

First line of title
second line of title

Master Thesis in Biostatistics (STA495)

by

Name of student
Matriculation number

supervised by

Name of responsible supervisor (with title)
Name of supervisor (with title and affiliation if external)

Zurich, month year

p-values:
their use, abuse and proper use
illustrated with seven facets

Mäxli Musterli

Version July 25, 2018

Contents

Preface	iii
1 Introduction	1
1.1 Bayesian Regression	6
2 Data	9
3 Examples	11
3.1 Simulated Data	11
3.2 Real World Data	26
4 Discussion and Outlook - Some extenstions	37
4.1 random intercept model	37
4.2 marginal model	42
5 Other variable importance metrics in the Bayesian framework	49

Preface

Howdy!

Max Muster
June 2018

Chapter 1

Introduction

The aim of this master thesis is to implement the variable importance measure LMG (named after the authors Lindeman, Merenda, and Gold ref groemping 2007) in linear models estimated with Bayesian methods.

Regression models are popular in many applied research areas. These models provide a tool to find an association between a response variable Y and a set of explanatory variables. These explanatory variables are also called predictors or covariates. Regression parameters provide us the information how much the response variable is expected to change when a predictor changes by one unit, given all other predictors in the model stay the same. The last subsentence is very important for the correct interpretation of the regression parameters. It shows also that the parameter value of a predictor is dependent on the other predictors in the model. Because predictors are often correlated to some degree to each other, it is clear that it is not an easy task to find the most important predictors in a model. The first question then is: What do we mean by the importance of a predictor? A question that is not easy answered and depending on the research question. Goering 2015 concludes that there may never be a unique accepted definition of what variable importance is. Different metrics exist to quantify the importance of predictors. These metrics focus on different aspects and with correlated predictors they lead to different conclusions. A distinction should be made between the importance of predictors in regression models that are used to predict future data and regression models who wish to find an association between predictors and the response variable. In the former case, the aim is only to reduce the error between the predicted values and the real observed values. It does not really matter how we get there. In the other case, we are interested in the strength of the relationship between the predictors and the response variable. A predictor may explain little of the response variable given two other correlated predictors are already included in a regression model. However, this from the regression output unimportant predictor may be the main cause of the other two predictor values. It therefore may somehow be the most important predictor in this model. The causal relationship between the variables is missing in the regression model. Regressing conditional on other variables or using univariate regression models only provide us some parts of the bigger picture about the predictor in a model. Some authors recommend that the variable importance metric is based on both components. Which variable importance metrics are the most useful ones is still an open debate. A convincing theoretical basis is still lacking for all of them. A summary of the metrics can be found in ref (groemping, 2015). Groemping 2015 recommends to use the existing best practices, until a more profound solution is found. For variance (or generally goodness of fit) decomposition based importance she recommend to use LMG enhanced with joint contributions or dominance analysis.

The focus of this master thesis is on the LMG variable importance metric. The LMG is a metric that is based on variance decomposition. The total R^2 of a model is decomposed onto the predictors. Many authors call this a desirable property of a variable importance metric. Marginal and conditional information are incorporated. The following formulas are taken from

Groemping 2015. The same mathematical notations are used.

The following notations for the explained variance (1) and sequentially added variance (2) simplify the notation of the LMG formula.

$$\text{evar}(S) = \text{Var}(Y) - \text{Var}(Y \mid X_j, j \in S), \quad (1.1)$$

$$\text{svar}(M \mid S) = \text{evar}(M \cup S) - \text{evar}(S), \quad (1.2)$$

, where S and M denote disjoint sets of predictors.

$R^2(S)$ can be written as $\text{evar}(S)/\text{Var}(Y)$.

The LMG formula is given below for the first predictor only. Because of exchangeable predictors, this is no loss of generality.

$$\begin{aligned} \text{LMG}(1) &= \frac{1}{p!} \sum_{\pi \text{permutation}} \text{svar}(\{1\} \mid S_1(\pi)), \\ &= \frac{1}{p!} \sum_{S \subseteq \{2, \dots, p\}} n(S)! (p - n(S) - 1)! \text{svar}(\{1\} \mid S) \\ &= \frac{1}{p} \sum_{i=0}^{p-1} \left(\sum_{\substack{S \subseteq \{2, \dots, p\} \\ n(S)=1}} \text{svar}(\{1\} \mid S) \right) / \binom{p-1}{i} \end{aligned} \quad (1.3)$$

$$= \frac{1}{p} \sum_{i=0}^{p-1} \frac{\sum_{\substack{S \subseteq \{2, \dots, p\} \\ n(S)=1}} \text{svar}(\{1\} \mid S)}{\binom{p-1}{i}} \quad (1.4)$$

$$(1.5)$$

, where $S_1(\pi)$ is the set of predecessors of predictor 1.

The different writings of the formulas help to better grasp what is calculated in the LMG metric. The R^2 of the model including all predictors is decomposed. In the top formula the LMG value of predictor 1 is represented as an unweighted average over all orderings of the sequential added variance contribution of predictor 1. The middle formula shows that the calculation can be done computationally more efficient. The orderings with the same set of predecessors S are combined into one summand. Instead of $p!$ summands only 2^{p-1} summands need to be calculated. The bottom formula shows that the LMG metric can also be seen as the unweighted average over average explained variance improvements when adding predictor 1 to a model of size i without predictor 1.

The LMG metric is implemented in the R packages `relaimpo` and `hier.part`.

Chevan and Sutherland propose that instead of the variances an appropriate goodness-of-fit metric can be used in the LMG formula. They name their proposal hierarchical partitioning. The requirements are simply: an initial measure of fit when no predictor variable is present, a final measure of fit when N predictor variables are present, all intermediate models when various combinations of predictor variables are present. Hierarchical partitioning is implemented in the `hier.part` package. The LMG component of each variable is named independent component (I). The sum of the independent components (I) results then in the overall goodness-of-fit metric. The difference between the goodness-of-fit when only the predictors itself is included in the model, compared to its independent component (I) is named the joint contribution (J).

For the linear model the R^2 is the most widely used goodness-of-fit metric. Different formulas for R^2 exist (Kvalseth, 1985), all leading to the same value when an intercept is included and the model is fitted by maximum likelihood.

Two popular definitions are:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (1.6)$$

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (1.7)$$

, where $\hat{y}_i = E(y | X_n, \hat{\theta})$.

When other estimation methods than maximum likelihood are used (4) can be negative and (5) can be bigger than 1. When the model is fitted by Bayesian methods and estimates of the posteriors are employed this can easily happen (Gelman). A model that explains more than 100% of the variance does not make sense. A negative R^2 is also difficult to interpret. A negative R^2 may be interpreted as a fit that is worse than the mean of the data. When new data from a test set is predicted (e.g. in leave-one-out crossvalidation), one has to be careful with the R^2 definition used. R^2 should then be calculated by equation(4). Negative R^2 can then also happen, when the model is fitted by maximum likelihood (Referenz Beware of R2). Because of R^2 bigger than one or the possibility of negative R^2 values Gelman(2018) proposes to use:

$$R_{Gelman}^2 = \frac{V(\sum_{n=1}^N \hat{y}_n^s)}{V(\sum_{n=1}^N \hat{y}_n^s) + V(\sum_{n=1}^N e_n^s)} \quad (1.8)$$

, where $\hat{y}_n^s = E(y | X_n, \theta^s)$ and the vector of errors $e_n^s = y_n - \hat{y}_n^s$, V stands for the variance defined as $V(\sum_{n=1}^N z_n) = \sum_{n=1}^N (z_n - \bar{z})^2 / (N - 1)$ for any vector z and $\theta^s, s = 1, \dots, S$ are draws from the posterior parameter distribution. The formula is then guaranteed to be between 0 and 1. However, they argue, that we can no longer interpret an increase in R^2 as a improved fit to a fixed target because the denominator of R^2 is no longer fixed. Especially for the LMG formula this may be problematic. The R^2 can then be interpreted as a data-based estimate of the proportion of variance explained for new data under the assumption that the predictors are held fixed.

The variance of the linear model can be written as

$$\text{Var}(y) = \boldsymbol{\beta}^\top \boldsymbol{\Sigma}_{\mathbf{XX}} \boldsymbol{\beta} + \sigma^2, \quad (1.9)$$

where

$\boldsymbol{\beta}^\top = (\beta_1 \dots \beta_p)$ are the regression parameters without the intercept. $\boldsymbol{\Sigma}_{\mathbf{XX}}$ is the covariance matrix of the regressors.

Writting it this way makes it clear that the predictors in the Gelman equation can also be taken as random (Gelman 2017).

Another possible R^2 for the classical linear model would be to sample the σ parameter from the posterior distribution instead of defining it as in equation (6). This would then yield to the following R^2 definition:

$$R_{Snyder}^2 = \frac{V(\sum_{n=1}^N \hat{y}_n^s)}{V(\sum_{n=1}^N \hat{y}_n^s) + \sigma^s} \quad (1.10)$$

, where $\hat{y}_n^s = E(y | X_n, \theta^s)$, V stands for the variance and $\theta^s, s = 1, \dots, S$ are draws from the posterior parameter distribution.

As a side note: Instead of using $V(\sum_{n=1}^N e_n^s)$ we could also use $\sum(y - \hat{y}^s)^2 / (n - 1)$ as an estimate for the error. For the maximum likelihood estimate $\text{var}(y_i - \hat{y}_i) = \sum(y_i - \hat{y}_i)^2 / (n - 1)$. This is because the mean of the residuals is 0. When the samples of the posterior parameters

are used instead, the mean of the residuals is not zero. $\text{var}(y_i - \hat{y}_i) = \sum(y_i - \hat{y}_i)^2/(n - 1)$ is than a little bit bigger than $\text{var}(y_i - \hat{y}_i)$. In practice the values should only differ by a very small amount. We do not expect the errors to have a systematic bias. However, the residuals are just a sample of the error. The mean of the residuals must not be exactly 0 when the samples of the posteriors are used for the regression coefficients. When we use $\text{var}(y_i - \hat{y}_i)$, we do not include this uncertainty and fix the mean to 0. In my understanding, because we do not estimate the mean and take the posterior sample as the possible true regression parameter we would not need to divide by $n - 1$ but only by n . Because this is also true for the fixed effects it does not matter if we divide by $n - 1$ or by n as long as we use the same for the fixed effects and the error term. However, when we think of the Bayes R^2 as an estimate of goodness-of-fit for new data negative values may not be something that should be strictly avoided. As in the case of predicting new data from a test set, equation (4) can reasonably be applied in the Bayes case, leading to:

$$R^2 = 1 - \frac{\sum_{n=1}^N (y - \hat{y}_n^s)^2}{\sum_{n=1}^N (y - \bar{y})^2}, \quad (1.11)$$

, where $\hat{y}_n^s = E(y | X_n, \theta^s)$ and $\theta^s, s = 1, \dots, S$ are draws from the posterior parameter distribution.

For two predictors equation 6 simplifies to

$$\text{Var}(y) = \beta_1^2 \text{Var}(X_1) + 2\beta_1\beta_2 \text{Cov}(X_1, X_2) + \beta_2^2 \text{Var}(X_2) + \sigma^2, \quad (1.12)$$

A benefit of writing the variance in this way, is that it is in the linear model possible to calculate the explained variance of the submodels from the full model including all predictors.

When predictor X_1 is alone in the model the explained variance includes the variance of the predictor itself, the whole covariance term and in addition some of the contribution of the variance of X_2 in equation 7. In mathematical notation that is

$$\text{svar}(X_1 | \emptyset) = \beta_1^2 \text{Var}(X_1) + 2\beta_1\beta_2 \text{Cov}(X_1, X_2) + \beta_2^2 \text{Var}(X_2)\rho_{12}^2 \quad (1.13)$$

The contribution of the second regressor is then simply the difference to the total explained variance.

