Bayesian linear regression with the blm package

Jasper Ginn

The `blm` package contains a simple Gibbs sampler to run a Bayesian Linear Regression model. It is written partly in R and partly in Julia. The bridge between these two languages is handled by the JuliaCall package.

Julia is a high-performing and high-level language that provides a +- 18x speedup compared to R. In the case of a Gibbs sampler, we repeatedly sample from conditional posterior distributions, which amplifies the need for speed and efficiency.

0.1 Setting up blm

To set up the blm package, we load in in R

```
library(blm)
```

```
## Loading required package: magrittr
```

This also loads the magrittr package, which is helpful because we can chain commands using the forward-operating pipe command (`%>%`).

Next, we load the Julia environment

```
blm_setup()
```

```
## Julia version 1.0.3 at location /home/jasper/julia-1.0.3/bin will be used.
```

```
## Loading setup script for JuliaCall...
```

```
## Finish loading setup script for JuliaCall.
```

This can take a couple of seconds.

0.2 The data

The data we will use comes from the exercises

```
## Exercise 2 data
d <- haven::read_sav("../testing/Exercise 2 - Data.sav")
df <- as.data.frame(d)
knitr::kable(head(df))
```

extraversion	agreeableness	attitude
-3.000000	0.6666667	2.7500000
-2.500000	0.3333333	0.5833333
-2.500000	1.3333333	2.1666667
-2.400000	1.8333333	1.6666667
-2.333333	0.1666667	1.5000000
-2.166667	3.0000000	1.4166667

This is simulated data. We can plot it with a simple 3D scatterplot

```
# Plot data
library(scatterplot3d)
scatterplot3d(df$extraversion, df$agreeableness, df$attitude, color="lightgreen", xlab="extraversion",
              pch=21, ylab="agreeableness", zlab="attitude", main="Attitude ~ agreeableness + extraversi
```

For comparison purposes, we run a linear regression (lm) model using Maximum Likelihood

```
# Linear model for comparison
ffit <- lm("attitude ~ extraversion + agreeableness", data=df)
summary(ffit)
```

```
##
## Call:
## lm(formula = "attitude ~ extraversion + agreeableness", data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5003 -0.5147 -0.0182  0.5889  1.7572
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.86919    0.15388   5.649 6.88e-08 ***
## extraversion  -0.08210    0.05639  -1.456  0.14731
```

```
## agreeableness  0.25971    0.09014   2.881  0.00449 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8441 on 166 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.04964,    Adjusted R-squared:  0.03819
## F-statistic: 4.335 on 2 and 166 DF,  p-value: 0.01461
```

0.3 Running a blm model

To fit a blm model, we call the `blm()` function, which is similar to the `lm()` function.

```
bfit <- blm("attitude ~ extraversion + agreeableness",
            data=df, center = TRUE)
```

The `center = TRUE` option centers the data so that we can remove traces of autocorrelation.

At this point, we can call `print()` and `summary()` on the data to check the model status.

```
print(bfit)
```

```
## Bayesian Linear Model (BLM) object:
##
## Data:
##  Predictors: 2
##  Outcome: attitude
##  Centered: TRUE
##
## Sampler:
##  Chains: 1
##  Iterations: 10000
##  Thinning: 1
##  Burn: 1000
##
```

```
## Priors (Coefficients) :
##       b0   b1   b2
## mu     0    0    0
## tau 1000 1000 1000
##
## Priors (Residuals) :
##       sigma
## rate   0.01
## scale  0.01
```

This gives us information about the data, the sampling options and the priors, which are uninformative at this point.

```
summary(bfit)
```

```
## Model results for blm object:
##
## Formula: 'attitude ~ extraversion + agreeableness'
##
## Sampled: FALSE
##
## Sampling settings :
##  Obs. Predictors Chains Iterations Thinning Burn
##   169          2      1      10000        1 1000
```

`summary()` gives us information about the model fit, MAP estimates and so on. Given that we have not sampled the posterior, however, these statistics are not yet supplied.

We can update sampling settings as follows

```
bfit <- bfit %>%
  # Update sampling settings
  sampling_options(., chains = 3, iterations = 15000, thinning = 2, burn = 2000)
```

Priors may be made informative as follows

```r
bfit <- bfit %>%
  set_priors("b2" = prior("normal", mu=0.5, sd=2))
print(bfit)
```

```
## Bayesian Linear Model (BLM) object:
##
## Data:
##  Predictors: 2
##  Outcome: attitude
##  Centered: TRUE
##
## Sampler:
##  Chains: 3
##  Iterations: 15000
##  Thinning: 2
##  Burn: 2000
##
## Priors (Coefficients) :
##       b0   b1  b2
## mu     0    0 0.5
## tau 1000 1000 2.0
##
## Priors (Residuals) :
##       sigma
## rate   0.01
## scale  0.01
```

