Replication of Figures 3–7 and Tables 3–4

Approximate Factor Models for Functional Time Series

Sven Otto and Nazarii Salish

Install the `dffm` package:

```
remotes::install_github("ottosven/dffm")
```

```
library(dffm)
library(knitr)
library(plot3D)

# Create figures directory if it doesn't exist
dir.create("figures", showWarnings = FALSE)
# Create tables directory if it doesn't exist
dir.create("tables", showWarnings = FALSE)
# Create data directory if it doesn't exist
dir.create("data", showWarnings = FALSE)
```

Run `01_retrieve_data.R` to download the data used in the empirical applications. The data will be saved in the `data` directory.

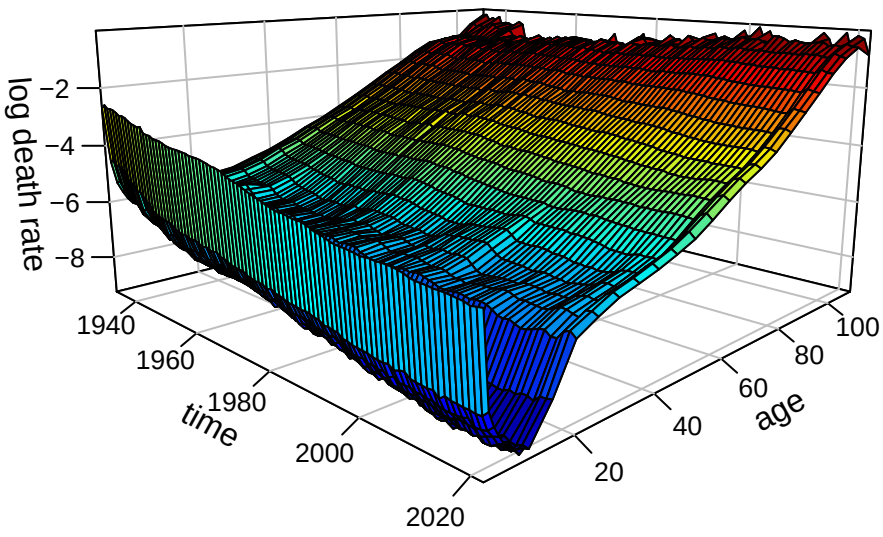
```
source("01_retrieve_data.R")
```

Replication of Figure 3

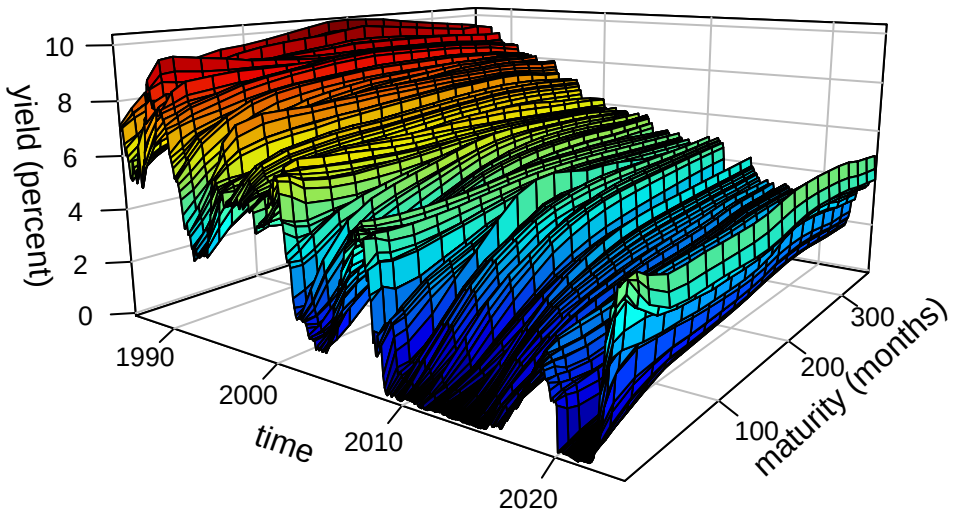
Log mortality rate curves for U.S. males and U.S. Treasury yield curves

```
# Load and prep data
mydata = readRDS("./data/mort.rds")
grid = c(1:9, seq(10, ncol(mydata), by=5))
mydata2 = ts(mydata[, grid], start=1933, frequency=1)
```

```r
colnames(mydata2) = grid
fda2 = fda.preprocess(mydata2)

# Set up plot variables
time = replicate(length(fda2$observationgrid), time(fda2$raw.data))

# Create 3D surface plot
plot3D::surf3D(
  x = time,
  y = t(replicate(dim(fda2$raw.data)[1], fda2$observationgrid)),
  z = fda2$raw.data,
  theta = 45, phi = 12,
  bty = "b2",
  colvar = fda2$raw.data,
  shade = 0.05,
  border = "black",
  zlim = range(fda2$raw.data) + c(-0.1, 0.1),
  xlab = 'time', ylab = "age", zlab = "log death rate",
  ticktype = "detailed",
  expand = 0.5,
  cex.axis = 0.8,
  col = NULL,
  colkey = FALSE
)
```



```r
# Load and prep data
mydata = readRDS("./data/LW360.rds")
grid = c(1, 10, seq(20, 360, by=20))
mydata2 = ts(mydata[, grid], start=c(1985,11), frequency=12)
colnames(mydata2) = grid
fda2 = fda.preprocess(mydata2)

# Set up plot variables
time = replicate(length(fda2$observationgrid), time(fda2$raw.data))

# Create 3D surface plot
plot3D::surf3D(
 x = time,
 y = t(replicate(dim(fda2$raw.data)[1], fda2$observationgrid)),
 z = fda2$raw.data,
 theta = 35, phi = 12,
 bty = "b2",
 colvar = fda2$raw.data,
 shade = 0.05,
 border = "black",
 zlim = range(fda2$raw.data) + c(-0.1, 0.1),
 xlab = 'time', ylab = "maturity (months)", zlab = "yield (percent)",
 ticktype = "detailed",
 expand = 0.5,
 cex.axis = 0.8,
 col = NULL,
 colkey = FALSE
)
```