In the general case with p regressors, the conditional variance formula can be used to calculate the R^2 of all submodels.

As an example the conditional distribution of a normal distribution: The elements of the vector \mathbf{Y} are reordered as

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix}, \mathbf{Y}_1 \in \mathbb{R}^q, \mathbf{Y}_2 \in \mathbb{R}^{p-q}.$$

The joint distribution is a multivariate normal distribution with elements

$$\begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}\right), \boldsymbol{\Sigma}_{21} = \boldsymbol{\Sigma}_{12}^T,$$

the conditional distribution is normally distributed again with mean

$$E(\mathbf{Y}_1 | \mathbf{y}_2) = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{y}_2 - \boldsymbol{\mu}_2)$$

and the conditional variance is

$$\text{Var}(\mathbf{Y}_1 | \mathbf{y}_2) = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}.$$

The conditional variance formula can be used to calculate the R^2 of the submodels. Lets assume we are interested in the R^2 of a model containing the predictors $\mathbf{X}_{q...p}$, and regression coefficients $\boldsymbol{\beta}^\top = (\beta_1 \dots \beta_p)$ without the intercept. The regression coefficients are further separated in $\boldsymbol{\beta}_{1\dots q-1}^\top = (\beta_1 \dots \beta_{q-1})$ and $\boldsymbol{\beta}_{q\dots p}^\top = (\beta_q \dots \beta_p)$.

As in the normal distribution example above we have the covariance matrix of p predictors written as

$$\boldsymbol{\Sigma}_{\mathbf{XX}} = \text{Cov}(\mathbf{X}) = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}^{p \times p}, \quad (1.14)$$

$$\boldsymbol{\Sigma}_{11} = \text{Cov}(\mathbf{X}_{1\dots q-1}, \mathbf{X}_{1\dots q-1}), \quad (1.15)$$

$$\boldsymbol{\Sigma}_{12} = \text{Cov}(\mathbf{X}_{1\dots q-1}, \mathbf{X}_{q\dots p}), \quad (1.16)$$

$$\boldsymbol{\Sigma}_{22} = \text{Cov}(\mathbf{X}_{q\dots p}, \mathbf{X}_{q\dots p})$$

The conditional variance of the predictors $\mathbf{X}_{1\dots q-1}$ given the predictors $\mathbf{X}_{q\dots p}$ is then

$$\text{Cov}(\mathbf{X}_{1\dots q-1} \mid \mathbf{x}_{q\dots p}) = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \quad (1.17)$$

The total explained variance of the model including predictors $\mathbf{X}_{1\dots p}$ omits simply the σ^2 parameter in equation , which is

$$\text{evar}(\mathbf{X}_{1\dots p}) = \boldsymbol{\beta}^\top \boldsymbol{\Sigma}_{\mathbf{XX}} \boldsymbol{\beta}. \quad (1.18)$$

The explained variance of a submodel can be calculated by subtracting from the total explained variance, the variance of the not-in-the-model-included-predictors that is not explained by in-the-model-included-predictors. The variance that is not explained by in-the-model-included-predictors is given by the variance of the not-in-the-model-included predictors conditional on the in-the-model-included-predictors. The explained variance of a submodel containing predictors $\mathbf{X}_{q\dots p}$ can therefore be written as

$$\text{evar}(\mathbf{X}_{q\dots p}) = \text{evar}(\mathbf{X}_{1\dots p}) - \boldsymbol{\beta}_{1\dots q-1}^\top \text{Cov}(\mathbf{X}_{1\dots q-1} \mid \mathbf{x}_{q\dots p}) \boldsymbol{\beta}_{1\dots q-1}. \quad (1.19)$$

To gain the the R^2 value of the submodel we need to divide the explained variance by the total variance,

$$\text{evar}(\mathbf{X}_{q\dots p}) / \text{Var}(\mathbf{Y}). \quad (1.20)$$

The LMG formula requires calculation of the R^2 values for all $2^p - 1$ submodels.

In the Bayesian setting we do have a whole probability distribution for each regression parameter. We can sample the regression parameters from the posterior joint distribution of the fullmodel and use the conditional variance formula to calculate the explained variance of all submodels for each parameter sample. As Gelman notes their R^2 can no longer be interpreted as a fit to a fixed target. For the LMG formula this may be problematic. However, using the conditional variance formula to calculate the R^2 of the submodels, the same total variance is used for a sample of the joint posterior distribution. The important property that all shares should be non-negative and the dependence of the submodels to each other is then respected for each sample. With dependence, i mean the interconnection of the R^2 values of the submodels. As an example for a violation of this interconnection lets assume we have uncorrelated predictors. Instead of fitting the full model and use the conditional mean formula to get the R^2 of the

submodels, it would be possible to fit a separate Bayesian model for each submodel. The LMG values could then be built by sampling a parameter from each submodel. The problem is then that the parameter values change in each submodel, even if the predictors are uncorrelated. We would have many possibly true parameter values of a predictor in the same LMG comparison. It would then also be possible that the R^2 decreases when adding predictors. A further of fitting one full model only is that we only need to fit one Bayesian model including all predictors. This makes it possible to calculate the LMG values also in the Bayesian framework in a reasonable amount of time.

1.1 Bayesian Regression

This Section provides a short introduction Bayesian regression and about some assumptions. It is summarized from the book (Bayesian Analysis for the Social Sciences, 2009). In regression analysis we are interested in the dependence of \mathbf{y} on \mathbf{X} . The conditional mean of a continuous response variable $\mathbf{y} = (y_1, \dots, y_n)^\top$ is related to a $n \times k$ predictor matrix \mathbf{X} via a linear model,

$$\mathbb{E}(\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}) = \mathbf{X}\boldsymbol{\beta}, \quad (1.21)$$

where $\boldsymbol{\beta}$ is a $k \times 1$ vector of unknown regression coefficients.

Under some assumptions about the density, conditional independence and homoskedastic variances, the regression can be written as

$$\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n) \quad (1.22)$$

Under the assumption of weak exogeneity and conditional independence the joint density of the data can be written as

$$p(\mathbf{y}, \mathbf{X} | \boldsymbol{\theta}) = p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}_{y|x}) p(\mathbf{X} | \boldsymbol{\theta}_x), \quad (1.23)$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_{y|x}, \boldsymbol{\theta}_x)^\top$. The weak exogeneity assumption implicates that the whole information about y_i is contained in x_i and $\boldsymbol{\theta}_{y|x}$. Knowledge of the parameters $\boldsymbol{\theta}_{x_i}$ provides no additional information about y_i . The interest of regression is mostly in the posterior parameters $\boldsymbol{\theta}_{y|x}$. These posterior densities are proportional to likelihood of the data multiplied by the prior density. The joint density $p(\mathbf{y}, \mathbf{X} | \boldsymbol{\theta})$ is used to learn about the posterior parameters, via Bayes Rule

$$p(\boldsymbol{\theta} | \mathbf{y}, \mathbf{X}) \propto p(\mathbf{y}, \mathbf{X} | \boldsymbol{\theta}) p(\boldsymbol{\theta}). \quad (1.24)$$

The dependence of \mathbf{y} on \mathbf{X} is captured in the parameters $\boldsymbol{\theta}_{y|x} = (\boldsymbol{\beta}, \sigma^2)$. Under the assumption of independent prior densities about $\boldsymbol{\theta}_{y|x}$ and $\boldsymbol{\theta}_x$ the posterior distribution of the parameters can be written as

$$p(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}_x | \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}, \sigma^2)}{p(\mathbf{y} | \mathbf{X})} \times \frac{p(\mathbf{X} | \boldsymbol{\theta}_x) p(\boldsymbol{\theta}_x)}{p(\mathbf{X})} \quad (1.25)$$

This factorization shows that under the above mentioned assumptions the posterior inference about the parameters $\boldsymbol{\theta}_{y|x} = (\boldsymbol{\beta}, \sigma^2)$ is independent from the inference about $\boldsymbol{\theta}_x$ given data \mathbf{X} . This also means that the assumptions about \mathbf{X} being fixed or stochastic result in the same posterior density of $\boldsymbol{\theta}_{y|x}$. In the case of fixed regressors $p(\mathbf{X})$ and $\boldsymbol{\theta}_x$ drop out of the calculations.

For stochastic predictors it means that given \mathbf{X} nothing more can be gained about $\theta_{y|x} = (\boldsymbol{\beta}, \sigma^2)$ from knowing $\boldsymbol{\theta}_x$.

In regression the focus is on $\boldsymbol{\theta}_{y|x} = (\boldsymbol{\beta}, \sigma^2)$, for which under some assumptions it does not matter whether we assume fixed or stochastic predictors. For the LMG formula the variance of the predictors is also incorporated. The LMG formula may be especially interesting for continuous predictors, which in most cases are stochastic. For stochastic predictors the information about $\boldsymbol{\theta}_x$ would therefore also be relevant. As seen in equation ... inference about $\boldsymbol{\theta}_x$ is independent from inference about $\boldsymbol{\theta}_{y|x}$. If we have stochastic predictors and ignore dependence we just use an estimate of the covariance matrix and do not incorporate this uncertainty. Because the explained variance is calculated by $\boldsymbol{\beta}^\top \boldsymbol{\Sigma}_{\mathbf{XX}} \boldsymbol{\beta}$, inference about $\boldsymbol{\theta}_x$ seems to be equally important as inference about $\boldsymbol{\theta}_{y|x}$ for stochastic predictors. If we know the distribution of the $p(\mathbf{X})$ the $\boldsymbol{\theta}_x$ could be estimated. However, the computation times are then much higher. We would need to do the whole LMG calculation for each posterior covariance sample of the predictors. Depending on the number of predictors this would quite take some time. In most cases the problem is that we do not know the distribution of the \mathbf{X} . As a practical solution we could then use nonparametric bootstrapping to include the uncertainty of the stochastic predictors in the LMG formula. We would then also need to do the whole LMG calculations for each bootstrap sample of the covariance matrix. There exist also different covariance estimator. The shrinkage method may be an interesting estimator with some interesting properties.

Chapter 2

Data

Maybe it is the methods section. Here however, we give a couple hints. Note that you can wisely use `preamble`-chunks. Minimal, is likely:

```
library(knitr)
opts_chunk$set(
  fig.path='figure/ch02_fig',
  self.contained=FALSE,
  cache=TRUE
)
```

Defining figure options is very helpful:

```
library(knitr)
opts_chunk$set(fig.path='figure/ch02_fig',
              echo=TRUE, message=FALSE,
              fig.width=8, fig.height=2.5,
              out.width='\\textwidth-3cm',
              message=FALSE, fig.align='center',
              background="gray98", tidy=FALSE, #tidy.opts=list(width.cutoff=60),
              cache=TRUE
)
options(width=74)
```

Notice how in Figure 2.1 everything is properly scaled.

