Minimal intro to sienaBayes

Johan Koskinen and T.A.B. Snijders

2024-06-22

Contents

Preamble

Package

```r
require(RSiena)
```

```
## Loading required package: RSiena
```

```r
require(multiSiena)
```

```
## Loading required package: multiSiena
```

Load mock dataset

Use a routine for creating synthetic dataset with 6 networks

```r
source("https://raw.githubusercontent.com/johankoskinen/Sunbelt2024/main/multiSiena/data.set.enzo.setup
```

This dataset, `enzo`,

```r
enzo <- create.enzo()
```

is a `sienaGroup` object.

What data

Plot the 6 networks for the two waves. For each group $j = 1, \dots, 6$, we have adjacency matrices

$$\boldsymbol{x}^{(j)}(t_0), \text{ and } \boldsymbol{x}^{(j)}(t_1)$$

For example, for group $j = 1$, $\boldsymbol{x}^{(j)}(t_0)$

```r
enzo$Data1$depvars[[1]][,,1]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    1    0    0    0    0
## [3,]    0    0    0    1    0
## [4,]    0    0    0    0    1
## [5,]    0    0    0    0    0
```

and $\boldsymbol{x}^{(j)}(t_1)$

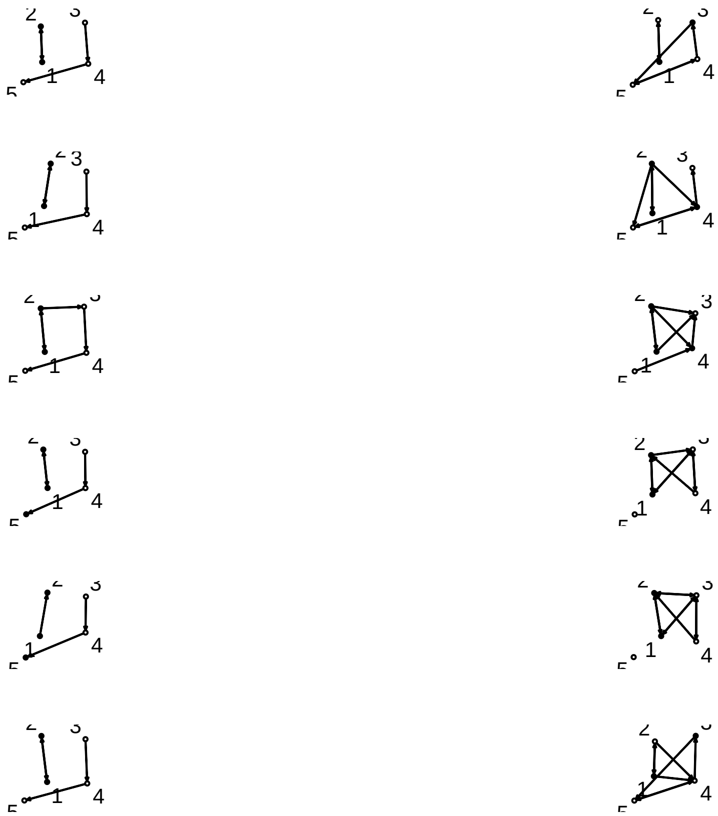
```r
enzo$Data1$depvars[[1]][,,2]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    1    0    0    0    0
## [3,]    0    0    0    0    1
## [4,]    0    0    1    0    1
## [5,]    0    0    0    1    0
```

For

```r
require(sna)
coord <- matrix(c(1,1,
1,2,
2,2,
2,1,
.5,.5),5,2,byrow=TRUE)
# === plot
par( mfrow = c(6,2) ,oma = c(0,1,0,0) + 0.1,
          mar = c(1,0,1,1) + 0.1)
# grp1
gplot( enzo$Data1$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data1$depvars[[2]][,1,1], label=c
gplot( enzo$Data1$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data1$depvars[[2]][,1,2], label=
# grp2
gplot( enzo$Data2$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data2$depvars[[2]][,1,1], label=c
gplot( enzo$Data2$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data2$depvars[[2]][,1,2], label=
# grp3
gplot( enzo$Data3$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data3$depvars[[2]][,1,1], label=c
gplot( enzo$Data3$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data3$depvars[[2]][,1,2], label=
# grp4
gplot( enzo$Data4$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data4$depvars[[2]][,1,1], label=c
gplot( enzo$Data4$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data4$depvars[[2]][,1,2], label=
# grp5
gplot( enzo$Data5$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data5$depvars[[2]][,1,1], label=c
```

```
gplot( enzo$Data5$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data5$depvars[[2]][,1,2], label=c
# grp6
gplot( enzo$Data6$depvars[[1]][,,1] ,coord = coord, vertex.col = enzo$Data6$depvars[[2]][,1,1], label=c
gplot( enzo$Data6$depvars[[1]][,,2] , coord = coord, vertex.col = enzo$Data6$depvars[[2]][,1,2], label=c
```



Basic Model

For eaxh group $j = 1, \ldots, 6$ we define a **stochastic actor-oriented model**

$$\boldsymbol{x}^{(j)}(t_1) \mid \boldsymbol{x}^{(j)}(t_0) \sim SAOM(\theta_j)$$

The $SAOM(\theta)$ is determined by its **objective** and **rate** functions

$$f_i(\boldsymbol{x}, \theta), \text{ and } \lambda_i(\theta)$$

that are assumed to be the same for all groups $j = 1, \ldots, N$.

Define model

You define the *effects* of the objective function of the model in the standard way. The effects structure is obtained from `getEffects()`

```
seed <- 1234
GroupEffects <- getEffects(enzo)
GroupsModel <- sienaAlgorithmCreate(projname = 'Enzo',seed=seed)
```

```
## If you use this algorithm object, siena07 will create/use an output file Enzo.txt .
```

Hierarchial model

All groups have the same effects and model, $SAOM(\theta_j)$, but some of the parameters, γ_j, vary across groups, and some, η, are the same for all groups.

$$\theta_j = \begin{bmatrix} \gamma_j \\ \eta \end{bmatrix}$$

Group-varying and fixed

By default

```r
summary(GroupEffects)$random
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE
```

```r
summary(GroupEffects)$effectName[summary(GroupEffects)$random]
```

```
## [1] "outdegree (density)"
```

```r
summary(GroupEffects)$effectName[summary(GroupEffects)$random==FALSE]
```

```
##  [1] "constant net rate (period 1)"  "constant net rate (period 3)"
##  [3] "constant net rate (period 5)"  "constant net rate (period 7)"
##  [5] "constant net rate (period 9)"  "constant net rate (period 11)"
##  [7] "reciprocity"                   "rate beh (period 1)"
##  [9] "rate beh (period 3)"           "rate beh (period 5)"
## [11] "rate beh (period 7)"           "rate beh (period 9)"
## [13] "rate beh (period 11)"          "beh linear shape"
```

Model for group-varying

The model assumes that the group-varying parameters follow a **multivariate normal distribution**

$$\gamma_j \sim N(\mu, \Sigma)$$

Target of inference

We want to estimate the population mean μ and the non-varying parameter η.

Running default

To estimate the model with default settings

```r
groupModel.e <- sienaBayes(GroupsModel, data = enzo,
                initgainGlobal=0.1, initgainGroupwise = 0.001,
                effects = GroupEffects,
             nrunMHBatches=40, silentstart=FALSE)
```

Results

What did we get?

