Ch 7: Fit state-space models with TVPs in dynr

Sy-Miin Chow

June 5, 2021

Contents

```
#Start fresh, set display options and load required libraries
rm(list=ls())
set.seed(12345678)
options(scipen = 999, digits=5)
require(mvtnorm)
require(dynr)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
require(xtable)
require(ggplot2)
```

Background

Introduction

This example mirrors the illustrative simulated example 7.5.1 in Chapter 7 of Gates, Chow, & Molenaar. The data generation model features two latent factors, each identified with 3 manifest indicators. The two latent factors conform to a common trend (changing local level or set-point) that follows the dual change score model (McArdle, & Hamagami, 2001), which includes as a special form sigmoid-shaped curves with individual differences in initial level and final asymptote. For this illustrative example, we generate data from $n = 10$ participants, each with 30 time points. To make estimation feasible given the relatively small T available from each participant, all parameters, except for the time-varying ones, are assumed and constrained to be invariant across individuals. Compared to the simulation example used in 7.5.1, substantially fewer participants and slightly larger T were used to reduce computational time.

We will illustrate how to use four models to represent the trend as a time-varying parameter (TVP) using an R package, dynr. The models include the:

(1) random walk model;

(2) integrated random walk model (with fixed and correctly specified initial condition or IC);

(3) integrated random walk model with freely estimated IC means and covariance matrix;

(4) smoothed integrated random walk model; and

(5) model ignoring time trends (TVPs constrained as invariant).

For first-time users of dynr

The dynr package compiles C code in response to user input, so more setup is required for the dynr package than for many others, including installation of the GSL library and C compiler. Steps for installation and automated installer for Windows can be downloaded here: https://dynrr.github.io/predownload.html

Reference

Gates, K., Chow, S-M., & Molenaar, P.C.M. Analysis of Intraindividual Variation. Systems Approaches to Human Process Analysis. New York, NY: Taylor & Francis.

McArdle, J. J., & Hamagami, F. (2001). Latent difference score structural models for linear dynamic analysis with incomplete longitudinal data. In L. Collins & A. Sayer (Eds.), New methods for the analysis of change (p. 139-175). Washington, DC: American Psychological Association.

Ou, L., Hunter, M. D., & Chow, S.-M. (2019). What's for dynr: A Package for Linear and Nonlinear Dynamic Modeling in R. The R Journal. https://journal.r-project.org/archive/2019/RJ-2019-012/index.html

Dynr development team. Dynamic modeling in R (dynr) website. https://dynrr.github.io/

Dynr development team. User Installation Instructions Manual for dynr.

https://cran.r-project.org/web/packages/dynr/vignettes/InstallationForUsers.pdf

Simulated Data Generation

Initial set-up

Here we define the number of time points, participants, true parameter values, other data simulation settings, and various place holders.

```r
time      <- 30 #Total number of time points
npad      <- 0 #Occasions to throw out to wash away the effects of initial condition
np        <- 10 #Total number of participants
ne        <- 4 #Number of latent variables
ny        <- 6 #Number of manifest variables
psi       <- matrix(c(2, .5, 0,0,  # Process noise variance-covariance matrix
                      .5, 1.5,0,0,
                      0,0,0,0,
                      0,0,0,0),
                    ncol = ne, byrow = T)
```

```
#Factor loading matrix
lambda    <- matrix(c(1, 0,1,0,
                      1.5,0,1,0,
                      1, 0,1,0,
                      0, 1, 1,0,
                      0, .8,1,0,
                      0, 1,1,0),
                    ncol = ne, byrow = TRUE)
theta     <- diag(.5, ncol = ny, nrow = ny)  # Measurement error variances.
beta      <- matrix(c(0.8, 0,0,0,    # Lagged directed relations among latent variables.
                      -.2, .7,0,0,
                      0,0,.8,1,
                      0,0,0,1),
                    ncol = ne, byrow = TRUE)
a0        <- c(0,0,1,5)
P0        <- matrix(c(1,0,0,0,
                      0,1,0,0,
                      0,0,.5,0,
                      0,0,0,.2),
                    ncol=ne,byrow=TRUE) #Initial covariance matrix for the trend elements
yall <- matrix(0,nrow = time*np, ncol = ny)
etaall <- matrix(0,nrow = time*np, ncol = ne)
```

Data generation

Data generation starts here. The observed data are written out to a file called PFALDS.txt.

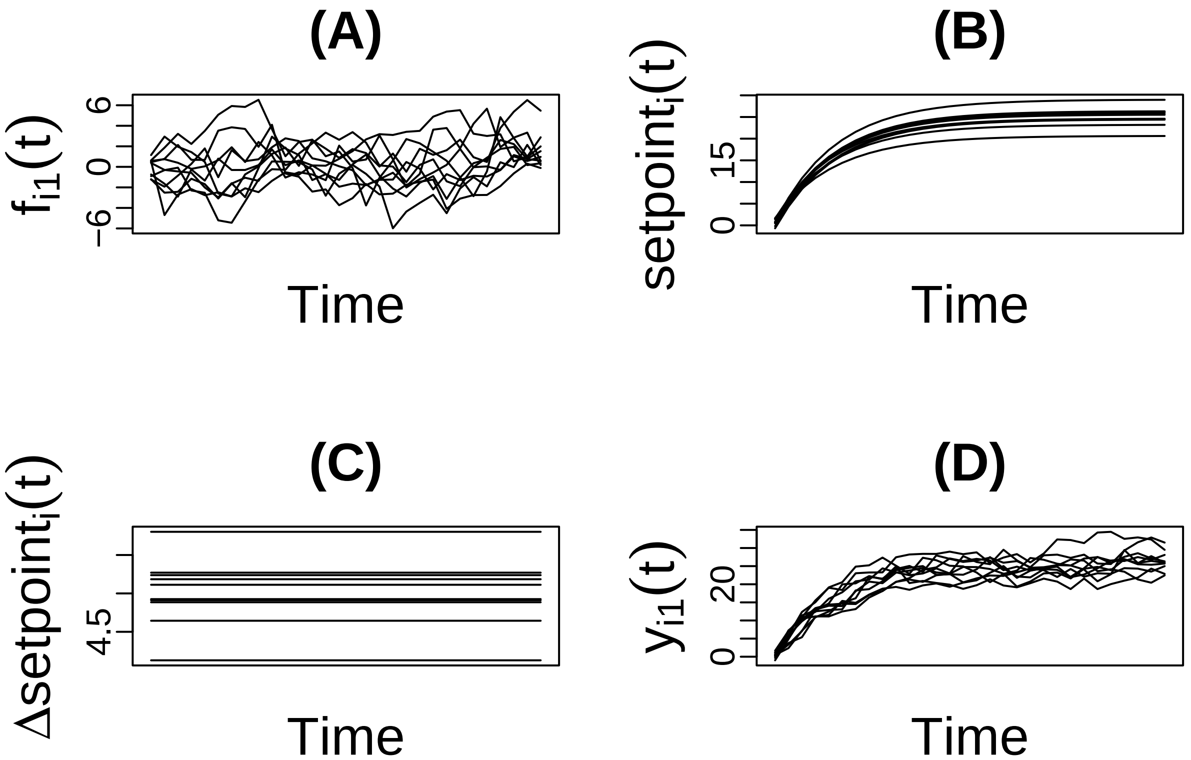
```
for (p in 1:np){
# Set up matrix for contemporaneous variables.
  etaC      <- matrix(0, nrow = ne, ncol = time + npad)
  etaC[,1]  <- a0 + chol(P0)%*%rnorm(ne)

# Latent variable residuals.
  zeta      <- mvtnorm::rmvnorm(time+npad, mean = c(0, 0,0,0), sigma = psi)

# Measurement errors.
  epsilon   <- rmvnorm(time+npad, mean = c(0, 0, 0, 0, 0, 0), sigma = theta)

# Generate latent factor scores
  for (i in 2:(time+npad)){
    etaC[ ,i]   <- beta %*% etaC[ ,i-1] + zeta[i, ]
  }#End of loop over time

  eta <- t(etaC[,(npad+1):(npad+time)])

  # generate observed series
  y   <- matrix(0, nrow = time, ncol = ny)
  for (i in 1:nrow(y)){
    y[i, ] <- lambda %*% eta[i, ] + epsilon[i, ]
  }
  yall[(1+(p-1)*time):(p*time),] = y
  etaall[(1+(p-1)*time):(p*time),] = eta
} #End of p loop
```

```r
yall = cbind(rep(1:np,each=time),rep(1:time,np),yall)
etaall = cbind(rep(1:np,each=time),rep(1:time,np),etaall)
colnames(etaall) = c("ID","time",paste0("LV",1:ne))
etaall = data.frame(etaall)
colnames(yall) = c("ID","time",paste0("y",1:ny))
yall = data.frame(yall)
file0 = paste0('PFALDS.txt')
write.table(yall,file0,row.names=FALSE,col.names=FALSE,sep=",")
```