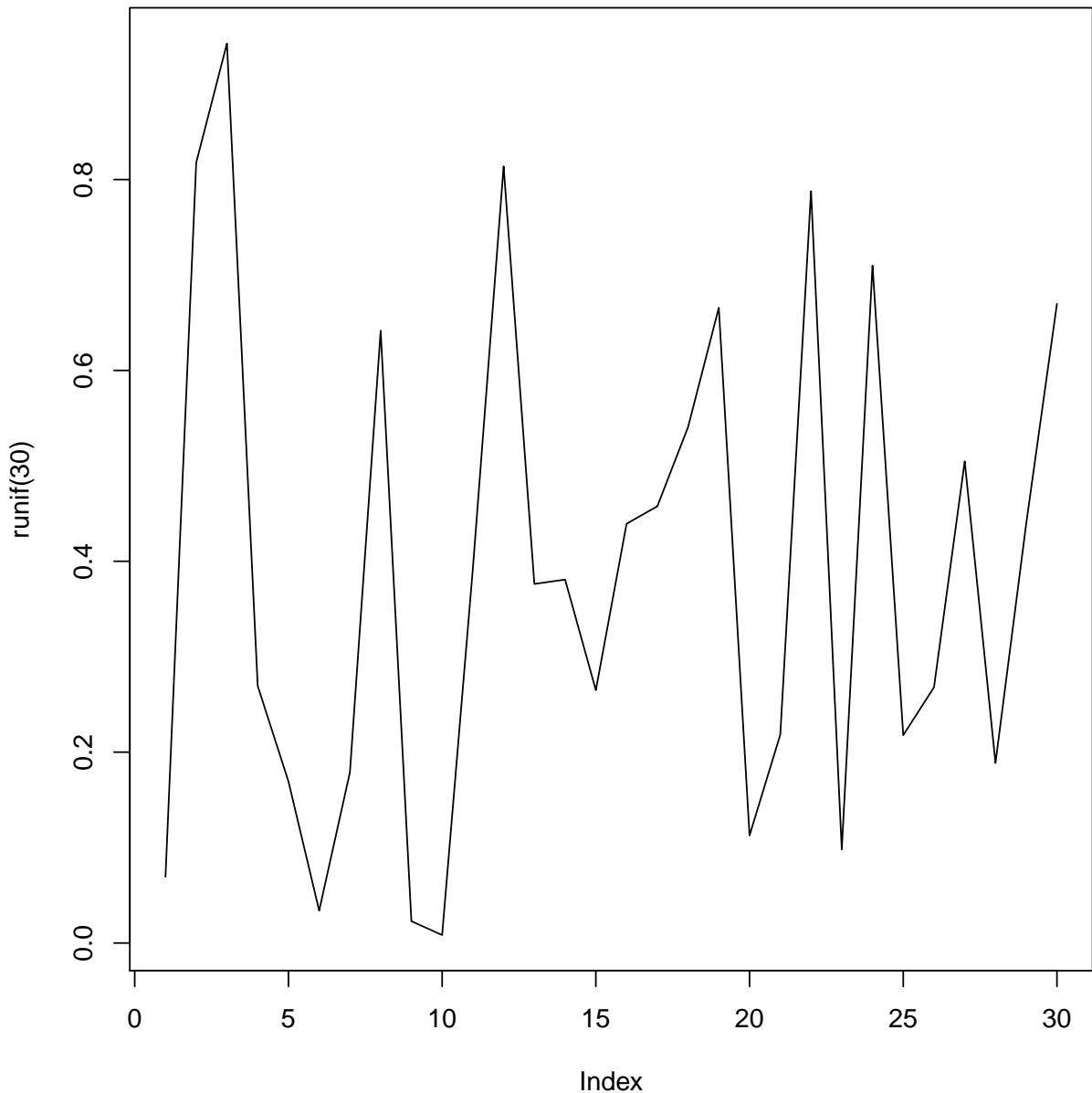


Figure 2.1: Test figure to illustrate figure options used by knitr.

Chapter 3

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.

3.1 Simulated Data

Lets assume a simple model:

$$Y_i \sim \mathcal{N}(\beta_0 + x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + x_4\beta_4, \sigma^2), \quad (3.1)$$

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 parameter. For computational reasons a small burning period of 1000 and a sample size of 1000 were chosen. 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.

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

## 
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 0.000127 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.27 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: 0.074254 seconds (Warm-up)
##                 0.065907 seconds (Sampling)
##                 0.140161 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 1.3e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 0.074243 seconds (Warm-up)
##                 0.066928 seconds (Sampling)
##                 0.141171 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,] 0.35178659 0.8655954 1.142714 0.8983258 0.9083553 0.9562517
## [2,] 0.06255488 0.8535117 1.049424 0.7477095 0.8104561 1.0594685
## [3,] 0.31597084 0.7650021 1.340354 0.9362496 1.1261895 1.2277133
## [4,] 0.06533558 1.0942983 0.951873 0.8730962 0.7628246 1.0079768
## [5,] 0.28055153 1.0896500 1.150824 0.9820826 0.8198639 1.4869377
## [6,] 0.40788418 0.9904042 1.312746 0.9577036 0.7945795 1.6462027
## [7,] 0.54576471 0.8962367 1.204006 0.9703716 1.0254554 1.3898373
## [8,] 0.28610365 0.8699938 1.110565 0.7824389 0.9131850 0.8861995
## [9,] 0.18966765 0.8408969 1.076889 0.8853991 0.8885238 1.2108844
## [10,] -0.04572765 1.0818832 1.255911 0.5915038 1.0709531 0.9675569

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

#data frame with all submodels
df.rtwos <- rtwos(df[,2:5], post.sample)

df.rtwos[,1:5]

##          X1      X2      X3      X4      X5
## none 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## x1   0.1155161 0.1265817 0.05861261 0.2214549 0.14825388
## x2   0.2743602 0.2539277 0.30650867 0.1766879 0.19380243
## x3   0.3583408 0.2997988 0.30910601 0.3549040 0.30995688
## x4   0.1549655 0.1350313 0.18054621 0.1127693 0.09532595
## x1 x2  0.4652522 0.4560452 0.42407258 0.4808169 0.41292796
## x1 x3  0.4251260 0.3791724 0.33656876 0.5076723 0.40604211
## x1 x4  0.2878599 0.2785842 0.25263665 0.3548052 0.25903292
## x2 x3  0.5829040 0.5098845 0.56668905 0.4920195 0.46490367
## x2 x4  0.4267990 0.3866899 0.48417204 0.2877273 0.28746299
## x3 x4  0.4514269 0.3819125 0.42701257 0.4159155 0.36089513
## x1 x2 x3 0.7078564 0.6499467 0.63719154 0.7115499 0.61751520
## x1 x2 x4 0.6397839 0.6100421 0.62046964 0.6158245 0.52523325
## x1 x3 x4 0.5327627 0.4761689 0.46519159 0.5866294 0.46992805
## x2 x3 x4 0.6781015 0.5939166 0.68714028 0.5543666 0.51713991
## all    0.8239334 0.7547871 0.77532252 0.7966452 0.68703340

```

After the R² for each posterior sample and their corresponding submodels is calculated, the package hier.part is used to calculate the LMG value for each posterior sample.

```

# prepare data frame for LMG values

LMG.Vals<-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] = obj.Gelman$IJ[,1]
}

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

##      2.5%      50%      97.5%
## 0.06083054 0.14395976 0.24118322

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

##      2.5%      50%      97.5%
## 0.1447740 0.2572612 0.3774172

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

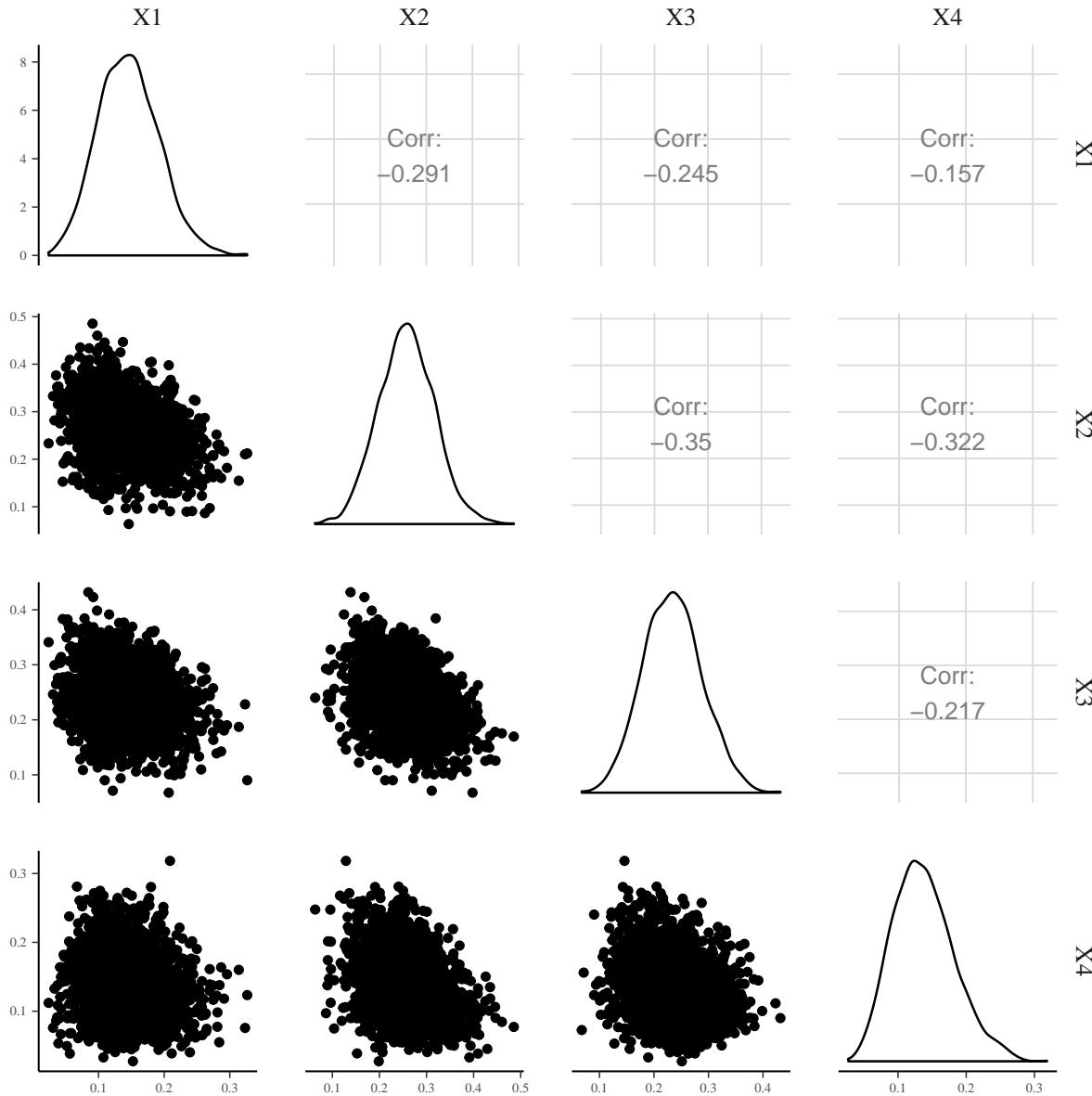
##      2.5%      50%      97.5%
## 0.1319720 0.2334742 0.3435360

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

##      2.5%      50%      97.5%
## 0.06183814 0.13457410 0.23772626

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

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: 4.355912
## Analysis based on 50 observations
##
## 4 Regressors:
```

```

## x1 x2 x3 x4
## Proportion of variance explained by model: 84.79%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##          lmg
## x1 0.1668528
## x2 0.1210657
## x3 0.2659347
## x4 0.2940836
##
## Average coefficients for different model sizes:
##
##          1X      2Xs      3Xs      4Xs
## x1 0.9679797 1.0043131 1.0461149 1.0971962
## x2 0.7726506 0.7890499 0.8151228 0.8548060
## x3 1.1604683 1.0873804 0.9968800 0.8819888
## x4 1.0909722 1.1062813 1.1261169 1.1526956
##
## 
## Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##          Lower   Upper
##      percentage 0.95  0.95
## x1.lmg 0.1669  0.0902 0.2557
## x2.lmg 0.1211  0.0619 0.2001
## x3.lmg 0.2659  0.1794 0.3596
## x4.lmg 0.2941  0.2076 0.3999
##
## 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 exogeneity and conditional independence the posterior distributions of the regression parameters β are valid 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 posterior 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: 4.355912
## Analysis based on 50 observations
##
## 4 Regressors:
## x1 x2 x3 x4
## Proportion of variance explained by model: 84.79%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##          lmg
## x1  0.1668528
## x2  0.1210657
## x3  0.2659347
## x4  0.2940836
##
## Average coefficients for different model sizes:
##
##          1X      2Xs      3Xs      4Xs
## x1  0.9679797 1.0043131 1.0461149 1.0971962
## x2  0.7726506 0.7890499 0.8151228 0.8548060
## x3  1.1604683 1.0873804 0.9968800 0.8819888
## x4  1.0909722 1.1062813 1.1261169 1.1526956
##
## 
## Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##          Lower   Upper
##          percentage 0.95  0.95
## x1.lmg 0.1669    0.0827 0.2833
## x2.lmg 0.1211    0.0418 0.2397
## x3.lmg 0.2659    0.1305 0.4015
## x4.lmg 0.2941    0.1560 0.4198
##
## CAUTION: Bootstrap confidence intervals can be somewhat liberal.