Replication of Table 3

In-sample MSFEs for log mortality curves with VAR(1) dynamics

```
# Load data and create forecasts
mydata = readRDS("./data/mort.rds")
fdaobj = readRDS("./data/mort-fda.rds")
cumacobj = fts.cumAC(fdaobj)

VAR1forecasts = function(K, obj) {
 forecast = fts.VARforecast(obj, K=K, p=1, AR=FALSE, h=0)
 colMeans((forecast$curve.predict - mydata)^2)
}

# Calculate metrics and create table
tab = round(rbind(
 Factor = colMeans(sapply(1:8, VAR1forecasts, obj=cumacobj)),
 PCA = colMeans(sapply(1:8, VAR1forecasts, obj=fdaobj))
), 5)
colnames(tab) = 1:8

# Export to LaTeX and display
latex_content = c(
 "\\begin{table}[t]",
 "\\centering",
 kable(tab, format="latex", booktabs=TRUE),
```

```
 "\\end{table}"
)
writeLines(latex_content, "tables/table3.tex")
kable(tab)
```

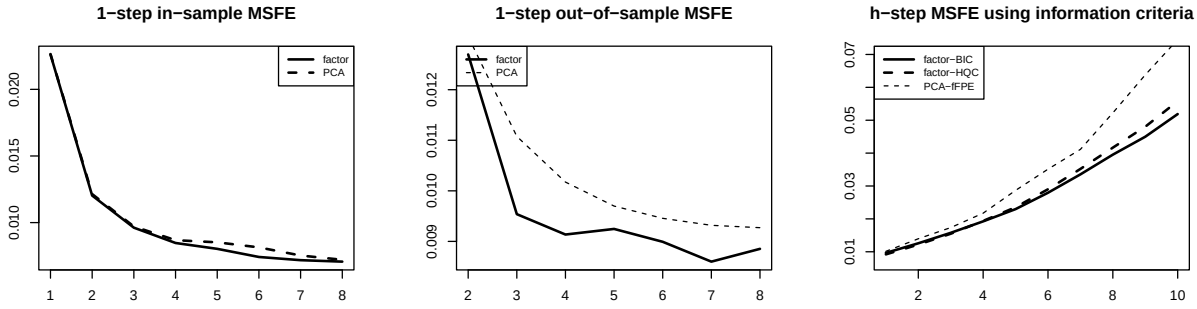
	1	2	3	4	5	6	7	8
Factor	0.02262	0.01203	0.00962	0.00847	0.00803	0.00742	0.00718	0.00707
PCA	0.02265	0.01214	0.00971	0.00868	0.00852	0.00814	0.00753	0.00721

