Bayesian Linear Regression

Bruna Wundervald

2018-10-14

The regression problem involves determining the relationship between some responde variable Y and a set of predictor variables $\mathbf{X} = (X_1, ...X_p)$. Usually, we assume that this relationship can be described by a deterministic function f and some additive random error that follows a Normal distribution centered in 0 and with variance equals to σ^2:

$$Y = f(\mathbf{X}) + \epsilon$$

The predictors $\mathbf{X}$ are assumed to be observed without error, so they're not considered **random**. We can check that

$$f(\mathbf{X}) = E[Y|\mathbf{X} = \mathbf{x}]$$

meaning that the $f(\mathbf{X})$ describes the expectation over Y when $\mathbf{X}$ is observed. The true regression function is unknown and we have no way of determining it its analytic form exactly, even if it exists. What we do is find approximations which are the closest to the truth as possible.

Basis functions

Assuming that f is made up of a linear combination of basis functions and the correspondent coefficients, it can be written as

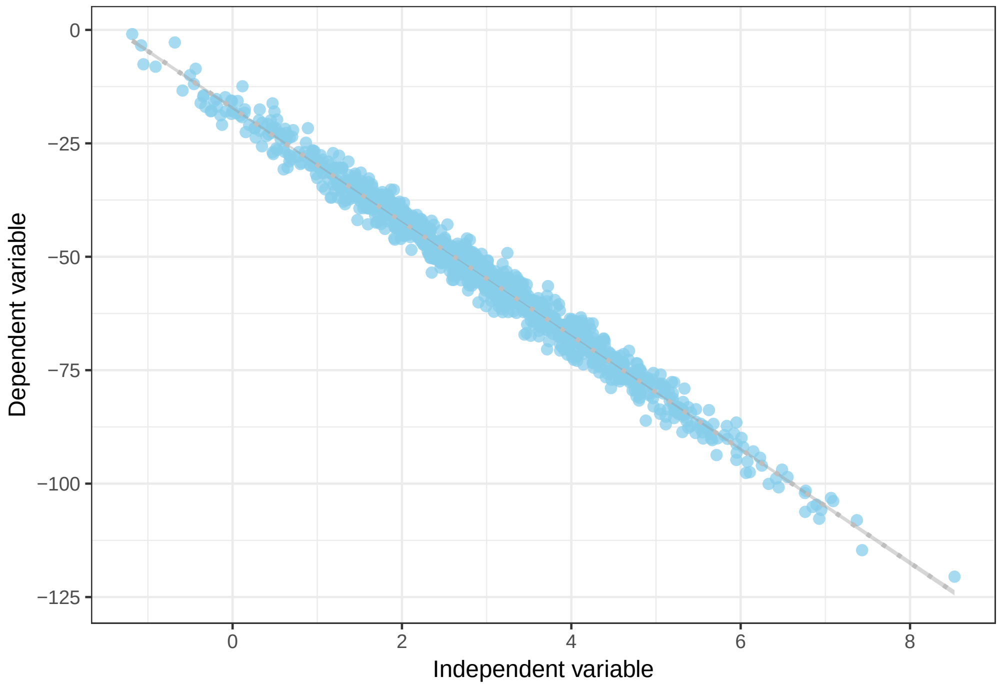
$$f(\mathbf{x}) = \sum_{i=1}^{k} \beta_i B_i(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X}$$

where $\beta = (\beta_i, ...\beta_k)$ is the set of coefficients corresponding to basis functions $\mathbf{B} = (Bi, ...Bk)$.