#-----


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

n.boot = 10

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

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

##      2.5%      50%     97.5%
## 0.03316721 0.10069290 0.19184795

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

##      2.5%      50%     97.5%
## 0.10355572 0.2154247 0.3460890

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

##      2.5%      50%     97.5%
## 0.1118686 0.2141213 0.3345831

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

##      2.5%      50%     97.5%
## 0.0514513 0.1660575 0.3295510

#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 0.000106 seconds  

## 1000 transitions using 10 leapfrog steps per transition would take 1.06 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: 0.098861 seconds (Warm-up)
##                 0.110365 seconds (Sampling)
##                 0.209226 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 3e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.3 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: 0.151875 seconds (Warm-up)
##                 0.08298 seconds (Sampling)
##                 0.234855 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,]  0.20052228 0.9872136 0.9970127 0.9899809 1.0063197 0.8903780
## [2,] -0.02297881 1.0088341 1.0072453 0.9957068 1.0011101 0.9512888

```

```

##      [3,] -0.11901835 0.9917888 0.9885906 0.9919268 0.9903575 0.8244199
##      [4,]  0.05730358 0.9964999 1.0092232 0.9950032 1.0033827 0.8633687
##      [5,] -0.08279075 0.9932238 0.9967296 0.9959667 0.9980719 0.8416048
##      [6,]  0.12616714 0.9866720 1.0102625 0.9743373 0.9988939 0.8295769
##      [7,] -0.04622525 0.9904358 0.9886503 1.0032408 1.0033684 0.9784942
##      [8,]  0.24497457 0.9992391 0.9813202 1.0080779 1.0063643 0.9014726
##      [9,]  0.02061810 1.0023034 0.9978158 0.9957526 0.9960182 0.9993591
##     [10,] -0.05240880 1.0111704 0.9991333 1.0007581 1.0042840 0.9187695

#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.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.boot[,i, b]=obj.Gelman$IJ[,1]
  }
}

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

##      2.5%      50%      97.5%
## 0.08282611 0.13231416 0.18596713

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

##      2.5%      50%      97.5%
## 0.1118092 0.1562988 0.1943982

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

##      2.5%      50%      97.5%
## 0.2990457 0.3597446 0.3972258

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

##      2.5%      50%      97.5%
## 0.08258831 0.18802788 0.24003283

```

In the following example we know that the \mathbf{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<-array(0, c(4,dim(df.rtwos)[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, b]=obj.Gelman$IJ[,1]
  }
}

```

```

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

##      2.5%      50%     97.5%
## 0.06495083 0.13227804 0.24036532

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

##      2.5%      50%     97.5%
## 0.09945212 0.15629146 0.21036735

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

##      2.5%      50%     97.5%
## 0.2084063 0.3135088 0.3675324

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

##      2.5%      50%     97.5%
## 0.1329764 0.2286050 0.2981260

```

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 parameter

#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, 1,5)

df.rtwos.boot <-rtwos.boot(df[,2:5], post.sample, 1000)

n.boot = 1000

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

}

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

##      2.5%      50%     97.5%
## 0.1455363 0.2294642 0.3134606

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

##      2.5%      50%     97.5%
## 0.06861673 0.14433177 0.22894050

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

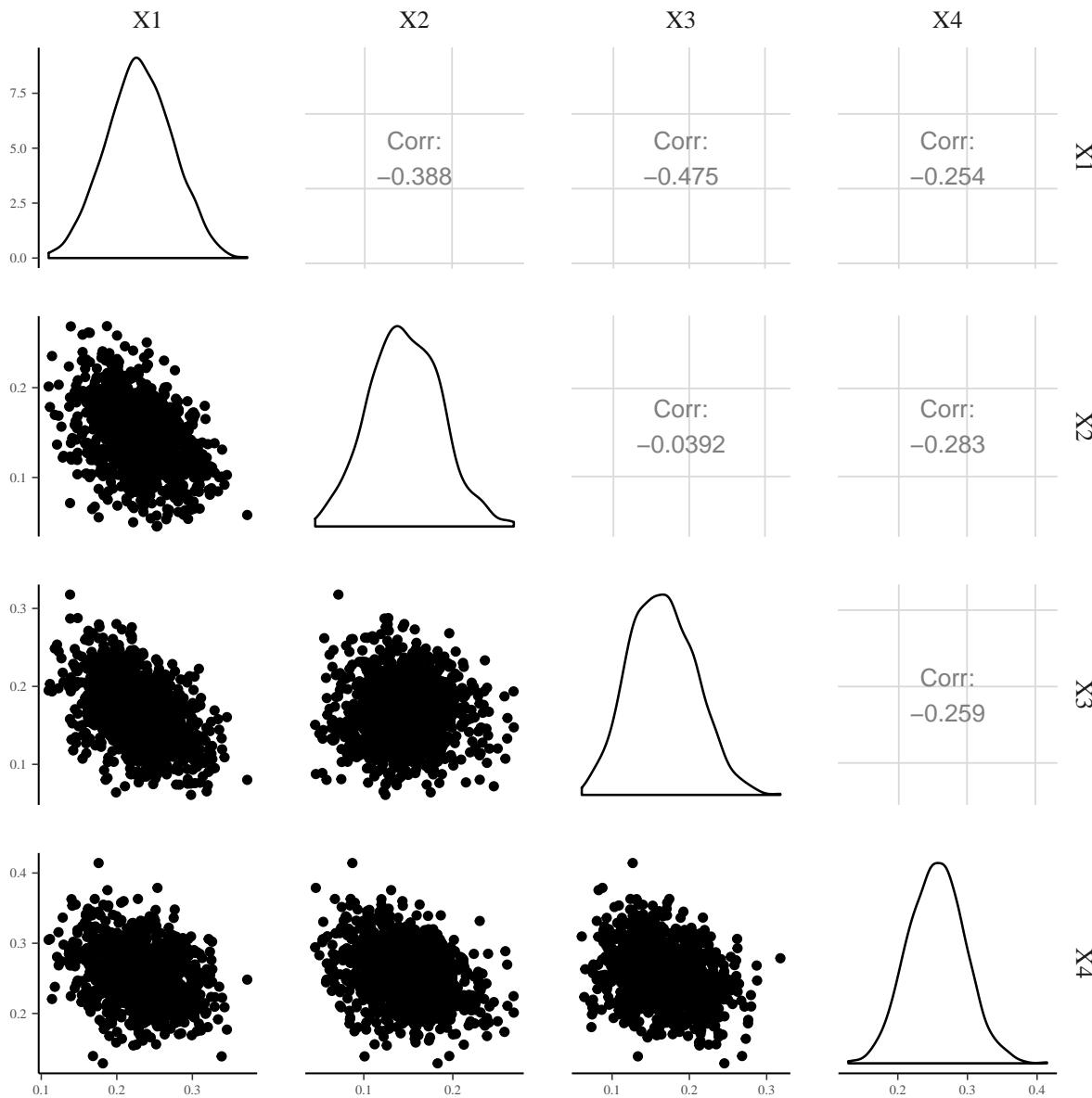
##      2.5%      50%     97.5%
## 0.08746488 0.16393586 0.24866457

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

##      2.5%      50%     97.5%
## 0.1782911 0.2554290 0.3366212

dat <- data.frame(t(LMG.Vals.boot[1:4,1,]))
pairs.chart <- ggpairs(dat, lower = list(list(combo = "facetdensity")), upper = list(list(c
pairs.chart

```



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

```
##          x1          x2          x3          x4
## x1  1.18967229 -0.2288990  0.01461313  0.2334858
## x2 -0.22889895  0.7701068  0.27285759 -0.0514278
## x3  0.01461313  0.2728576  0.71861703 -0.1451502
## x4  0.23348578 -0.0514278 -0.14515015  1.2953711
```

#Comparison of sample covariance and shrink covariance matrix

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