Plots of simulated data

This is the code for creating Figure 7.2(A)-(D).

```r
par(mfrow=c(2,2))
par(cex.axis=1.3,cex.lab=2,mgp=c(2.0,.5,0),cex.main=2)
interaction.plot(etaall$time,etaall$ID,etaall$LV1,legend=F,lty=1,
                 ylab=expression(f[i1](t)),xlab="Time",
                 main=("(A)"),xaxt = "n")

par(cex.axis=1.3,cex.lab=2,mgp=c(2.0,.5,0),cex.main=2)
interaction.plot(etaall$time,etaall$ID,etaall$LV3,legend=F,lty=1,
                 ylab=expression(setpoint[i](t)),xlab="Time",
                 main=("(B)"),xaxt = "n")

par(cex.axis=1.3,cex.lab=2,mgp=c(2.0,.5,0),cex.main=2)
interaction.plot(etaall$time,etaall$ID,etaall$LV4,legend=F,lty=1,
                 ylab=expression(paste(Delta,setpoint[i](t))),
                 xlab="Time",main=("(C)"),xaxt = "n")

par(cex.axis=1.3,cex.lab=2,mgp=c(2.0,.5,0),cex.main=2)
interaction.plot(yall$time,yall$ID,yall$y1,legend=F,lty=1,
                 ylab=expression(paste(y[i1](t))),xlab="Time",
                 main=("(D)"),xaxt = "n")
```



Fit State-Space Models with TVPs

Now we illustrate how to fit a series of state-space models with TVPs using dynr.

Read in data and set up data structure in dynr

```
# Read in data - normally you would start right here.
ch7 <- read.table(file0,header=FALSE,sep=",")
colnames(ch7) <- c("ID","Time",paste0("V",1:ny))

# Set up dynr Data
ch72 <- dynr.data(ch7, id="ID", time="Time",
                  observed=paste0("V",1:ny))
```

Model 1: Random walk model for the TVP

Now we prepare the dynr recipes for a two-factor model that follows a vector-autoregressive model of order 1, as subjected to the influence of a common trend (time-varying set-point) that follows a random walk model. This model is written as: $\vartheta_i(t) = \vartheta_i(t-1) + \zeta_{\vartheta i}(t)$

```
ne        <- 3 #Number of latent variables

#Define the dynamic model
dynamics <- prep.matrixDynamics(
  values.dyn=matrix(c(.6, -0.1, 0,
                      -.1, .5,0,
                      0,0,1), ncol=ne,byrow=TRUE),
  params.dyn=matrix(c('phi11', 'phi12', 'fixed',
                      'phi21', 'phi22', 'fixed',
                      rep('fixed',ne)), ncol=ne,byrow=TRUE),
  isContinuousTime=FALSE)

#Meausurement models
meas <- prep.measurement(
  values.load=matrix(c(1,0,1,
                       2,0,1,
                       1,0,1,
                       0,1,1,
                       0,2,1,
                       0,1,1),ncol=ne,byrow=TRUE), #Starting values for entries in Lambda
  params.load=matrix(c(rep('fixed',ne),
                       'lambda21',rep('fixed',ne-1),
                       'lambda31',rep('fixed',ne-1),
                       rep('fixed',ne),
                       'fixed','lambda52',rep('fixed',ne-2),
                       'fixed','lambda62',rep('fixed',ne-2)),
                     ncol=ne,byrow=TRUE), #Labels for fixed and freed parameters
  state.names=c("eta1","eta2","mu1"), #Labels for latent variables in eta(t)
  obs.names=paste0('V',1:ny) #Labels for observed variables in y(t)
)

#Initial condition means and covariance matrix. Here we free them up.
initial <- prep.initial(
  values.inistate=a0[1:3],
  params.inistate=c('m1','m2','m3'),
  values.inicov=P0[1:3,1:3],
  params.inicov=diag(c('v1','v2','v3')))


#Process and measurement noise covariance matrices
mdcov <- prep.noise(
  values.latent=matrix(c(1,-.2,0,
                         -.2,1,0,
                         0,0,.5),ncol=ne,byrow=TRUE),
  params.latent=matrix(c('psi_11','psi_12','fixed',
                         'psi_12','psi_22','fixed',
                         'fixed','fixed','psi_mu1'),ncol=ne,byrow=TRUE),
  values.observed=diag(rep(.5,ny),ny),
  params.observed=diag(paste0('var_e',1:ny),ny)
)

#Put recipes and data together to prepare the full model
model <- dynr.model(dynamics=dynamics, measurement=meas,
                    noise=mdcov, initial=initial, data=ch72,
                    outfile="LDS1.c")

# Cook it!
res <- dynr.cook(model,debug_flag=TRUE,verbose = FALSE)
```

```
## Optimization function called.
## Starting Hessian calculation ...
## Finished Hessian calculation.
## Original exit flag:  3
## Modified exit flag:  3
## Optimization terminated successfully: ftol_rel or ftol_abs was reached.
## Original fitted parameters:  0.78489 -0.20993 0.055648 0.80452 1.4398 0.95014
## 0.79571 0.95531 0.58535 0.070611 0.015071 1.0482 -0.7341 -0.6809 -0.62214
## -0.62568 -0.71599 -0.48133 -2.1718 -2.057 3.1333 -1.1225 -2.8542 -1.9177
##
## Transformed fitted parameters:  0.78489 -0.20993 0.055648 0.80452 1.4398
## 0.95014 0.79571 0.95531 1.7956 0.12679 1.0241 2.8524 0.47994 0.50616 0.53679
## 0.5349 0.48871 0.61796 -2.1718 -2.057 3.1333 0.32548 0.057602 0.14694
##
## Doing end processing
## Successful trial
## Total Time: 8.2411
## Backend Time: 7.5027
```

```
coef(res)
```

```
##     phi11     phi21     phi12     phi22  lambda21  lambda31  lambda52  lambda62
##  0.784894 -0.209933  0.055648  0.804523  1.439775  0.950141  0.795714  0.955313
##    psi_11    psi_12    psi_22   psi_mu1    var_e1    var_e2    var_e3    var_e4
##  1.795617  0.126791  1.024138  2.852428  0.479935  0.506163  0.536793  0.534899
##    var_e5    var_e6        m1        m2        m3        v1        v2        v3
##  0.488706  0.617959 -2.171836 -2.057034  3.133308  0.325479  0.057602  0.146942
```

```
summary(res)
```

```
## Coefficients:
##          Estimate Std. Error t value ci.lower ci.upper           Pr(>|t|)
## phi11      0.7849     0.0478   16.41   0.6911   0.8787 <0.0000000000000002 ***
## phi21     -0.2099     0.0495   -4.24  -0.3070  -0.1129 <0.0000000000000002 ***
## phi12      0.0556     0.0524    1.06  -0.0471   0.1584              0.145
## phi22      0.8045     0.0523   15.38   0.7020   0.9070 <0.0000000000000002 ***
## lambda21   1.4398     0.0634   22.71   1.3155   1.5640 <0.0000000000000002 ***
## lambda31   0.9501     0.0284   33.49   0.8945   1.0057 <0.0000000000000002 ***
## lambda52   0.7957     0.0323   24.63   0.7324   0.8590 <0.0000000000000002 ***
## lambda62   0.9553     0.0341   27.97   0.8884   1.0222 <0.0000000000000002 ***
## psi_11     1.7956     0.3078    5.83   1.1923   2.3989 <0.0000000000000002 ***
## psi_12     0.1268     0.1925    0.66  -0.2506   0.5042              0.255
## psi_22     1.0241     0.2162    4.74   0.6004   1.4479 <0.0000000000000002 ***
## psi_mu1    2.8524     0.3231    8.83   2.2191   3.4857 <0.0000000000000002 ***
## var_e1     0.4799     0.0576    8.33   0.3670   0.5929 <0.0000000000000002 ***
## var_e2     0.5062     0.1204    4.20   0.2702   0.7421 <0.0000000000000002 ***
## var_e3     0.5368     0.0600    8.94   0.4191   0.6545 <0.0000000000000002 ***
## var_e4     0.5349     0.0676    7.92   0.4025   0.6673 <0.0000000000000002 ***
## var_e5     0.4887     0.0587    8.32   0.3736   0.6038 <0.0000000000000002 ***
## var_e6     0.6180     0.0705    8.76   0.4798   0.7562 <0.0000000000000002 ***
## m1        -2.1718     0.7336   -2.96  -3.6097  -0.7339              0.002 **
## m2        -2.0570     0.8387   -2.45  -3.7009  -0.4131              0.007 **
## m3         3.1333     0.7617    4.11   1.6404   4.6263 <0.0000000000000002 ***
## v1         0.3255     0.2326    1.40  -0.1303   0.7813              0.081 .
## v2         0.0576     0.2433    0.24  -0.4193   0.5345              0.407
## v3         0.1469     0.1844    0.80  -0.2144   0.5083               0.213
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log-likelihood value at convergence = 5811.23
## AIC = 5859.23
## BIC = 5948.12
```