The Classic Linear Model

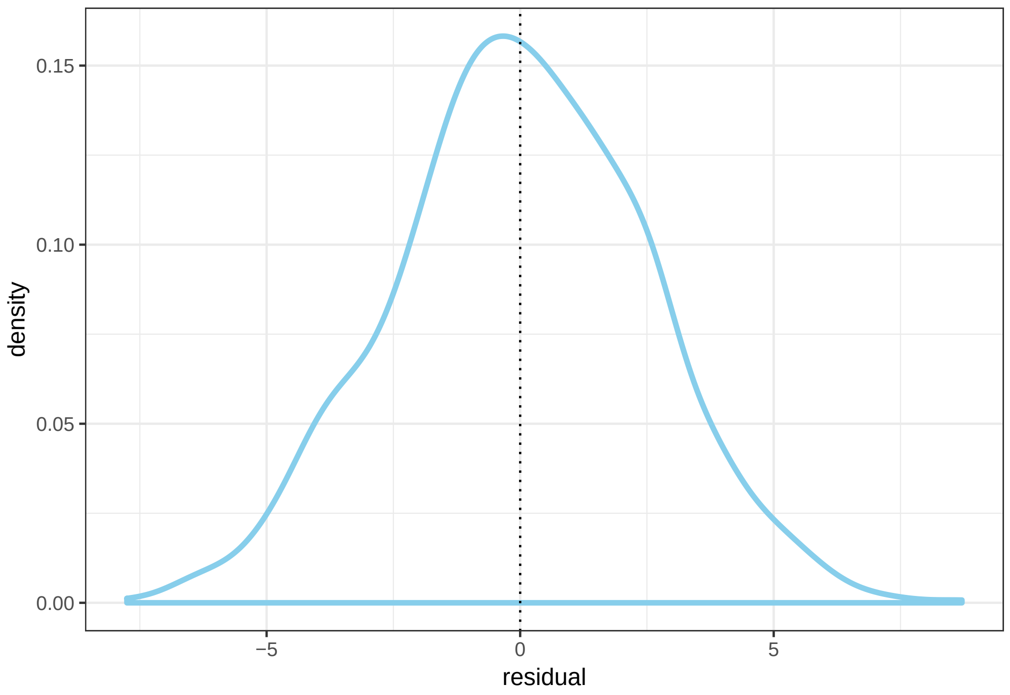
```r
library(tidyverse) # Always

# Simulating  some data ----------------------------------
set.seed(2018) # reprodutibility

# Simulating the independent variable (arbitrarily
# chosen as ~Normal and the error ~ N(0, sigma^2)
sigma <- 1/rgamma(n = 1, shape = 0.5, rate = 1)
x <- rnorm(n = 1000, mean = 3, sd = 1.5)
e <- rnorm(n = 1000, mean = 0, sd = sqrt(sigma))
V <- matrix(sigma*c(10, 0, 0, 10), ncol = 2, nrow = 2)

# Priors of the parameters - intercept and slope
betas <- MASS::mvrnorm(n = 1, mu = c(0, 0),
                       Sigma = V)


# The regression model
y <- betas[1] + (betas[2] * x) + e
```

```r
da <- data.frame(y, x)

# Plotting our data
da %>%
  ggplot(aes(x, y)) +
  geom_point(colour = 'skyblue', size = 2, alpha = 0.75) +
  geom_smooth(method = 'lm', colour = 'grey', linetype = 'dotted') +
  theme_bw() +
  labs(x = 'Independent variable', y = 'Dependent variable')
```



```r
# The classical regression model
# With functions:
lm(y ~ x) %>% summary()
#>
#> Call:
#> lm(formula = y ~ x)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -7.7544 -1.5793 -0.0066  1.7400  8.7116
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -17.21746    0.17521  -98.27   <2e-16 ***
#> x           -12.53167    0.05199 -241.06   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.502 on 998 degrees of freedom
#> Multiple R-squared:  0.9831, Adjusted R-squared:  0.9831
#> F-statistic: 5.811e+04 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
# Vanilla flavour:
mm <- model.matrix( ~ x, data = da)
k <-  ncol(mm)
n <- nrow(mm)

# Estimating betas
v <- solve(t(mm) %*% mm)
betas_hat <- c(v %*% t(mm) %*% y)
betas_hat
#> [1] -17.21746 -12.53167

# y_hat
y_hat <- mm %*% betas_hat

da$res <- y - y_hat

da %>%
  ggplot(aes(res)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  geom_vline(xintercept = 0, linetype = 'dotted') +
  theme_bw() +
  labs(x = 'residual')
```



```
# Residual sum of squares
rss <- sum((y - y_hat)^2)

# Mean squared errors
mse <- mean((y - y_hat)^2)

# Rs - Multiple correlation coefficients
```

```r
(R <- sum((y_hat - mean(y))^2)/sum((y - mean(y))^2))
#> [1] 0.983115
(R_adj <- 1 - (1 - R)*((n-1)/(n-k)))
#> [1] 0.9830981

# Estimating the variance of the parameters
var_betas <- solve(t(mm) %*% mm) * mse
var_betas
#>              (Intercept)            x
#> (Intercept)  0.030635895 -0.008110332
#> x           -0.008110332  0.002697209

# t-values
t1 <- betas_hat[2]/sqrt(var_betas[2, 2])
t2 <- betas_hat[1]/sqrt(var_betas[1, 1])

# p-values
# pt(t1, df = n - k) # ?
# pt(t2, df = n - k)
```

The Bayesian Linear Model

The Bayesian approach consists in: 1. assign prior distributions to all the unknown parameters; 2. write down the likelihood of the data given the parameters; 3. determine the posterior distributions of the parameters given the data using Bayes' Theorem.

