Chapter 1

Examples

In the following section the Bayesian LMG implementation is presented on two examples. The first examples simulates data, the second examples uses real data.

1.1 Simulated Data

We assume a simple model for the first example:

$$
\begin{aligned}
&Y_i \sim \mathcal{N}(\beta_0 + x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + x_4\beta_4, \sigma^2),\\
&\beta_1 = 0.5, \beta_2 = 1, \beta_3 = 2, \beta_4 = 0, \sigma^2 = 1\\
&\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4 \sim \mathcal{N}(0, 1)
\end{aligned}
$$

The values of the four predictors are sampled from a standard normal distribution. These values are then multiplied by the regression coefficients to obtain the dependent variable. A standard normal distributed error is added. Fifty observations were sampled.

The following Code was used to simulate the data :

```
x1 <- rnorm(50, 0, 1); x2 <- rnorm(50, 0, 1)
x3 <- rnorm(50, 0, 1); x4 <- rnorm(50, 0, 1)
#b1 <- 0.5; b2 <- 1; b3 <- 2; b4 <- 0
b1 <- 1; b2 <- 1; b3 <- 1; b4 <- 1

y <- b1*x1 + x2*b2 + b3*x3 + b4*x4 + rnorm(50, 0, 1)

df <- data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, x4 = x4)
```

The model is fitted using the rstanarm package with the default priors for the regression and σ^2 parameters. A burn-in periode of 1000, a sample size of 20000 and a thining of 20 were chosen, resulting in a posterior sample size of 1000. For each posterior sample of the parameters the R^2 value is calculated. The R^2 of the submodels is then calculated by the conditional variance formula for each posterior sample. The thinning makes sense in this case to reduce the computational burden and still obtain an appropriate posterior of the R^2 values (Link and Eaton, 2012).

Table 1.1: Samples from the posterior distributions of the regression parameters

	x1	x2	x3	x4	sigma
sample 1	1.16	1.005	0.849	1.214	0.889
sample 2	1.09	1.125	0.884	1.228	0.859
sample 3	1.27	1.131	0.782	1.152	0.797
sample 4	1.57	0.706	0.940	1.199	0.984
sample 5	1.24	0.967	0.991	1.137	0.852
sample 6	1.37	0.854	0.820	1.127	0.737
sample 7	1.35	0.854	1.113	1.230	0.997
sample 8	1.33	0.847	0.876	1.238	0.880
sample 9	1.15	1.087	0.897	1.274	0.893
sample 10	1.13	1.201	0.908	0.905	0.830

Table 1.2: R^2 for all submodels for the first six posterior samples

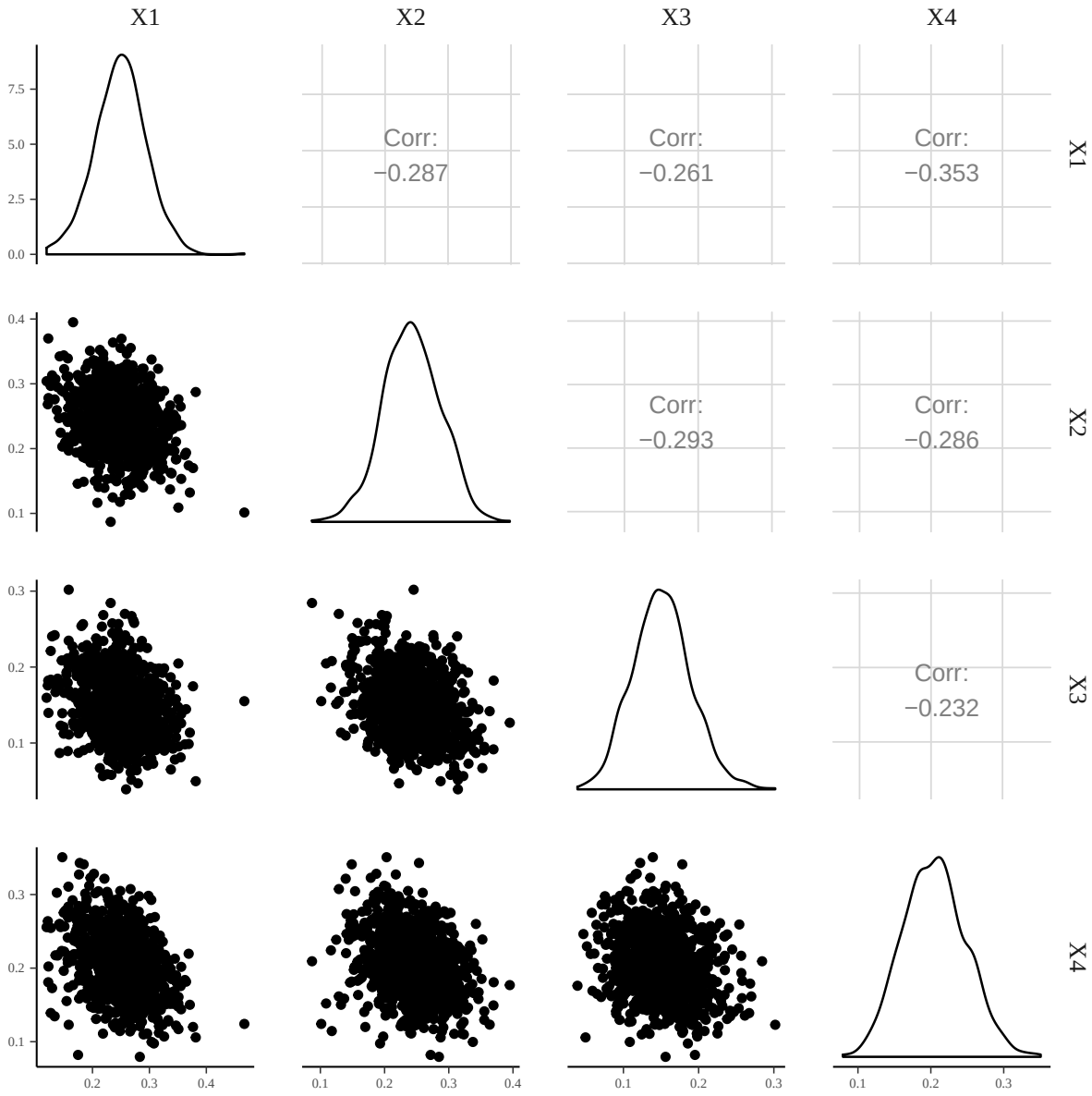
	sample 1	sample 2	sample 3	sample 4	sample 5	sample 6
none	0.000	0.000	0.000	0.000	0.000	0.000
x1	0.268	0.233	0.320	0.406	0.288	0.384
x2	0.131	0.165	0.163	0.052	0.125	0.096
x3	0.115	0.131	0.092	0.089	0.145	0.093
x4	0.390	0.385	0.351	0.353	0.355	0.370
x1 x2	0.409	0.408	0.495	0.465	0.422	0.490
x1 x3	0.464	0.444	0.493	0.585	0.526	0.566
x1 x4	0.587	0.553	0.597	0.676	0.573	0.671
x2 x3	0.221	0.265	0.230	0.127	0.242	0.169
x2 x4	0.556	0.588	0.550	0.427	0.512	0.495
x3 x4	0.466	0.475	0.410	0.410	0.458	0.429
x1 x2 x3	0.572	0.580	0.633	0.625	0.625	0.644
x1 x2 x4	0.758	0.762	0.804	0.754	0.736	0.802
x1 x3 x4	0.726	0.705	0.720	0.805	0.751	0.801
x2 x3 x4	0.608	0.649	0.587	0.469	0.587	0.535
all	0.864	0.875	0.893	0.862	0.877	0.904

```
post2 <- stan_glm(y ~ 1 + x1 + x2 + x3 + x4,
                  data = df,
                  chains = 1, cores =1, iter=40000, thin=20)
```