Check objects returned

```r
names(groupModel.e)
```

Non-varying

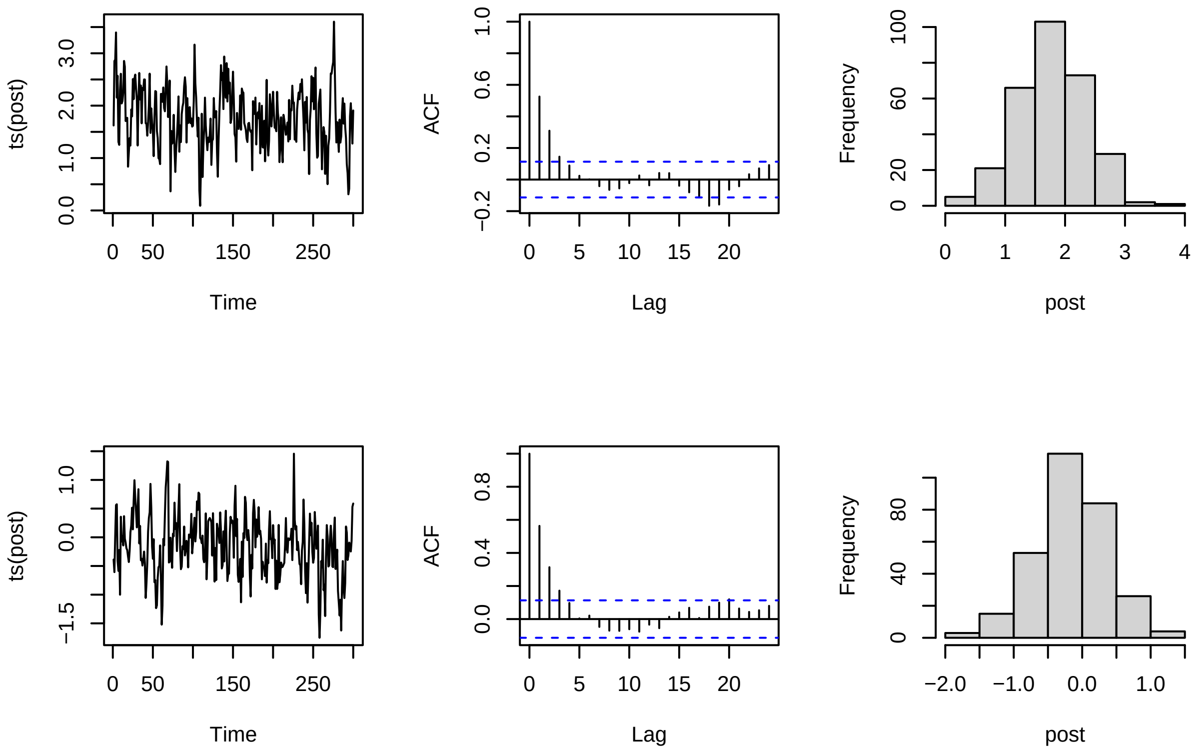
The `nmain`: 300 posterior draws of η are in `ThinPosteriorEta`

```
dim(groupModel.e$ThinPosteriorEta)
```

```
## [1] 300   2
```

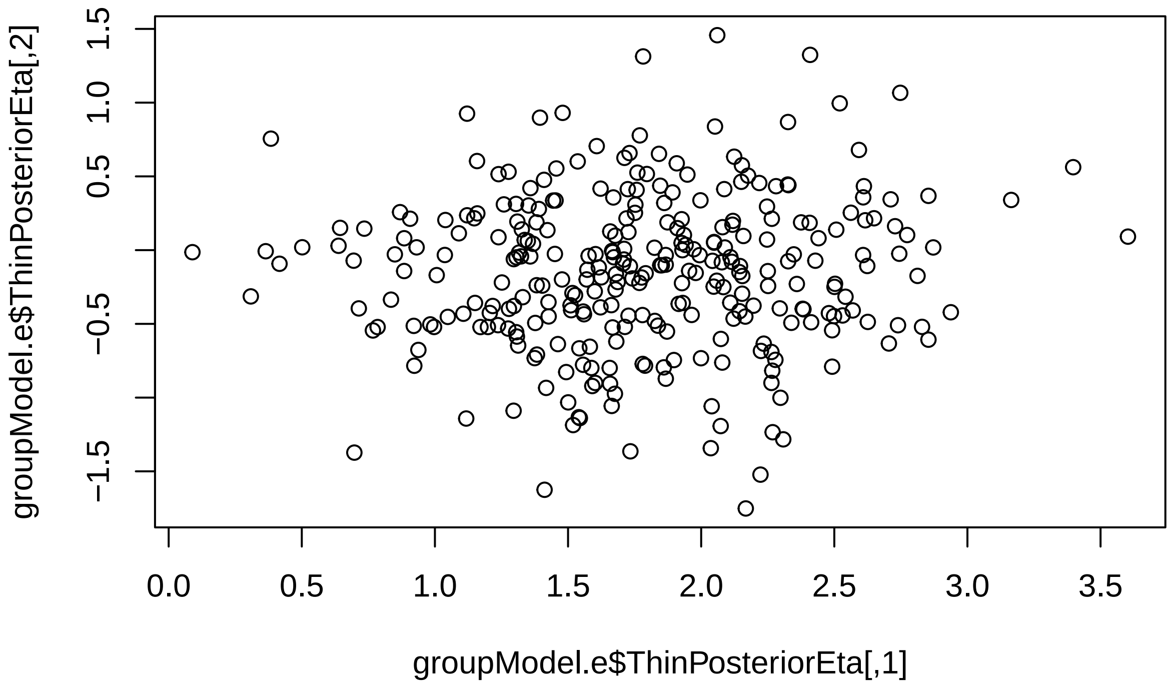
This corresponds to `reciprocity` and the `linear shape` for behaviour

```
par(mfrow=c(2,3))
for (k in c(1:2)){
  post <- groupModel.e$ThinPosteriorEta[,k]
plot(ts(post))# draws in each iteration
acf(post,main='')# the autocorrelation function - how much dependence
hist(post,main='')# (empirical) posterior distribution
}
```



Note that these are draws from a bivariate distribution

```
plot(groupModel.e$ThinPosteriorEta)
```



We can calculate probabilites for the parameters using the values simulated from the posterior

For example, we can calculate the probability that the reciprocity parameter is positive **given data** as

```
mean(groupModel.e$ThinPosteriorEta[,1]>0)
```

```
## [1] 1
```

Given these 6 (synthetic) networks, the reciprocity parameter is positive with a posterior probability of 1

Population parameters

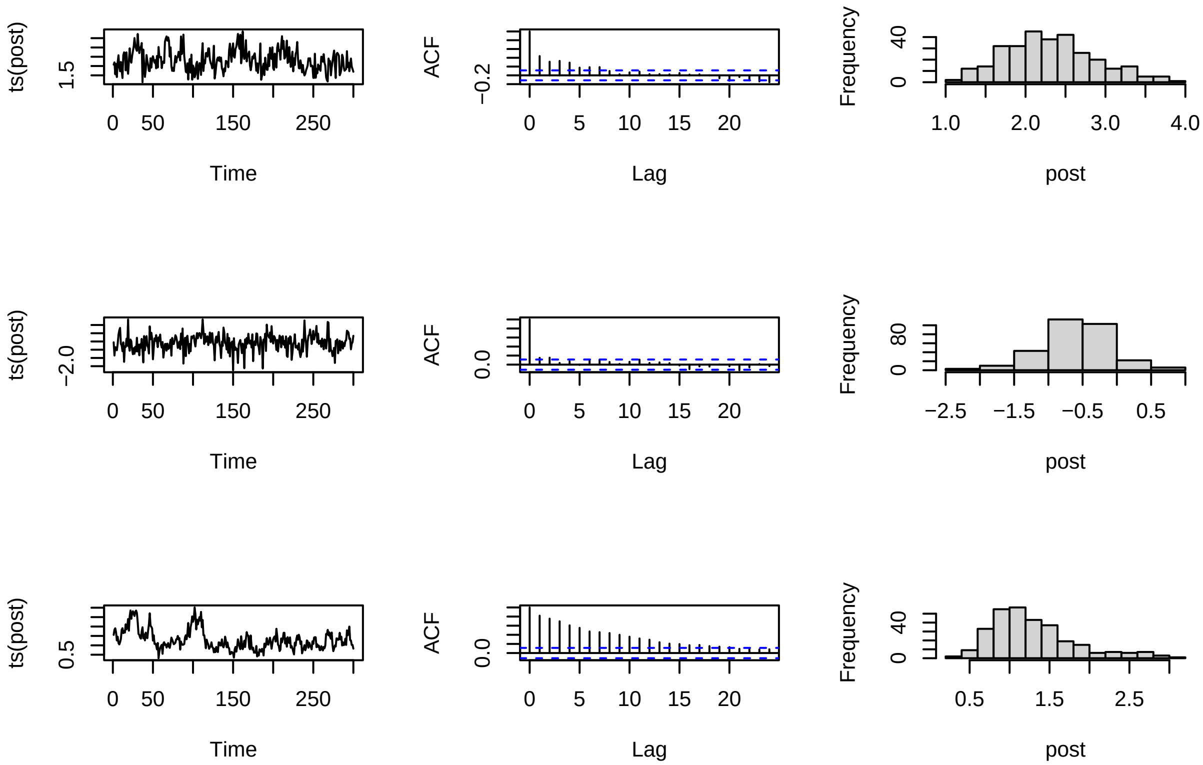
The posteriors for μ are found in `ThinPosteriorMu`

```
dim(groupModel.e$ThinPosteriorMu)
```