Model 2: Integrated random walk (IRW) Model

This model is written as:

$$\vartheta_i(t) = \vartheta_i(t-1) + \Delta_i(t-1)$$

$\Delta_i(t) = \Delta_i(t-1) + \zeta_{\Delta i}(t)$

```
ne        <- 4 #Number of latent variables
#Define the dynamic model
dynamics.IRW <- prep.matrixDynamics(
  values.dyn=matrix(c(.6, -0.1, 0,0,
                    -.1, .5,0,0,
                    0,0,1,1,
                    0,0,0,1), ncol=ne,byrow=TRUE),
  params.dyn=matrix(c('phi11', 'phi12', 'fixed','fixed',
                    'phi21', 'phi22', 'fixed','fixed',
                    rep('fixed',ne),
                    rep('fixed',ne)), ncol=ne,byrow=TRUE),
  isContinuousTime=FALSE)

#Measurement model
meas.IRW <- prep.measurement(
  values.load=matrix(c(1,0,1,0,
                     2,0,1,0,
                     1,0,1,0,
                     0,1,1,0,
                     0,2,1,0,
                     0,1,1,0),ncol=ne,byrow=TRUE), #Starting values for entries in Lambda
  params.load=matrix(c(rep('fixed',ne),
                     'lambda21',rep('fixed',ne-1),
                     'lambda31',rep('fixed',ne-1),
                     rep('fixed',ne),
                     'fixed','lambda52',rep('fixed',ne-2),
                     'fixed','lambda62',rep('fixed',ne-2)),
                   ncol=ne,byrow=TRUE), #Labels for fixed and freed parameters
  state.names=c("eta1","eta2","mu1","slope1"), #Labels for latent variables in eta(t)
  obs.names=paste0('V',1:ny) #Labels for observed variables in y(t)
)

#Initial condition. Here set to the known and correctly specified
#mean vector and covariance matrix
initial.IRW <- prep.initial(
  values.inistate=a0,
  params.inistate=c('fixed','fixed','fixed','fixed'),
  values.inicov=P0,
  params.inicov=matrix(rep('fixed',ne*ne),ncol=ne,byrow=TRUE))
```

```
#Process and measurement noise covariance matrices
mdcov.IRW <- prep.noise(
  values.latent=matrix(c(.5,-.2,0,0,
                         -.2,.5,0,0,
                         0,0,0,0,
                         0,0,0,.05),ncol=ne,byrow=TRUE),
  params.latent=matrix(c('psi_11','psi_12','fixed','fixed',
                         'psi_12','psi_22','fixed','fixed',
                         'fixed','fixed','fixed','fixed',
                         'fixed','fixed','fixed','psi_sl1'),ncol=ne,byrow=TRUE),
  values.observed=diag(rep(.5,ny),ny),
  params.observed=diag(paste0('var_e',1:ny),ny)
)

#Put recipes and data together to prepare the full model
model2 <- dynr.model(dynamics=dynamics.IRW, measurement=meas.IRW,
                     noise=mdcov.IRW, initial=initial.IRW, data=ch72,
                     outfile="LDS2.c")

#Cook it!
res2 <- dynr.cook(model2,debug_flag=TRUE,verbose = FALSE)
```

```
## Optimization function called.
## Starting Hessian calculation ...
## Finished Hessian calculation.
## Original exit flag:  3
## Modified exit flag:  3
## Optimization terminated successfully: ftol_rel or ftol_abs was reached.
## Original fitted parameters:  0.71964 -0.26082 0.0020613 0.80594 1.5586 0.95163
## 0.80257 0.9527 0.62156 0.17648 0.2562 -0.97369 -0.71472 -1.4354 -0.58216
## -0.6207 -0.72016 -0.49822
##
## Transformed fitted parameters:  0.71964 -0.26082 0.0020613 0.80594 1.5586
## 0.95163 0.80257 0.9527 1.8618 0.32858 1.35 0.37769 0.48933 0.23803 0.55869
## 0.53757 0.48667 0.60761
##
## Doing end processing
## Successful trial
## Total Time: 5.5227
## Backend Time: 4.7651
```

```
coef(res2)
```

```
##      phi11      phi21      phi12      phi22   lambda21   lambda31   lambda52
##  0.7196414 -0.2608162  0.0020613  0.8059370  1.5585736  0.9516312  0.8025658
##   lambda62     psi_11     psi_12     psi_22    psi_sl1     var_e1     var_e2
##  0.9526972  1.8618389  0.3285807  1.3500004  0.3776864  0.4893308  0.2380288
##     var_e3     var_e4     var_e5     var_e6
##  0.5586898  0.5375686  0.4866747  0.6076126
```

```
summary(res2)
```

```
## Coefficients:
##          Estimate Std. Error t value ci.lower ci.upper            Pr(>|t|)
## phi11     0.71964    0.05254   13.70  0.61666  0.82263 <0.0000000000000002 ***
## phi21    -0.26082    0.05330   -4.89 -0.36528 -0.15635 <0.0000000000000002 ***
## phi12     0.00206    0.04358    0.05 -0.08335  0.08747               0.481
## phi22     0.80594    0.04066   19.82  0.72624  0.88563 <0.0000000000000002 ***
## lambda21  1.55857    0.05966   26.13  1.44165  1.67550 <0.0000000000000002 ***
## lambda31  0.95163    0.03128   30.42  0.89031  1.01295 <0.0000000000000002 ***
## lambda52  0.80257    0.02791   28.75  0.74785  0.85728 <0.0000000000000002 ***
## lambda62  0.95270    0.02972   32.06  0.89445  1.01095 <0.0000000000000002 ***
## psi_11    1.86184    0.22358    8.33  1.42363  2.30005 <0.0000000000000002 ***
## psi_12    0.32858    0.13148    2.50  0.07088  0.58628               0.007 **
## psi_22    1.35000    0.17041    7.92  1.01601  1.68399 <0.0000000000000002 ***
## psi_sl1   0.37769    0.06443    5.86  0.25141  0.50396 <0.0000000000000002 ***
## var_e1    0.48933    0.05324    9.19  0.38499  0.59368 <0.0000000000000002 ***
## var_e2    0.23803    0.08821    2.70  0.06515  0.41091               0.004 **
## var_e3    0.55869    0.05678    9.84  0.44740  0.66998 <0.0000000000000002 ***
## var_e4    0.53757    0.06807    7.90  0.40414  0.67099 <0.0000000000000002 ***
## var_e5    0.48667    0.05426    8.97  0.38033  0.59302 <0.0000000000000002 ***
## var_e6    0.60761    0.06929    8.77  0.47181  0.74341 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log-likelihood value at convergence = 5629.43
## AIC = 5665.43
## BIC = 5732.10
```