For each posterior sample the R^2 of the full model and the submodels is calculated via the conditional variance formula. The first few posterior samples are shown in Table 1.1. a the package hier.part is used to calculate the LMG value for each posterior sample.

```r
# prepare data frame for LMG values

LMG.Vals.I<-matrix(0, 4, dim(df.rtwos)[2])

LMG.Vals.J<-matrix(0, 4, dim(df.rtwos)[2])

LMG.Vals.T<-matrix(0, 4, dim(df.rtwos)[2])


for(i in 1:dim(df.rtwos)[2]){

  gofn<-df.rtwos[,i]

  obj.Gelman<-partition(gofn, pcan = 4, var.names = names(df[,2:5]))

  LMG.Vals.I[,i]=obj.Gelman$IJ[,1]
  LMG.Vals.J[,i]=obj.Gelman$IJ[,2]
        LMG.Vals.T[,i]=obj.Gelman$IJ[,3]
}

varnames <- row.names(obj.Gelman$IJ)


# posterior LMG distribution of each variable
quantile(LMG.Vals.I[1,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.157 0.250 0.339

quantile(LMG.Vals.I[2,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.152 0.241 0.326

quantile(LMG.Vals.I[3,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0792 0.1504 0.2330

quantile(LMG.Vals.I[4,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.124 0.205 0.293

quantile(LMG.Vals.J[1,], c(0.025, 0.5, 0.975))
```

```
##   2.5%    50%  97.5%
## 0.0344 0.0515 0.0691

quantile(LMG.Vals.J[2,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0618 0.0803 0.0969

quantile(LMG.Vals.J[3,], c(0.025, 0.5, 0.975))

##    2.5%     50%   97.5%
## -0.0397 -0.0334 -0.0250

quantile(LMG.Vals.J[4,], c(0.025, 0.5, 0.975))

##     2.5%      50%    97.5%
## -0.00203  0.00352  0.00887

quantile(LMG.Vals.T[1,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.198 0.302 0.397

quantile(LMG.Vals.T[2,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.216 0.323 0.419

quantile(LMG.Vals.T[3,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0541 0.1167 0.1935

quantile(LMG.Vals.T[4,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.128 0.208 0.296

# some example how it could be displayed
dat <- data.frame(t(LMG.Vals.I))

pairs.chart <- ggpairs(dat, lower = list(list(combo = "facetdensity")), upper = list(list(c
pairs.chart
```




```
# Comparison to relaimpo package

fit <- lm(y~., data=df)

######## compare to relimp package

run<-boot.relimp(fit, fixed=TRUE)

booteval.relimp(run, bty = "perc", level = 0.95,
                sort = FALSE, norank = TRUE, nodiff = TRUE,
                typesel = c("lmg"))

## Response variable: y
## Total response variance: 4.88
```

```
## Analysis based on 50 observations
##
## 4 Regressors:
## x1 x2 x3 x4
## Proportion of variance explained by model: 86.6%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##      lmg
## x1 0.256
## x2 0.248
## x3 0.153
## x4 0.209
##
## Average coefficients for different model sizes:
##
##       1X   2Xs   3Xs   4Xs
## x1 1.439 1.357 1.284 1.219
## x2 1.239 1.142 1.034 0.915
## x3 0.736 0.813 0.873 0.920
## x4 1.185 1.174 1.172 1.178
##
##
##  Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##                   Lower  Upper
##        percentage 0.95   0.95
## x1.lmg 0.2557     0.1777 0.3500
## x2.lmg 0.2483     0.1701 0.3406
## x3.lmg 0.1527     0.0870 0.2344
## x4.lmg 0.2094     0.1388 0.3002
##
## CAUTION: Bootstrap confidence intervals can be somewhat liberal.
## NOTE: X-matrix has been considered as fixed for bootstrapping.
```

Using the default uninformative priors, the LMG distributions obtained from the Bayesian framework are very similar to the bootstrap confidence intervals of the LMG estimates obtained from the relaimpo package. In both cases fixed regressors are assumed. In the example above the predictors were sampled from a normal distribution. It would therefore be more reasonable to assume stochastic predictors. As noted in ... under the assumption of weak exogeinity and conditional independence the posterior distributions of the regression paramters $\boldsymbol{\beta}$ are valid

Variable	I	J	Total
x1	0.25 (0.157, 0.339)	0.052 (0.034, 0.069)	0.302 (0.198, 0.397)
x2	0.241 (0.152, 0.326)	0.08 (0.062, 0.097)	0.323 (0.216, 0.419)
x3	0.15 (0.079, 0.233)	-0.033 (-0.04, -0.025)	0.117 (0.054, 0.193)
x4	0.205 (0.124, 0.293)	0.004 (-0.002, 0.009)	0.208 (0.128, 0.296)