```
##          x1          x2          x3          x4
## x1  1.18967229 -0.2288990  0.01461313  0.2334858
## x2 -0.22889895  0.7701068  0.27285759 -0.0514278
## x3  0.01461313  0.2728576  0.71861703 -0.1451502
## x4  0.23348578 -0.0514278 -0.14515015  1.2953711
```

Table 3.1: Variable description of air pollution data

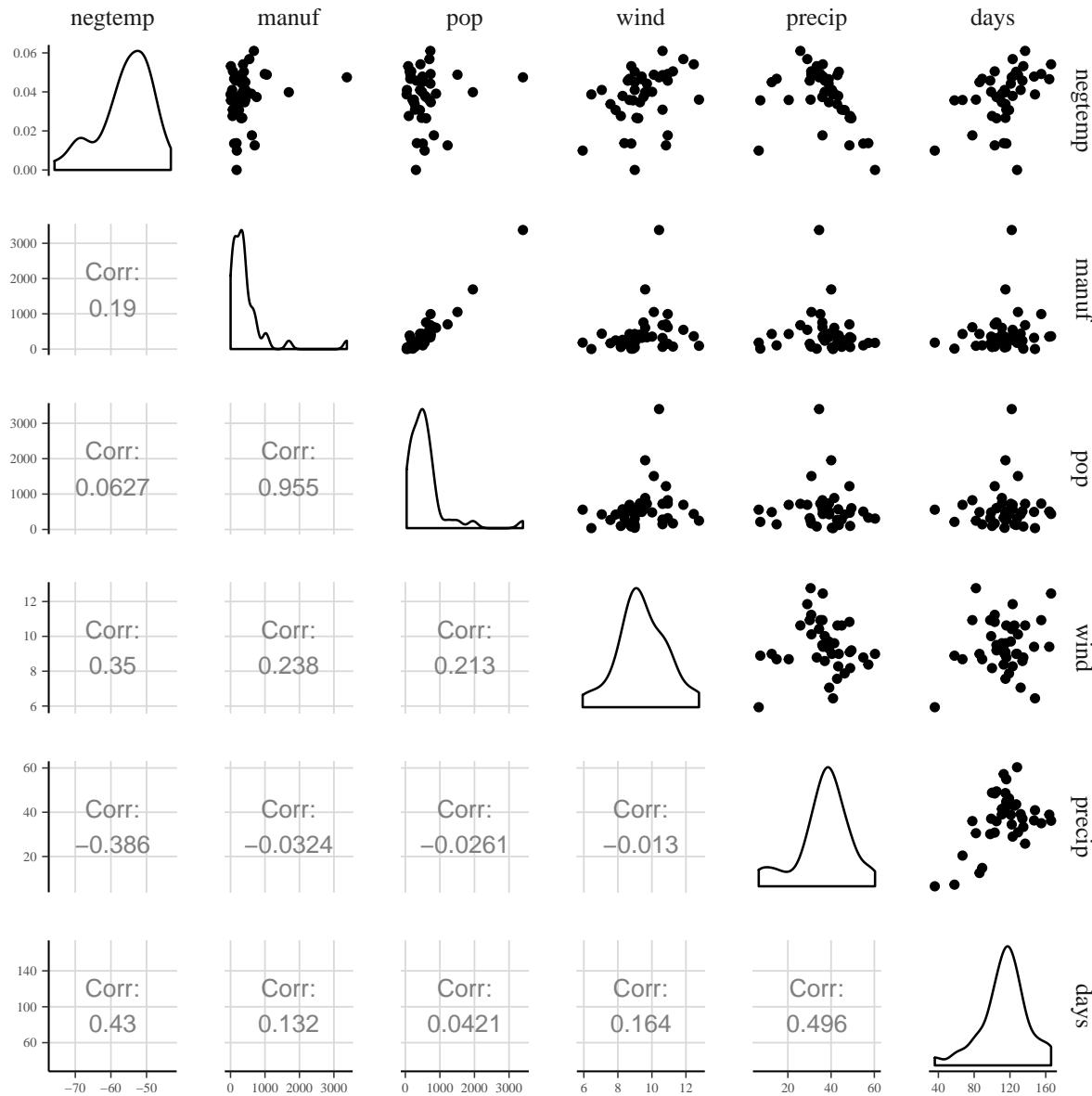
Variable Name	Description	Codes/Values
SO2	sulfur dioxide content of air	microgrammes per cubic meter
negtemp	negative value of average	fahrenheit
manuf	number of manufacturing enterprises employing 20 or more workers	integers
pop	population size in thousands (1970 census)	numbers
wind	average wind speed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

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

## Estimating optimal shrinkage intensity lambda.var (variance vector): 0.6037
##
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.4525
##          x1          x2          x3          x4
## x1  1.063018555 -0.12783778  0.008351944  0.11160357
## x2 -0.127837782  0.89676052  0.178027260 -0.02806219
## x3  0.008351944  0.17802726  0.876357065 -0.08105320
## x4  0.111603568 -0.02806219 -0.081053201  1.10490300
## attr(,"lambda")
## [1] 0.4524851
## attr(,"lambda.estimated")
## [1] TRUE
## attr(,"class")
## [1] "shrinkage"
## attr(,"lambda.var")
## [1] 0.6037376
## attr(,"lambda.var.estimated")
## [1] TRUE
```

3.2 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 consists of the SO2 level as the dependent variable and six explanatory variables. Two of the explanatory variables 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: 550.9476
## Analysis based on 41 observations
##
## 6 Regressors:
```

```

## negtemp manuf pop wind precip days
## Proportion of variance explained by model: 66.95%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##          lmg
## negtemp 0.11112440
## manuf   0.29645486
## pop     0.16612543
## wind    0.01670048
## precip  0.01867491
## days    0.06043172
##
## Average coefficients for different model sizes:
##
##          1X      2Xs      3Xs      4Xs      5Xs
## negtemp 1.40813251 1.334422627 1.281176139 1.25765114 1.28261316
## manuf   0.02685872 0.037288506 0.046426172 0.05392217 0.05982752
## pop     0.02001359 0.004397813 -0.009249699 -0.02072002 -0.03035031
## wind    1.55574078 -0.024806053 -1.384826041 -2.37946774 -2.98034367
## precip  0.10826200 0.114668437 0.164271600 0.26014506 0.40567633
## days    0.32726028 0.297872739 0.241114488 0.15574609 0.04248847
##          6Xs
## negtemp 1.26794109
## manuf   0.06491817
## pop     -0.03927674
## wind    -3.18136579
## precip  0.51235896
## days    -0.05205019
##
##
## Confidence interval information ( 1000 bootstrap replicates, bty= perc ):
## Relative Contributions with confidence intervals:
##
##          Lower  Upper
##          percentage 0.95  0.95
## negtemp.lmg 0.1111  0.0355 0.2138
## manuf.lmg   0.2965  0.1900 0.4115
## pop.lmg     0.1661  0.1098 0.2403
## wind.lmg    0.0167  0.0089 0.0728
## precip.lmg  0.0187  0.0081 0.0712
## days.lmg    0.0604  0.0203 0.1389
##
## 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
## [1,]    135.83097 1.6205034 0.05559092 -0.024963609 -3.368546
## [2,]    148.61733 1.9921319 0.05209391 -0.033695296 -3.379457
## [3,]    134.21933 1.8223345 0.05344065 -0.030226013 -3.321580
## [4,]    109.72730 0.9034187 0.06408435 -0.036752478 -4.075358
## [5,]    169.22540 1.8980336 0.07106017 -0.042167151 -3.707558
## [6,]    148.96289 1.5976024 0.03715969 -0.002440436 -5.353039
## [7,]    143.43058 1.9582751 0.04677972 -0.023647799 -3.424771
## [8,]    138.12934 1.9767608 0.04680241 -0.020412746 -3.164154
## [9,]    31.62113 0.4178926 0.07519204 -0.047230985 -1.559374
## [10,]   67.76305 0.3463465 0.08937092 -0.059099658 -2.171377
##          parameters
## iterations     precip       days     sigma
## [1,] 0.61867890 -0.135979662 13.71905
## [2,] 0.58114542  0.009231195 13.98130
## [3,] 0.63406863  0.004519797 13.94855
## [4,] 0.40657783 -0.096770264 15.49280
## [5,] 1.09928658 -0.419331341 13.29130
## [6,] 0.72383580 -0.181001660 17.26865
## [7,] 0.54949515 -0.026075784 13.07193
## [8,] 0.55819691 -0.022390552 12.83345
## [9,] 0.13001486  0.259908026 16.58099
## [10,] 0.07051633 -0.038521238 14.73657

#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 0.000000000 0.000000000 0.000000000
## x1   0.1851174789 0.354757878 0.286512432
## x2   0.5159090754 0.246527846 0.342081676
## x3   0.3550144090 0.113969863 0.191841855
## x4   0.0142971366 0.015067873 0.016850648
## x5   0.0002522025 0.001754583 0.006335778
## x6   0.0976191698 0.245652389 0.230862537
## x1 x2 0.6054322912 0.507200513 0.528696856
## x1 x3 0.5099993760 0.445270827 0.450735709
## x1 x4 0.1862057657 0.363098436 0.290265657
## x1 x5 0.2240787525 0.441680841 0.382884967
## x1 x6 0.2050125946 0.425074111 0.363324206

```

```

## x2 x3      0.6091586524 0.460226343 0.508703591
## x2 x4      0.5187029528 0.246550343 0.342174538
## x2 x5      0.5174445778 0.249893437 0.351805476
## x2 x6      0.5641635721 0.434853872 0.507672952
## x3 x4      0.3550676397 0.116690195 0.193250427
## x3 x5      0.3560037606 0.116542643 0.200135346
## x3 x6      0.4377402491 0.346152318 0.405710107
## x4 x5      0.0146011543 0.016958947 0.023458917
## x4 x6      0.1024127968 0.247415066 0.233531366
## x5 x6      0.1232918763 0.300851144 0.264298788
## x1 x2 x3   0.6456106231 0.584255292 0.592992648
## x1 x2 x4   0.6299578857 0.535105745 0.552953372
## x1 x2 x5   0.6337285916 0.584173288 0.611906572
## x1 x2 x6   0.6163720923 0.566495167 0.591070960
## x1 x3 x4   0.5332435173 0.469598807 0.472349588
## x1 x3 x5   0.5494288477 0.532562443 0.547631148
## x1 x3 x6   0.5272929582 0.512954480 0.523845393
## x1 x4 x5   0.2278603389 0.459725409 0.394161398
## x1 x4 x6   0.2062563925 0.434216088 0.367645926
## x1 x5 x6   0.2247710032 0.444241717 0.385372831
## x2 x3 x4   0.6138551538 0.460582565 0.509634363
## x2 x3 x5   0.6111122982 0.464530328 0.519795613
## x2 x3 x6   0.6282292054 0.559388261 0.600408251
## x2 x4 x5   0.5202160480 0.249919029 0.351888289
## x2 x4 x6   0.5712100901 0.437947107 0.512078169
## x2 x5 x6   0.5710755760 0.469718024 0.523015285
## x3 x4 x5   0.3560535555 0.119303587 0.201596553
## x3 x4 x6   0.4406228328 0.346764294 0.407037274
## x3 x5 x6   0.4543353180 0.393715673 0.431450645
## x4 x5 x6   0.1259845839 0.301113141 0.265303596
## x1 x2 x3 x4 0.6648538323 0.604437815 0.610664917
## x1 x2 x3 x5 0.6641871676 0.638957779 0.655086937
## x1 x2 x3 x6 0.6516045505 0.627040581 0.639878133
## x1 x2 x4 x5 0.6665049380 0.626816987 0.650558389
## x1 x2 x4 x6 0.6410854774 0.594866718 0.615773494
## x1 x2 x5 x6 0.6360323883 0.585606086 0.613042441
## x1 x3 x4 x5 0.5829666007 0.572906477 0.585314068
## x1 x3 x4 x6 0.5510641756 0.538350903 0.546506715
## x1 x3 x5 x6 0.5513361785 0.534281256 0.549041548
## x1 x4 x5 x6 0.2292035985 0.460591928 0.395268558
## x2 x3 x4 x5 0.6157810155 0.464875234 0.520697018
## x2 x3 x4 x6 0.6358059358 0.562980536 0.605322254
## x2 x3 x5 x6 0.6294242541 0.573880076 0.604750245
## x2 x4 x5 x6 0.5794657832 0.474944249 0.529029029
## x3 x4 x5 x6 0.4588355009 0.395931365 0.434254757
## x1 x2 x3 x4 x5 0.6903033120 0.671422793 0.685128213
## x1 x2 x3 x4 x6 0.6712607148 0.648350860 0.658656123
## x1 x2 x3 x5 x6 0.6666325206 0.640247140 0.656109414
## x1 x2 x4 x5 x6 0.6728797311 0.626825334 0.650558535
## x1 x3 x4 x5 x6 0.5887423878 0.572960223 0.585334637
## x2 x3 x4 x5 x6 0.6375962715 0.578877182 0.610545875

```

```

## all          0.6963888732 0.671452014 0.685134958

# prepare data frame for LMG values

LMG.Vals<-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]=obj.Gelman$IJ[,1]
}