Model 3: IRW model with empirically determined initial conditions (ICs)

Here we use the empirical means and variances from the first time point and the difference between the second and first time point to set the initial conditions for the local level and slope in the IRW model. This is in contrast to Model 2, in which these IC-related parameters are freely estimated. The code model for this model is essentially identical to that for Model 2:

$$\vartheta_i(t) = \vartheta_i(t-1) + \Delta_i(t-1)$$

$$\Delta_i(t) = \Delta_i(t-1) + \zeta_{\Delta i}(t)$$

, except for the IC specification using prep.initial.

```
ytemp <- yall[yall$time <= 2,]
m3<-mean(unlist(ytemp[ytemp$time==1,3:8]),na.rm=T)
m4<-mean(unlist(ytemp[ytemp$time==2,3:8]-ytemp[ytemp$time==1,3:8]),na.rm=T)
v3<-var(rowMeans(ytemp[ytemp$time==1,3:8],na.rm=TRUE),na.rm=TRUE)
v4<-var(rowMeans(ytemp[ytemp$time==2,3:8]- ytemp[ytemp$time==1,3:8],na.rm=TRUE),na.rm=TRUE)

initial.free <- prep.initial(
  values.inistate=c(0, 0,m3,m4),
  params.inistate=c('m1','m2','m3','fixed'),
  values.inicov=matrix(c(1,0,0,0,
                       0,1,0,0,
                       0,0,v3,0,
                       0,0,0,v4),ncol=ne,byrow=TRUE),
  params.inicov=diag(c('v1','v2','v3','fixed')))

#Put recipes and data together to prepare the full model
model3 <- dynr.model(dynamics=dynamics.IRW, measurement=meas.IRW,
                     noise=mdcov.IRW, initial=initial.free, data=ch72,
                     outfile="LDS3.c")
```

```r
#Cook it!
res3 <- dynr.cook(model3,debug_flag=TRUE,verbose = FALSE)
```

```
## Optimization function called.
## Starting Hessian calculation ...
## Finished Hessian calculation.
## Original exit flag:  3
## Modified exit flag:  3
## Optimization terminated successfully: ftol_rel or ftol_abs was reached.
## Original fitted parameters:  0.72055 -0.26666 0.0020091 0.79828 1.5392 0.95196
## 0.80306 0.95369 0.6401 0.18839 0.27051 -1.3219 -0.72243 -1.2282 -0.59388
## -0.62437 -0.71724 -0.48736 -0.93341 -0.60372 1.7927 -0.90919 -2.205 -2.5252
##
## Transformed fitted parameters:  0.72055 -0.26666 0.0020091 0.79828 1.5392
## 0.95196 0.80306 0.95369 1.8967 0.35731 1.378 0.26664 0.48557 0.29282 0.55218
## 0.5356 0.4881 0.61424 -0.93341 -0.60372 1.7927 0.40285 0.11025 0.08004
##
## Doing end processing
## Successful trial
## Total Time: 8.8308
## Backend Time: 8.0848
```

```r
coef(res3)
```

```
##      phi11      phi21      phi12      phi22   lambda21   lambda31   lambda52
##  0.7205470 -0.2666619  0.0020091  0.7982764  1.5392233  0.9519646  0.8030626
##   lambda62     psi_11     psi_12     psi_22    psi_sl1     var_e1     var_e2
##  0.9536929  1.8966773  0.3573119  1.3779506  0.2666406  0.4855703  0.2928234
##     var_e3     var_e4     var_e5     var_e6         m1         m2         m3
##  0.5521784  0.5355995  0.4880954  0.6142450 -0.9334063 -0.6037211  1.7926916
##         v1         v2         v3
##  0.4028521  0.1102466  0.0800402
```

```r
summary(res3)
```

```
## Coefficients:
##          Estimate Std. Error t value ci.lower ci.upper             Pr(>|t|)
## phi11     0.72055    0.05225   13.79  0.61814  0.82296 <0.0000000000000002 ***
## phi21    -0.26666    0.05333   -5.00 -0.37119 -0.16213 <0.0000000000000002 ***
## phi12     0.00201    0.04435    0.05 -0.08492  0.08893                0.482
## phi22     0.79828    0.04303   18.55  0.71394  0.88262 <0.0000000000000002 ***
## lambda21  1.53922    0.05949   25.87  1.42262  1.65583 <0.0000000000000002 ***
## lambda31  0.95196    0.03072   30.99  0.89176  1.01217 <0.0000000000000002 ***
## lambda52  0.80306    0.02814   28.54  0.74791  0.85822 <0.0000000000000002 ***
## lambda62  0.95369    0.02991   31.89  0.89507  1.01231 <0.0000000000000002 ***
## psi_11    1.89668    0.22607    8.39  1.45358  2.33978 <0.0000000000000002 ***
## psi_12    0.35731    0.13475    2.65  0.09321  0.62141                0.004 **
## psi_22    1.37795    0.17170    8.03  1.04142  1.71449 <0.0000000000000002 ***
## psi_sl1   0.26664    0.05598    4.76  0.15692  0.37636 <0.0000000000000002 ***
## var_e1    0.48557    0.05354    9.07  0.38064  0.59050 <0.0000000000000002 ***
## var_e2    0.29282    0.09311    3.15  0.11034  0.47531                0.001 ***
## var_e3    0.55218    0.05657    9.76  0.44131  0.66305 <0.0000000000000002 ***
## var_e4    0.53560    0.06718    7.97  0.40392  0.66728 <0.0000000000000002 ***
## var_e5    0.48810    0.05401    9.04  0.38224  0.59395 <0.0000000000000002 ***
## var_e6    0.61424    0.06932    8.86  0.47838  0.75011 <0.0000000000000002 ***
## m1       -0.93341    0.41804   -2.23 -1.75275 -0.11406              0.013 *
## m2       -0.60372    0.50412   -1.20 -1.59177  0.38433              0.116
## m3        1.79269    0.44239    4.05  0.92563  2.65976 <0.0000000000000002 ***
## v1        0.40285    0.23791    1.69 -0.06344  0.86914              0.046 *
## v2        0.11025    0.26877    0.41 -0.41654  0.63703              0.341
## v3        0.08004    0.17546    0.46 -0.26385  0.42393              0.324
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log-likelihood value at convergence = 5617.45
## AIC = 5665.45
## BIC = 5754.34
```

Model 4: Smoothed Integrated Random Walk (SIRW) Model

Note that this model is very similar to the IRW, with the exception that a parameter, α_1, is estimated. This would induce some smoothing effect on the time-varying set-point when the absolute value of alpha1 is < 1. This model is written as:

$$\vartheta_i(t) = \alpha\vartheta_i(t-1) + \Delta_i(t-1)$$

$$\Delta_i(t) = \Delta_i(t-1) + \zeta_{\Delta i}(t).$$

```r
ne        <- 4 #Number of latent variables
#Prepare dynr recipes
#Define the dynamic model
dynamics.SIRW <- prep.matrixDynamics(
  values.dyn=matrix(c(.6, -0.1, 0,0,
                      -.1, .5,0,0,
                      0,0,.9,1,
                      0,0,0,1), ncol=ne,byrow=TRUE),
  params.dyn=matrix(c('phi11', 'phi12', 'fixed','fixed',
                      'phi21', 'phi22', 'fixed','fixed',
                      rep('fixed',2),'alpha1','fixed',
                      rep('fixed',ne)), ncol=ne,byrow=TRUE),
  isContinuousTime=FALSE)

## Cook it!
#Put recipes and data together to prepare the full model
#We use fixed and correctly specified IC here. So all
#subrecipes from this model are identical to those from
#Model 2 (IRW with correctly specified IC), except for
#the dynamic model (dynamics.SIRW in this case).
model4 <- dynr.model(dynamics=dynamics.SIRW, measurement=meas.IRW,
                     noise=mdcov.IRW, initial=initial.IRW, data=ch72,
                     outfile="LDS4.c")
res4 <- dynr.cook(model4,debug_flag=TRUE,verbose = FALSE)
```

```
## Optimization function called.
## Starting Hessian calculation ...
## Finished Hessian calculation.
## Original exit flag:  3
## Modified exit flag:  3
## Optimization terminated successfully: ftol_rel or ftol_abs was reached.
## Original fitted parameters:  0.77309 -0.20972 0.012371 0.74757 0.80014 1.4588
## 0.95372 0.76261 0.94547 0.76666 0.22176 0.24831 -8.2009 -0.72786 -1.0625
## -0.61161 -0.64784 -0.71728 -0.50342
##
## Transformed fitted parameters:  0.77309 -0.20972 0.012371 0.74757 0.80014
## 1.4588 0.95372 0.76261 0.94547 2.1526 0.47734 1.3877 0.00027441 0.48294 0.34558
## 0.54248 0.52318 0.48808 0.60446
##
## Doing end processing
## Successful trial
## Total Time: 7.7303
## Backend Time: 7.0026
```

```
coef(res4)
```

```
##       phi11       phi21       phi12       phi22      alpha1    lambda21
##  0.77309168 -0.20972317  0.01237146  0.74757265  0.80013964  1.45880062
##    lambda31    lambda52    lambda62      psi_11      psi_12      psi_22
##  0.95372477  0.76260688  0.94546802  2.15255649  0.47734350  1.38770669
##     psi_sl1      var_e1      var_e2      var_e3      var_e4      var_e5
##  0.00027441  0.48294242  0.34558327  0.54247520  0.52317720  0.48807830
##      var_e6
##  0.60445773
```

```
summary(res4)
```

```
## Coefficients:
##           Estimate Std. Error t value  ci.lower  ci.upper           Pr(>|t|)
## phi11     0.773092   0.041737   18.52  0.691288  0.854896 <0.0000000000000002
## phi21    -0.209723   0.036143   -5.80 -0.280563 -0.138884 <0.0000000000000002
## phi12     0.012371   0.047433    0.26 -0.080596  0.105339               0.40
## phi22     0.747573   0.043923   17.02  0.661486  0.833660 <0.0000000000000002
## alpha1    0.800140   0.003868  206.87  0.792559  0.807721 <0.0000000000000002
## lambda21  1.458801   0.036305   40.18  1.387644  1.529958 <0.0000000000000002
## lambda31  0.953725   0.026146   36.48  0.902479  1.004970 <0.0000000000000002
## lambda52  0.762607   0.029445   25.90  0.704896  0.820318 <0.0000000000000002
## lambda62  0.945468   0.033119   28.55  0.880556  1.010380 <0.0000000000000002
## psi_11    2.152556   0.207737   10.36  1.745400  2.559713 <0.0000000000000002
## psi_12    0.477343   0.122469    3.90  0.237309  0.717378 <0.0000000000000002
## psi_22    1.387707   0.156593    8.86  1.080790  1.694624 <0.0000000000000002
## psi_sl1   0.000274   0.000892    0.31 -0.001473  0.002022               0.38
## var_e1    0.482942   0.052823    9.14  0.379411  0.586474 <0.0000000000000002
## var_e2    0.345583   0.079636    4.34  0.189500  0.501667 <0.0000000000000002
## var_e3    0.542475   0.054900    9.88  0.434873  0.650078 <0.0000000000000002
## var_e4    0.523177   0.068415    7.65  0.389087  0.657268 <0.0000000000000002
## var_e5    0.488078   0.052078    9.37  0.386007  0.590149 <0.0000000000000002
## var_e6    0.604458   0.069559    8.69  0.468124  0.740791 <0.0000000000000002
##
## phi11    ***
## phi21    ***
## phi12
## phi22    ***
## alpha1   ***
## lambda21 ***
## lambda31 ***
## lambda52 ***
## lambda62 ***
## psi_11   ***
## psi_12   ***
## psi_22   ***
## psi_sl1
## var_e1   ***
## var_e2   ***
## var_e3   ***
## var_e4   ***
## var_e5   ***
## var_e6   ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log-likelihood value at convergence = 5393.82
## AIC = 5431.82
## BIC = 5502.19
```

Model 5: Time-invariant trend

Here to specify a time-invariant model, we use Model 1, the random walk model. All subrecipes are identical to Model 1, except that the process noise variances for the TVPs are constrained to be zero.

```r
ne        <- 3 #Number of latent variables

#Process and measurement noise covariance matrices
mdcov.inv <- prep.noise(
  values.latent=matrix(c(.5,-.2,0,
                         -.2,.5,0,
                         0,0,0),ncol=ne,byrow=TRUE),
  params.latent=matrix(c('psi_11','psi_12','fixed',
                         'psi_12','psi_22','fixed',
                         'fixed','fixed','fixed'),ncol=ne,byrow=TRUE),
  values.observed=diag(rep(.5,ny),ny),
  params.observed=diag(paste0('var_e',1:ny),ny)
)

#Initial condition means and covariance matrix.
initial.inv <- prep.initial(
  values.inistate=a0[1:3],
  params.inistate=c('m1','m2','m3'),
  values.inicov=P0[1:3,1:3],
  params.inicov=diag(c('v1','v2','v3')))

model5 <- dynr.model(dynamics=dynamics, measurement=meas,
                     noise=mdcov.inv, initial=initial.inv, data=ch72,
                     outfile="LDS5.c")
# Cook it!
res5 <- dynr.cook(model5,debug_flag=TRUE,verbose = FALSE)
```

```
## Optimization function called.
## Starting Hessian calculation ...
## Finished Hessian calculation.
## Original exit flag:  3
## Modified exit flag:  3
## Optimization terminated successfully: ftol_rel or ftol_abs was reached.
## Original fitted parameters:  0.76838 -0.10598 0.04493 0.91146 1.035 0.99366
## 0.98901 0.99094 0.90149 0.23041 0.20286 -0.92601 0.42437 -0.55221 -0.43296
## -0.6764 -0.41482 -24.558 -24.138 25.152 -1.4599 -2.1434 -3.4799
##
## Transformed fitted parameters:  0.76838 -0.10598 0.04493 0.91146 1.035 0.99366
## 0.98901 0.99094 2.4633 0.56755 1.3557 0.39613 1.5286 0.57568 0.64859 0.50844
## 0.66046 -24.558 -24.138 25.152 0.23226 0.11726 0.03081
##
## Doing end processing
## Successful trial
## Total Time: 11.703
## Backend Time: 10.91
```

```
round(coef(res5),5)
```

```
##     phi11     phi21     phi12     phi22  lambda21  lambda31  lambda52  lambda62
##   0.76838  -0.10598   0.04493   0.91146   1.03499   0.99366   0.98901   0.99094
##    psi_11    psi_12    psi_22    var_e1    var_e2    var_e3    var_e4    var_e5
##   2.46328   0.56755   1.35567   0.39613   1.52863   0.57568   0.64859   0.50844
##    var_e6        m1        m2        m3        v1        v2        v3
##   0.66046 -24.55766 -24.13825  25.15225   0.23226   0.11726   0.03081
```

```
summary(res5)
```

```
## Coefficients:
##           Estimate Std. Error t value  ci.lower  ci.upper           Pr(>|t|)
## phi11      0.76838    0.03087   24.89   0.70788   0.82888 <0.0000000000000002
## phi21     -0.10598    0.02324   -4.56  -0.15154  -0.06042 <0.0000000000000002
## phi12      0.04493    0.03230    1.39  -0.01838   0.10824              0.083
## phi22      0.91146    0.02479   36.76   0.86287   0.96005 <0.0000000000000002
## lambda21   1.03499    0.01039   99.64   1.01464   1.05535 <0.0000000000000002
## lambda31   0.99366    0.00713  139.40   0.97969   1.00763 <0.0000000000000002
## lambda52   0.98901    0.00804  123.02   0.97325   1.00476 <0.0000000000000002
## lambda62   0.99094    0.00850  116.52   0.97427   1.00761 <0.0000000000000002
## psi_11     2.46328    0.23160   10.64   2.00935   2.91720 <0.0000000000000002
## psi_12     0.56755    0.13219    4.29   0.30847   0.82663 <0.0000000000000002
## psi_22     1.35567    0.14447    9.38   1.07252   1.63881 <0.0000000000000002
## var_e1     0.39613    0.06699    5.91   0.26483   0.52743 <0.0000000000000002
## var_e2     1.52863    0.14503   10.54   1.24437   1.81289 <0.0000000000000002
## var_e3     0.57568    0.07742    7.44   0.42394   0.72742 <0.0000000000000002
## var_e4     0.64859    0.07332    8.85   0.50487   0.79230 <0.0000000000000002
## var_e5     0.50844    0.06371    7.98   0.38358   0.63331 <0.0000000000000002
## var_e6     0.66046    0.07382    8.95   0.51578   0.80513 <0.0000000000000002
## m1       -24.55766    0.46098  -53.27 -25.46116 -23.65416 <0.0000000000000002
## m2       -24.13825    0.44140  -54.69 -25.00337 -23.27313 <0.0000000000000002
## m3        25.15225    0.40109   62.71  24.36612  25.93837 <0.0000000000000002
## v1         0.23226    0.21324    1.09  -0.18568   0.65019              0.139
## v2         0.11726    0.15418    0.76  -0.18493   0.41945              0.224
## v3         0.03081    0.08890    0.35  -0.14342   0.20504              0.365
##
## phi11    ***
## phi21    ***
## phi12    .
## phi22    ***
## lambda21 ***
## lambda31 ***
## lambda52 ***
## lambda62 ***
## psi_11   ***
## psi_12   ***
## psi_22   ***
## var_e1   ***
## var_e2   ***
## var_e3   ***
## var_e4   ***
## var_e5   ***
## var_e6   ***
## m1       ***
## m2       ***
## m3       ***
## v1
## v2
## v3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log-likelihood value at convergence = 5706.23
## AIC = 5752.23
## BIC = 5837.42
```