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

Corresponding to the rate for the network, density, and rate for behaviour

```
par(mfrow=c(3,3))
for (k in c(1:3)){
  post <- groupModel.e$ThinPosteriorMu[,k]
plot(ts(post))
acf(post,main='')
hist(post,main='')
}
```



Summary of results

To get a table to numerical summaries of the posteriors

```r
summary(groupModel.e)
```

```
## Bayesian estimation.
## Prior distribution:
##
## Mu      basic rate parameter net    2.5161
##         outdegree (density)         0.0000
##         rate beh period 1           0.7323
##
## Sigma     0.5812   0.0000  -0.1585
##           0.0000   1.0000   0.0000
##          -0.1585   0.0000   0.1310
##
## Prior Df  5
##
## Kappa     0.0000
##
## Eta
## For the fixed parameters, constant prior.
##
## Algorithm specifications were  nprewarm = 50 , nwarm = 50 , nmain = 250 , nrunMHBatches = 40 , nImpro
##  nSampVarying = 1 , nSampConst = 1 , mult = 5 .
## Posterior means and standard deviations are averages over the last 250 runs.
##
## Proportion of acceptances in MCMC proposals after warming up:
## 0.19 0.20 0.23 0.24 0.26 0.31 0.25 0.25
```

```
##
## This should ideally be between 0.15 and 0.50.
## Note: this summary does not contain a convergence check.
## Note: the print function for sienaBayesFit objects can also use a parameter nfirst,
##       indicating the first run from which convergence is assumed.
##
## Groups:
##  Data1   Data2   Data3   Data4   Data5   Data6
##
## Posterior means and standard deviations for global mean parameters
##
## Total number of runs in the results is  300 .
## Posterior means and standard deviations are averages over 250 MCMC runs (excluding warming, after thi
##
##                                          Post.      Post.     cred.    cred.   p      varying   Post
##                                          mean       s.d.m.    from     to                       s.d.
## Network Dynamics
##    1. rate constant net rate (period 1)   2.1459  ( 0.7326 )  0.8441  3.4185
##    2. rate constant net rate (period 3)   2.1719  ( 0.7819 )  0.9366  3.9436
##    3. rate constant net rate (period 5)   2.3410  ( 0.7160 )  1.2694  3.7795
##    4. rate constant net rate (period 7)   2.3381  ( 0.7119 )  1.1655  3.7801
##    5. rate constant net rate (period 9)   2.5540  ( 0.7121 )  1.2057  3.9846
##    6. rate constant net rate (period 11)  2.2758  ( 0.6683 )  1.1175  3.7375
##    7. eval outdegree (density)           -0.5602  ( 0.4986 ) -1.6841  0.4112  0.09       +      0.979
##    8. eval reciprocity                    1.7251  ( 0.5568 )  0.6394  2.7458  1.00       -
##
## Behavior Dynamics
##    9. rate rate beh (period 1)            1.2257  ( 0.5262 )  0.4060  2.4141
##   10. rate rate beh (period 3)            1.1665  ( 0.5480 )  0.2743  2.6184
##   11. rate rate beh (period 5)            1.1340  ( 0.5262 )  0.3071  2.4207
##   12. rate rate beh (period 7)            1.1159  ( 0.5089 )  0.3590  2.4234
##   13. rate rate beh (period 9)            1.0906  ( 0.5191 )  0.2869  2.3592
##   14. rate rate beh (period 11)           1.2065  ( 0.5260 )  0.3976  2.5006
##   15. eval beh linear shape              -0.1760  ( 0.5260 ) -1.2722  0.8481  0.36       -
##
## Posterior mean of global covariance matrix (varying parameters)
##   0.6371  -0.0077  -0.1424
##  -0.0077   0.9590   0.0054
##  -0.1424   0.0054   0.1591
##
## Posterior standard deviations of elements of global covariance matrix
##   0.3928   0.3808   0.1483
##   0.3808   0.8117   0.1823
##   0.1483   0.1823   0.0921
##
## For the rate parameters across all groups:
##                          Post.mean Post.sd
## basic rate parameter net   2.30241 0.55871
## rate beh period 1          1.16753 0.44662
```

Changing hierarchial model

In the previous model, `reciprocity` was assumed to have the same parameter $\theta_{j,rec} = \eta_{rec}$, for all $j = 1, \ldots, 6$. To allow the `reciprocity` parameter to *vary* across groups j, set

```
GroupEffects <- setEffect( GroupEffects, recip,random=TRUE)
```

```
##   effectName  shortName include fix   test  initialValue parm randomEffects
## 1 reciprocity recip     TRUE    FALSE FALSE            0    0     TRUE
```

so that now

$$\theta_j = \begin{bmatrix} \gamma_{j,rate_x} \\ \gamma_{j,dens} \\ \gamma_{j,rec} \\ \gamma_{j,rate_z} \\ \eta_{shape,z} \end{bmatrix}, \text{ and } \gamma_j \sim \mathcal{N}_4(\mu, \Sigma)$$

Running new model

To estimate the model with one more random effect

```
groupModel.e2 <- sienaBayes(GroupsModel, data = enzo,
                initgainGlobal=0.1, initgainGroupwise = 0.001,
                effects = GroupEffects,
             nrunMHBatches=40, silentstart=FALSE)
```

Output

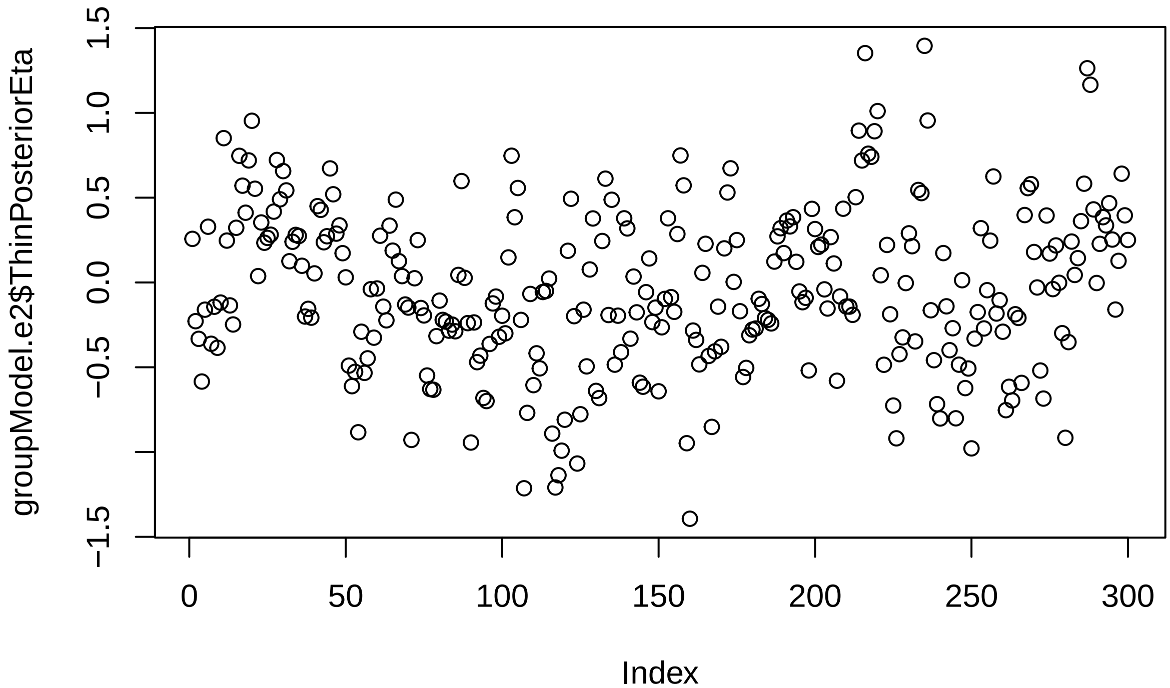
To highlight the fact that our posterior inference is represented by *simulated draws of random variables*, we will now examine and plot these random parameters using standard `R` functions.

