Matrix model selection package

Jonathan Walter, Thomas Anderson, Lei Zhao, Daniel Reuman

2018-05-02

The mms package provides tools to do model selection among matrix models via a leave-n-out cross validation and resampling approach.

Brief introduction to matrix regression

If $i = 1, \ldots, P$ are sampling locations or other identifiers and matrices $M_

Example data

Model selection on matrix models and the package functions

Details of the method

Simulation study of effectiveness

The effectiveness of matrix model selection using leave-n-out cross-validation (lnocv) can be demonstrated by applying mms to data generated by a theoretical model designed to simulate geographies of synchrony arising from one or more mechanisms (Walter et al. 2017). Here, we describe the model and its implementation for a test-case showing that lnocv can discriminate among a true predictor and multiple spurious predictors representing mechanisms not operating in the model. A more comprehensive suite of tests of effectiveness are described in Walter et al. (2017).

The model is an extension of a vector autoregressive moving-average (VARMA) model that generates time series of abundance that may be correlated across multiple locations. The model simulates populations at locations $i = 1, \ldots, P$, with populations linked by dispersal and subject to spatially correlated environmental fluctuations (Moran Effects). Fluctuations about the within-patch carrying capacity are denoted $w_i(t)$ for time t. The $\epsilon_i^{(j)}(t)$ are environmental conditions for potential Moran drivers. Dispersal is implemented via a P by P connectivity matrix, D. Population density dependence is controlled by the autoregressive coefficients m_{il}, while the influence of environmental variables is given by the moving average coefficients $q_{il}^{(j)}$. The model is:

$$\begin{pmatrix} w_1(t) \\ \vdots \\ w_P(t) \end{pmatrix} \approx D \left[\begin{pmatrix} \sum_{l=1}^{a} m_{1l} w_1(t-l) + \sum_{j=1}^{b} \sum_{l=0}^{c} q_{1l}^{(j)} \epsilon_1^{(j)}(t-l) \\ \vdots \\ \sum_{l=1}^{a} m_{Pl} w_P(t-l) + \sum_{j=1}^{b} \sum_{l=0}^{c} q_{Pl}^{(j)} \epsilon_P^{(j)}(t-l) \end{pmatrix} \right], \tag{1}$$

Here we consider a scenario in which spatial structure in population dynamics (i.e., geography of synchrony) arises from spatial structure in an environmental driver. The data produced are identical to those used to present "mechanism A"" in Walter et al. (2017). We simulated population dynamics in $P = 16$ locations divided between two disjoint eight-location sets, S_1 and S_2. S_1 was {1,...,8} and S_2 was {9,...,16}. All locations had identical 2nd-order density dependence with $m_{i1} \approx 0.375$ and $m_{i2} \approx -0.368$. These

parameters produce periodic oscillations with a dominant period length of ≈ 5 time steps. We simulated three environmetnal drivers: an "operating" driver that strongly influences population dynamics (i.e., $q_{i1}^{(1)} = 1$) and is spatially structured; a "latent" driver that is also spatially structured but does not influence population dynamics (i.e., $q_{i1}^{(2)} = 0$); and a third driver representing local variability that is spatially unstructured (random) and has a small influence on population dynamics (i.e., $q_{i1}^{(2)} = 0.1$). Spatial structure in environmental drivers was given by the covariance matrices Ω_j, which were block matrices having off-diagona entries 0.6 within S_1 or S_2 and 0.3 between groups. Organisms dispersed among populations such that 5% of each population dispersed evenly among all other populations. The model was run for 150 time steps with a 100 time step burn-in period.

We use simulated data to define a response matrix and multiple predictors, including a true predictor representing the operating mechanism in the simulation described above, and two spurious predictors representing latent (non-operating) possible mechanisms. The data are available using:

```
library(mms)
data(simdat)
attach(simdat)
```

The object 'simdat' is a list containing four matrices. The first three are P by t matrices giving, respectively, abundances $w_i(t)$ and environmental fluctuations $\epsilon_i^{(j)}(t)$ for the operating ($j = 1$) and latent ($j = 2$) drivers. The fourth is the P by P dispersal matrix, D. We then compute synchrony matrices using Pearson correlations:

```
popsynch<-cor(t(simdat$pop), method="pearson")
driversynch<-cor(t(simdat$driver), method="pearson")
latentsynch<-cor(t(simdat$latent), method="pearson")
```

The lnocv score for an individual model is obtained using `mmsscore`, e.g. for a model in which dispersal is the sole predictor of population synchrony:

```
modmats<-list(popsynch=popsynch, driversynch=driversynch, latentsynch=latentsynch, dispersal=simdat$disp
modscore<-mmsscore(mats=list(modmats$popsynch, modmats$dispersal), pred=2, n=2, maxruns=500)
print(modscore)
```

```
## $lno.score
## [1] 0.03830168
##
## $num.pos
## [1] 120
##
## $num.att
## [1] 120
##
## $num.rnk
## [1] 120
##
## $num.usd
## [1] 120
```