Compare Estimation Results

Here we show how estimation results returned by dynr's cooking can be compared with the true values used in the data generation. We use the R package, xtable, to compile table of results.

Extract estimates from different models and combine them

```
# Extract true values of the latent variables, including the time-varying set-point
true <- etaall[,c(3:5)]

# True values of all parameters that are reasonable to be compared
truepar <- c(.8,-.2,0,.7,1.5,1,.8,1,2,.5,1.5,rep(.5,ny))

# Extract the corresponding estimated parameter values from different models
allPar <- cbind(truepar,
coef(res)[!model$'param.names' %in% c(paste0('m',1:3),paste0('v',1:3),'psi_mu1')],
coef(res2)[!model2$'param.names' %in% c('psi_sl1')],
coef(res3)[!model3$'param.names' %in% c('psi_sl1',paste0('m',1:4),paste0('v',1:4))],
coef(res4)[!model4$'param.names' %in% c('psi_sl1','alpha1')],
coef(res5)[!model5$'param.names' %in% c(paste0('m',1:3),paste0('v',1:3),'psi_mu1')])

colnames(allPar) <- c("True","RW","IRW (fixed IC)","IRW (free IC)",
                   "SIRW","Invariant")
rownames(allPar ) <- c("$\\phi_{11}$","$\\phi_{21}$","$\\phi_{12}$","$\\phi_{22}$",
                       "$\\lambda_{21}$","$\\lambda_{31}$",
                       "$\\lambda_{52}$","$\\lambda_{62}$",
                       "$\\psi_{11}$","$\\psi_{12}$","$\\psi_{22}$",
                       "$\\Var(\\epsilon_{1})$","$\\Var(\\epsilon_{2})$",
                        "$\\Var(\\epsilon_{3})$","$\\Var(\\epsilon_{4})$",
                        "$\\Var(\\epsilon_{5})$","$\\Var(\\epsilon_{6})$")

sum = xtable(allPar,include.rownames=FALSE,
             caption="Parameter Estimates for Illustrative Example 1.",
             digits=rep(2,dim(allPar)[2]+1))
write(paste(print(sum,sanitize.text.function = function(x) x)),
      file = "demo1Est.txt")
```

```
## % latex table generated in R 4.1.1 by xtable 1.8-4 package
## % Fri Dec 31 10:56:04 2021
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrrrr}
##   \hline
##  & True & RW & IRW (fixed IC) & IRW (free IC) & SIRW & Invariant \\
##   \hline
## $\phi_{11}$ & 0.80 & 0.78 & 0.72 & 0.72 & 0.77 & 0.77 \\
##   $\phi_{21}$ & -0.20 & -0.21 & -0.26 & -0.27 & -0.21 & -0.11 \\
##   $\phi_{12}$ & 0.00 & 0.06 & 0.00 & 0.00 & 0.01 & 0.04 \\
##   $\phi_{22}$ & 0.70 & 0.80 & 0.81 & 0.80 & 0.75 & 0.91 \\
##   $\lambda_{21}$ & 1.50 & 1.44 & 1.56 & 1.54 & 1.46 & 1.03 \\
##   $\lambda_{31}$ & 1.00 & 0.95 & 0.95 & 0.95 & 0.95 & 0.99 \\
##   $\lambda_{52}$ & 0.80 & 0.80 & 0.80 & 0.80 & 0.76 & 0.99 \\
##   $\lambda_{62}$ & 1.00 & 0.96 & 0.95 & 0.95 & 0.95 & 0.99 \\
##   $\psi_{11}$ & 2.00 & 1.80 & 1.86 & 1.90 & 2.15 & 2.46 \\
##   $\psi_{12}$ & 0.50 & 0.13 & 0.33 & 0.36 & 0.48 & 0.57 \\
##   $\psi_{22}$ & 1.50 & 1.02 & 1.35 & 1.38 & 1.39 & 1.36 \\
##   $\Var(\epsilon_{1})$ & 0.50 & 0.48 & 0.49 & 0.49 & 0.48 & 0.40 \\
##   $\Var(\epsilon_{2})$ & 0.50 & 0.51 & 0.24 & 0.29 & 0.35 & 1.53 \\
##   $\Var(\epsilon_{3})$ & 0.50 & 0.54 & 0.56 & 0.55 & 0.54 & 0.58 \\
##   $\Var(\epsilon_{4})$ & 0.50 & 0.53 & 0.54 & 0.54 & 0.52 & 0.65 \\
##   $\Var(\epsilon_{5})$ & 0.50 & 0.49 & 0.49 & 0.49 & 0.49 & 0.51 \\
##   $\Var(\epsilon_{6})$ & 0.50 & 0.62 & 0.61 & 0.61 & 0.60 & 0.66 \\
##    \hline
## \end{tabular}
## \caption{Parameter Estimates for Illustrative Example 1.}
## \end{table}
```

```
# Extract smoothed latent variables estimates

#First check the dimension of the smoothed latent variable estimates
#stored in the cooked dynr object. This is of dimension ne x np*time.
dim(res@eta_smooth_final)
```