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

##      2.5%      50%     97.5%
## 0.03089074 0.10414155 0.20513745

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

##      2.5%      50%     97.5%
## 0.1492509 0.2632596 0.3615762

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

##      2.5%      50%     97.5%
## 0.08674675 0.15221081 0.21173554

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

##      2.5%      50%     97.5%
## 0.007556802 0.017508171 0.071477479

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

##      2.5%      50%     97.5%
## 0.006626489 0.020800713 0.070602786

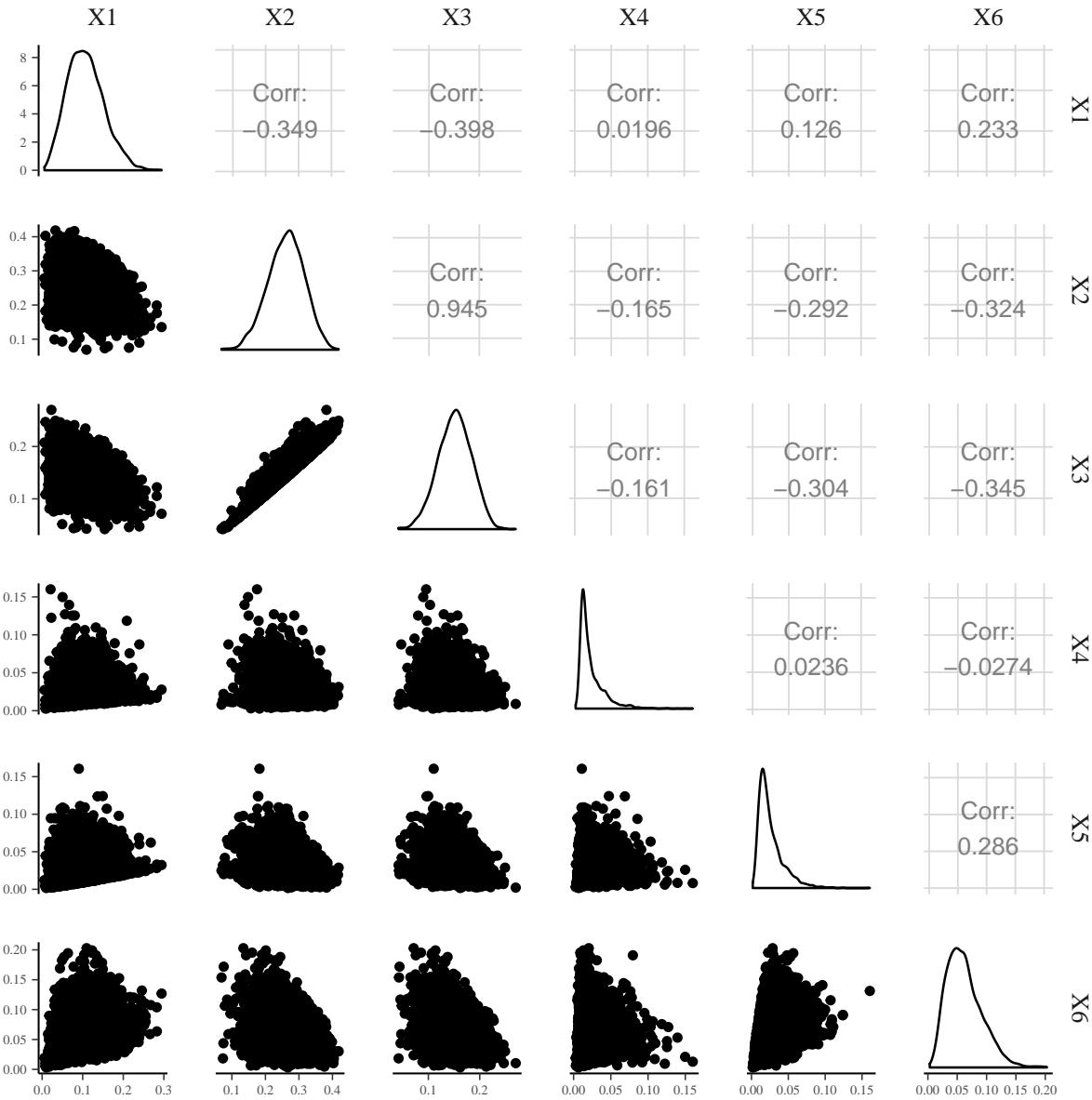
quantile(LMG.Vals[6,], c(0.025, 0.5, 0.975))

##      2.5%      50%     97.5%
## 0.01776118 0.05803545 0.13121868

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

pairs.chart <- ggpairs(dat, lower = list(list(combo = "facetdensity")), upper = list(list(conti
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
## [1,]     93.15510 0.9598408 0.05076080 -0.02230947 -2.974473
```

```

##      [2,]    85.33896 0.9767056 0.04873335 -0.02762700 -2.993646
##      [3,]   115.76574 1.0501914 0.04800659 -0.02754002 -2.960514
##      [4,]   105.13833 1.0937662 0.06237204 -0.03862311 -2.755087
##      [5,]   100.17739 0.9218288 0.06397271 -0.03741000 -3.209142
##      [6,]    94.81891 0.8699693 0.06460313 -0.03974537 -3.058839
##      [7,]   107.25037 1.1169629 0.04871286 -0.02439039 -2.983201
##      [8,]    85.37864 0.8471359 0.05783901 -0.03187420 -2.868974
##      [9,]   104.06738 0.9644024 0.05936805 -0.03575935 -3.024624
##     [10,]   92.09087 1.0109323 0.05776583 -0.03223990 -2.961939
##      parameters
## iterations      precip      days      sigma
##      [1,] 0.14680578 0.01154576 10.73355
##      [2,] 0.30231176 0.07928726 11.73207
##      [3,] -0.06167479 -0.00771524 15.35028
##      [4,] 0.14574409 -0.01068613 13.39733
##      [5,] -0.05972290 0.10561934 13.20576
##      [6,] 0.04507632 0.07356100 13.71082
##      [7,] 0.16243033 -0.03046557 15.03457
##      [8,] 0.13903104 0.09494624 15.20654
##      [9,] -0.06792343 0.05425036 15.97530
##     [10,] 0.27630906 0.04133327 12.82158

#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<-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]=obj.Gelman$IJ[,1]
}

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

##      2.5%      50%     97.5%
## 0.0725409 0.1143731 0.1584069

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

##      2.5%      50%     97.5%
## 0.1656552 0.2578780 0.3334933

```

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

##      2.5%      50%     97.5%
## 0.09132594 0.15029884 0.20709797

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

##      2.5%      50%     97.5%
## 0.01266590 0.01641026 0.02130490

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

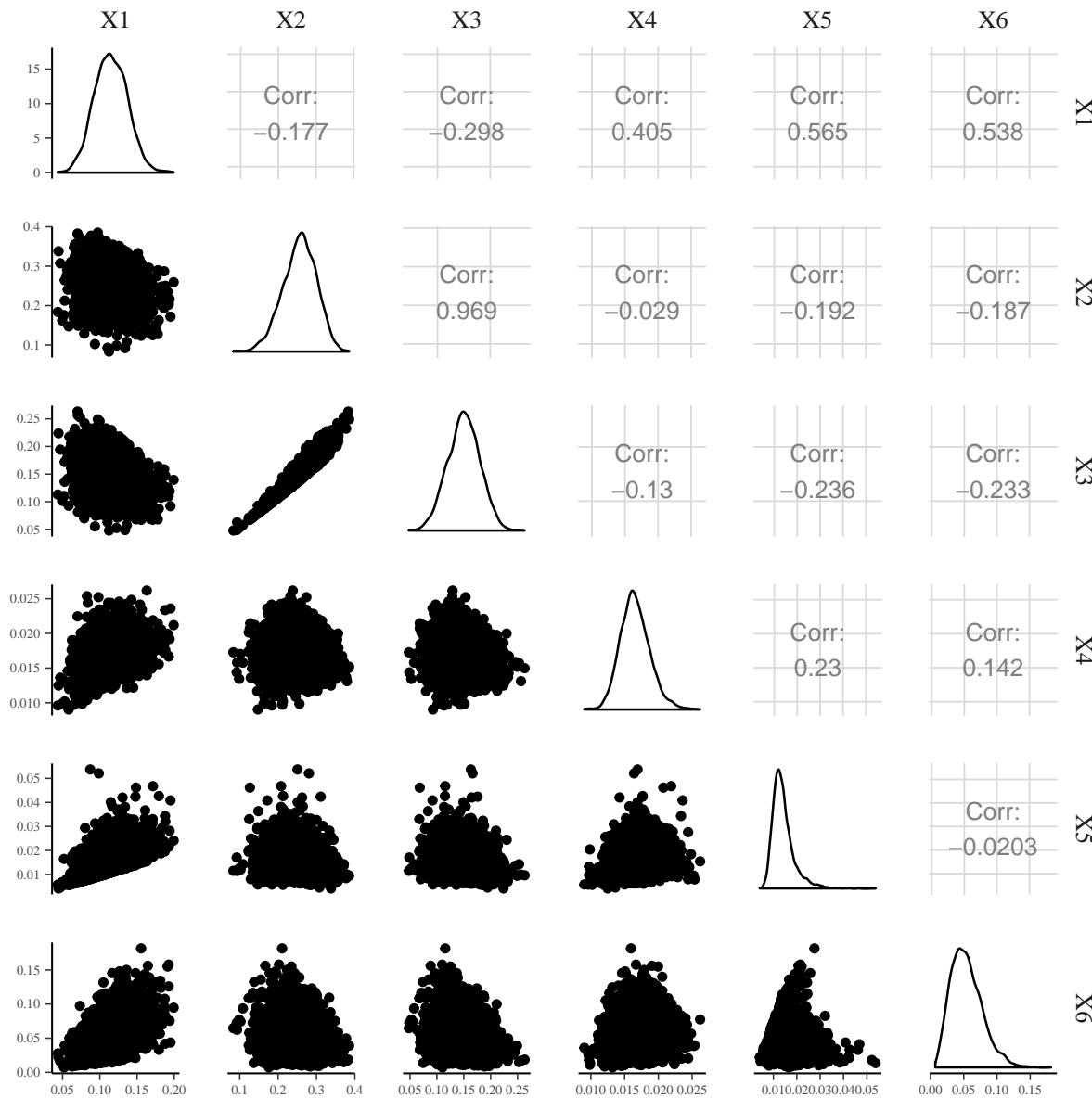
##      2.5%      50%     97.5%
## 0.007922485 0.013181413 0.026273074

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

##      2.5%      50%     97.5%
## 0.01718799 0.05162126 0.10892648

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

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



Chapter 4

Discussion and Outlook - Some extenstions

In the following chapter some extenstions of the LMG formula in the Bayesian framework beyond the simple linear regression model are shown. The focus is on repeated measurements models. These models extend the simple linear regression by allowing intra subject correlation between repeated measures.

The dependence within subjects can be modeled by including random effects (mixed model) or by assuming correlated errors within a subject (marginal model). Using a random intercept model or a compound symmetry matrix for the error will result in the same model for the fixed predictors. A mixed model can be extended by including a random slope per subject, allowing for less restrictive longitudinal shapes. The marginal approach can get more freedom by different specified covariance matrices of the error terms. An unstructured covariance matrix, where no restriction are imposed, allows for the most freedom. However, depending on the number of repeated measurements and the sample size the covariance matrix can get too large to make reasonable inference about it.

The extenstion of the LMG formula in the Bayesian framework presented in chapter is restricted to models where the conditional variance formula can easily be applied to get the explained variance of the submodel from the regression parameters of the full model. The focus is on the fixed predictors and not on the random effects. Using the conditional variance formula to get the explained variance of the fixed predictors of the submodels should be applicable in the marginal models, where only the fixed effects are modelled anyway. In the mixed model framework the conditional variance formula is applicable to models including only random intercepts and the focus lies in the explained variance of the fixed predictors. For random-slope models there are atleast some difficulties involved, if it is possible at all the get the expalined variance of the submodel. This chapter shows a random intercept model and a repeated measurement model with an unstructured covariance matrix.

The first example concerns a simple random intercept model with time varying predictors.

4.1 random intercept model

Different R^2 metrics exist for linear mixed models. The variance of a random intercept model with regression parameter β can be written as

$$\text{Var}(y) = \sigma_f^2 + \sigma_\alpha^2 + \sigma_\epsilon^2, \quad (4.1)$$

where $\sigma_f^2 = \text{Var}(\mathbf{X}\beta) = \beta^\top \Sigma_{\mathbf{XX}} \beta$, σ_α^2 is the random intercept and σ_ϵ^2 is the error term.

An R^2 that is guaranteed to be positive can be defined as

$$R_{LMM}^2 = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_\alpha^2 + \sigma_\epsilon^2}, \quad (4.2)$$

Referenz Naka, Snyder.... It is theoretical possible that the R_{LMM}^2 decreases when adding predictors. This may be problematic for the LMG metric, because of violation of the non-negative property. This should rarely be the case with real data. The R^2 can not decrease when adding predictor by using the conditional variance formula on the full model to calculate the R^2 of the submodels. In the Bayesian framework we would draw samples from the posterior distributions of the parameters.

The total variance of the full can be calculated as in equation 6 or by using the samples of the random intercept for each subject directly. The same total variance is then used for one sample of the posterior. (In a repeated measure study we often have within and between subject predictors. If we use the total variance of the full model the random intercept is fitted including all predictors. If a between subject predictor is excluded (e.g. Sex) and we would fit a new random intercept model, the random intercept parameter will in addition explain the variance that was explained by the excluded predictor. In other words it means that the model with the excluded between subject predictor will explain almost as much as the model where the predictor is included when each time a new model and therefore a new random intercept term is fitted in each model.

When using the conditional variance formula for the R^2 of the submodels, it only takes into account the explained variance of the fixed predictors.)

In repeated measurement studies the focus is often in within subject changes. The between subject variance estimated with the random intercept term may not be so important. The more important question may be how much the fixed predictors explain compared to the within subject error, which is

$$R_{\text{repeated}}^2 = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_\epsilon^2}, \quad (4.3)$$

The square root of this term is known under the name correlation within subjects by ref(bland Altman 1995). Although in the paper the subject term is fitted as a factor. If we are interested in the within subject effects we can use the model including only the between subject predictor as the null model.

The following example shows a simple random intercept model with time-varying predictors. The main question is which within subject predictors are the most important ones. The between subject variance is of lower importance.

The data are simulated from the following regression setting with $m = 4$ timepoints,

$$Y_{i,j} \sim \mathcal{N}(\beta_0 + x_{1,i,j}\beta_1 + x_{2,i,j}\beta_2 + x_{3,i,j}\beta_3 + x_{4,i,j}\beta_4 + \alpha_i, \sigma^2), \quad i = 1, \dots, n \quad j = 1, \dots, m \quad (4.4)$$

where $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 2$, $\beta_4 = 2$, $\sigma^2 = 1$, $\alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2)$, $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma)$.

The following R code is used to simulate the data:

```
sub<- 1:20
subi <- rnorm(20, 0, 4)
subi<-rep(subi, 4)
t <- c(0, 1, 2,3)
t <- c(rep(0, 20), rep(1,20), rep(2, 20), rep(3,20))
```

```
mu <- rep(0,4)
sig <- matrix(0.4, 4, 4)
diag(sig) <- 1
sig[3,4] <- 0.9
sig[4,3] <- 0.9
sig[1,2] <- 0.3
sig[2,1] <- 0.3

rawvars <- mvrnorm(n=80, mu=mu, Sigma=sig)

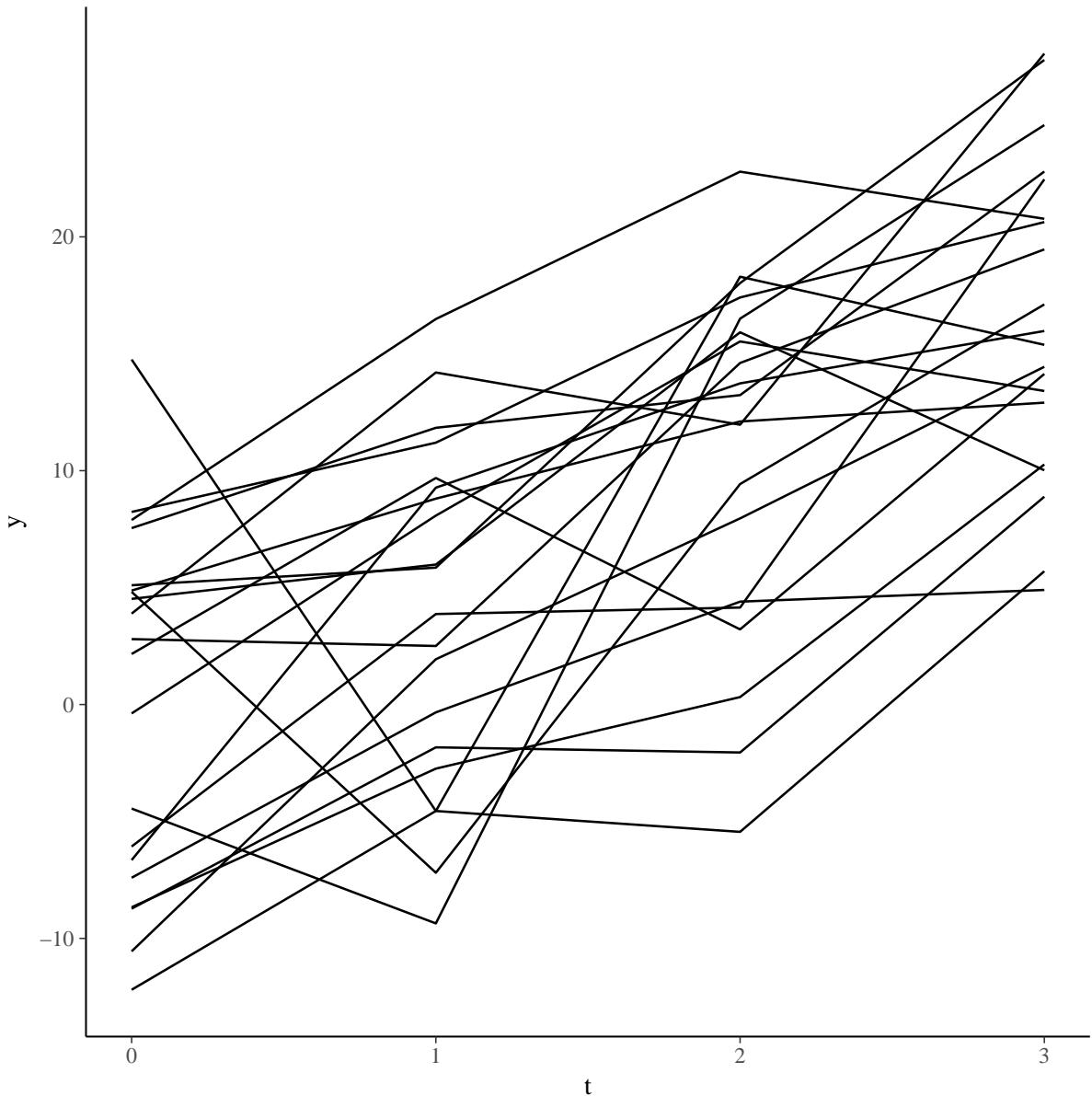
x1 <- t+rawvars[,1]
x2 <- t+rawvars[,2]
x3 <- t+rawvars[,3]
x4 <- t+rawvars[,4]

b1 <- b2 <-1
b3 <- b4 <- 2

y<- x1*b1 +x2*b2 +x3*b3+ x4*b4 + subi+ rnorm(80, 0, 0.1)

df <- data.frame(y=y, x1 = x1, x2=x2, x3 = x3, x4 = x4, sub = rep(sub,4))

p <- ggplot(data = df, aes(x = t, y = y, group = sub))
p + geom_line()
```



The R^2 of the submodels were first calculated according to the formula in equation, The random intercept effect is not of interest. We see that most of the within subject variance is explained by the predictors. The credible intervals are very narrow. For the information about the between subject variance we can look at the random intercept variance directly. In the second part the random intercept is included in the total variance calculation and the calculation of the R^2 values. We see that there is a large between subject variance in this dataset. The LMG values including the between subject variance are much lower. The credible intervals are also much wider, because the uncertainty about the between subject variance is also included. In my opinion we can get more useful information from separating the between and within variance components when it is easy possible. Note that we assumed non stochastic predictors otherwise the credible intervals would be larger. In general it seems more reasonable to assume stochastic time-varying predictors. The variance could then be estimated by non-parameteric bootstrap, resampling whole subjects (all repeated measurements of a subject).

```
fit <- stan_glmer(y ~ x1+x2+x3+x4 + (1|sub) ,
  data = df,
  chains = 4, cores =
```

```

post.sample <- as.matrix(fit)
post.sample.r <- post.sample[,c(2:5,(ncol(post.sample)-1):ncol(post.sample))]

df.rtwos <- rtwos.ri.r(df[,2:5], post.sample.r)

LMG.Vals<-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]=obj.Gelman$IJ[,1]
}

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

##      2.5%      50%     97.5%
## 0.1759721 0.1778704 0.1797420

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

##      2.5%      50%     97.5%
## 0.1805498 0.1824204 0.1841952

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

##      2.5%      50%     97.5%
## 0.3213042 0.3228832 0.3244004

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

##      2.5%      50%     97.5%
## 0.3149818 0.3165701 0.3180643

# explained compared to total variance

df.rtwos <- rtwos.ri.a(df[,2:5], post.sample)

LMG.Vals<-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]=obj.Gelman$IJ[,1]
}

```

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

##          2.5%      50%      97.5%
## 0.009191661 0.019669347 0.047444048

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

##          2.5%      50%      97.5%
## 0.01137392 0.03076685 0.06097521

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

##          2.5%      50%      97.5%
## 0.02238598 0.05498246 0.10306689

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

##          2.5%      50%      97.5%
## 0.02545344 0.06020207 0.11190935
```

4.2 marginal model

The next example concerns a repeated measurement model with an unstructured covariance error structure. The data are generated from the following model:

$$Y_i \sim \mathcal{N}(\mathbf{X}_i\boldsymbol{\beta}, \boldsymbol{\Sigma}), \quad i = 1, \dots, n \quad (4.5)$$

where $\boldsymbol{\Sigma}$ represents an unstructured error covariance matrix.

