Universiteit Utrecht

Erik-Jan van Kesteren
Methodology & Statistics

Bayesian Linear Models
The **blim** Package

```
> library(devtools)
> install_github("vankesteren/blim")
> library(blim)
```

1. The blim Framework

Anyone who can use the lm function in R can use blim! This is how to quickly make a **blimfit** object and see the results stored inside this object:

```
> fit <- blim(formula = y~x1+x2,
                data = dataset)
> summary(fit)
```

A blimfit object contains:
1. The **trace**: a matrix of all the sampled parameters
2. The **summary**: parameter estimates, standard errors, credible intervals, R squared and markov chain error
3. The **priors**: the prior information about the parameters
4. The **data**: matrix representations of the input data
5. The **autocorrelation**: to lag 40 for easy plotting

2. Wrapper Function

The blim function is a wrapper function that performs the following tasks:
1. Translate the model in formula form to model matrices readable for the samplers (see 3)
2. Optionally calculate a minimum training sample for an ANOVA model (option mtsprior = TRUE)
3. Parse priors input in the form "dnorm(0,1000)" for processing in the samplers
4. Call the sampler specified in the method-argument on the model matrices to perform analysis (see 3)
5. Process the trace returned from the sampler:
   a. Apply burnin
   b. Apply thinning
   c. Create a summary with mean, se, CI and mc error
   d. Give column names to the trace
6. Return a blimfit object with the indicated elements.

3. The Samplers

blim's core is built around **4 samplers**:

1. CPPBR
The standard method is "cppbr": Bayes Regression in C++. This method is near-instant because of its use of the Armadillo linear algrebra library in the C++ language via the Rcpp package. Bayes Regression samples the beta parameters at once from a multivariate normal distribution:

$$\boldsymbol{\beta}_n \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n), \text{ where}$$
$$\boldsymbol{\Lambda}_n = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Lambda}_0), \text{ and}$$
$$\boldsymbol{\mu}_n = (\boldsymbol{\Lambda}_n)^{-1}(\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \boldsymbol{\Lambda}_0\boldsymbol{\mu}_0)$$

```
// Sample from multivariate normal
arma::mat Y = arma::randn(1, k);
arma::vec betan = (posterior_mu.t() +
    Y * arma::chol(posterior_sigma)).t();
```

2. RBR
The "rbr" method is an R implementation of the "cppbr" method and is included for compatibility.

3. RGS
Method "rgs" is the Gibbs sampler of blim. It can handle any amount (k) of predictors (v) due to the following for-loop within each iteration:

```
for (v in 1:k){
  p_tau <- 1/(t(X[,v])%*%X[,v]/var[i+1] +
         (prior_tau[v]^2))
  p_mu <- (prior_mu[v]*(prior_tau[v]^2) +
         (y-bhat[-v]%*%t(X[,-v]))%*%X[,v]/var[i+1])
         * p_tau
  bhat[v] <- rnorm(1, mean = p_mu, sd = sqrt(p_tau))
}
```

4. RMHS
The Metropolis-Hastings procedure is explained in 4.

4. RMHS

blim's workhorse "rmhs" method uses **Gibbs sampling** just like in method "rgs" for conjugate priors and a **random-walk Metropolis** algorithm for evaluating the posterior in case of **nonconjugate priors**:

```
for (v in 1:k){
  if (fun_prior_b[v] == "dnorm"){
    # Gibbs procedure if prior is conjugate
  } else {
    # Likelihood variance and mean
    l_tau <- var[i+1]/t(X[,v])%*%X[,v]
    l_mu <- (y-bhat[-v]%*%t(X[,-v]))%*%X[,v]/
              var[i+1] * l_tau
    # posterior <- prior * likelihood
    fx <- function(x){
      eval(parse(text=prior_beta[v])) *
      dnorm(x, mean = l_mu, sd = sqrt(l_tau))}
    # Sample a candidate value
    bstar <- bhat[v] + rnorm(1, 0, tune[v])
    # Compute the acceptance ratio
    r <- min(1,fx(bstar)/fx(beta[i,v]))
    # Assign when accepted
    if (runif(1) <= r) bhat[v] <- bstar
  }
}
```

5. The Example

Birthweight data. 189 babies were weighed. Their mothers were checked for **smoking** during pregnancy and history of **hypertension**.