> The script `BayesPlots.R` contains functions for plots that are **custom made** for exaxamining the estimation results from `sienaBayes`

You will find `BayesPlots.R` here: https://www.stats.ox.ac.uk/~snijders/siena/BayesPlots.r; and explanations and examples of their usage are found here: https://www.stats.ox.ac.uk/~snijders/siena/SienaBayesExample5_s.pdf.

Now we only have one common parameter, one η

```
plot(groupModel.e2$ThinPosteriorEta)
```



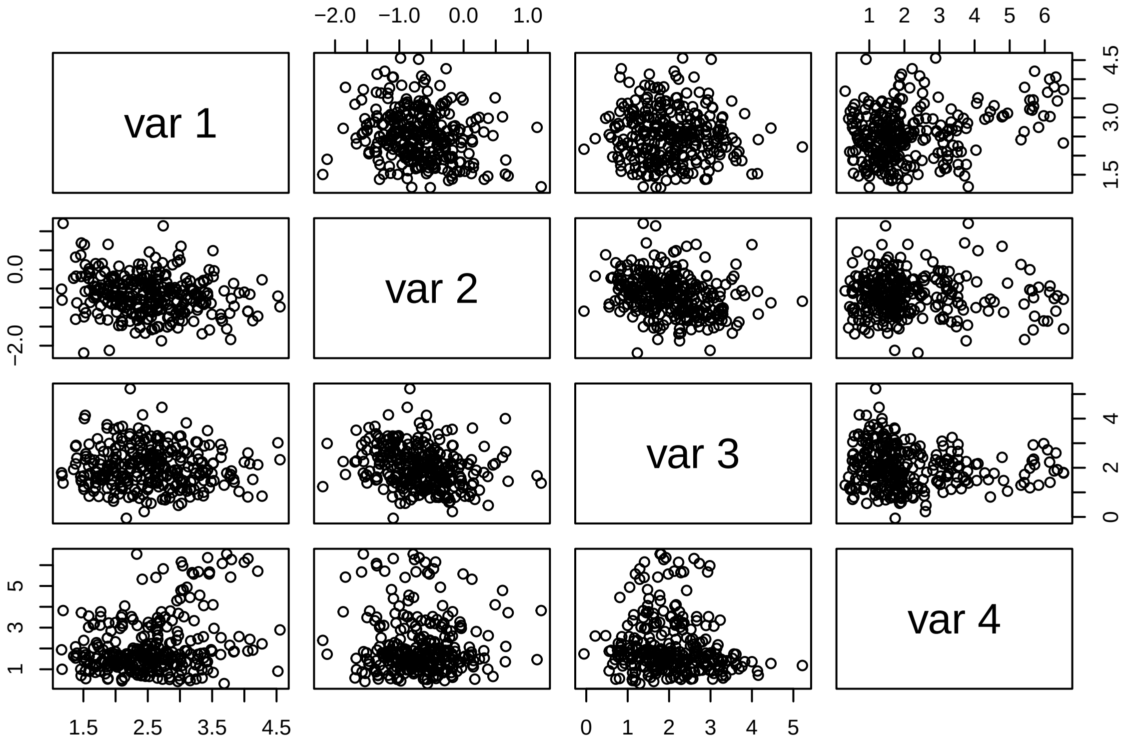
In `BayesPlots.R`, the function `GlobalNonRateParameterPlots`, provides custom-made plots of these posteriors.

In the plot we see the 300 simulated values of η that are drawn using MCMC from the posterior distribution of η given data.

> Given these 6 (synthetic) networks, the reciprocity parameter is positive with a posterior probability of 1

But we have μ has four dimentions (parameters)

```
pairs(groupModel.e2$ThinPosteriorMu)
```



We can calculate the probability that the reciprocity parameter is positive **given data** as

```
mean(groupModel.e2$ThinPosteriorMu[,3]>0)
```

```
## [1] 0.9966667
```

Group-level analysis

For reciprocity, we saw in the previous example that

$$\Pr(\mu_{rec} > 0 \mid Data) \approx 1$$

but what about the group-level parameters $\gamma_{1,rec}, \gamma_{2,rec}, \ldots, \gamma_{6,rec}$ - are they always positive for all of the groups?

Group and population-level

The posterior (predictive) distributions of the γ_j's are stored in `ThinParameters` as **iteration** by **group** by **parameter**

```
dim(groupModel.e2$ThinParameters)
```

```
## [1] 300   6   5
```

and within each group

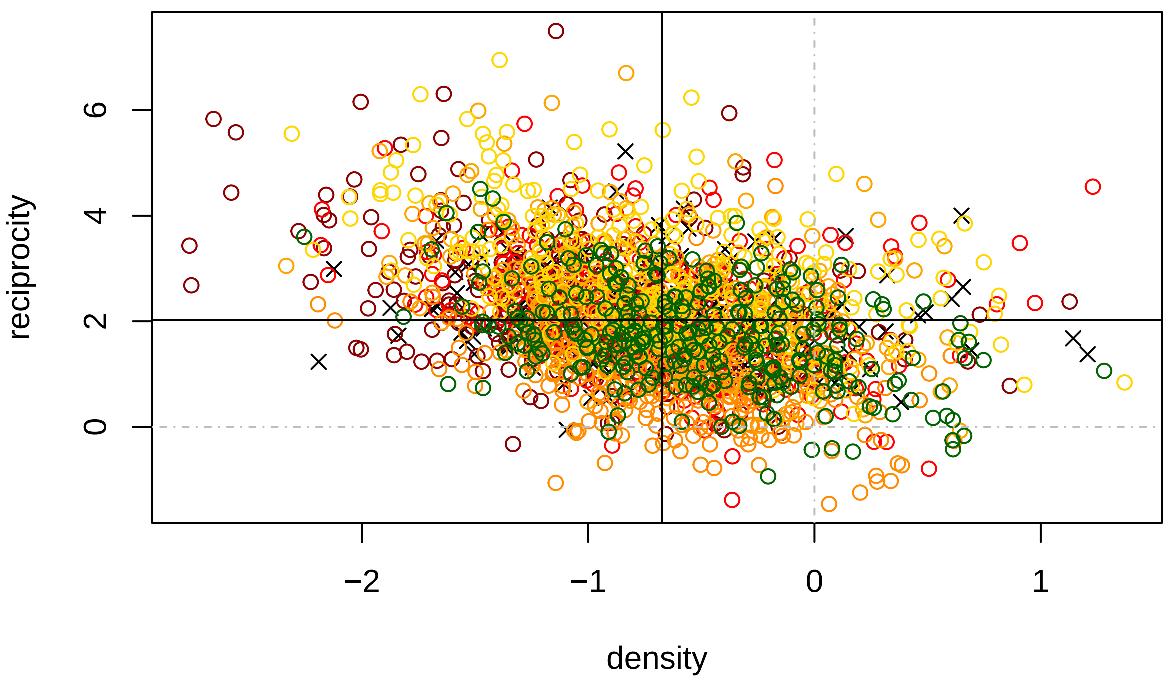
```
head(groupModel.e2$ThinParameters[,1,])
```

```
##      constant net rate outdegree (density) reciprocity rate beh
## [1,]          3.084515          -1.8143328  2.38747125 1.188040
## [2,]          2.534277          -0.8095603  2.10497307 1.392641
## [3,]          3.319430          -0.5882144  0.66770252 1.857220
## [4,]          2.482389          -1.6900750  1.84137145 2.622582
```

```
## [5,]          3.122526          -1.6681452  1.25374655 1.728405
## [6,]          3.727048          -0.9129924  0.06870421 1.269398
##       beh linear shape
## [1,]         0.2569260
## [2,]        -0.2283378
## [3,]        -0.3325189
## [4,]        -0.5843580
## [5,]        -0.1605018
## [6,]         0.3287635
```