If we are happy with these settings, we can sample from the posterior distribution. This process takes the longest during the first time you call the `sample_posterior()` function. This happens because Julia's Just-In-Time (JIT) compiler compiles the code when it is called the first time. In subsequent evaluations of the code, it then uses the already compiled code.

```
# First time we call the Julia code
t1 <- Sys.time()
bfit <- bfit %>%
  sample_posterior()
print(Sys.time() - t1)
```

```
## Time difference of 4.634263 secs
```

```
# Subsequent iterations
t1 <- Sys.time()
bfit <- bfit %>%
  sample_posterior()
print(Sys.time() - t1)
```

```
## Time difference of 1.411839 secs
```

To observe the diagnostic information about the posterior sampling, we call `summary()`

```
summary(bfit)
```

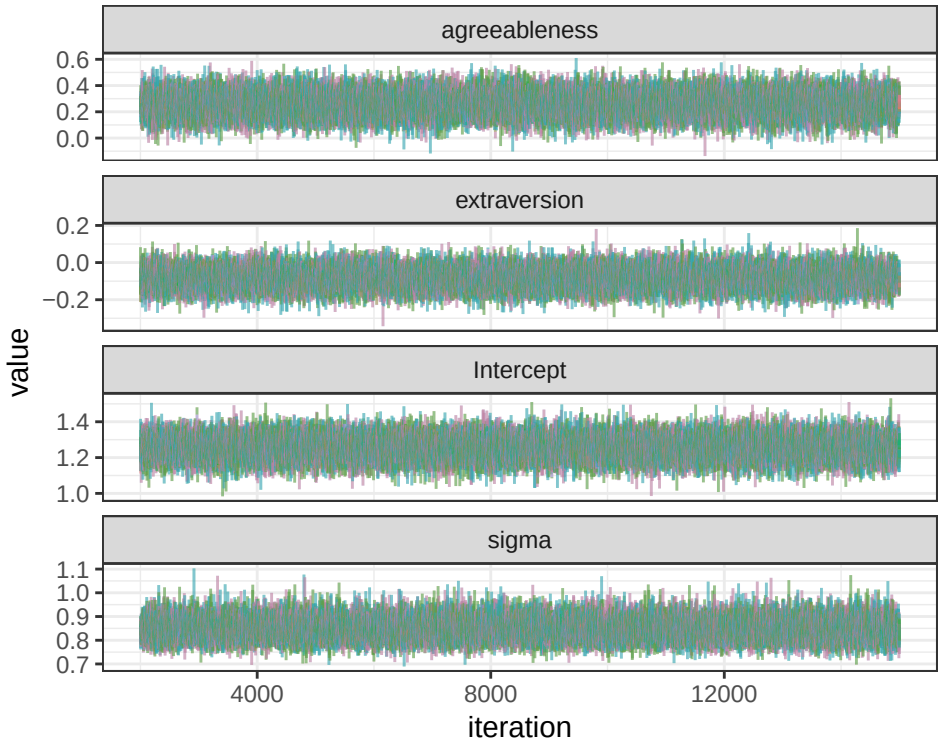
```
## Model results for blm object:
##
## Formula: 'attitude ~ extraversion + agreeableness'
##
## Sampled: TRUE
##
## Sampling settings :
##  Obs. Predictors Chains Iterations Thinning Burn
##   169          2      3      15000        2 2000
##
## Maximum a posteriori (MAP) estimates :
##       Est. (mean)    SD MCERR.
## b0          1.256 0.065  0.001
## b1         -0.083 0.057  0.000
```

```
## b2          0.260 0.090  0.001
## sigma       0.848 0.047  0.000
##
## 95% credible interval :
##          2.5%    25%    50%    75% 97.5%
## b0      1.128  1.212  1.256  1.300 1.384
## b1     -0.193 -0.121 -0.082 -0.045 0.029
## b2      0.082  0.200  0.260  0.321 0.438
## sigma   0.763  0.815  0.846  0.878 0.946
```