Table 1.3: Variance decomposition for stochastic predictors using bootstrap. I = LMG values, J = joint contribution, Total = total explained variance in one-predictor only model

for fixed and stochastic predictors. Inference about the covariance matrix can be seen as an independent problem. G recommends in most cases to use the non fixed regressor option when calculating bootstrap confidence intervals. The confidence intervals will then in general be a bit larger. If we want to include this uncertainty in the Bayesian framework, we would need some ideas about the distribution of the predictor variables $\mathbf{X}$. It is then possible to obtain poserior distributions of their corresponding covariance matrix. As a practical solution nonparametric bootstrap may be used to include the uncertainty of the covariance matrix. The following code includes the uncertainty of the stochastic predictors.

```
#Code assuming stochastic predictors
run.stochastic<-boot.relimp(fit, fixed=FALSE)

booteval.relimp(run.stochastic, bty = "perc", level = 0.95,
                sort = FALSE, norank = TRUE, nodiff = TRUE,
                typesel = c("lmg"))

## Response variable: y
## Total response variance: 4.88
## Analysis based on 50 observations
##
## 4 Regressors:
## x1 x2 x3 x4
## Proportion of variance explained by model: 86.6%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##      lmg
## x1 0.256
## x2 0.248
## x3 0.153
## x4 0.209
##
## Average coefficients for different model sizes:
```

```
##
##       1X   2Xs   3Xs   4Xs
## x1 1.439 1.357 1.284 1.219
## x2 1.239 1.142 1.034 0.915
## x3 0.736 0.813 0.873 0.920
## x4 1.185 1.174 1.172 1.178
##
##
##  Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##                   Lower  Upper
##        percentage 0.95   0.95
## x1.lmg 0.2557     0.1198 0.4229
## x2.lmg 0.2483     0.1210 0.3833
## x3.lmg 0.1527     0.0710 0.2835
## x4.lmg 0.2094     0.0630 0.3721
##
## CAUTION: Bootstrap confidence intervals can be somewhat liberal.

#-------------------------------------------------------------------------------------------


df.rtwos.boot <-rtwos.boot(df[,2:5], post.sample, 10)

n.boot = 10

LMG.Vals.I.boot<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))
LMG.Vals.J.boot<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))
LMG.Vals.T.boot<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))

for (b in 1:n.boot){

for(i in 1:dim(df.rtwos.boot)[2]){

        gofn<-df.rtwos.boot[,i,b]

        obj.Gelman<-partition(gofn, pcan = 4, var.names = names(df[,2:5]))

        LMG.Vals.I.boot[,i, b]=obj.Gelman$IJ[,1]
        LMG.Vals.J.boot[,i, b]=obj.Gelman$IJ[,2]
        LMG.Vals.T.boot[,i, b]=obj.Gelman$IJ[,3]
```

```
}

}

quantile(c(LMG.Vals.I.boot[1,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.123 0.245 0.364

quantile(c(LMG.Vals.I.boot[2,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.122 0.263 0.369

quantile(c(LMG.Vals.I.boot[3,1:1000,]), c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0744 0.1498 0.2460

quantile(c(LMG.Vals.I.boot[4,1:1000,]), c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0676 0.1974 0.3295

#very similar values as in the confidence intervals for stochastic predictors


#what if we have prior knowledge
my_prior <- normal(location = c(1, 1,1,1), scale = c(0.01, 0.01,0.01,0.01), autoscale = FALSE)

post2 <- stan_glm(y ~ 1 + x1 + x2 + x3 + x4,
                  data = df, prior = my_prior,
                  chains = 2, cores = 1)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 2.5e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.25 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.093746 seconds (Warm-up)
##                0.064849 seconds (Sampling)
##                0.158595 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 2.1e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.099348 seconds (Warm-up)
##                0.065183 seconds (Sampling)
##                0.164531 seconds (Total)

#posterior sample
post.sample <- as.matrix(post2)

#example of the first 10 posterior samples
```

```
post.sample[1:10,]
```

```
##           parameters
## iterations (Intercept)    x1    x2    x3    x4 sigma
##       [1,]    -0.00684 0.979 0.995 1.015 1.001 0.907
##       [2,]     0.29375 1.018 0.994 0.994 1.006 0.979
##       [3,]    -0.21283 0.984 1.006 1.004 0.996 0.849
##       [4,]     0.08490 1.016 0.994 0.997 0.997 0.898
##       [5,]     0.12661 1.002 1.015 1.004 0.983 0.831
##       [6,]    -0.37427 0.977 0.991 0.997 1.000 0.977
##       [7,]    -0.15713 1.012 0.998 1.003 1.003 0.800
##       [8,]     0.27994 1.002 0.999 0.986 0.997 0.861
##       [9,]    -0.32156 0.998 0.998 1.015 1.009 0.845
##      [10,]     0.28413 0.995 1.008 0.988 0.991 0.901
```

```
#no need for the intercept, last parameter is sigma
post.sample <- post.sample[,-1]


#data frame with all submodels


df.rtwos.boot <-rtwos.boot(df[,2:5], post.sample, 10)

n.boot = 10

LMG.Vals.I.boot.p<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))

for (b in 1:n.boot){

for(i in 1:dim(df.rtwos.boot)[2]){

        gofn<-df.rtwos.boot[,i,b]

        obj.Gelman<-partition(gofn, pcan = 4, var.names = names(df[,2:5]))

        LMG.Vals.I.boot.p[,i, b]=obj.Gelman$IJ[,1]
}

}

quantile(c(LMG.Vals.I.boot.p[1,1:1000,]), c(0.025, 0.5, 0.975))
```

```
##   2.5%    50%  97.5%
```

```
## 0.0954 0.1965 0.2638

quantile(c(LMG.Vals.I.boot.p[2,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.220 0.288 0.347

quantile(c(LMG.Vals.I.boot.p[3,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.149 0.190 0.284

quantile(c(LMG.Vals.I.boot.p[4,1:1000,]), c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0743 0.1574 0.3017
```

Variable	I	J	Total
x1	0.245 (0.123, 0.364)	0.088 (-0.026, 0.159)	0.331 (0.109, 0.505)
x2	0.263 (0.122, 0.369)	0.13 (0.014, 0.217)	0.39 (0.139, 0.581)
x3	0.15 (0.074, 0.246)	-0.009 (-0.074, 0.035)	0.13 (0.044, 0.253)
x4	0.197 (0.068, 0.33)	-0.002 (-0.026, 0.058)	0.203 (0.053, 0.374)

Table 1.4: Variance decomposition. I = LMG values, J = joint contribution, Total = total explained variance in one-predictor only model