1. We find thta the conjugate choice of (joint) prior for

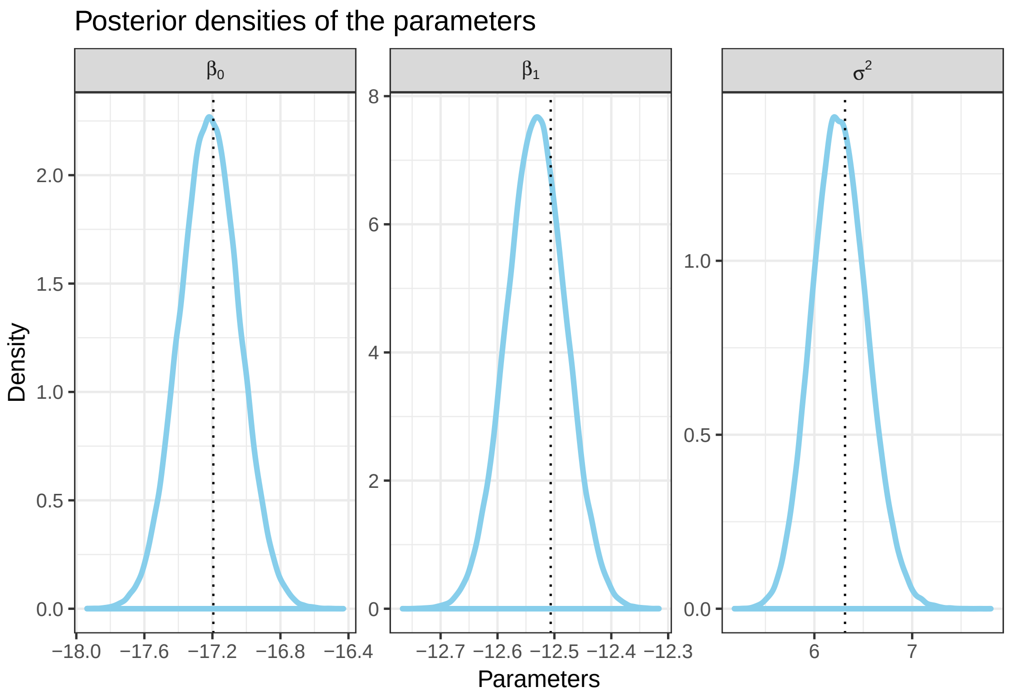
β and σ^2 is the normal inverse-gamma (NIG), denoted by $NIG(\mathbf{m}, \mathbf{V}, a, b)$, with probability density function:

$$p(\beta, \sigma^2) = p(\beta|\sigma^2)p(\sigma^2)$$
$$p(\beta, \sigma^{\mathbf{2}}) = N(\mathbf{m}, \sigma^2\mathbf{V}) \times IG(a, b)$$