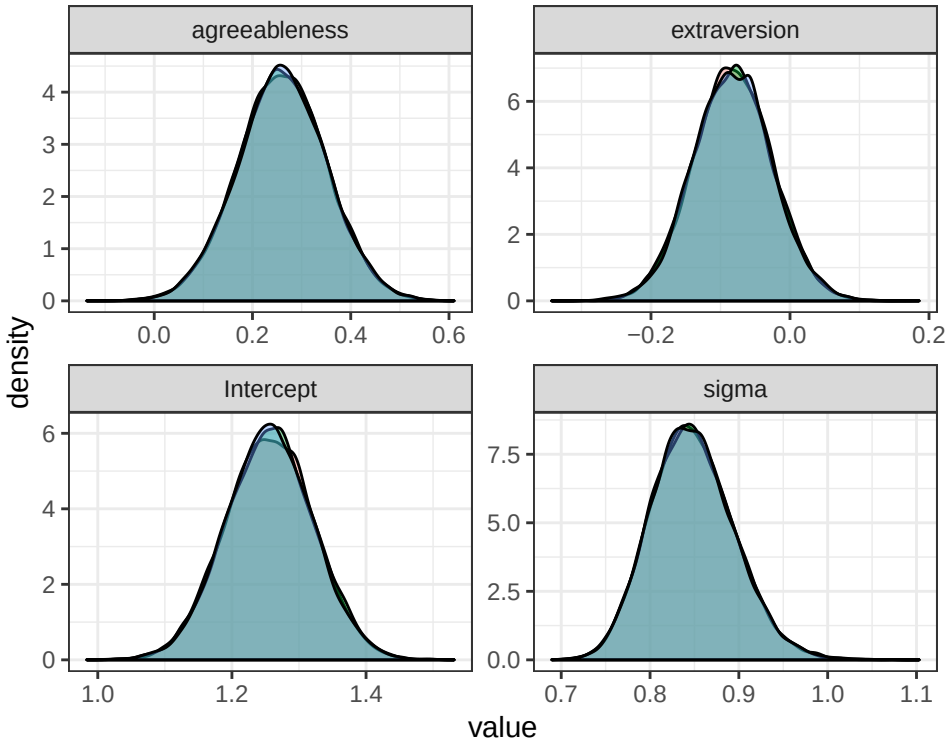
These diagnostics give us the MAP estimates, credible intervals and Gelman-Rubin statistic for each of the coefficients. The Burn-in diagnostics are the coefficient of a linear regression performed on the squared posterior values against the iteration values. If there is a trend in the posterior values, the coefficient will be removed further away from 0.

Next, we can plot the history plots, density plots for the posterior values and autocorrelation plots.

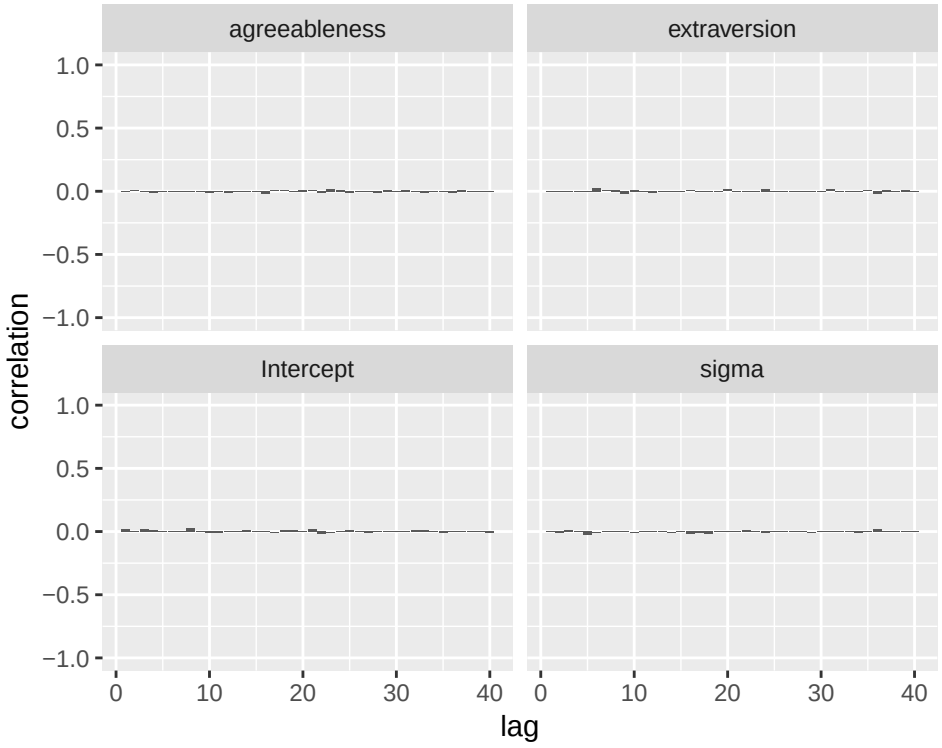
```
plot(bfit, "history")
```



```
plot(bfit, "density")
```