In the following example we know that the **X** are coming from a normal distribution. The covariance matrix of the distribution is estimated in a Bayesian way. The package Jags is used.

```
zy = df[,2:5]


#------------------------------------------------------------------------------
# The rest can remain unchanged, except for the specification of difference of
# correlations at the very end.
#------------------------------------------------------------------------------

# Load some functions used below:
# Install the ellipse package if not already:
# Standardize the data:

#zy = apply(y,2,function(yVec){(yVec-mean(yVec))/sd(yVec)})
# Assemble data for sending to JAGS:
dataList = list(
        zy = zy ,
        Ntotal =  nrow(zy) ,
```

```
        Nvar = ncol(zy) ,
        # Include original data info for transforming to original scale:
        # For wishart (dwish) prior on inverse covariance matrix:
        zRscal = ncol(zy) ,  # for dwish prior
        zRmat = diag(x=1,nrow=ncol(zy))  # Rmat = diag(apply(y,2,var))
)

# Define the model:
modelString = "
model {
for ( i in 1:Ntotal ) {
zy[i,1:Nvar] ~ dmnorm( zMu[1:Nvar] , zInvCovMat[1:Nvar,1:Nvar] )
}
for ( varIdx in 1:Nvar ) { zMu[varIdx] ~ dnorm( 0 , 1/2^2 ) }
zInvCovMat ~ dwish( zRmat[1:Nvar,1:Nvar] , zRscal )
# Convert invCovMat to sd and correlation:
zCovMat <- inverse( zInvCovMat )

}
" # close quote for modelString
writeLines( modelString , con="Jags-MultivariateNormal-model.txt" )

# Run the chains:
nChain = 3
nAdapt = 500
nBurnIn = 500
nThin = 10
nStepToSave = 20000
require(rjags)
jagsModel = jags.model( file="Jags-MultivariateNormal-model.txt" ,


## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 50
##    Unobserved stochastic nodes: 5
##    Total graph size: 82
##
## Initializing model

update( jagsModel , n.iter=nBurnIn )
```

```
codaSamples = coda.samples( jagsModel ,




# Convergence diagnostics:
parameterNames = varnames(codaSamples) # get all parameter names


# Examine the posterior distribution:
mcmcMat = as.matrix(codaSamples)
chainLength = nrow(mcmcMat)

covMat <- array(NA, c(4,4,chainLength))

for (i in 1:chainLength){
covMat[1:4,1:4,i]<-matrix(mcmcMat[i,], 4, 4)
}

covMat <- covMat[1:4,1:4,sample(1:20000, replace=F)]


df.rtwos <-rtwos.covm(df[,2:5], post.sample, covMat, 10)


n.boot = 10

LMG.Vals.I.covm<-array(0, c(4,dim(df.rtwos)[2], n.boot))
LMG.Vals.J.covm<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))
LMG.Vals.T.covm<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))


for (b in 1:n.boot){

        for(i in 1:dim(df.rtwos)[2]){

                gofn<-df.rtwos[,i,b]

                obj.Gelman<-partition(gofn, pcan = 4, var.names = names(df[,2:5]))

                LMG.Vals.I.covm[,i, b]=obj.Gelman$IJ[,1]
                LMG.Vals.J.covm[,i, b]=obj.Gelman$IJ[,2]
```

```
                LMG.Vals.T.covm[,i, b]=obj.Gelman$IJ[,3]

        }

}

quantile(c(LMG.Vals.I.covm[1,1:1000,]), c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0998 0.1918 0.2720

quantile(c(LMG.Vals.I.covm[2,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.188 0.261 0.384

quantile(c(LMG.Vals.I.covm[3,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.145 0.206 0.289

quantile(c(LMG.Vals.I.covm[4,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.120 0.175 0.222
```

Variable	I	J	Total
x1	0.192 (0.1, 0.272)	0.045 (-0.059, 0.106)	0.245 (0.042, 0.343)
x2	0.261 (0.188, 0.384)	0.054 (-0.026, 0.167)	0.332 (0.164, 0.521)
x3	0.206 (0.145, 0.289)	-0.025 (-0.12, 0.093)	0.182 (0.035, 0.383)
x4	0.175 (0.12, 0.222)	0.012 (-0.043, 0.09)	0.168 (0.092, 0.289)

Table 1.5: Variance decomposition. I = LMG values, J = joint contribution, Total = total explained variance in one-predictor only model

Using the bootstrap samples of the covariance matrix or samples from the posterior covariance matrix resulted in very similar LMG values. Bootstrap seems to be a valuable option for stochastic predictors when the distribution of the predictors is unknown. Even when the distribution is known the difference seems to be tiny. A benefit of going the full Bayesian way is that we can also include prior knowledge of the covariance matrix. For stochastic predictors the uncertainty about the covariance matrix is reflected in the large credible intervals. Even when we would knew the exact regression parameters, there is alot of uncertainty in the LMG values caused by the uncertainty about the covariance matrix. Code xx shows the uncertainty about the LMG values caused by the uncertainty about the covariance matrix.

Another interesting option in some cases would be to use the shrinkage estimate of the

covariance matrix.

```
#How much variance is effectively in the bootstrap matrix when we know the regression param

#fake post sample

x1 <- rnorm(50, 0, 1); x2 <- rnorm(50, 0, 1)
x3 <- rnorm(50, 0, 1); x4 <- rnorm(50, 0, 1)
#b1 <- 0.5; b2 <- 1; b3 <- 2; b4 <- 0
b1 <- 1; b2 <- 1; b3 <- 1; b4 <- 1

y <- b1*x1 + x2*b2 + b3*x3 + b4*x4 + rnorm(50, 0, 1)

df <- data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, x4 = x4)

post.sample <- matrix(1, 1000,5)

df.rtwos.boot <-rtwos.boot(df[,2:5], post.sample, 10)

n.boot = 10

LMG.Vals.I.boot<-array(0, c(4,dim(df.rtwos.boot)[2], n.boot))

for (b in 1:n.boot){

for(i in 1:dim(df.rtwos.boot)[2]){

        gofn<-df.rtwos.boot[,i,b]

        obj.Gelman<-partition(gofn, pcan = 4, var.names = names(df[,2:5]))

        LMG.Vals.I.boot[,i, b]=obj.Gelman$IJ[,1]
}

}

quantile(c(LMG.Vals.I.boot[1,1:1000,]), c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.142 0.174 0.249

quantile(c(LMG.Vals.I.boot[2,1:1000,]), c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0938 0.1376 0.1696
```

```r
quantile(c(LMG.Vals.I.boot[3,1:1000,]), c(0.025, 0.5, 0.975))
```

```
##  2.5%   50% 97.5%
## 0.128 0.182 0.334
```

```r
quantile(c(LMG.Vals.I.boot[4,1:1000,]), c(0.025, 0.5, 0.975))
```

```
##  2.5%   50% 97.5%
## 0.162 0.277 0.305
```

```r
cov(df[,2:5])
```

```
##        x1      x2       x3       x4
## x1  0.805  0.0370  0.20597 -0.30659
## x2  0.037  0.6865 -0.05344 -0.08940
## x3  0.206 -0.0534  0.71539  0.00982
## x4 -0.307 -0.0894  0.00982  1.50946
```

```r
#Comparison of sample covariance and shrink covariance matrix

cov(df[,2:5])
```

```
##        x1      x2       x3       x4
## x1  0.805  0.0370  0.20597 -0.30659
## x2  0.037  0.6865 -0.05344 -0.08940
## x3  0.206 -0.0534  0.71539  0.00982
## x4 -0.307 -0.0894  0.00982  1.50946
```

```r
cov.shrink(df[,2:5])
```

```
## Estimating optimal shrinkage intensity lambda.var (variance vector): 0.281
##
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.7835
##          x1       x2       x3       x4
## x1  0.79278  0.00806  0.04464 -0.06110
## x2  0.00806  0.70728 -0.01185 -0.01823
## x3  0.04464 -0.01185  0.72804  0.00199
## x4 -0.06110 -0.01823  0.00199  1.29900
## attr(,"lambda")
## [1] 0.783
## attr(,"lambda.estimated")
## [1] TRUE
## attr(,"class")
## [1] "shrinkage"
```