Replication of Figure 4

VAR(1) forecasting results for the mortality data

```
par(mfrow = c(1,3))

# Plot 1: MSFE comparison
plot(1:8, tab[1,], type="l", lwd=2, ylab="", xlab="", main="1-step in-sample MSFE")
lines(1:8, tab[2,], lty=2, lwd=2)
legend("topright", c("factor", "PCA"), lty=c(1,2), lwd=2, seg.len=2.5, cex=0.8)

# Plot 2: MSFE comparison
results = readRDS("./mortality-results/1step-rolling.rds")
MSEs = colMeans(results)
plot_msfe = function(idx1, idx2, steps, title) {
  plot(steps, MSEs[idx1], type="l", lwd=2, main=title, ylab="", xlab="")
  lines(steps, MSEs[idx2], lty=2)
  legend("topleft", c("factor", "PCA"), lty=c(1,2), lwd=c(2,1), cex=0.8)
}
plot_msfe(4:10, 42:48, 2:8, "1-step out-of-sample MSFE")

# Plot 3: Information criteria comparison
get_results = function(steps) {
  sapply(steps, function(i) {
    res = colMeans(readRDS(sprintf("./mortality-results/%dstep-rolling.rds", i)))
    c(res["BIC.VAR.AC"], res["HQC.VAR.AC"], res["fFPE.VAR.PC"])
  })
}

results = get_results(1:12)
```

```
plot(1:10, results[1,1:10], type="l", lwd=2, ylim=c(0.007,0.07),
     ylab="", xlab="", main="h-step MSFE using information criteria")
lines(1:10, results[2,1:10], lwd=2, lty=2)
lines(1:10, results[3,1:10], lty=2)
legend("topleft", c("factor-BIC", "factor-HQC", "PCA-fFPE"),
       lwd=c(2,2,1), lty=c(1,2,2), cex=0.8, seg.len=3)
```

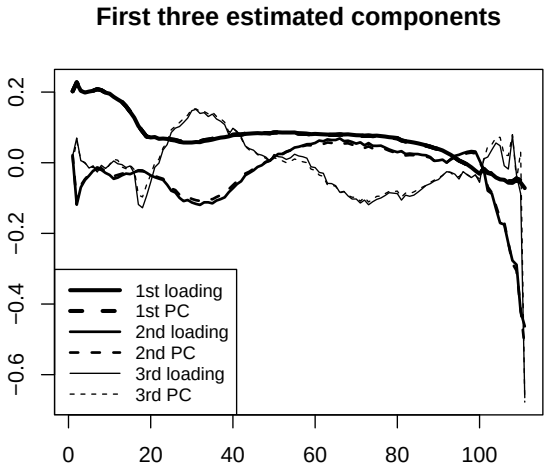


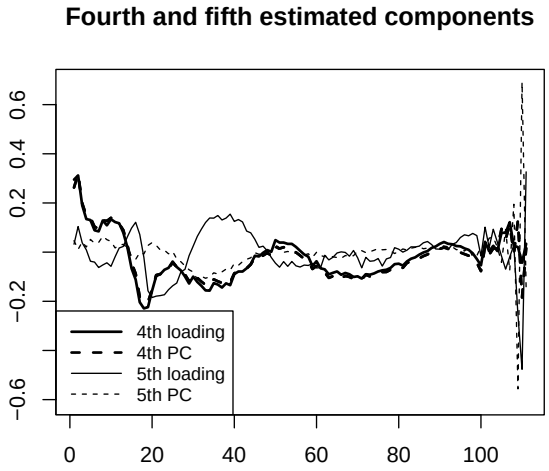
Replication of Figure 5

Factor loadings and principal components for the mortality data

```
par(mfrow = c(1,2))

plot_components = function(comps, title, yrange=NULL) {
  if (is.null(yrange)) {
    yrange = range(c(fdaobj$eigenfunctions[,comps], cumacobj$eigenfunctions[,comps]))
  }
  plot(fdaobj$eigenfunctions[,comps[1]], type="l", lty=2, ylab="", xlab="",
       main=title, lwd=length(comps), ylim=yrange)
  for (i in seq_along(comps)) {
    lines(cumacobj$eigenfunctions[,comps[i]], lwd=length(comps)+1-i)
    lines(fdaobj$eigenfunctions[,comps[i]], lty=2, lwd=length(comps)+1-i)
  }
}

plot_components(1:3, "First three estimated components")
legend("bottomleft",
       c("1st loading", "1st PC", "2nd loading", "2nd PC", "3rd loading", "3rd PC"),
       lty=rep(c(1,2), 3), lwd=rep(c(3,2,1), each=2), cex=0.8, seg.len=3.2)
```

```
plot_components(4:5, "Fourth and fifth estimated components", c(-0.61, max(c(fdaobj$eigenfun
legend("bottomleft",
       c("4th loading", "4th PC", "5th loading", "5th PC"),
       lty=rep(c(1,2), 2), lwd=rep(c(2,1), each=2), cex=0.8, seg.len=3.2)
```