	No Smoking	Smoking
No HY	μ1	μ2
HY	μ3	μ4

In addition, **mother's weight** at the last menstrual period was measured. This could act as a covariate. Using the table above, I can specify an **informative hypothesis** about the data: I expect **μ1 > μ2 > μ3 > μ4**, based on my "extensive" medical background. To test the hypothesis, I specify an **ANOVA**-like model with dummy variables:

$$\text{bwt}_i = \mu_1 \cdot \text{no}_i + \mu_2 \cdot \text{sm}_i + \mu_3 \cdot \text{hy}_i + \mu_4 \cdot \text{smhy}_i + \varepsilon_i$$

In the blim package, this simply translates to:

```
mod1 <- blim(bwt~0+no+sm+hy+smhy,
             data = birthwt,
             iter = 99999,
             mtsprior = TRUE,
             method = "rmhs")
```

6. Model Selection

I specify a second model with mother's weight (lwt) as a standardised covariate. I give it a **cauchy prior** with location 0 and scale 100, as the effect will be in terms of grams of birthweight. This seems a reasonable prior, as I don't know whether the effect is positive or negative, but I do have expectations about the approximate range.

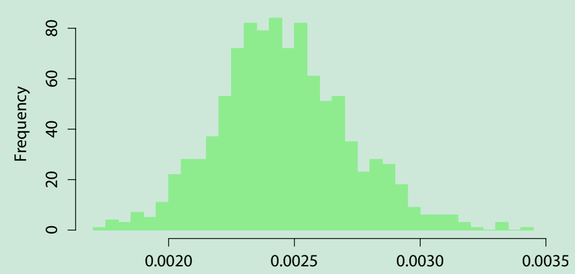
```
mod2 <- blim(bwt~0+no+sm+hy+smhy+scale(lwt),
          data = birthwt,
          # Mu priors the same as mtsprior for mod1
          prior_b = c(rep("dnorm(2950,402403)",4)
                    ,"dcauchy(0,100)"),
          method = "rmhs",
          # Dynamic tuning every 100th iter (see 10)
          dtuning = 100)
```

Unlike frequentist statistics, where model selection too often happens through p-values, Bayesian analysis allows for model selection with **Bayes Factors**, quantifying the amount of evidence there is for model 1 relative to model 2. blim can calculate a Bayes Factor comparing two models using the **Laplace-Metropolis estimator**:

$$f(\mathrm{D}) \approx (2\pi)^{P/2} \left|\mathbf{H}^*\right|^{1/2} f(\theta^*) f(\mathrm{D}|\theta^*)$$

Where the quantities **H*** and θ^* are estimated via the trace covariance matrix and the conditional medians, respectively. It can also quantify the **precision** of this estimate using a **bootstrap**. The precision increases when the amount of iterations in the sampler increases.

```
> BF(mod1, mod2, bootstrap = TRUE, plot = TRUE)
```

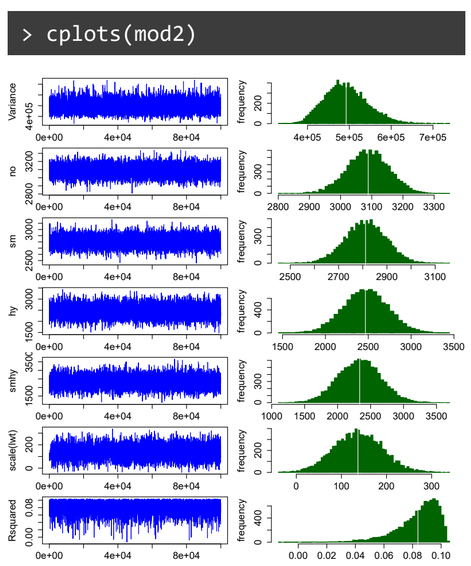


BF12 is around 0.0025. This indicates that model 2 is 400 times as likely as model 1. Here, **DIC** difference ≈ 15.7