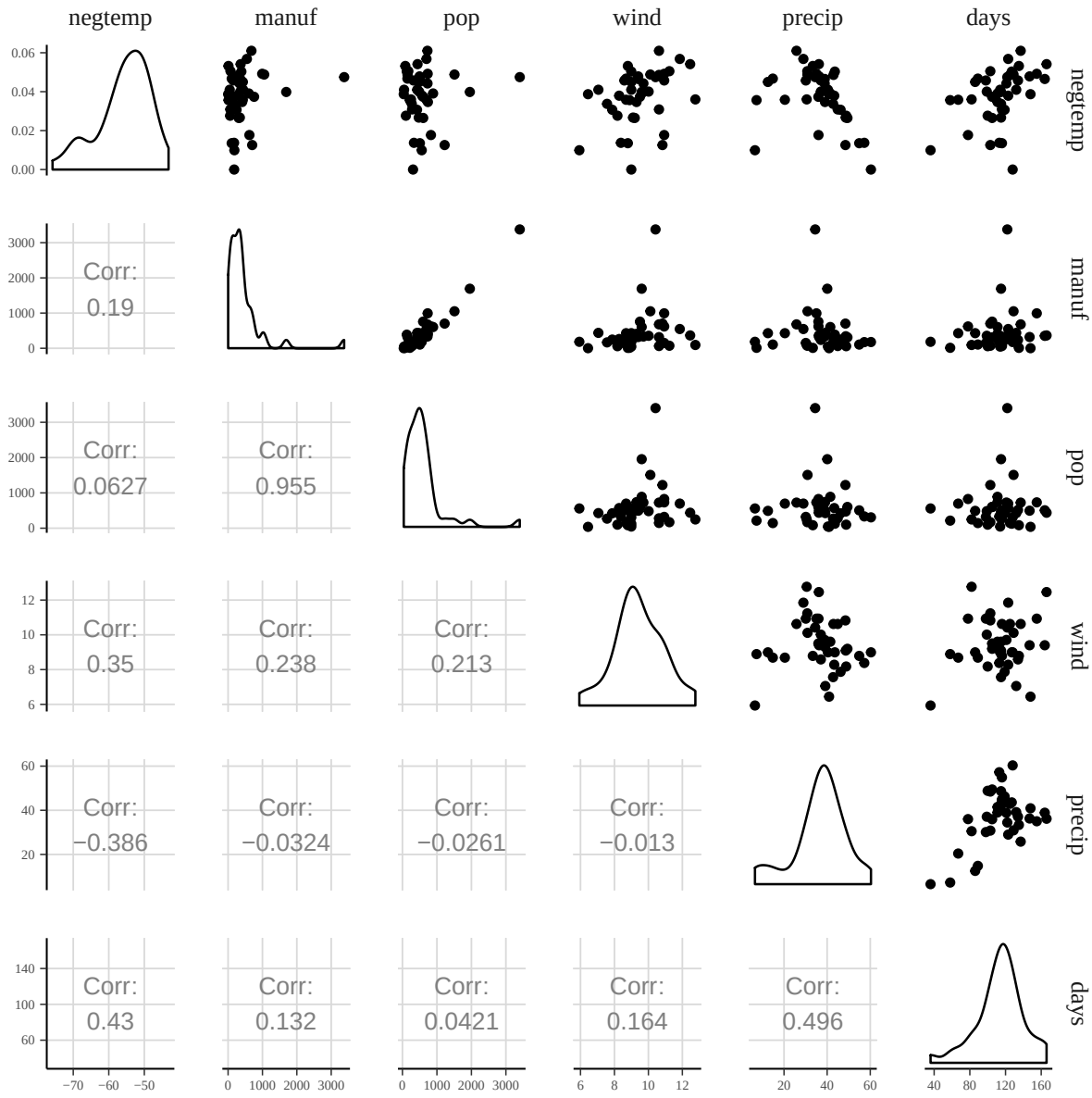
Table 1.6: Variable description of air pollution data

Variable Name	Description	Codes/Values
SO2	sulfur dioxide content of air	micrograms per cubic meter
negtemp	negative value of average	fahrenheit
manuf	number of manufacturing enterprises employing 20 or more workers	integers
pop	population size in thousands (1970 census)	numbers
wind	average wind speed	miles per hour
precip	average annual percipitation	inches
days	average munber of days with precipitation per year	integers

```
## attr(,"lambda.var")
## [1] 0.281
## attr(,"lambda.var.estimated")
## [1] TRUE
```

1.2 Real World Data

The following example data are taken from the book Bayesian Regression Modeling with INLA. The data were about air pollution in 41 cities in the United States originally published in Everitt (2006). The data consits of the SO2 level as the dependent variable and six explanatory variables Two of the explanatory variables are are related to human ecology (pop, manuf) and four others are related to climate (negtemp, wind, precip, days).