So, the β, given σ^2 are assumed to have a Normal distribution, since its domain goes from $-\infty$ to $+\infty$, and its mean and variance can be adjusted accordingly to the expertise of the one who is building the model. The σ^2 is assumed to have the IG(a, b) distribution as its domain goes from 0 to $+\infty$.

```r
# Bayesian model -----------------------------------------------------

# Priors were already assigned as -------------
# Bs ~ Normal(m, V) = Normal(0, sigma^2 * 10)
# Sigma ~ IG(a, b) = IG(0.5, 1)

# Posterior distribution ----------------------
# Betas | Sigma, y ~ N(m*, V*)
# Sigma | y ~ IG(a*, b*), where:
#
# m* = (V^-1 + B'B)^-1 * (V^-1*m + B'Y)
# V* = (V^-1 + B'B)^-1
# a* = a + n/2
# b* = b + (m' V^-1 m + Y'Y -  (m*)'(V*)^-1m*)/2
```

```r
#---------------------------------------------

v_star <- solve(solve(V) + t(mm) %*% mm)
m_star <- v_star %*% (solve(V) %*% c(0, 0) + t(mm) %*% y)
a_star <- 0.5 + n/2
b_star <- 1 + (t(c(0, 0)) %*% solve(V) %*% c(0, 0) +
                 (t(y) %*% y) - t(m_star) %*% solve(v_star) %*% m_star)/2

# Sampling from the posteriors ---------------------------------------------------
sim <- 100000
gamma <- rgamma(sim, shape = a_star,
                rate = b_star)

# For the variance
sigma_sim <- 1/gamma


# Consider that  you have a random variable
# Y ~ Normal(mu, variance), if you multiply it
# by a constant 'c' the variance is multiplied
# by c squared, aka, cY ~ Normal(mu, variance * c^2)

# For the random error, we used that, if Y ~ Normal(0, v*),
# sqrt(sigma) * Y ~ Normal(0, v*sigma), which is our
# target distribution

err <- sqrt(sigma_sim)*MASS::mvrnorm(n = sim, mu = c(0, 0),  v_star)

# consider now that we are adding the random
# error ~ Normal(0, v*sigma) to a constant
# (the estimated mean for the betas), which will lead us to have
# beta ~ Normal(m_star, v*sigma), as we just added a
# constant to the location of the distribution

params <- data.frame(par = c(rep(c(m_star), each = sim) +
                         c(err[,1], err[,2]), sigma_sim))

params$groups <- as.factor(rep(c(1:3), each = sim))
params$groups_label <- factor(params$groups, labels =
                          c('beta[0]', 'beta[1]', 'sigma^2'))


params_prior <- c(betas, sigma)

vline <- function(group){
  geom_vline(data = dplyr::filter(params,
                           groups == group),
             aes(xintercept = params_prior[group]), linetype = 'dotted')
}

params %>%
  ggplot(aes(par)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  1:3 %>% purrr::map(vline) +
  facet_wrap(~groups_label, scales = 'free',
             labeller = label_parsed) +
  theme_bw() +
  labs(x = 'Parameters', y = 'Density',
       title = 'Posterior densities of the parameters')
```



Real Data

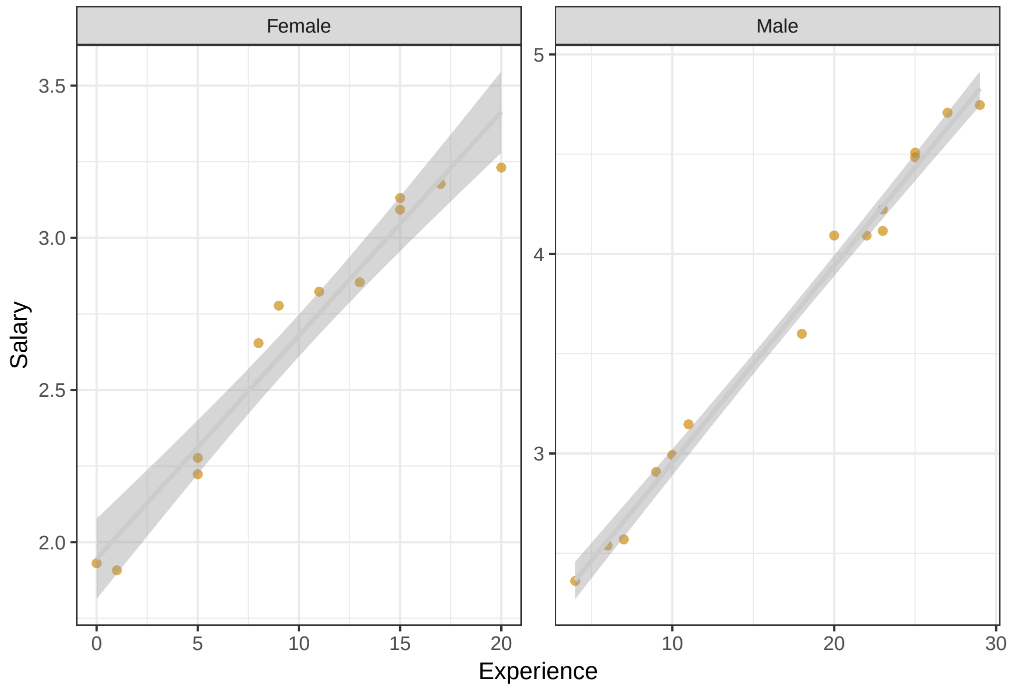
Data with the salary, years of experience and gender of bank branch managers of a big bank.

```
# devtools::install_github('pet-estatistica/labestData')
da <- labestData::CharnetEg7.3 %>%
  mutate(gender = factor(sexo, labels = c('Female', 'Male'))) %>%
  select(-sexo)

head(da) %>% knitr::kable()
```