To illustrate how we have inference for μ for density and reciprocity, but also for γ_j for density and reciprocity for each group, let us plot density against reciprocity for both levels

```
plot(groupModel.e2$ThinPosteriorMu[,2],
     groupModel.e2$ThinPosteriorMu[,3],
     xlim=range(groupModel.e2$ThinParameters[,,2]),
     ylim=range(groupModel.e2$ThinParameters[,,3]),
     pch=4, xlab='density',ylab='reciprocity')
grp.cols <- c('darkred','red','darkorange','orange','gold','darkgreen')
for (j in c(1:6))
{
  lines( groupModel.e2$ThinParameters[,j,2], groupModel.e2$ThinParameters[,j,3],type='p',pch=1,col=grp.
}
abline(v=mean(groupModel.e2$ThinPosteriorMu[,2]))
abline(h=mean(groupModel.e2$ThinPosteriorMu[,3]))
abline(v=0,col='grey',lty=4)
abline(h=0,col='grey',lty=4)
```

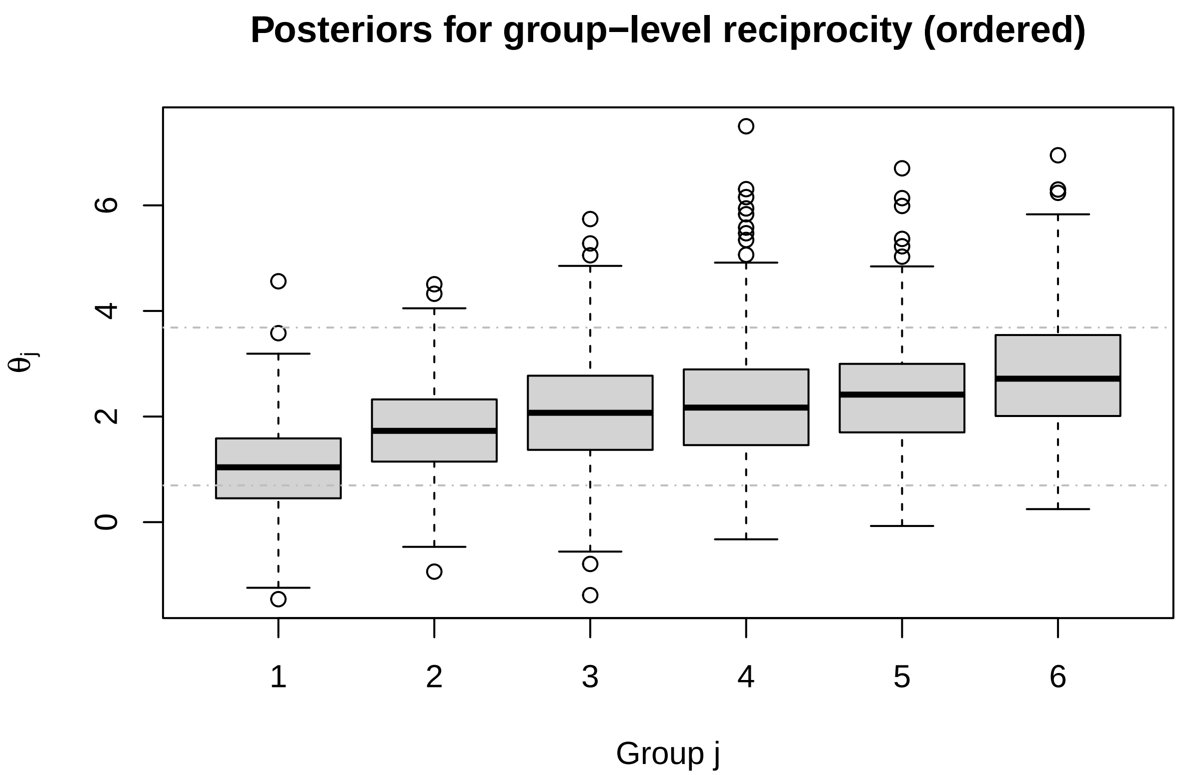


For an individual parameter it is common to look at the catepilar plot

```
require(HDInterval)
```

```
## Loading required package: HDInterval
```

```r
grp.means <- colMeans(groupModel.e2$ThinParameters[,,3])
boxplot(groupModel.e2$ThinParameters[,order(grp.means),3],
        ylab=expression(theta[j]),xlab='Group j',main='Posteriors for group-level reciprocity (ordered)
CI <- hdi(groupModel.e2$ThinPosteriorMu[,3],1-.05)
abline(h=CI[1],col='grey',lty=4)
abline(h=CI[2],col='grey',lty=4)
```



In the catepillar plot, we have the (posterior) predictive distributions for the group-level reciprocity parameters, ordered accourding to the group-mean. Grey lines represent the credibility interval for the population-level reciprocity parameter μ_{rec}

Prior and posterior

To get a posterior distribution for μ and η (and Σ), we need to have **prior distributions** for these parameters

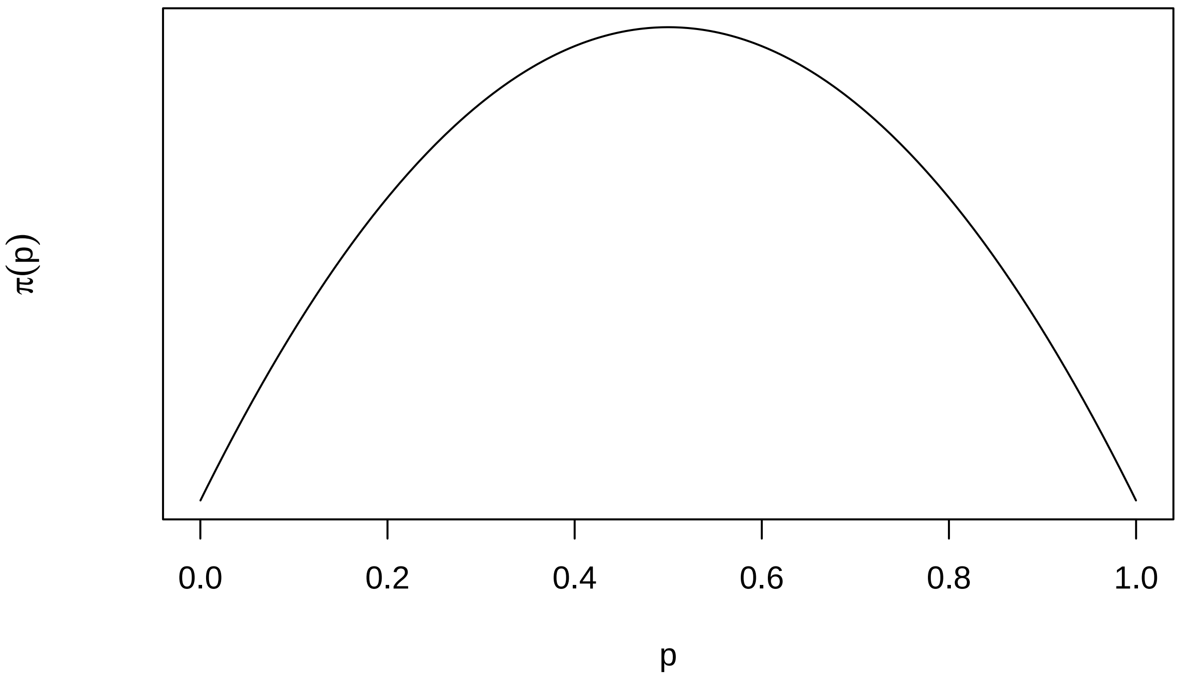
> A prior distribution for a parameter quantifies the uncertainty that we have about a parameter before we collect and observe DATA

