Demonstration of selected features

James T. Thorson

```r
library(dsem)
#> Loading required package: TMB
#> Loading required package: Matrix
#> Warning: package 'Matrix' was built under R version 4.3.1
```

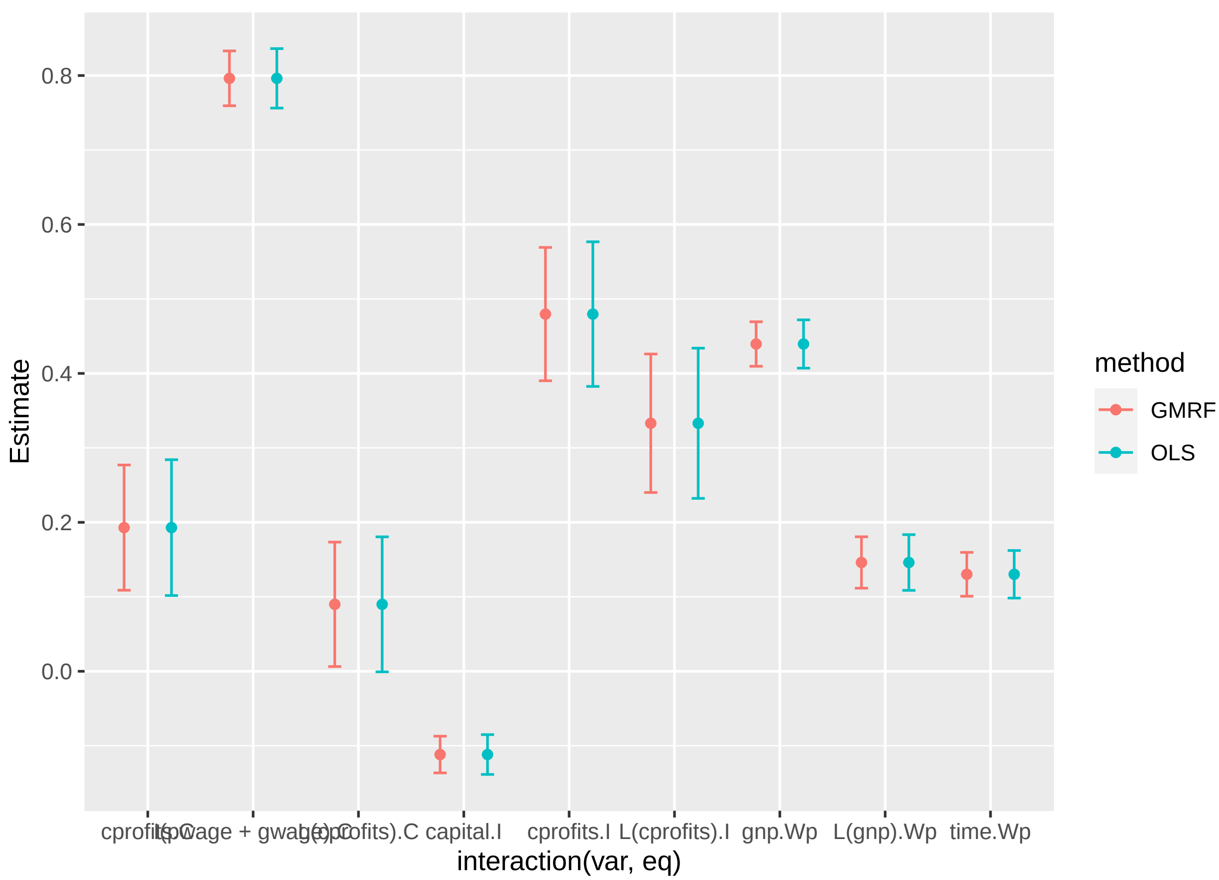
`dsem` is an R package for fitting dynamic structural equation models (DSEMs) with a simple user-interface and generic specification of simultaneous and lagged effects in a non-recursive structure. We here highlight a few features in particular.

