Box 1 - Example DFA workflow

Model specification

For this example we will assume that the observations at time t ($\mathbf{y}_t$) arise from a multivariate normal distribution wherein the mean vector is the sum of a vector of intercepts ($\boldsymbol{\alpha}$) and latent factors ($\mathbf{L}\boldsymbol{\gamma}_t$), and the covariance matrix $\boldsymbol{\Phi}$ is diagonal and unequal. Eqn (1) then becomes:

$$\mathbf{y}_t \sim \text{MVN}(\boldsymbol{\alpha} + \mathbf{L}\boldsymbol{\gamma}_t, \boldsymbol{\Phi}).$$

The latent factors are themselves a multivariate random walk with normal errors. Here we assume $\mathbf{Q}$ equals the identity matrix $\mathbf{I}$, such that Eqn (3) is

$$\boldsymbol{\gamma}_t \sim \text{MVN}(\boldsymbol{\gamma}_{t-1}, \mathbf{Q}).$$

Simulated data

Here are some simulated data for 5 time series with 30 measurements each based upon 2 latent factors. The observation variances are also unequal. We will set the covariance of the process errors $\mathbf{Q}$ to the identity matrix $\mathbf{I}$. Note that we will also subtract the mean of the data so as to aid in parameter estimation (see below).

```
set.seed(123)
## matrix dims
nn <- 5; mm <- 2; tt <- 30
## zero-mean latent factors
gamma1 <- cumsum(rnorm(tt)); gamma1 <- gamma1 - mean(gamma1)
gamma2 <- cumsum(rnorm(tt)); gamma2 <- gamma2 - mean(gamma2)
## factor loadings
LL <- matrix(c(0.2, 0.9,
               0.4, 0.7,
               0.6, 0.5,
               0.8, 0.3,
               1.0, 0.1),
             nn, mm, byrow=TRUE)
## covariance matrix for observation errors
Phi <- diag(seq(1,5)/5)
## observation errors
vv <- t(MASS::mvrnorm(tt, matrix(0,nn,1), Phi))
## simulated data
yy <- LL %*% rbind(gamma1,gamma2) + vv
## zero-mean data
yy <- t(scale(t(yy), scale=FALSE))
```

Fitting the model

There are several options for fitting factor models, but for this example we make use of the **MARSS** package (Holmes et al. 2012) for **R** to estimate the parameters in the DFA model. The notation of

Holmes et al. is somewhat different than ours, however, so we must write out our DFA model in a form based on the following specification of a state-space model:

$$\mathbf{y}_t \sim \text{MVN}(\mathbf{a} + \mathbf{Z}\mathbf{x}_t, \mathbf{R})$$
$$\mathbf{x}_t \sim \text{MVN}(\mathbf{u} + \mathbf{B}\mathbf{x}_{t-1}, \mathbf{Q}).$$

Thus, for our DFA model we want $\mathbf{a} = \boldsymbol{\alpha}$, $\mathbf{Z} = \mathbf{L}$ $\mathbf{x}_t = \boldsymbol{\gamma}_t$, $\mathbf{R} = \boldsymbol{\Phi}$, $\mathbf{u} = \mathbf{0}$, and $\mathbf{B} = \mathbf{I}$. Furthermore, because we substracted the mean of each time series, we will set $\mathbf{a} = \mathbf{0}$. Note that this will not affect our estimates of the factors and their loadings.

The MARSS function makes use of list matrices, which allow one to mix character and numeric classes within the same matrix. Any entries of class character are interpreted as parameters to be estimated; any numeric entries are fixed at their value.

```
library(MARSS)
## model list
mod_list <- list(
  ## observation eqn
  A = matrix(0,nn,1),         # zero vector
  Z = matrix(list(0),nn,mm), # all 0's for now
  R = matrix(list(0),nn,nn), # all 0's for now
  ## state eqn
  U = matrix(0,mm,1),         # zero vector
  B = diag(mm),               # identity matrix
  Q = diag(mm)                # identity matrix
)
## specify observation parameters
## upper right corner of Z stays zero
for(cc in 1:mm) {
  for(rr in cc:nn) {
    mod_list$Z[rr,cc] <- paste0(rr,cc)
  }
}
## R is diagonal and unequal
diag(mod_list$R) <- paste0(seq(nn),seq(nn))
## fit the model
dfa <- MARSS(yy, model=mod_list, inits=list(x0=matrix(0,2,1)))
```