Example: Bernoulli graph

For a cross-section $n = 5$ network, a Bernoulli graph says that each of the $5(5-1)/2 = 10$ possible ties all have a probability of p of being present, independently of each other. The parameter $0 \leq p \leq 1$ and hence a prior for p might be $p \sim Beta(\alpha, \beta)$

```r
alpha.p <- 2
beta.p <- 2
```

```r
p <- seq(from=.0001,to =.9999, length.out = 1000)
plot( p , dbeta(p, alpha.p, beta.p) , type = 'l', yaxt='n',ylab=expression(pi(p)))
```

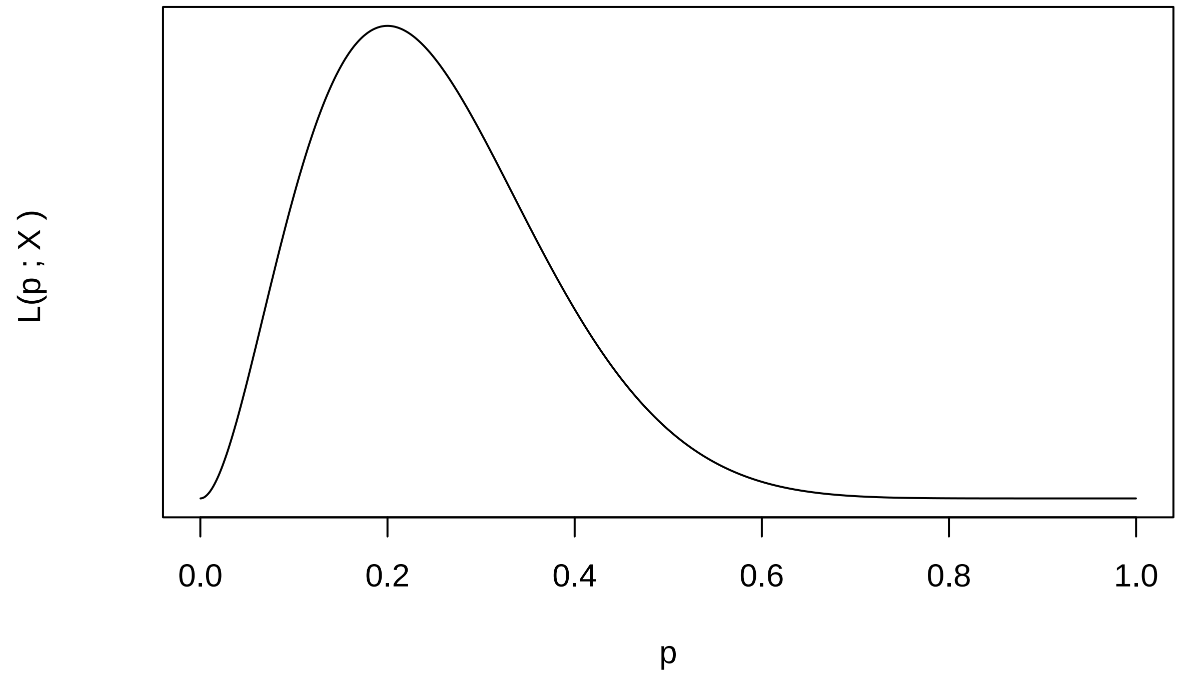


Assume that we observe the network

$$\mathbf{X} = \begin{bmatrix} - & 1 & 0 & 0 & 0 \\ - & - & 0 & 1 & 0 \\ - & - & - & 0 & 0 \\ - & - & - & - & 0 \\ - & - & - & - & - \end{bmatrix}$$

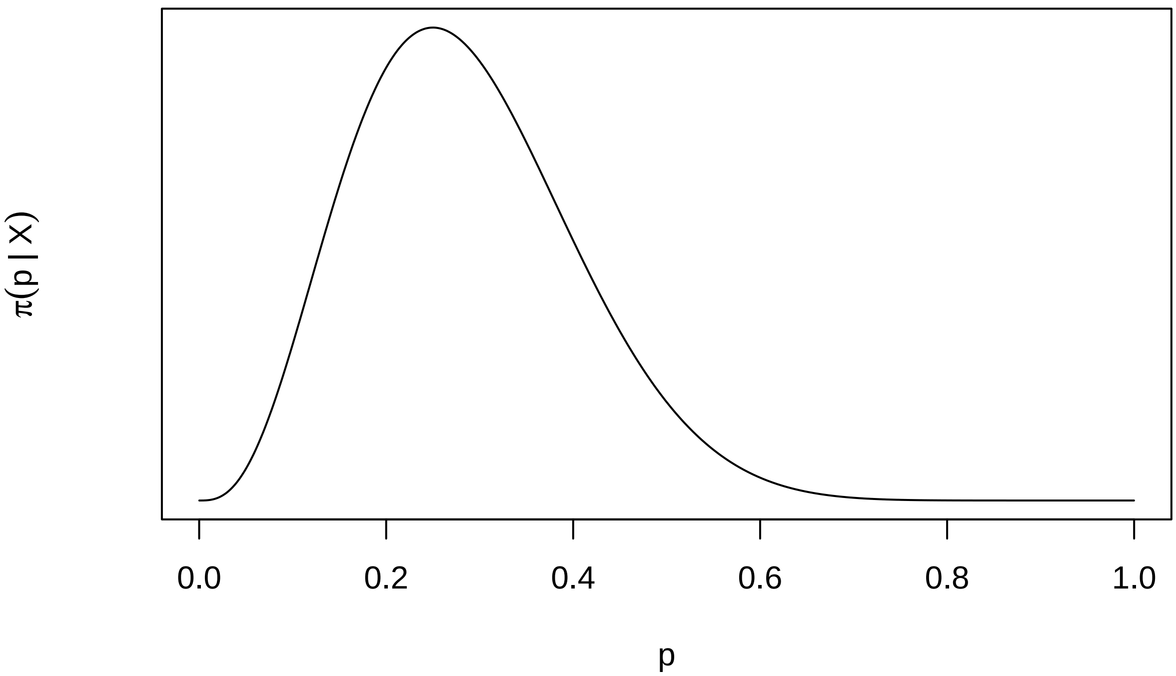
This has $L = \sum_{i<j} x_{ij} = 2$, which gives the **likelihood** function for p

```
plot(p,p^2*(1-p)^8, type = 'l', yaxt='n',ylab='L(p ; X ) ' )
```



The posterior distribution will be $p \mid \mathbf{X} \sim Beta(2+\alpha, 8+\beta)$

```
plot( p , dbeta(p, alpha.p+2, beta.p+8) , type = 'l', yaxt='n',ylab=expression(pi(p ~"|"~ X)))
```



Try with different values of the hyperparameters α and β in the prior for p

Even better, try Mattias Villani's widget that illustates Bayesian inference https://observablehq.com/@mattiasvillani/bayesian-inference-for-bernoulli-iid-data?collection=@mattiasvillani/bayesian-learning[Beta-Binomial]

Example: SAOM

To understand the hierarchical SAOM, have a look at what group-level parameters our model assumed for a given μ. You may investigate this using simulation.