Comparison with dynamic linear models

We first demonstrate that `dsem` gives identical results to `dynlm` for a well-known econometric model, the Klein-1 model.

```r
library(dynlm)
#> Warning: package 'dynlm' was built under R version 4.3.1
#> Warning: package 'zoo' was built under R version 4.3.1

data(KleinI, package="AER")
KleinI = ts(data.frame(KleinI, "time"=time(KleinI) - 1931))

# dynlm
fm_cons <- dynlm(consumption ~ cprofits + L(cprofits) + I(pwage + gwage), data = KleinI)
fm_inv <- dynlm(invest ~ cprofits + L(cprofits) + capital, data = KleinI)                #
fm_pwage <- dynlm(pwage ~ gnp + L(gnp) + time, data = KleinI)

# dsem
sem = "
  cprofits -> consumption, 0, a1
  cprofits -> consumption, -1, a2
  pwage -> consumption, 0, a3
  gwage -> consumption, 0, a3

  cprofits -> invest, 0, b1
  cprofits -> invest, -1, b2
  capital -> invest, 0, b3

  gnp -> pwage, 0, c2
  gnp -> pwage, -1, c3
  time -> pwage, 0, c1
"
tsdata = KleinI[,c("time","gnp","pwage","cprofits",'consumption',"gwage","invest","capital")]
tsdata[1,c('consumption','pwage','invest')] = NA
```

```r
fit = dsem( sem=sem, tsdata=tsdata, newtonsteps=0, quiet=TRUE )   #
#> 1 regions found.
#> Using 1 threads
#> 1 regions found.
#> Using 1 threads

# Compile
m1 = rbind( summary(fm_cons)$coef[-1,], summary(fm_inv)$coef[-1,], summary(fm_pwage)$coef[-1,] )[,1:2]
m2 = summary(fit$opt$SD)[1:9,]
m = rbind(
  data.frame("var"=rownames(m1),m1,"method"="OLS","eq"=rep(c("C","I","Wp"),each=3)),
  data.frame("var"=rownames(m1),m2,"method"="GMRF","eq"=rep(c("C","I","Wp"),each=3))
)
m = cbind(m, "lower"=m$Estimate-m$Std..Error, "upper"=m$Estimate+m$Std..Error )

# ggplot estimates
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 4.3.1
ggplot(data=m, aes(x=interaction(var,eq), y=Estimate, color=method)) +
  geom_point( position=position_dodge(0.9) ) +
  geom_errorbar( aes(ymax=as.numeric(upper),ymin=as.numeric(lower)),
                 width=0.25, position=position_dodge(0.9))  #
```



Results show that both packages provide (almost) identical estimates and standard errors.

Comparison with vector autoregressive models

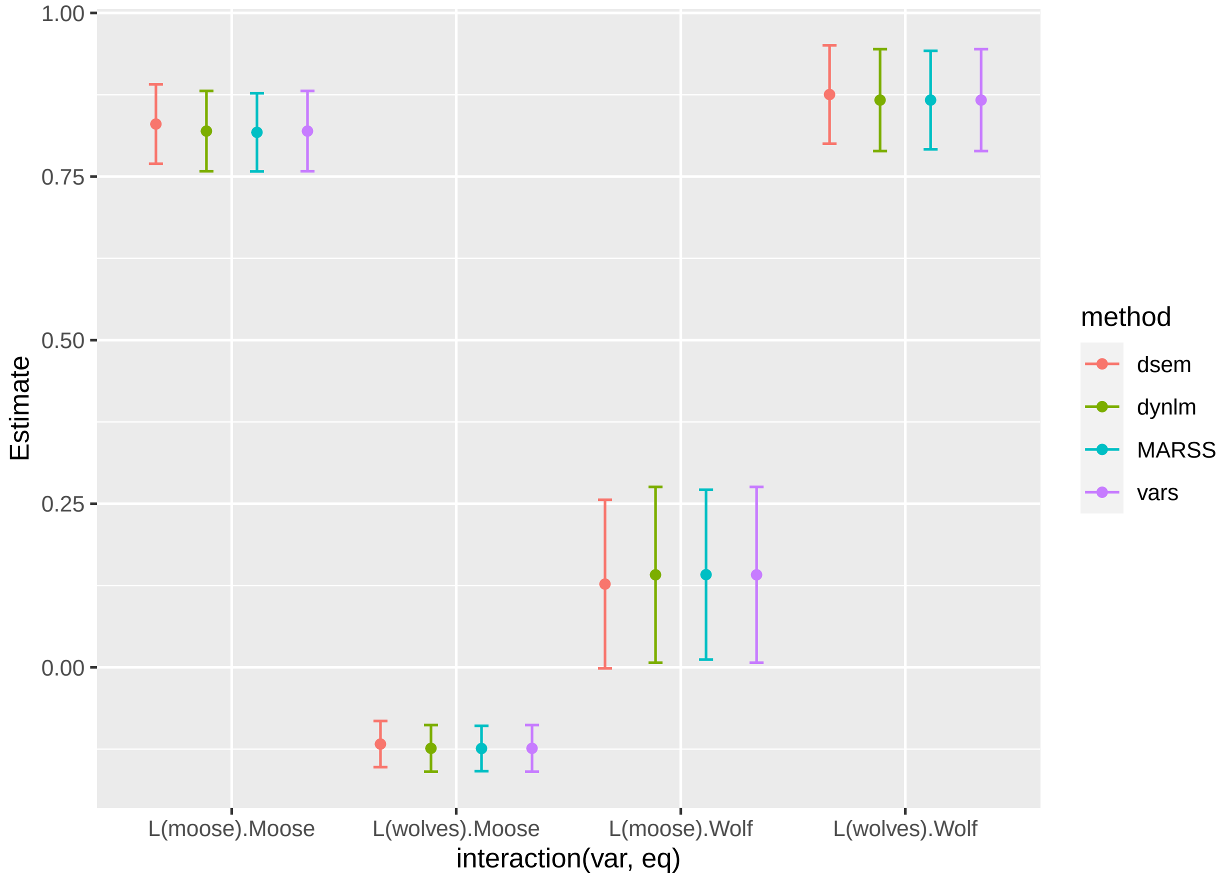
We next demonstrate that `dsem` gives similar results to a vector autoregressive model (VAM). To do so, we analyze population abundance of wolf and moose populations on Isle Royale from 1959 to 2019, downloaded from their website (Vucetich, JA and Peterson RO. 2012. The population biology of Isle Royale wolves and moose: an overview. URL: www.isleroyalewolf.org).