Factor rotations

For this example we will use a varimax rotation of the factors and loadings, where we seek an $m \times m$ non-singular rotation matrix $\mathbf{H}$, such that the following state-space forms of our DFA model are equivalent ($\mathbf{v}_t$ and $\mathbf{w}_t$ are vectors of observation and process errors, respectively):

$$\mathbf{y}_t = \mathbf{L}\boldsymbol{\gamma}_t + \boldsymbol{\alpha} + \mathbf{v}_t \Leftrightarrow \mathbf{y}_t = \mathbf{L}\mathbf{H}^{-1}\boldsymbol{\gamma}_t + \boldsymbol{\alpha} + \mathbf{v}_t$$
$$\boldsymbol{\gamma}_t = \boldsymbol{\gamma}_{t-1} + \mathbf{w}_t \Leftrightarrow \mathbf{H}\boldsymbol{\gamma}_t = \mathbf{H}\boldsymbol{\gamma}_{t-1} + \mathbf{H}\mathbf{w}_t.$$

This is easily done in **R** via the varimax function:

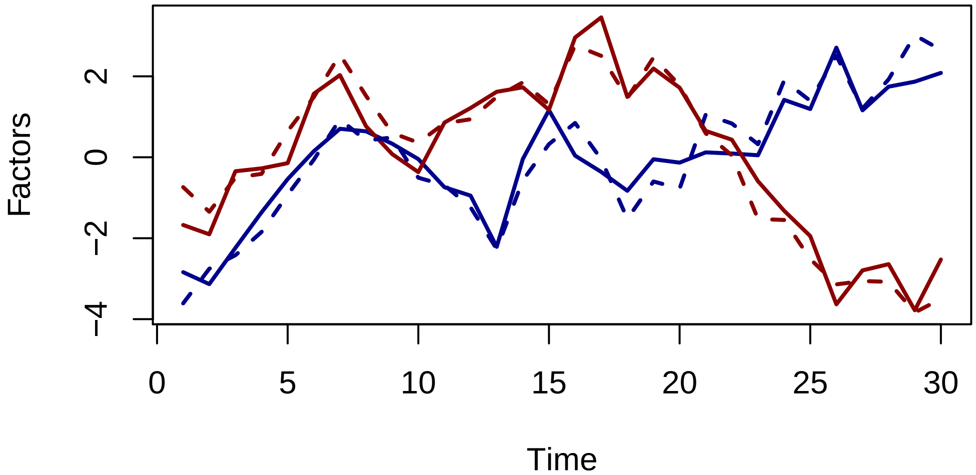
```
## loadings matrix
L_hat <- coef(dfa, type="matrix")$Z
```

```
## inverse of the rotation matrix
H_inv <- varimax(L_hat)$rotmat
## rotate loadings matrix
L_rot <- L_hat %*% H_inv
## rotate factors
gamma_rot <- solve(H_inv) %*% dfa$states
## model fits
yhat <- L_rot %*% gamma_rot
```

Estimated parameters

Here are plots of the true (solid) and estimated factors (dashed). Note that **MARSS** has no way of ordering the factors, so care should be used when examining them. In this case, the estimated factors were interchanged.

```
matplot(cbind(gamma2,gamma1,t(gamma_rot)), type="l", lty=c(1,1,2,2), lwd=2,
        col=c("darkblue", "darkred"), ylab="Factors", xlab="Time")
```



Here is a comparison of the true factor loadings and the estimates in $\hat{\mathbf{L}}$.

```
## true values
LL
```

```
##      [,1] [,2]
## [1,]  0.2  0.9
## [2,]  0.4  0.7
## [3,]  0.6  0.5
## [4,]  0.8  0.3
## [5,]  1.0  0.1
```

```
## estimates
round(L_rot[,2:1],2)
```

```
##      [,1] [,2]
## [1,] 0.27 0.86
## [2,] 0.44 0.63
## [3,] 0.64 0.47
## [4,] 0.78 0.28
```

```
## [5,] 0.91 0.34
```

It looks like the estimated loadings are generally pretty close to the true values, with the exception of the loading of factor 2 on data series 5.

Model fits

Here is a plot of the simulated data (numbers) and the fits (lines) from the estimated DFA model.

```
matplot(t(yy), type="p", cex=0.8, col=rainbow(nn,start=0.55, end=0.85),
        ylab="Observations", xlab="Time")
matlines(t(yhat), type="l", lty=1, lwd=2, col=rainbow(nn,start=0.55, end=0.85))
```