Drawing θ_j

If we *knew* μ and Σ, we can draw $\theta_j \sim \mathcal{N}_4(\mu, \Sigma)$. The (implied) model in our estimated model is given by

```
mu <- groupModel.e2$priorMu
Sigma <- (1/groupModel.e2$priorDf)*groupModel.e2$priorSigma
```

For Σ, we plug in the prior expected value $E(\Sigma) = \nu^{-1}\Lambda_0$.

We can draw M sets of parameters

```
require(mvtnorm)
```

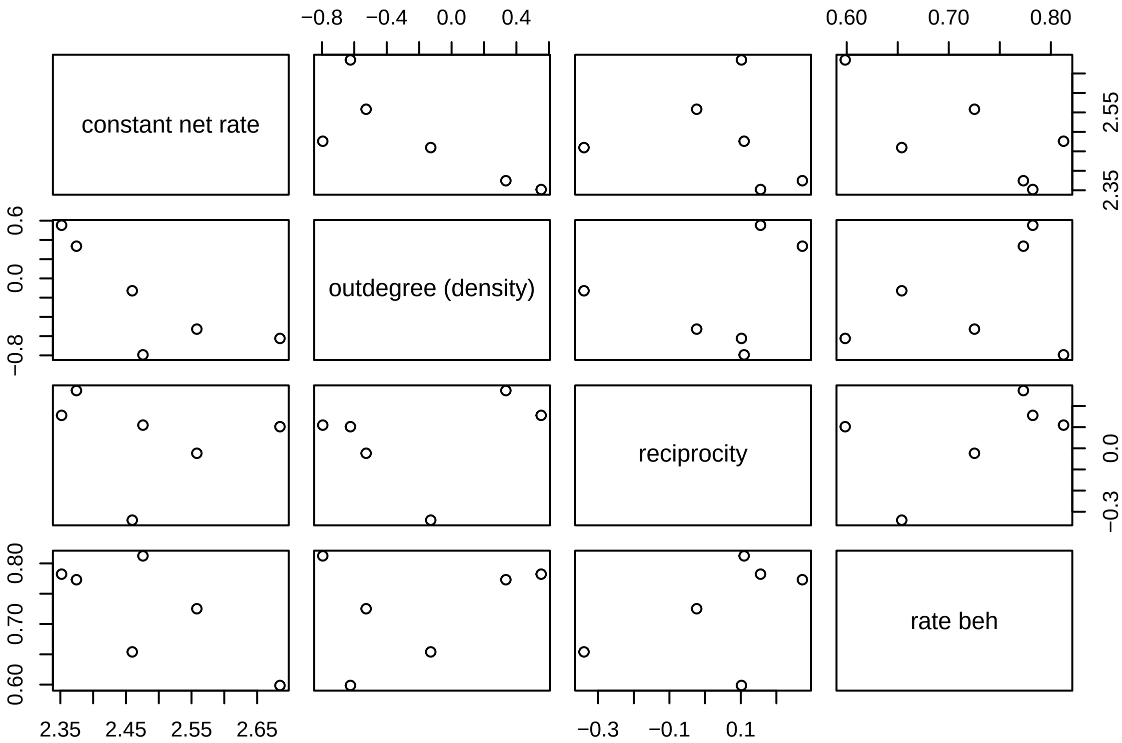
```
## Loading required package: mvtnorm
```

```
M <- 6
thetas <- rmvnorm(M, mean =mu, sigma = Sigma)
```

So, these are M vectors, each with 4 parameters.

```
colnames(thetas) <- colnames(groupModel.e2$ThinParameters[,1,1:4])
```

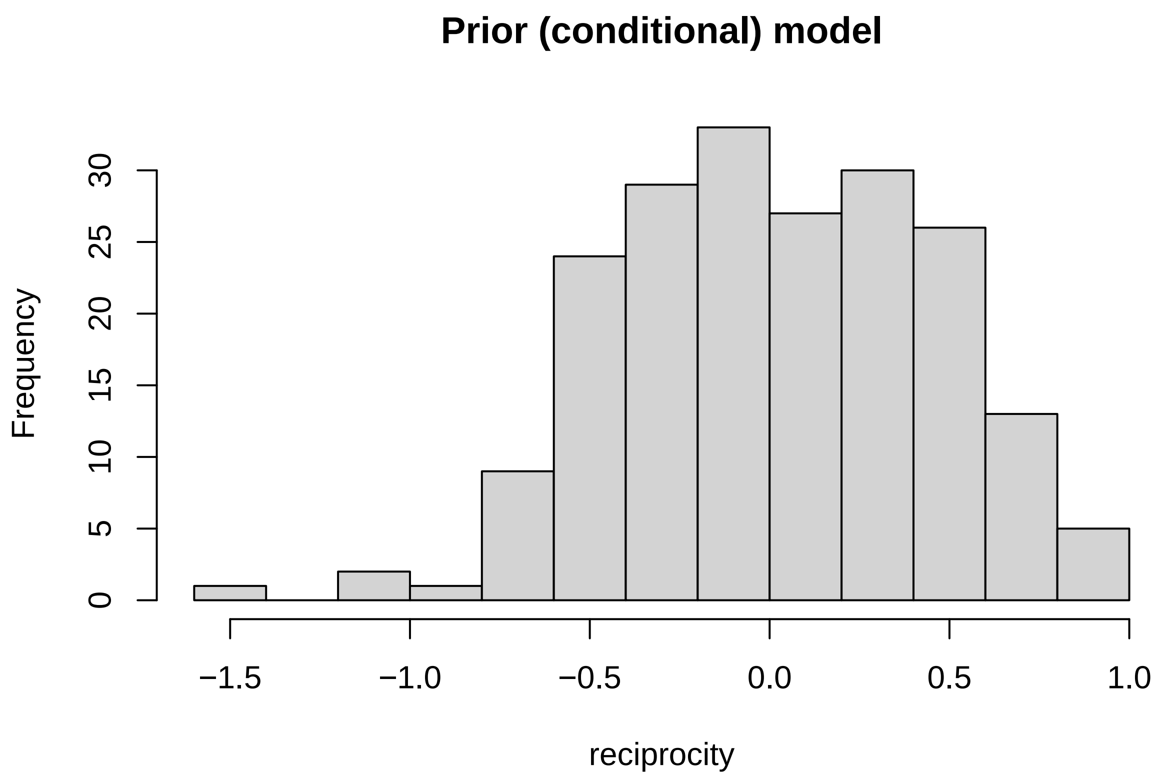
```r
pairs(thetas)
```



We can do this many times to get an idea of the model means. We can also set M large, to get an idea of the distribution

For a given value of μ and Σ, each $\theta_{j,k}$ will be normally distributed. For example, here, the distribution for reciprocity

```r
M <- 200
thetas <- rmvnorm(M, mean =mu, sigma = Sigma)
colnames(thetas) <- colnames(groupModel.e2$ThinParameters[,1,1:4])
k <- 3
hist(thetas[,k],main='Prior (conditional) model',xlab=colnames(thetas)[k])
```