A simple linear regression model including the dependent variable SO2 and the six explanatory variables is fitted with the lm command in R. The R^2 of the full model is 0.670.

```
usair.lm <- lm(SO2~., data = usair)


#Code assuming stochastic predictors
run<-boot.relimp(usair.lm, fixed=TRUE)

booteval.relimp(run, bty = "perc", level = 0.95,
                sort = FALSE, norank = TRUE, nodiff = TRUE,
                typesel = c("lmg"))

## Response variable: SO2
## Total response variance: 551
```

```
## Analysis based on 41 observations
##
## 6 Regressors:
## negtemp manuf pop wind precip days
## Proportion of variance explained by model: 67%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##            lmg
## negtemp 0.1111
## manuf   0.2965
## pop     0.1661
## wind    0.0167
## precip  0.0187
## days    0.0604
##
## Average coefficients for different model sizes:
##
##             1X     2Xs      3Xs     4Xs     5Xs     6Xs
## negtemp 1.4081  1.3344  1.28118  1.2577  1.2826  1.2679
## manuf   0.0269  0.0373  0.04643  0.0539  0.0598  0.0649
## pop     0.0200  0.0044 -0.00925 -0.0207 -0.0304 -0.0393
## wind    1.5557 -0.0248 -1.38483 -2.3795 -2.9803 -3.1814
## precip  0.1083  0.1147  0.16427  0.2601  0.4057  0.5124
## days    0.3273  0.2979  0.24111  0.1557  0.0425 -0.0521
##
##
##  Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##                         Lower  Upper
##             percentage 0.95   0.95
## negtemp.lmg 0.1111     0.0417 0.2179
## manuf.lmg   0.2965     0.1960 0.4085
## pop.lmg     0.1661     0.1116 0.2380
## wind.lmg    0.0167     0.0088 0.0678
## precip.lmg  0.0187     0.0082 0.0693
## days.lmg    0.0604     0.0218 0.1468
##
## CAUTION: Bootstrap confidence intervals can be somewhat liberal.
## NOTE: X-matrix has been considered as fixed for bootstrapping.
```

```
bayes.usair <- stan_glm(SO2 ~ . ,
                  data = usair,
                  chains = 4, cores = 4)

#posterior sample
post.sample <- as.matrix(bayes.usair)

#example of the first 10 posterior samples
post.sample[1:10,]

##           parameters
## iterations (Intercept) negtemp  manuf     pop  wind  precip    days sigma
##       [1,]       118.7   1.583 0.0541 -0.0278 -3.93  0.6724  0.0127  14.9
##       [2,]       125.1   1.234 0.0632 -0.0357 -3.64  0.3548 -0.0919  12.0
##       [3,]       111.9   1.266 0.0666 -0.0403 -3.13  0.7228 -0.1107  12.1
##       [4,]       131.3   1.669 0.0530 -0.0340 -2.29  0.5455 -0.0834  15.5
##       [5,]       133.2   1.445 0.0665 -0.0365 -4.23  0.3336 -0.0349  13.4
##       [6,]       166.2   1.800 0.0739 -0.0504 -4.80  0.9482 -0.2504  15.2
##       [7,]        61.7   0.705 0.0471 -0.0223 -1.48 -0.0573  0.1497  16.9
##       [8,]       155.1   1.913 0.0731 -0.0484 -3.35  1.1421 -0.2888  14.4
##       [9,]       174.9   2.186 0.0622 -0.0305 -3.91  0.9105 -0.2629  16.2
##      [10,]       167.6   1.896 0.0630 -0.0393 -3.94  0.7348 -0.2471  18.6

#no need for the intercept, last parameter is sigma
post.sample <- post.sample[,-1]

#data frame with all submodels
df.rtwos <-rtwos(usair[,2:7], post.sample)

df.rtwos[,1:3]

##                     X1      X2      X3
## none           0.00000 0.00000 0.00000
## x1             0.19813 0.19301 0.14768
## x2             0.38052 0.49226 0.45173
## x3             0.23454 0.30945 0.26985
## x4             0.00719 0.00464 0.00856
## x5             0.01546 0.00206 0.01719
## x6             0.20727 0.08160 0.13928
## x1 x2          0.49206 0.58940 0.52002
## x1 x3          0.40724 0.47368 0.39405
## x1 x4          0.20385 0.20134 0.14968
## x1 x5          0.30130 0.21118 0.23953
## x1 x6          0.28350 0.20447 0.20070
```

```
## x2 x3          0.50652 0.64071 0.62351
## x2 x4          0.38459 0.50262 0.45654
## x2 x5          0.40138 0.49277 0.47513
## x2 x6          0.52283 0.53023 0.53416
## x3 x4          0.23488 0.31209 0.27019
## x3 x5          0.25332 0.31040 0.29079
## x3 x6          0.42400 0.37835 0.39351
## x4 x5          0.02293 0.00662 0.02607
## x4 x6          0.20737 0.08206 0.14029
## x5 x6          0.22093 0.12802 0.14315
## x1 x2 x3       0.54928 0.66918 0.63360
## x1 x2 x4       0.52441 0.63451 0.54604
## x1 x2 x5       0.58025 0.60072 0.59605
## x1 x2 x6       0.56069 0.59453 0.55848
## x1 x3 x4       0.43702 0.51516 0.41628
## x1 x3 x5       0.51103 0.49215 0.48653
## x1 x3 x6       0.48821 0.48332 0.44332
## x1 x4 x5       0.31625 0.22360 0.24735
## x1 x4 x6       0.28996 0.21312 0.20305
## x1 x5 x6       0.30497 0.21118 0.23985
## x2 x3 x4       0.51325 0.65549 0.63173
## x2 x3 x5       0.52910 0.64098 0.64905
## x2 x3 x6       0.58931 0.64852 0.65446
## x2 x4 x5       0.40535 0.50315 0.47983
## x2 x4 x6       0.53650 0.54710 0.54626
## x2 x5 x6       0.52568 0.54990 0.53424
## x3 x4 x5       0.25363 0.31306 0.29109
## x3 x4 x6       0.43190 0.38723 0.39916
## x3 x5 x6       0.43243 0.41302 0.39473
## x4 x5 x6       0.22094 0.12802 0.14378
## x1 x2 x3 x4    0.57440 0.70421 0.65071
## x1 x2 x3 x5    0.61703 0.67272 0.68287
## x1 x2 x3 x6    0.60259 0.67024 0.65629
## x1 x2 x4 x5    0.62959 0.65312 0.63623
## x1 x2 x4 x6    0.59358 0.63982 0.58486
## x1 x2 x5 x6    0.58207 0.60118 0.59750
## x1 x3 x4 x5    0.56010 0.54309 0.52461
## x1 x3 x4 x6    0.51928 0.52532 0.46642
## x1 x3 x5 x6    0.51320 0.49242 0.48761
## x1 x4 x5 x6    0.31797 0.22394 0.24840
## x2 x3 x4 x5    0.53572 0.65578 0.65714
## x2 x3 x4 x6    0.60373 0.66649 0.66751
```

```
## x2 x3 x5 x6    0.58932 0.65411 0.66057
## x2 x4 x5 x6    0.54059 0.57027 0.54626
## x3 x4 x5 x6    0.44218 0.42595 0.40099
## x1 x2 x3 x4 x5 0.65740 0.71220 0.71073
## x1 x2 x3 x4 x6 0.62912 0.70551 0.67415
## x1 x2 x3 x5 x6 0.61872 0.67328 0.68451
## x1 x2 x4 x5 x6 0.62961 0.65686 0.64157
## x1 x3 x4 x5 x6 0.56018 0.54619 0.52913
## x2 x3 x4 x5 x6 0.60393 0.67403 0.67221
## all            0.65745 0.71559 0.71560
```

```
# prepare data frame for LMG values

LMG.Vals.I<-matrix(0, 6, dim(df.rtwos)[2])

for(i in 1:dim(df.rtwos)[2]){

  gofn<-df.rtwos[,i]

  obj.Gelman<-partition(gofn, pcan = 6, var.names = names(usair[,2:7]))

  LMG.Vals.I[,i]=obj.Gelman$IJ[,1]
}

# posterior LMG distribution of each variable
quantile(LMG.Vals.I[1,], c(0.025, 0.5, 0.975))
```

```
##   2.5%    50%  97.5%
## 0.0324 0.1035 0.2020
```

```
quantile(LMG.Vals.I[2,], c(0.025, 0.5, 0.975))
```

```
##  2.5%   50% 97.5%
## 0.150 0.264 0.360
```

```
quantile(LMG.Vals.I[3,], c(0.025, 0.5, 0.975))
```