This dataset was previously analyzed by in Chapter 14 of the User Manual for the R-package MARSS (Holmes, E. E., M. D. Scheuerell, and E. J. Ward (2023) Analysis of multivariate time-series using the MARSS package. Version 3.11.8. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112, DOI: 10.5281/zenodo.5781847).

Here, we compare fits using `dsem` with `dynlm`, as well as a vector autoregressive model package `vars`, and finally with `MARSS`.

```r
data(isle_royale)
data = ts( log(isle_royale[,2:3]), start=1959)

sem = "
  wolves -> wolves, -1, arW
  moose -> wolves, -1, MtoW
  wolves -> moose, -1, WtoM
  moose -> moose, -1, arM
  moose <-> moose, 0, sdM
  wolves <-> wolves, 0, SDW
"
fit = dsem( sem=sem, tsdata=data, estimate_delta0=TRUE, upper=0.99, quiet=TRUE )

# dynlm
fm_wolf = dynlm( wolves ~ 1 + L(wolves) + L(moose), data=data )   #
fm_moose = dynlm( moose ~ 1 + L(wolves) + L(moose), data=data )   #

# MARSS
library(MARSS)
#> Warning: package 'MARSS' was built under R version 4.3.1
z.royale.dat <- t(scale(data.frame(data),center=TRUE,scale=FALSE))
royale.model.1 <- list(
  Z = "identity",
  B = "unconstrained",
  Q = "diagonal and unequal",
  R = "zero",
  U = "zero"
)
kem.1 <- MARSS(z.royale.dat, model = royale.model.1)
#> Success! abstol and log-log tests passed at 16 iterations.
#> Alert: conv.test.slope.tol is 0.5.
#> Test with smaller values (<0.1) to ensure convergence.
#>
#> MARSS fit is
#> Estimation method: kem
#> Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
#> Estimation converged in 16 iterations.
#> Log-likelihood: -3.21765
#> AIC: 22.4353   AICc: 23.70964
#>
```

```r
#>                          Estimate
#> B.(1,1)                    0.8669
#> B.(2,1)                   -0.1240
#> B.(1,2)                    0.1417
#> B.(2,2)                    0.8176
#> Q.(X.wolves,X.wolves)      0.1341
#> Q.(X.moose,X.moose)        0.0284
#> x0.X.wolves                0.2324
#> x0.X.moose                -0.6476
#> Initial states (x0) defined at t=0
#>
#> Standard errors have not been calculated.
#> Use MARSSparamCIs to compute CIs and bias estimates.
SE <- MARSSparamCIs( kem.1 )

# Using VAR package
library(vars)
#> Warning: package 'vars' was built under R version 4.3.1
#> Warning: package 'strucchange' was built under R version 4.3.1
#> Warning: package 'sandwich' was built under R version 4.3.1
#> Warning: package 'urca' was built under R version 4.3.1
#> Warning: package 'lmtest' was built under R version 4.3.1
var = VAR( data, type="const" )

# Compile
m1 = rbind( summary(fm_wolf)$coef[-1,], summary(fm_moose)$coef[-1,] )[,1:2]
m2 = summary(fit$opt$SD)[1:4,]
m3 = cbind( SE$parMean[c(1,3,2,4)], SE$par.se$B[c(1,3,2,4)] )
colnames(m3) = colnames(m2)
m4 = rbind( summary(var$varresult$wolves)$coef[-3,], summary(var$varresult$moose)$coef[-3,] )[,1:2]
m = rbind(
  data.frame("var"=rownames(m1),m1,"method"="dynlm","eq"=rep(c("Wolf","Moose"),each=2)),
  data.frame("var"=rownames(m1),m2,"method"="dsem","eq"=rep(c("Wolf","Moose"),each=2)),
  data.frame("var"=rownames(m1),m3,"method"="MARSS","eq"=rep(c("Wolf","Moose"),each=2)),
  data.frame("var"=rownames(m1),m4,"method"="vars","eq"=rep(c("Wolf","Moose"),each=2))
)
m = cbind(m, "lower"=m$Estimate-m$Std..Error, "upper"=m$Estimate+m$Std..Error )

# ggplot estimates
library(ggplot2)
ggplot(data=m, aes(x=interaction(var,eq), y=Estimate, color=method)) +
  geom_point( position=position_dodge(0.9) ) +
  geom_errorbar( aes(ymax=as.numeric(upper),ymin=as.numeric(lower)),
                 width=0.25, position=position_dodge(0.9))  #
```



Results again show that `dsem` can estimate parameters for a vector autoregressive model (VAM), and it exactly matches results from `vars` and using `dynlm`. Results using `MARSS` differ somewhat for reasons we don't understand.