Drawing μ

When we estimate the model, μ and Σ are not fixed and, recall, we are not necessarily interested in the θ_j's, but the parameters μ and Σ. Our prior for Σ is

$$\Sigma \sim InvWish(\Lambda_0, \nu)$$

```
M <- 200

priorSigDraws <- rWishart(M,groupModel.e2$priorDf,
                groupModel.e2$priorSigma)
```

and

$$\mu \mid \Sigma \sim \mathcal{N}_4(\mu_0, \Sigma/\kappa_0)$$

NB: the current default for κ_0 is

```
groupModel.e2$priorKappa
```

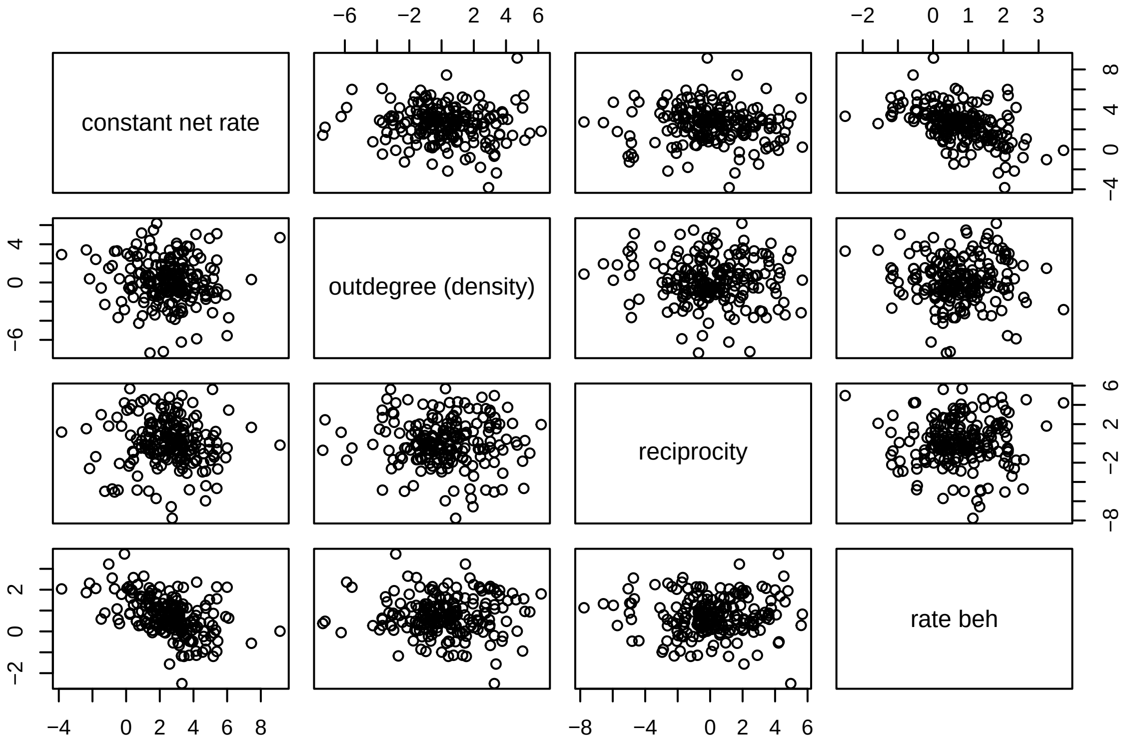
```
## [1] 0
```

For now, let us set it to 1 for the purpose of illustration.

```
priorMuDraws <- matrix(0,M,length(groupModel.e2$priorMu))
colnames(priorMuDraws) <- colnames(groupModel.e2$ThinParameters[,1,1:4])
for (k in c(1:M))
{
  priorMuDraws[k,] <-  rmvnorm(1, mean =groupModel.e2$priorMu, sigma = priorSigDraws[,,k])
}
```

A plot of our prior distribution of μ is given by

```
pairs(priorMuDraws)
```



In practice, people do not investigate their prior but you may in order to get an idea of what you are assuming when estimating the model

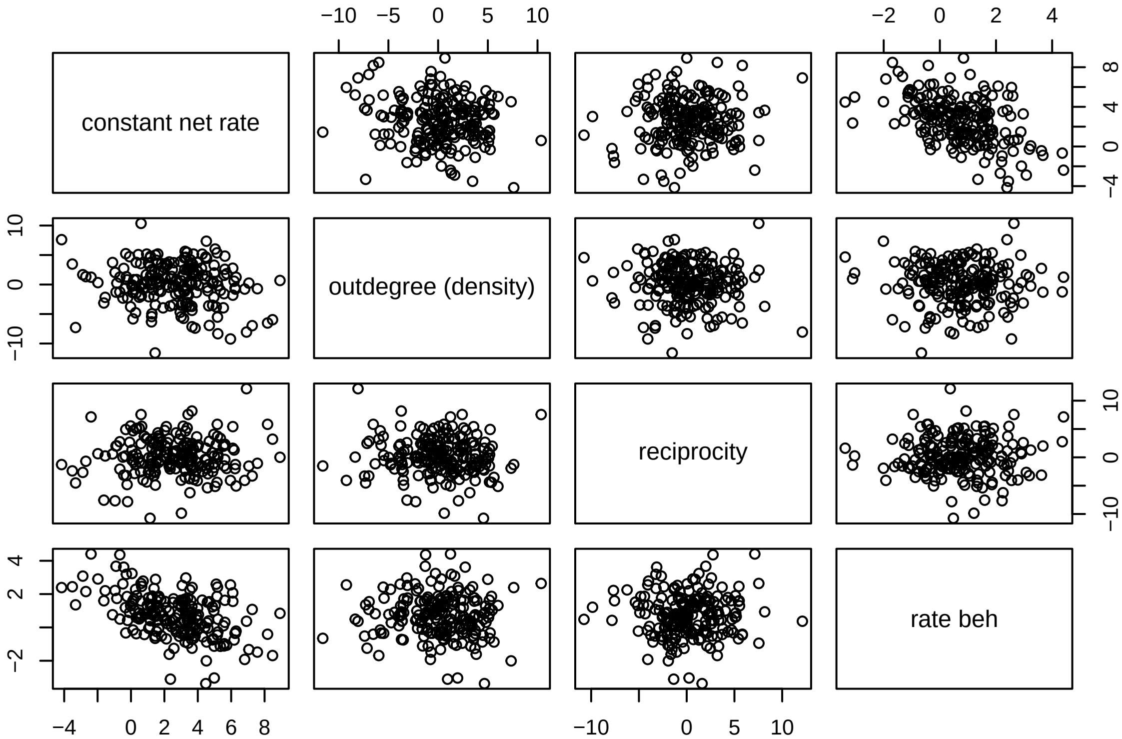
Drawing θ_j unconditionally

Previously, we drew from the distribution of $\theta_j \mid \mu, \Sigma$, i.e. *conditionally* on the unknown parameters. Taking our prior uncertainty about μ and Σ into account, we can now draw θ_j, *un-conditionally* on the unknown parameters

```
thetas <- matrix(0,M,length(groupModel.e2$priorMu))
colnames(thetas) <- colnames(groupModel.e2$ThinParameters[,1,1:4])
for (k in c(1:M))
{
  thetas[k,] <-  rmvnorm(1, mean = priorMuDraws[k,], sigma = priorSigDraws[,,k])
}
```

The uncertainty about μ will propagate into the uncertainty about θ_j

```
pairs(thetas)
```



More and bigger

Load routies

Create two waves for M networks, all of size n

```
n <- 10 # you could make this larger or smaller
M <- 5 # this is very few groups and you could try to produce many
```

The `Karen` dataset has as many groups as you like

```
Karen <- data.set.karen.set.up(n=10,# number of nodes
                               M=5,# number of networks
                               seed=123,nbrNodes=1,nmain=20,nwarm=20)
```

The networks are simulated from a model with effects, that in addition to the defaults have

- `transTrip`
- `simX` with `interaction1 = "beh"`

Now, figure out what inputs you need. The `sienaGroup` object is `Karen$my.Karen`

```
Karen$my.Karen
```

```
## Dependent variables:
## net : oneMode
##  beh : behavior
## Total number of groups: 5
## Total number of periods: 5
```

> For different model specifications, think of what priors you need to specify. Which ones should be random and which fixed?
>
> With the same prior, how does increasing M affect inference?

Increasing the prior variance (Σ), how does that affect inference?

```
groupModel.e <- sienaBayes(Karen$GroupsModel, data = Karen$my.Karen,
                           initgainGlobal=0.1, initgainGroupwise = 0.001,
                           effects = Karen$effects, priorMu = Karen$priorMu, priorSigma = Karen$priorSi
                           priorKappa = 0.01,
                           nwarm=Karen$nwarm, nmain=Karen$nmain, nrunMHBatches=40,
                           nbrNodes=Karen$nbrNodes, silentstart=FALSE)
```