salario	exp	gender
1.93077	0	Female
3.17692	17	Female
2.27692	5	Female
3.13077	15	Female
2.77692	9	Female
3.09231	15	Female

```
da %>%
  ggplot(aes(y = salario, x = exp)) +
  geom_point(colour = 'orange3', alpha = 0.65, size = 1.5) +
```

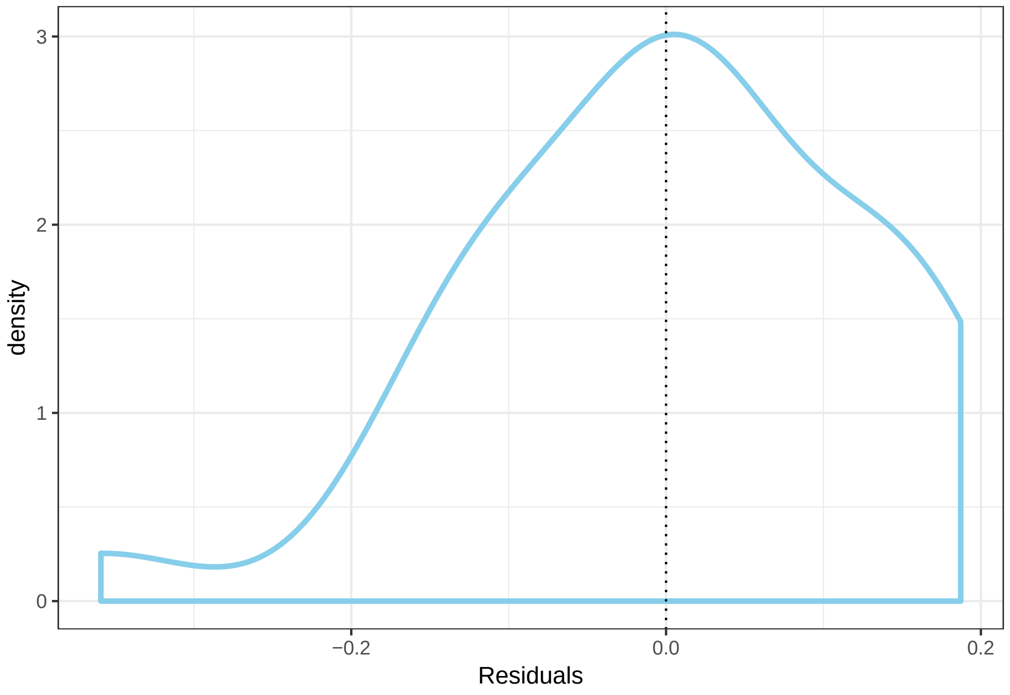
```r
  geom_smooth(method = 'lm', colour = 'grey80') +
  facet_wrap(~gender, scales = 'free') +
  theme_bw() +
  labs(y = 'Salary', x = 'Experience')
```



```r
# The classic model
lm(salario ~ exp + gender, data = da) %>% summary()
#>
#> Call:
#> lm(formula = salario ~ exp + gender, data = da)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.35905 -0.08887 -0.00299  0.08037  0.18718
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 1.771484   0.050981  34.748  < 2e-16 ***
#> exp         0.090917   0.003436  26.461  < 2e-16 ***
#> genderMale  0.330990   0.056801   5.827 5.22e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.1314 on 24 degrees of freedom
#> Multiple R-squared:  0.9784, Adjusted R-squared:  0.9766
#> F-statistic:   543 on 2 and 24 DF,  p-value: < 2.2e-16
```

```r
# The classic model by hand -----------------------------------------------
mm <- model.matrix(~ exp + gender, data = da)
k <-  ncol(mm)
n <- nrow(mm)
v <- solve(t(mm) %*% mm)
```