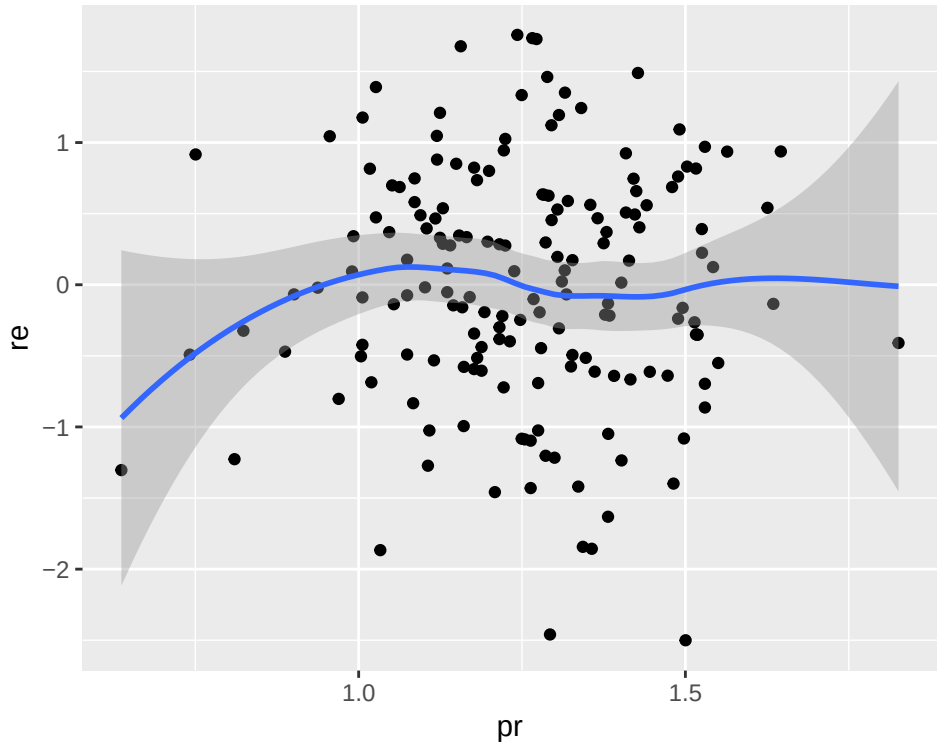

```
plot(bfit, "autocorrelation", chain=1)
```



If these values seem OK, we can move on to the posterior predictive checks. These are simulations that check certain assumptions of the data, in this case normality and heteroskedasticity, both of which are violated by our data, as we can see from the residual plot from the linear regression model

```
library(ggplot2)
ggplot(data.frame("pr" = predict(ffit),
                  "re" = resid(ffit)),
       aes(x=pr, y=re)) +
    geom_point() +
    geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



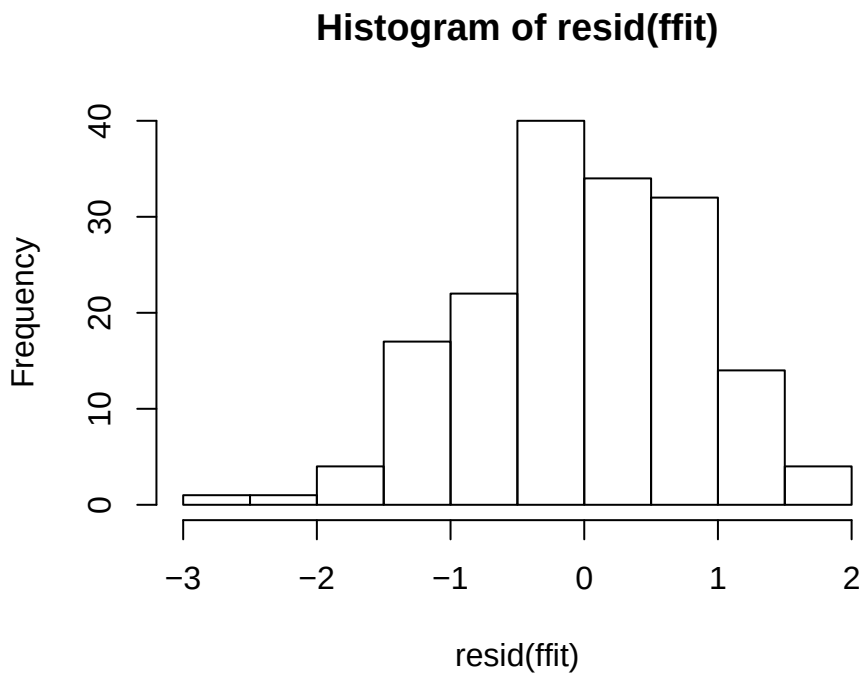
We can run the posterior predictive checks to test these assumptions as follows

```
ppc <- bfit %>%
  posterior_predictive_checks()
summary(ppc)
```

```
## Posterior Predictive Checks (PPC) for blm object:
##
## Bayesian p-value :
##   Normality Heteroskedasticity Independence
## p     0.786               0.68            0
```

We see that we are mainly violating the normality or errors assumption

```
hist(resid(ffit))
```



Finally, we can check the fit of the model against an intercept-only model using the DIC

```
bfit %>%
  model_fit()
```

```
## Model fit for blm object:
##
## Model DIC :
##               LL     DIC Eff. P
## (Model) -209.675 423.372  4.023
```