Note that in this case, and in those following, `maxruns` (and later, `nrand`) is set relatively low so that these examples can be run quickly. We recommend increasing these. A selection of models can be ranked using:

```
modranks<-mmsrank(mats=modmats, model.names=NA, n=2, maxruns=200)
print(modranks)
```

```
##   model.names   lno.score num.pos num.att num.rnk num.usd
## 1           2 0.004491566     120     120     120     120
## 2           3 0.039231228     120     120     120     120
```

```
## 3           4 0.038301679     120     120     120     120
## 4         2:3 0.004633066     120     120     120     120
## 5     c(2, 4) 0.004550706     120     120     120     120
## 6         3:4 0.038963419     120     120     120     120
## 7         2:4 0.004695749     120     120     120     120
```

Here, we use `model.nameds=NA` (the default) to test all combinations of predictors. The first matrix in `mats` is taken to be the response variable. Recall that in our example there is one operating environmental driver producing geography of synchrony. The model reflecting this receives the highest rank.

Model weights, corresponding to the frequency with which a model is the best performing in the candidate set are obtained using:

```
modweights<-mmsmodwts(mats=modmats, n=2, maxruns=200, nrand=50)
```

```
## [1] "mmsmodwts working on resampling 1 of 50"
## [1] "mmsmodwts working on resampling 2 of 50"
## [1] "mmsmodwts working on resampling 3 of 50"
## [1] "mmsmodwts working on resampling 4 of 50"
## [1] "mmsmodwts working on resampling 5 of 50"
## [1] "mmsmodwts working on resampling 6 of 50"
## [1] "mmsmodwts working on resampling 7 of 50"
## [1] "mmsmodwts working on resampling 8 of 50"
## [1] "mmsmodwts working on resampling 9 of 50"
## [1] "mmsmodwts working on resampling 10 of 50"
## [1] "mmsmodwts working on resampling 11 of 50"
## [1] "mmsmodwts working on resampling 12 of 50"
## [1] "mmsmodwts working on resampling 13 of 50"
## [1] "mmsmodwts working on resampling 14 of 50"
## [1] "mmsmodwts working on resampling 15 of 50"
## [1] "mmsmodwts working on resampling 16 of 50"
## [1] "mmsmodwts working on resampling 17 of 50"
## [1] "mmsmodwts working on resampling 18 of 50"
## [1] "mmsmodwts working on resampling 19 of 50"
## [1] "mmsmodwts working on resampling 20 of 50"
## [1] "mmsmodwts working on resampling 21 of 50"
## [1] "mmsmodwts working on resampling 22 of 50"
## [1] "mmsmodwts working on resampling 23 of 50"
## [1] "mmsmodwts working on resampling 24 of 50"
## [1] "mmsmodwts working on resampling 25 of 50"
## [1] "mmsmodwts working on resampling 26 of 50"
## [1] "mmsmodwts working on resampling 27 of 50"
## [1] "mmsmodwts working on resampling 28 of 50"
## [1] "mmsmodwts working on resampling 29 of 50"
## [1] "mmsmodwts working on resampling 30 of 50"
## [1] "mmsmodwts working on resampling 31 of 50"
## [1] "mmsmodwts working on resampling 32 of 50"
## [1] "mmsmodwts working on resampling 33 of 50"
## [1] "mmsmodwts working on resampling 34 of 50"
## [1] "mmsmodwts working on resampling 35 of 50"
## [1] "mmsmodwts working on resampling 36 of 50"
## [1] "mmsmodwts working on resampling 37 of 50"
## [1] "mmsmodwts working on resampling 38 of 50"
## [1] "mmsmodwts working on resampling 39 of 50"
## [1] "mmsmodwts working on resampling 40 of 50"
## [1] "mmsmodwts working on resampling 41 of 50"
```

```
## [1] "mmsmodwts working on resampling 42 of 50"
## [1] "mmsmodwts working on resampling 43 of 50"
## [1] "mmsmodwts working on resampling 44 of 50"
## [1] "mmsmodwts working on resampling 45 of 50"
## [1] "mmsmodwts working on resampling 46 of 50"
## [1] "mmsmodwts working on resampling 47 of 50"
## [1] "mmsmodwts working on resampling 48 of 50"
## [1] "mmsmodwts working on resampling 49 of 50"
## [1] "mmsmodwts working on resampling 50 of 50"
```

```
print(modweights)
```

```
##   model.names freq.top num.pos num.att num.rnk num.usd
## 1           2       29    6000    6000    6000    5621
## 2           3        0    6000    6000    6000    5621
## 3           4        0    6000    6000    6000    5621
## 4         2:3        4    6000    6000    6000    5621
## 5     c(2, 4)       13    6000    6000    6000    5621
## 6         3:4        0    6000    6000    6000    5621
## 7         2:4        4    6000    6000    6000    5621
```

Like for rankings, the correct model receives the highest weight. The model weights can be used further to generate variable importance weights:

```
varimp<-mmsvarwts(pred=2:4, weights=modweights, prednames=names(modmats)[-1])
print(varimp)
```

```
##     prednames summed.weights
## 1 driversynch           1.00
## 2 latentsynch           0.16
## 3   dispersal           0.34
```

The algorithm correctly identifies the operating environmental driver as the most important predictor.