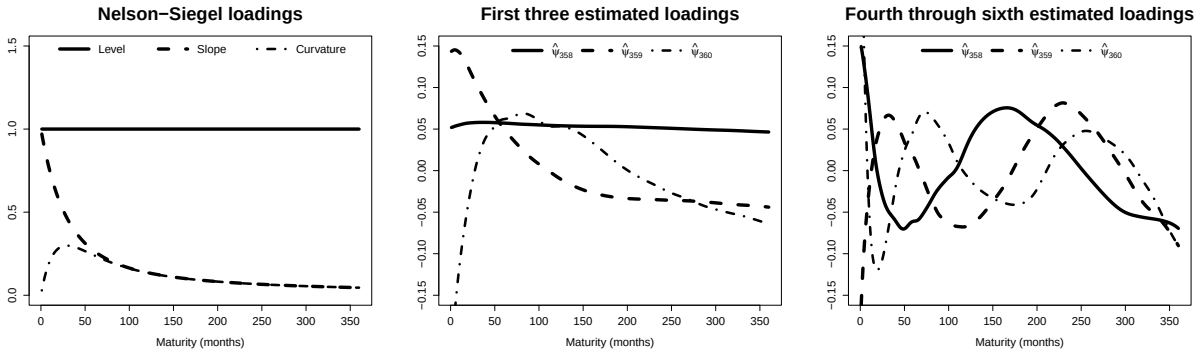


Replication of Figure 6

Loading Functions of the DNS model and yield curve data

```
par(mfrow=c(1,3))

# Nelson-Siegel loadings plot
figure.DNS = function(lambda=0.0609, data.length=360, maturity.length=120) {
 NSline = sapply(1:data.length, function(i) {
   c(1,
     (1-exp(-lambda*i))/(lambda*i),
     (1-exp(-lambda*i))/(lambda*i) - exp(-lambda*i))
 }) |> t()

 plot(1:maturity.length, NSline[1:maturity.length,1],
      type='l', ylim=c(0,1.5), lty=1, lwd=4,
      xlab="Maturity (months)", ylab="", main="Nelson-Siegel loadings",
      cex.lab=1.3, cex.axis=1.3, cex.main=2)
 lines(1:maturity.length, NSline[1:maturity.length,2], lty=2, lwd=4)
 lines(1:maturity.length, NSline[1:maturity.length,3], lty=4, lwd=3)
```

