)

for (v in 1:nVar){
  cvfit <- cv.glmnet(XSt, YSt[,v],foldid = BlockedCVInd)
  TempPhi <- as.matrix(coef(cvfit, s = "lambda.min"))
  Phi_lassoVAR[,v] <- TempPhi[-1,] # Remove intercept
  remove(cvfit,TempPhi)
}


# 3. Graphical VAR

res <- graphicalVAR(dataSt)
PCC <- res$PCC
PDC <- res$PDC


# 4. PC-VAR(1)

PCA_varimax <- principal(dataSt, nfactors = nComp, rotate="varimax")
F_rot <- PCA_varimax$scores
B_rot <- PCA_varimax$loadings
PCVARfit <- lsfit(head(F_rot,-1),tail(F_rot,-1),intercept = FALSE) # VAR(1) analysis on rotated
scores
Phi_PCVAR <- PCVARfit$coefficients
```