In the variance calculation we need to take into account that we do not have just one σ^2 parameter, but a covariance matrix $\boldsymbol{\Sigma}$. The diagonal elements of $\boldsymbol{\Sigma}$ represent the variance of each timepoint. The sum of the diagonal elements of $\boldsymbol{\Sigma}$ represents the variance for a whole subject. When there are no missing timepoints in each subject, we can take the mean of $\text{diag}(\boldsymbol{\Sigma})$ to make the formula compatible with the $\boldsymbol{\beta}^\top \boldsymbol{\Sigma} \mathbf{x} \boldsymbol{\beta}$ of equation resulting in the total variance term

$$\text{Var}(\mathbf{Y}) = \boldsymbol{\beta}^\top \boldsymbol{\Sigma} \mathbf{x} \boldsymbol{\beta} + \text{mean}(\text{diag}(\boldsymbol{\Sigma})), \quad (4.6)$$

The following R-code is used to generate the data:

```
sub<- 1:20
subi <- rnorm(20, 0, 1)
subi<-rep(subi, 4)

mu <- rep(0,4)
sig <- matrix(0.4, 4, 4)
diag(sig) <- 1
sig[3,4] <- 0.9
sig[4,3] <- 0.9
```

```

sig[1,2] <- 0.3
sig[2,1] <- 0.3

rawvars <- mvrnorm(n=80, mu=mu, Sigma=sig)
cov(rawvars)

## [,1]      [,2]      [,3]      [,4]
## [1,] 0.8185433 0.1544539 0.4858672 0.4857820
## [2,] 0.1544539 0.9208422 0.2848089 0.3715627
## [3,] 0.4858672 0.2848089 1.0513613 1.0061088
## [4,] 0.4857820 0.3715627 1.0061088 1.1152070

t <- c(rep(1, 20), rep(2, 20), rep(3, 20), rep(4, 20))
x1 <- t+rawvars[,1]
x2 <- t+rawvars[,2]
x3 <- t+rawvars[,3]
x4 <- t+rawvars[,4]

Sig<- matrix(3, 4,4)
diag(Sig) <- 10
u <- rep(0, 4)
Sig[1,1] <- 5
Sig[2,2] <- 7
Sig[3,4] <- 8
Sig[4,3] <- 8

Sig[1,2] <- 4
Sig[2,1] <-4

Sig

## [,1] [,2] [,3] [,4]
## [1,] 5 4 3 3
## [2,] 4 7 3 3
## [3,] 3 3 10 8
## [4,] 3 3 8 10

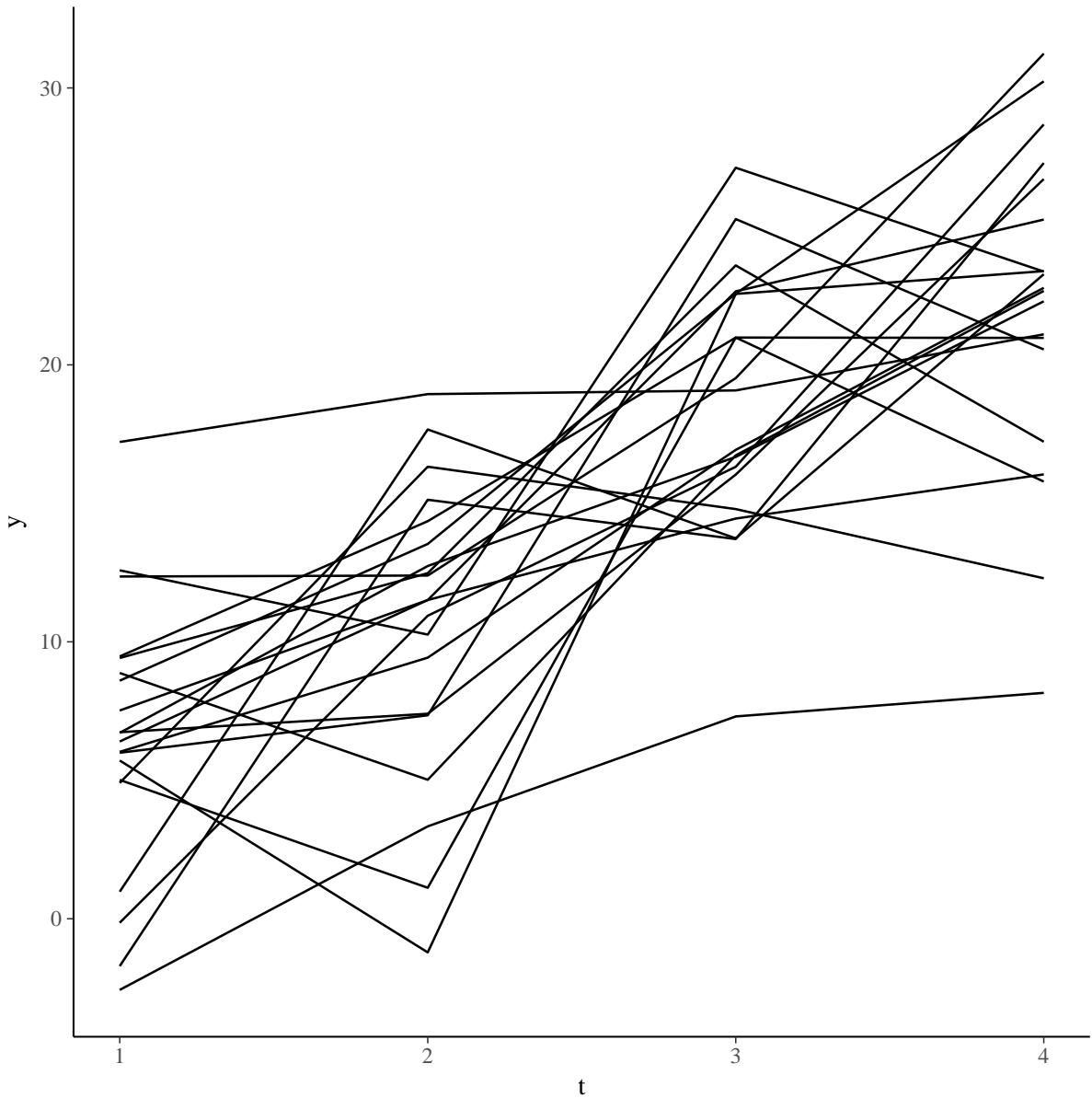
error <- mvrnorm(20, u, Sig)

y<- x1*b1 +x2*b2 + x3*b3 +x4*b4 +c(error)

t <- c(rep(1, 20), rep(2, 20), rep(3, 20), rep(4, 20))
df <- data.frame(y=y, x1 = x1, x2=x2 , x3 = x3, x4 = x4, sub = rep(sub,4), t =t)

p <- ggplot(data = df, aes(x = t, y = y, group = sub))
p + geom_line()

```



```
# Bayesian framework

Y <- matrix(df[, 'y'], 20, 4, byrow=F)
x1 <- matrix(df[, 'x1'], 20, 4, byrow=F)
x2 <- matrix(df[, 'x2'], 20, 4, byrow=F)
x3 <- matrix(df[, 'x3'], 20, 4, byrow=F)
x4 <- matrix(df[, 'x4'], 20, 4, byrow=F)

N = 20 #subjects
M = 4 # repeated measures
```

```
#-----
modelString <- "model{

# Likelihood
```

```

for(i in 1:N){
Y[i,1:M] ~ dmnorm(mu[i,1:M], Omega[1:M,1:M])
for(j in 1:M){
mu[i,j] <- beta0 + beta1*x1[i,j]+ beta2*x2[i,j]+ beta3*x3[i,j] + beta4*x4[i,j]
} }

# Priors

Omega[1:M, 1:M] ~ dwish(zRmat[1:M,1:M] , zRscal)
Sigma[1:M, 1:M] <- inverse(Omega)

beta0      ~ dnorm(0,0.001)
beta1      ~ dnorm(0,0.001)
beta2      ~ dnorm(0,0.001)
beta3      ~ dnorm(0,0.001)
beta4      ~ dnorm(0,0.001)

}"
```

writeLines(modelString , con="Jags-MultivariateNormal-model.txt")

```

model <- jags.model(textConnection(modelString),
data = list(Y=Y

n.chains=3)
```

samp <- coda.samples(model,
variable.names
n.iter=20000,

```

summary(samp)
```

```

##          Mean      SD  Naive SE Time-series SE
## Sigma[1,1] 5.975 2.3299 0.0095117      0.013015
## Sigma[2,1] 4.563 2.2608 0.0092298      0.011957
## Sigma[3,1] 2.735 2.4134 0.0098527      0.011506
## Sigma[4,1] 3.503 2.1681 0.0088512      0.010519
## Sigma[1,2] 4.563 2.2608 0.0092298      0.011957
## Sigma[2,2] 8.088 3.1867 0.0130098      0.015612
## Sigma[3,2] 1.281 2.7224 0.0111140      0.012150
## Sigma[4,2] 3.547 2.7235 0.0111188      0.012225
## Sigma[1,3] 2.735 2.4134 0.0098527      0.011506
## Sigma[2,3] 1.281 2.7224 0.0111140      0.012150
## Sigma[3,3] 12.252 4.8900 0.0199633     0.024465
## Sigma[4,3] 7.755 4.1055 0.0167605     0.019731
## Sigma[1,4] 3.503 2.1681 0.0088512      0.010519
## Sigma[2,4] 3.547 2.7235 0.0111188      0.012225
## Sigma[3,4] 7.755 4.1055 0.0167605     0.019731
## Sigma[4,4] 8.214 4.4759 0.0182729     0.022625
## beta1       1.222 0.2538 0.0010361     0.002673
## beta2       1.031 0.2019 0.0008242     0.001725
## beta3       1.259 0.5302 0.0021646     0.013629
## beta4       2.436 0.5739 0.0023429     0.015022
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75%   97.5%
## Sigma[1,1] 2.9398 4.3823 5.513  7.035 11.701
## Sigma[2,1] 1.5039 3.0429 4.149  5.595 10.107
## Sigma[3,1] -0.9407 1.2825 2.451  3.857 7.991
## Sigma[4,1] 0.5590 2.1827 3.181  4.438 8.269
## Sigma[1,2] 1.5039 3.0429 4.149  5.595 10.107
## Sigma[2,2] 4.0913 5.9917 7.483  9.469 15.617
## Sigma[3,2] -3.3030 -0.1920 1.136  2.584 6.625
## Sigma[4,2] 0.1516 2.0918 3.212  4.606 8.808
## Sigma[1,3] -0.9407 1.2825 2.451  3.857 7.991
## Sigma[2,3] -3.3030 -0.1920 1.136  2.584 6.625
## Sigma[3,3] 6.2917 9.1680 11.367 14.303 23.299
## Sigma[4,3] 3.2965 5.4609 7.080  9.269 16.033
## Sigma[1,4] 0.5590 2.1827 3.181  4.438 8.269
## Sigma[2,4] 0.1516 2.0918 3.212  4.606 8.808
## Sigma[3,4] 3.2965 5.4609 7.080  9.269 16.033
## Sigma[4,4] 4.1248 6.0422 7.556  9.628 16.021
## beta1       0.7320 1.0504 1.220  1.389 1.729
## beta2       0.6440 0.8936 1.027  1.162 1.441
## beta3       0.2234 0.9061 1.257  1.612 2.314
## beta4       1.2867 2.0572 2.442  2.822 3.547

#LMG calculations

samp <- coda.samples(model,
variable.n
n.itter=200

```

```
post.sample <- samp[[1]][,c(5:8, 1:4)]  
  
df.rtwos <- rtwos.marg(df[,2:5], post.sample, 4) # 4 repeated measures  
  
LMG.Vals<-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]=obj.Gelman$IJ[,1]  
}  
  
# posterior LMG distribution of each variable  
quantile(LMG.Vals[1,], c(0.025, 0.5, 0.975))  
  
##      2.5%      50%     97.5%  
## 0.1064941 0.1384998 0.1692021  
  
quantile(LMG.Vals[2,], c(0.025, 0.5, 0.975))  
  
##      2.5%      50%     97.5%  
## 0.07780784 0.10375754 0.13057235  
  
quantile(LMG.Vals[3,], c(0.025, 0.5, 0.975))  
  
##      2.5%      50%     97.5%  
## 0.1439328 0.1857861 0.2175383  
  
quantile(LMG.Vals[4,], c(0.025, 0.5, 0.975))  
  
##      2.5%      50%     97.5%  
## 0.1498266 0.1935178 0.2277630
```


Chapter 5

Other variable importance metrics in the Bayesian framework

Different variable importance metrics exist. The conditional variance formula allowed us to calculate the R^2 of the submodels from the posterior sample of the full model. The focus of this master thesis was on the LMG formula. For each posterior sample the LMG formula can be applied for the submodels of each posterior sample. A lot of the variable importance metrics are based on the R^2 of the full model compared to the submodels. Instead of the LMG formula we could as well have used another variable importance metric after we have calculated the R^2 of all the submodels. Commonality and dominance analysis seem to be interesting. Both provide besides the LMG information some other information about the variance decomposition of the predictors.

A disadvantage of the variable importance metric .