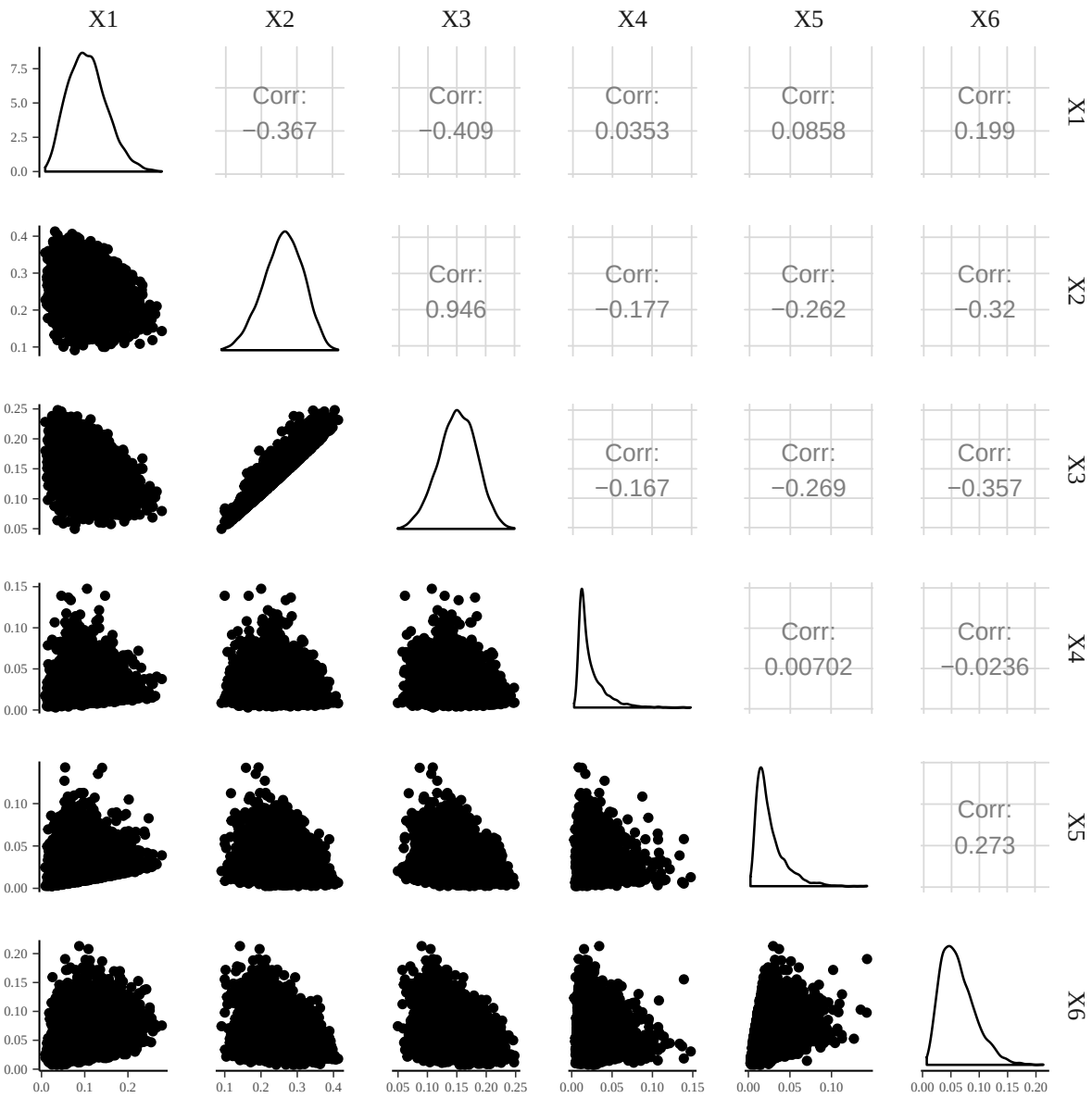
```
##  2.5%   50% 97.5%
## 0.086 0.151 0.212
```

```
quantile(LMG.Vals.I[4,], c(0.025, 0.5, 0.975))
```

```
##    2.5%     50%   97.5%
## 0.00748 0.01737 0.06933
```