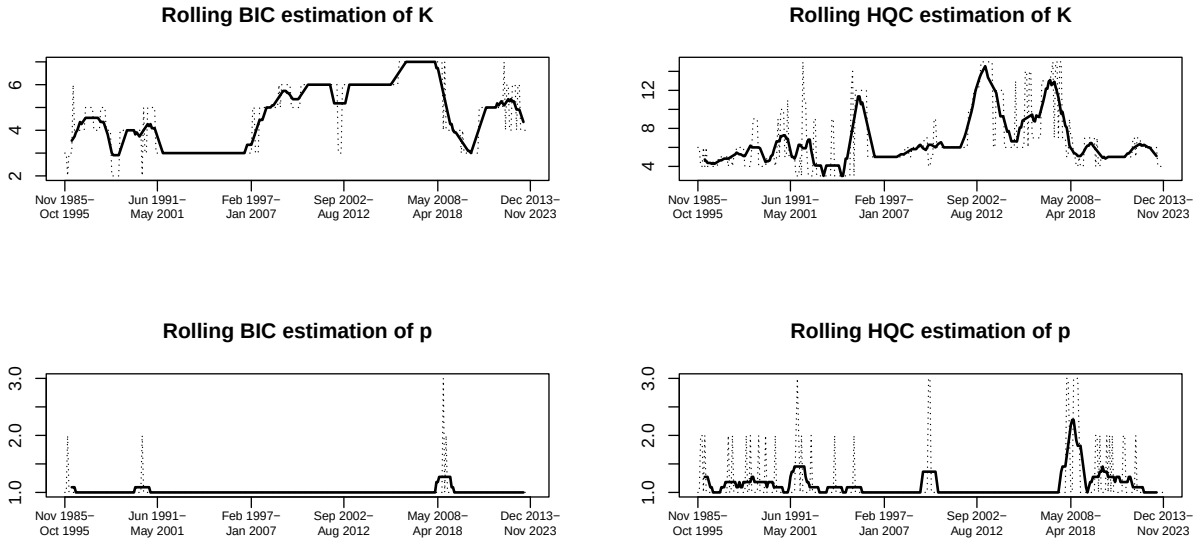
```
 legend("top", c('Level', 'Slope', 'Curvature'),
        lty=c(1,2,4), lwd=c(4,4,3), seg.len=3, cex=1.2,
        pt.cex=2, horiz=TRUE, bty="n")
}

# Plot loadings function
plot_loadings = function(loadings, title, signs=c(1,1,-1)) {
 plot(FDAdata$workinggrid, signs[1]*loadings[,1],
      ylim=c(-0.15,0.15), type='l', lty=1, lwd=4,
      xlab='Maturity (months)', ylab='', main=title,
      cex.lab=1.3, cex.axis=1.3, cex.main=2)
 lines(FDAdata$workinggrid, signs[2]*loadings[,2], lty=2, lwd=4)
 lines(FDAdata$workinggrid, signs[3]*loadings[,3], lty=4, lwd=3)

 psi_nums = (nrow(loadings)-2):nrow(loadings)
 legend("top", parse(text=sprintf("hat(psi)[%d]", psi_nums)),
        lty=c(1,2,4), lwd=c(4,4,3), pt.cex=2, cex=1.2,
        seg.len=3, horiz=TRUE, bty="n")
}

# Create all plots
figure.DNS(maturity.length=360)
FDAdata = readRDS("./data/LW360-fda.rds")
plot_loadings(FDAdata$eigenfunctions[,1:3], "First three estimated loadings", c(1,1,-1))
plot_loadings(FDAdata$eigenfunctions[,4:6], "Fourth through sixth estimated loadings", c(1,-1
```



Replicate Figure 7

Rolling Estimation of K and p for the yield curve data

```r
# Load and prep data
K.all = readRDS("./yield-results/rollingK-w120.rds")
time_points = c(1985.833, 1991.417, 1997.083, 2002.667, 2008.333, 2013.917)
time_labels = c(
 "Nov 1985-\nOct 1995", "Jun 1991-\nMay 2001", "Feb 1997-\nJan 2007",
 "Sep 2002-\nAug 2012", "May 2008-\nApr 2018", "Dec 2013-\nNov 2023"
)

# Create filtered versions
get_filtered = function(x) na.omit(stats::filter(x, rep(1/11, 11), sides = 2))
filters = lapply(1:4, function(i) get_filtered(K.all[,i]))

# Plotting function
plot_rolling = function(data, filtered, title) {
 plot(data, xaxt="n", col="black", lty=3, xlab="", ylab="", main=title)
 lines(filtered, lwd=2)
 axis(1, at=time_points, labels=time_labels, las=0, cex.axis=0.7)
}

# Create all plots
par(mfrow=c(2,2))
plot_rolling(K.all[,1], filters[[1]], "Rolling BIC estimation of K")
plot_rolling(K.all[,2], filters[[2]], "Rolling HQC estimation of K")
plot_rolling(K.all[,3], filters[[3]], "Rolling BIC estimation of p")
plot_rolling(K.all[,4], filters[[4]], "Rolling HQC estimation of p")
```