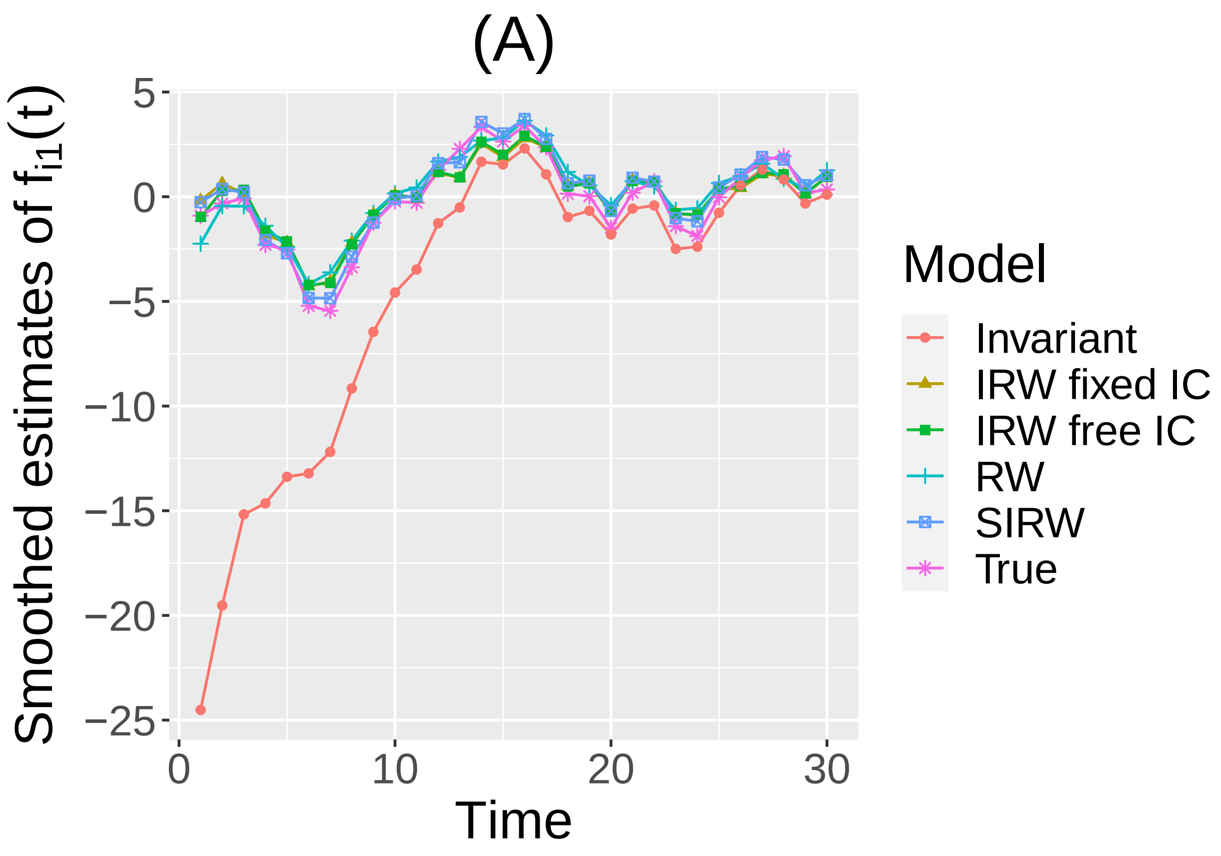
```
## [1]   3 300
```

```
# Column-bind estimates of latent variable scores from different models.
etaSmoothAll <- cbind(rep(1:time,np),
                      rep(1:np,each=time),
                      etaall[,c(3:5)],
                      t(res@eta_smooth_final[1:3,]),
                      t(res2@eta_smooth_final[c(1:3),]),
                      t(res3@eta_smooth_final[c(1:3),]),
                      t(res4@eta_smooth_final[c(1:3),]),
```

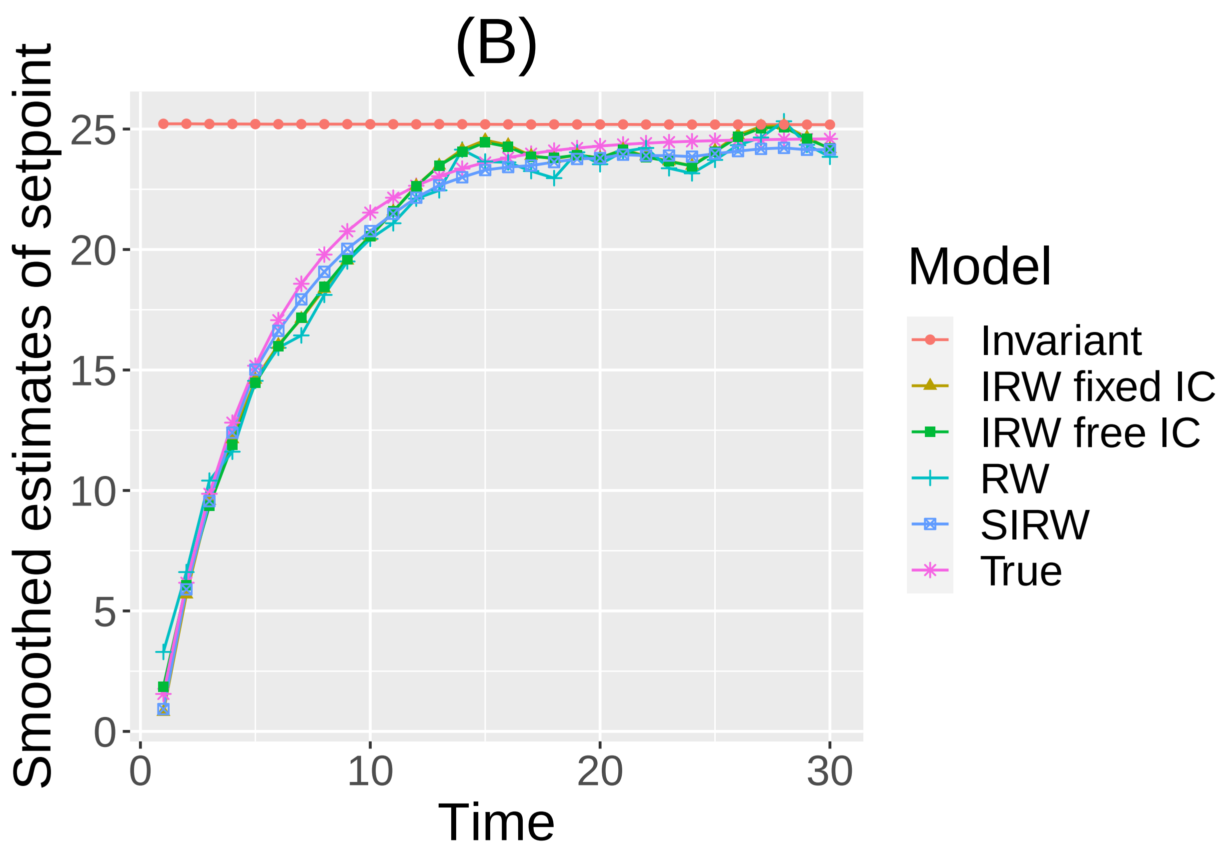
```r
                      t(res5@eta_smooth_final[1:3,]))

#Convert into a data frame and apply column names
etaSmoothAll <- data.frame(etaSmoothAll)
colnames(etaSmoothAll) <- c("Time","ID",
                            paste0("Model0LV",1:3),
                            paste0("Model1LV",1:3),
                            paste0("Model2LV",1:3),
                            paste0("Model3LV",1:3),
                            paste0("Model4LV",1:3),
                            paste0("Model5LV",1:3))


#Reshape from wide to long format
eta2 <- reshape(etaSmoothAll, direction='long',
                varying=3:dim(etaSmoothAll)[2],
                timevar='Model',
                times=c("True","RW","IRW fixed IC","IRW free IC",
                        "SIRW", "Invariant"),
                v.names=paste0('LV',1:3),
                idvar=c("Time","ID"))

#Compile a data frame of biases to make it easier to compare estimates
etabiasAll <- cbind(etaSmoothAll$Time,etaSmoothAll$ID,
  etaSmoothAll[,6:20] - do.call(cbind,
                                replicate(5, etaSmoothAll[,3:5],
                                          simplify=FALSE)))
#Convert into a data frame and apply column names
etabiasAll <- data.frame(etabiasAll)
colnames(etabiasAll) <- c("Time","ID",
                            paste0("Model1LV",1:3),
                            paste0("Model2LV",1:3),
                            paste0("Model3LV",1:3),
                            paste0("Model4LV",1:3),
                            paste0("Model5LV",1:3))


#Reshape from wide to long format
eta3 <- reshape(etabiasAll, direction='long',
                varying=3:dim(etabiasAll)[2],
                timevar='Model',
                times=c("RW","IRW fixed IC","IRW free IC",
                        "SIRW", "Invariant"),
                v.names=paste0('BiasLV',1:3),
                idvar=c("Time","ID"))
```

##Plot and summarize comparisons of results Code for this section can be used to replicate the results summarized in Tables 7.2-7.3, and Figures 7.3(A)-(D)