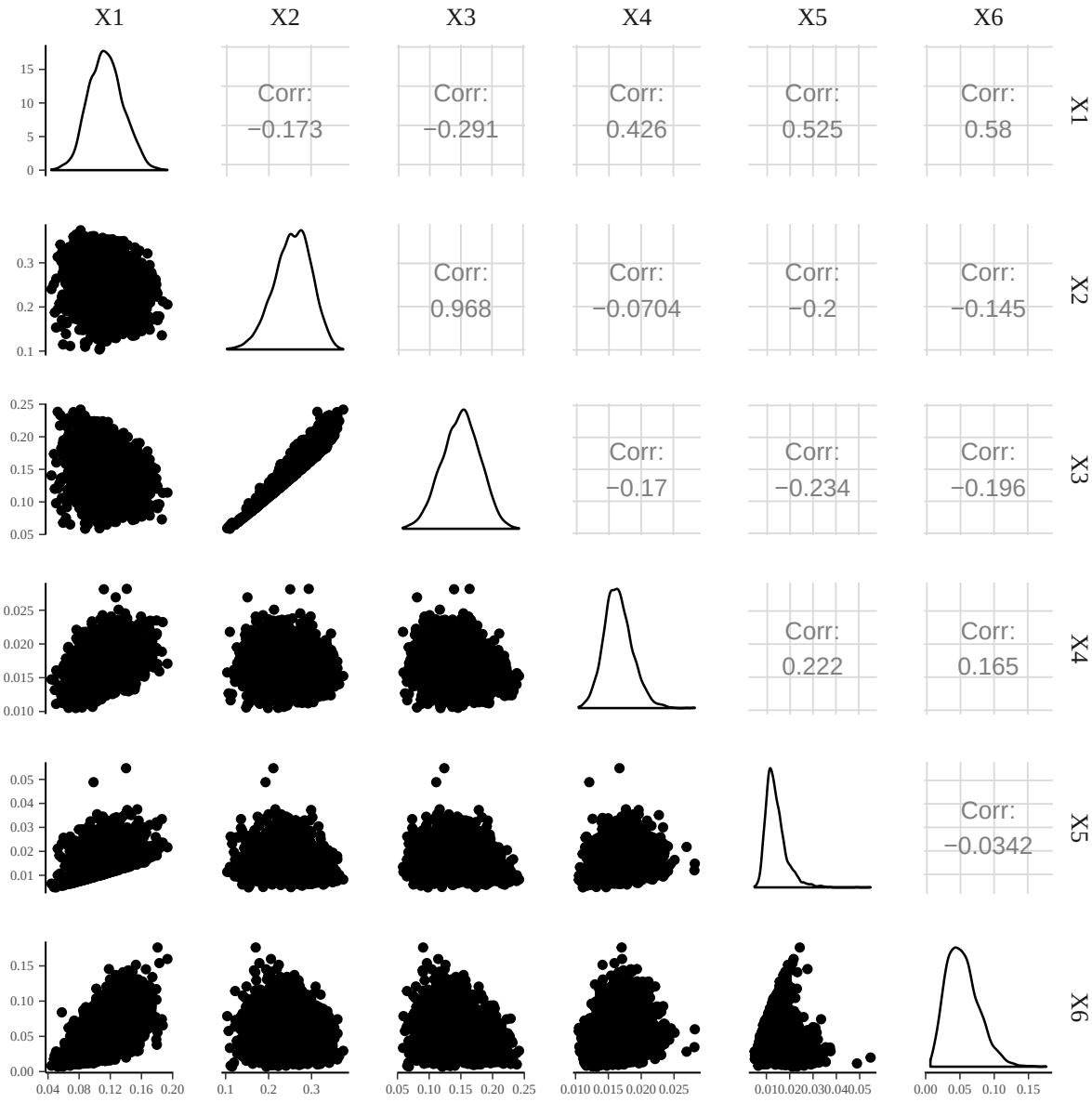
```
quantile(LMG.Vals.I[5,], c(0.025, 0.5, 0.975))
```

```
##    2.5%     50%   97.5%
## 0.00666 0.02069 0.07236

quantile(LMG.Vals.I[6,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0186 0.0578 0.1328

#Visualization
dat <- data.frame(t(LMG.Vals.I))

pairs.chart <- ggpairs(dat, lower = list(list(combo = "facetdensity")), upper = list(list(c
pairs.chart
```



```
#use prior knowledge
my_prior <- normal(location = c(1, 0.05,-0.03,-3, 0, 0), scale = c(0.1, 0.01,0.01,0.1, 0.1, 0.1


bayes.usair <- stan_glm(SO2 ~ . ,
                  data = usair, prior = my_prior,
                  chains = 4, cores = 4)

#posterior sample
post.sample <- as.matrix(bayes.usair)

#example of the first 10 posterior samples
post.sample[1:10,]

##           parameters
## iterations (Intercept) negtemp  manuf     pop  wind   precip   days sigma
##       [1,]        88.2   0.966 0.0525 -0.0289 -3.13  0.07741 0.1127  13.1
##       [2,]        81.7   0.991 0.0571 -0.0289 -3.10  0.03120 0.1967  14.6
##       [3,]        91.7   0.953 0.0508 -0.0297 -3.22  0.05613 0.1542  15.2
##       [4,]        98.1   1.004 0.0578 -0.0309 -2.85  0.03121 0.0247  12.4
##       [5,]        96.0   0.939 0.0641 -0.0406 -3.19  0.13547 0.0864  15.4
##       [6,]        87.8   1.033 0.0589 -0.0353 -3.24  0.22403 0.1259  13.4
##       [7,]        85.4   1.020 0.0634 -0.0376 -3.14  0.21619 0.1312  14.4
##       [8,]        87.0   0.983 0.0548 -0.0326 -3.06  0.14817 0.1268  15.3
##       [9,]       102.9   0.965 0.0451 -0.0271 -3.02 -0.00531 0.0491  14.3
##      [10,]       100.3   0.936 0.0524 -0.0272 -2.96 -0.02459 0.0369  14.4

#no need for the intercept, last parameter is sigma
post.sample <- post.sample[,-1]

#data frame with all submodels
df.rtwos <-rtwos(usair[,2:7], post.sample)


# prepare data frame for LMG values

LMG.Vals.I<-matrix(0, 6, dim(df.rtwos)[2])

for(i in 1:dim(df.rtwos)[2]){

  gofn<-df.rtwos[,i]

  obj.Gelman<-partition(gofn, pcan = 6, var.names = names(usair[,2:7]))
```

```
  LMG.Vals.I[,i]=obj.Gelman$IJ[,1]
}

# posterior LMG distribution of each variable
quantile(LMG.Vals.I[1,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0738 0.1137 0.1593

quantile(LMG.Vals.I[2,], c(0.025, 0.5, 0.975))

##  2.5%   50% 97.5%
## 0.166 0.258 0.332

quantile(LMG.Vals.I[3,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0914 0.1512 0.2056

quantile(LMG.Vals.I[4,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0126 0.0163 0.0211

quantile(LMG.Vals.I[5,], c(0.025, 0.5, 0.975))

##    2.5%     50%   97.5%
## 0.00805 0.01314 0.02565

quantile(LMG.Vals.I[6,], c(0.025, 0.5, 0.975))

##   2.5%    50%  97.5%
## 0.0163 0.0512 0.1076

#Visualization
dat <- data.frame(t(LMG.Vals.I))

pairs.chart <- ggpairs(dat, lower = list(list(combo = "facetdensity")), upper = list(list(c
pairs.chart
```


X1
X2
X3
X4
X5
X6
Corr: −0.173
Corr: −0.291
Corr: 0.426
Corr: 0.525
Corr: 0.58
Corr: 0.968
Corr: −0.0704
Corr: −0.2
Corr: −0.145
Corr: −0.17
Corr: −0.234
Corr: −0.196
Corr: 0.222
Corr: 0.165
Corr: −0.0342

Bibliography

Link, W. A. and Eaton, M. J. (2012). On thinning of chains in MCMC. *Methods in Ecology and Evolution*, **3**, 112–115. 1