Replication of Table 4

Rolling out-of-sample MSFEs for the yield curve data

```
# Helper functions
get.MSEs = function(results) {
 indices = list(short=1:12, med=13:24, long=25:120)
 sapply(indices, function(i) colMeans(results[i, 1:12])) |> t()
}

# Function to process rolling windows
process_window = function(window_size) {
 relMSEs = matrix(ncol=11, nrow=12,
   dimnames=list(
     paste(rep(c("short", "med", "long"), 4),
           rep(c("1step", "3step", "6step", "12step"), each=3)),
     c("DNS.VAR1", "DNS.VAR2", "BIC.ols", "HQC.ols", "FPC.ols",
       "BIC.lasso", "BIC.ridge", "HQC.lasso", "HQC.ridge", "FPC.lasso", "FPC.ridge")
   ))

 # Calculate relative MSEs for each step
 for(step in c(1,3,6,12)) {
   results = readRDS(sprintf("./CheResults/YieldRolling%d-%dstep.rds", window_size, step))
   rel.MSEs = get.MSEs(results)[,2:12] / get.MSEs(results)[,1]
   idx = ((step == 1) * 0 + (step == 3) * 3 + (step == 6) * 6 + (step == 12) * 9)
   relMSEs[(idx+1):(idx+3),] = round(rel.MSEs, 3)
 }

 # Select columns and save final table
 tab = relMSEs[,c(3,4,5,6,8,10,1,2)]
 saveRDS(tab, sprintf("tables/table4-%d.rds", window_size))

 # Create and save LaTeX table
 latex_table = knitr::kable(tab, format="latex",
   caption=sprintf("Out-of-sample MSFEs with rolling window of %d months", window_size),
   booktabs=TRUE,
   label=sprintf("tbl:table4-%d", window_size))

 writeLines(c(
   "\\begin{table}[htbp]",
   "\\centering",
   latex_table,
```

```
   sprintf("\\label{tbl:table4-%d}", window_size),
   "\\end{table}"
 ), sprintf("tables/table4-%d.tex", window_size))

 return(tab)
}

# Process both windows and display tables
tab1 = process_window(120)
tab2 = process_window(240)
knitr::kable(tab1)
```

	BIC.ols	HQC.ols	FPC.ols	BIC.lasso	HQC.lasso	FPC.lasso	DNS.VAR1	DNS.VAR2
short 1step	0.918	0.903	1.332	0.983	0.994	1.103	1.701	1.710
med 1step	1.213	1.185	1.748	1.215	1.208	1.206	1.758	1.672
long 1step	1.159	1.190	1.836	1.120	1.157	1.181	1.327	1.375
short 3step	0.915	0.936	1.317	0.882	0.876	0.946	1.113	1.076
med 3step	1.255	1.260	1.671	1.039	1.010	1.035	1.337	1.284
long 3step	1.362	1.416	1.872	1.027	1.031	1.049	1.265	1.359
short 6step	1.120	1.127	1.569	0.887	0.899	0.934	1.067	1.077
med 6step	1.470	1.485	2.009	0.997	0.978	0.988	1.284	1.295
long 6step	1.615	1.691	2.337	0.992	0.996	1.006	1.296	1.424
short 12step	2.627	2.626	4.142	0.922	0.928	0.934	1.092	1.086
med 12step	3.194	3.219	5.154	0.998	0.975	0.970	1.256	1.248
long 12step	3.291	3.406	5.433	0.999	0.990	1.001	1.369	1.478

```
knitr::kable(tab2)
```

	BIC.ols	HQC.ols	FPC.ols	BIC.lasso	HQC.lasso	FPC.lasso	DNS.VAR1	DNS.VAR2
short 1step	0.897	0.858	1.161	0.986	0.939	1.081	1.805	1.803
med 1step	1.039	0.945	1.280	0.992	0.934	1.176	2.033	1.833
long 1step	1.090	1.090	1.373	1.047	1.028	1.102	1.295	1.312
short 3step	0.803	0.816	0.982	0.915	0.914	0.923	1.038	0.939
med 3step	1.000	1.022	1.220	0.951	0.942	1.008	1.302	1.156
long 3step	1.178	1.195	1.393	0.988	0.989	1.020	1.158	1.179
short 6step	0.797	0.899	1.019	0.922	0.924	0.937	0.936	0.883
med 6step	1.028	1.172	1.396	0.957	0.955	0.993	1.156	1.091

	BIC.ols	HQC.ols	FPC.ols	BIC.lasso	HQC.lasso	FPC.lasso	DNS.VAR1	DNS.VAR2
long 6step	1.324	1.403	1.776	0.986	0.989	1.013	1.162	1.211
short 12step	0.835	0.978	1.173	0.945	0.943	0.944	0.907	0.878
med 12step	1.037	1.227	1.665	0.966	0.950	0.970	1.080	1.050
long 12step	1.430	1.564	2.344	0.991	0.986	0.998	1.183	1.233