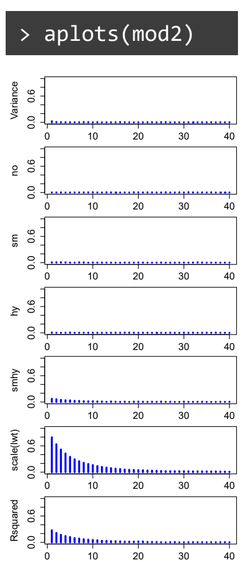
7. Convergence

Convergence and autocorrelation for the parameters and R^2 of model 2 are shown below. Note that the trace plot indicates no lack of convergence, the autocorrelation for the μs is low, and the autocorrelation for the covariate parameter with nonconjugate prior drops off quickly.

```
> cplots(mod2)
```




```
> aplots(mod2)
```



8. Posterior Predictive Check

An important assumption of the ANOVA is the **normality of residuals**. blim can construct a p-value for this using the kolmogorov-smirnov statistic as its **discrepancy measure**. In R the observed discrepancy vector is calculated in this way:

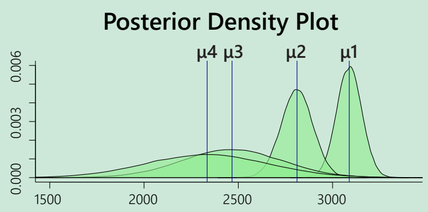
```
ObsD <- apply(trace,1,function(b){
     resid <- blimfit$y-blimfit$X%*%b
     D <- ks.test(resid,pnorm)$statistic
     return(D)
}
```

For the repD, yrep is calculated as follows for each iter:

```
y <- apply(blimfit$X%*%b,1,function(mean){
                              rnorm(1,mean,sqrt(var))}
```

Calling `PPC(mod2, type = "norm")` yields a posterior predictive p-value of around 0.5, indicating no violation of the residual normality assumption.

9. Conclusion



Covariate: mean 136, 95% CI [32;245]. For 1 sd increase in mother's last weight, birthwt increases by 136 grams.

Birthwt nothing (μ1): 3089 grams, 95% CI [2957; 3221]
Birthwt smoke (μ2): 2811 grams, 95% CI [2645; 2978]
Birthwt hypert (μ3): 2469 grams, 95% CI [1946; 2992]
Birthwt sm+hy (μ4): 2337 grams, 95% CI [1696; 2971]

Covariate: 136, 95% CI [32;245]. (For 1 sd increase in mother's weight, birthweight increases by 136 grams)

The means seem to have the right direction, but I want to test the hypothesis! bllim's inequality constraint BF estimator evaluates the hypothesis in the trace of the posterior (**fit**) and a randomly generated prior trace (**complexity**):

```
model = "par[1] > par[2] & par[2] > par[3] & par[3] > par[4]"
fit <- mean(apply(trace, 1,
                  function(par) eval(parse(text=model))))
com <- mean(apply(prior, 1,
                  function(par) eval(parse(text=model))))
BF <- fit/com
```

My hypothesis has a Bayes Factor of around 12, which indicates **strong evidence** relative to the unconstrained hypothesis!

10. Dynamic Tuning

blim performs tuning of the Metropolis algorithm acceptance rate by dynamically optimising the **variance of the proposal distribution**. This variance starts at 2.38 for each conditional posterior and is regulated based on keeping the acceptance rate within a margin, where the optimal acceptance rate is set at 0.45. blim reports final variance.

```
# pcur is current acceptance rate, popt=0.45
if (pcur > 0.5 || pcur < 0.15){
tune[v] <- (tune[v]*(1/pnorm(popt/2)))/
                (1/pnorm(pcur/2))
}
```