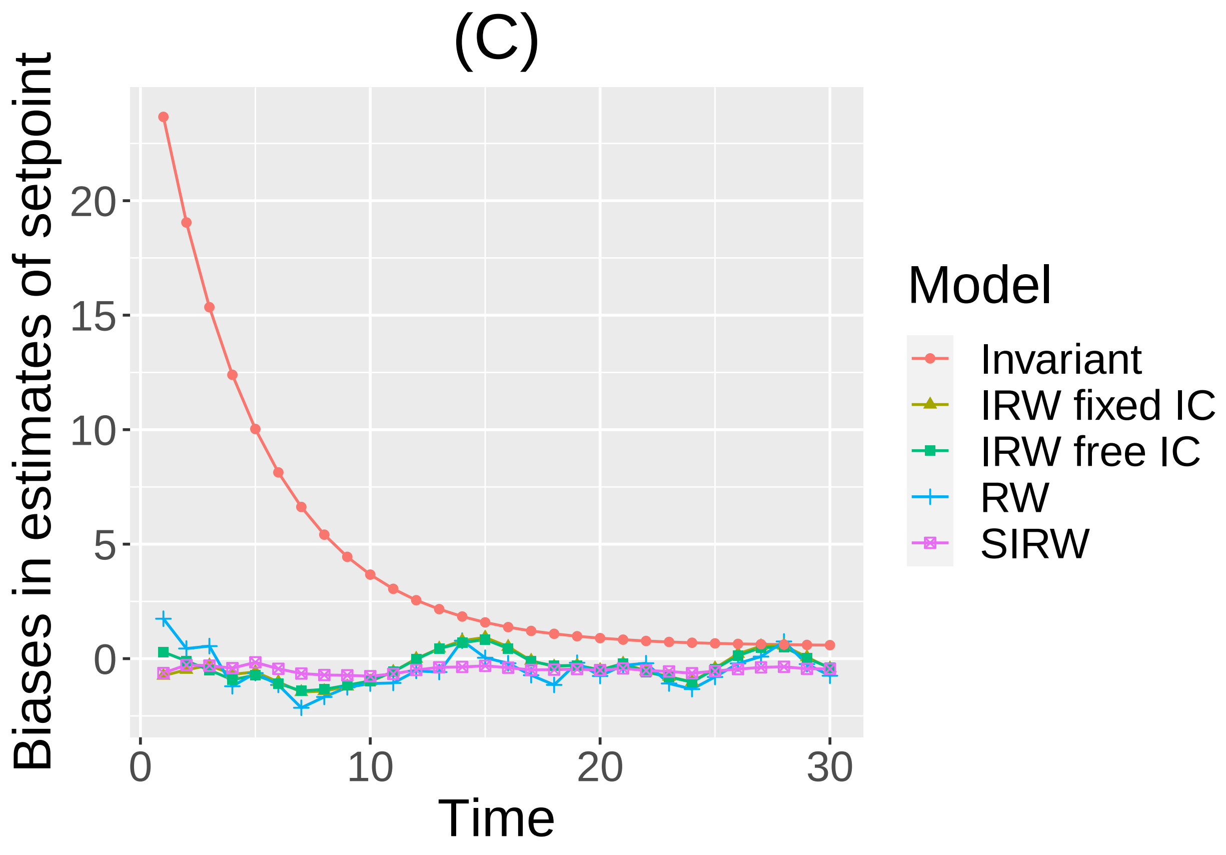
```r
#Use ggplot to compare the latent variables estimates - only one ID is used here.
eta.temp <- eta2[eta2$ID==10,]
eta.temp2 <- eta3[eta3$ID==10,]

ggplot(data = eta.temp, aes(x = Time, y = LV1, colour = Model, shape=Model)) +
  geom_line() + geom_point() + ggtitle("(A)")+
  ylab(expression(paste("Smoothed estimates of ",f[i1](t))))+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(text = element_text(size=20))
```



```
ggplot(data = eta.temp, aes(x = Time, y = LV3, colour = Model, shape=Model)) +
  geom_line() + geom_point() + ggtitle("(B)")+
  ylab("Smoothed estimates of setpoint")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(text = element_text(size=20))
```



```
ggplot(data = eta.temp2, aes(x = Time, y = BiasLV3, colour = Model, shape=Model)) +
  geom_line() + geom_point() + ggtitle("(C)")+
  ylab("Biases in estimates of setpoint")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(text = element_text(size=20))
```



```
#Calculate average biases for the latent variables
eta.bias <- data.frame(
  biasLV1 = colMeans(etaSmoothAll[,grep("LV1",colnames(etaSmoothAll))] - true[,1]),
  biasLV2 = colMeans(etaSmoothAll[,grep("LV2",colnames(etaSmoothAll))] - true[,2]),
  biasLV3 = colMeans(etaSmoothAll[,grep("LV3",colnames(etaSmoothAll))] - true[,3])
)
eta.bias <- eta.bias[2:6,]
row.names(eta.bias) <- c("RW","IRW fixed IC","IRW free IC","SIRW", "Invariant")

#Calculate biases in parameter estimates for a single replication
parEst.bias <- data.frame(
  RW = mean(coef(res)[!model$'param.names' %in%
       c(paste0('m',1:3),paste0('v',1:3),'psi_mu1')]-truepar),
  IRW_fixedIC = mean(coef(res2)[!model2$'param.names' %in%
                c('psi_sl1')]-truepar),
  IRW_freeIC = mean(coef(res3)[!model3$'param.names' %in%
               c('psi_sl1',paste0('m',1:4),paste0('v',1:4))]-truepar),
  SIRW = mean(coef(res4)[!model4$'param.names' %in%
         c('psi_sl1','alpha1')]-truepar),
  TimeInvariant = mean(coef(res5)[!model5$'param.names' %in%
                  c(paste0('m',1:3),paste0('v',1:3),'psi_mu1')]-truepar)
)

sum2 = xtable(eta.bias,include.rownames=FALSE,
             caption="Biases in Latent Variable Biases for Illustrative Example 1.",
             digits=rep(2,dim(eta.bias)[2]+1))
```

```
write(paste(print(sum2,sanitize.text.function = function(x) x)),
      file = "demo1Est.txt",append=TRUE)
```

```
## % latex table generated in R 4.1.1 by xtable 1.8-4 package
## % Fri Dec 31 10:56:04 2021
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrr}
##   \hline
##  & biasLV1 & biasLV2 & biasLV3 \\
##   \hline
## RW & 0.00 & -0.00 & 0.02 \\
##   IRW fixed IC & -0.05 & -0.08 & 0.08 \\
##   IRW free IC & -0.07 & -0.10 & 0.11 \\
##   SIRW & 0.03 & 0.03 & -0.01 \\
##   Invariant & -4.11 & -4.16 & 4.16 \\
##    \hline
## \end{tabular}
## \caption{Biases in Latent Variable Biases for Illustrative Example 1.}
## \end{table}
```

```
write(paste(print(parEst.bias,sanitize.text.function = function(x) x)),
      file = "demo1Est.txt",append=TRUE)
```

```
##          RW IRW_fixedIC IRW_freeIC       SIRW TimeInvariant
## 1 -0.053701   -0.035846  -0.029261 -0.0089274       0.10187
```

```
parEst.bias
```

```
##          RW IRW_fixedIC IRW_freeIC       SIRW TimeInvariant
## 1 -0.053701   -0.035846  -0.029261 -0.0089274       0.10187
```

```
eta.bias
```

```
##                  biasLV1    biasLV2    biasLV3
## RW             0.0044668 -0.0022691  0.0232456
## IRW fixed IC -0.0491182 -0.0774854  0.0836648
## IRW free IC  -0.0661933 -0.1013666  0.1061593
## SIRW           0.0299274  0.0280709 -0.0051651
## Invariant    -4.1086368 -4.1642291  4.1555131
```

```
sum2
```

```
## % latex table generated in R 4.1.1 by xtable 1.8-4 package
## % Fri Dec 31 10:56:04 2021
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrr}
##   \hline
##  & biasLV1 & biasLV2 & biasLV3 \\
##   \hline
## RW & 0.00 & -0.00 & 0.02 \\
##   IRW fixed IC & -0.05 & -0.08 & 0.08 \\
##   IRW free IC & -0.07 & -0.10 & 0.11 \\
##   SIRW & 0.03 & 0.03 & -0.01 \\
##   Invariant & -4.11 & -4.16 & 4.16 \\
##    \hline
## \end{tabular}
## \caption{Biases in Latent Variable Biases for Illustrative Example 1.}
## \end{table}
```