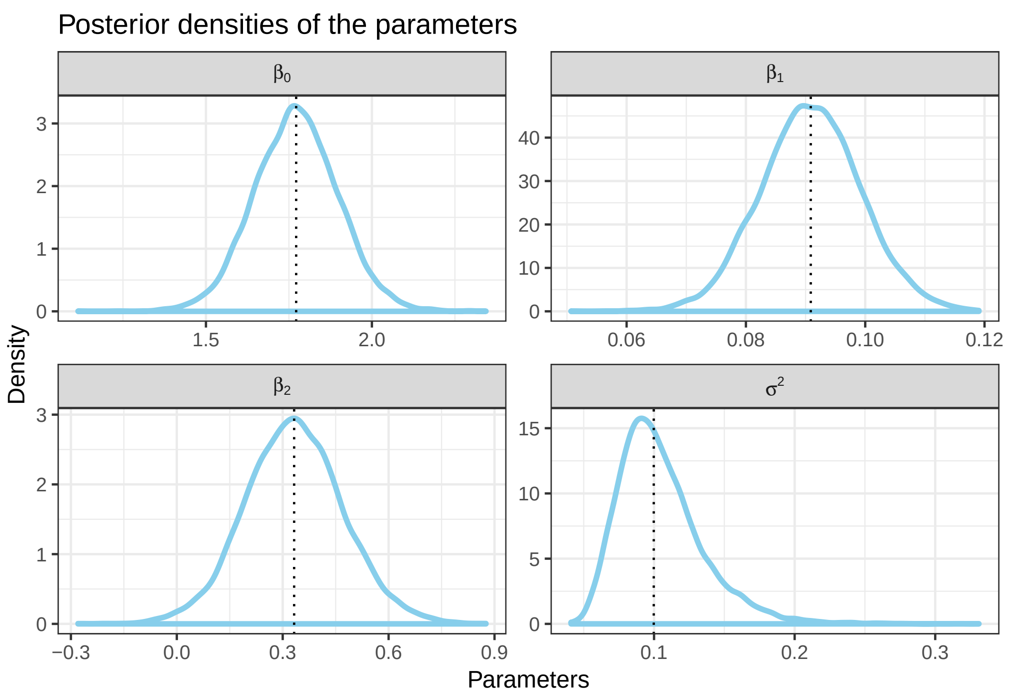
```
betas_hat <- v %*% t(mm) %*% da$salario
betas_hat
#>                   [,1]
#> (Intercept) 1.77148445
#> exp         0.09091695
#> genderMale  0.33099028

y_hat <- mm %*% betas_hat
da$res <- da$salario - y_hat

da %>%
  ggplot(aes(res)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  geom_vline(xintercept = 0, linetype = 'dotted') +
  theme_bw() +
  labs(x = 'Residuals')
```



```
# Bayesian model ---------------------------------------------------

# Priors were already assigned as -------------
# Bs ~ Normal(m, V) = Normal(0, sigma^2 * 10)
# Sigma ~ IG(a, b) = IG(0.5, 1)

# Posterior distribution ---------------
# Betas | Sigma, y ~ N(m*, V*)
# Sigma | y ~ IG(a*, b*), where:
#
# m* = (V^-1 + B'B)^-1 * (V^-1*m + B'Y)
# V* = (V^-1 + B'B)^-1
# a* = a + n/2
```

```r
# b* = b + (m' V^-1 m + Y'Y -  (m*)'(V*)^-1m*)/2
#----------------------------------------

V <-  diag(10, nrow = dim(mm)[2], ncol = dim(mm)[2])
v_star <- solve(solve(V) + t(mm) %*% mm)
m_star <- v_star %*% (solve(V) %*% c(0, 0, 0) + t(mm) %*% da$salario)
a_star <- 0.5 + n/2
b_star <- 1 + (t(c(0, 0, 0)) %*% solve(V) %*% c(0, 0, 0) +
                 (t(da$salario) %*% da$salario) -
                 t(m_star) %*% solve(v_star) %*% m_star)/2

# Sampling from the posteriors -------------------------------
sim <- 10000
gamma <- rgamma(sim, shape = a_star, rate = b_star)

# For the variance
sigma_sim <- 1/gamma


# Consider that  you have a random variable
# Y ~ Normal(mu, variance), if you multiply it
# by a constant 'c' the variance is multiplied
# by c squared, aka, cY ~ Normal(mu, variance * c^2)

# For the random error, we used that, if Y ~ Normal(0, v*),
# sqrt(sigma) * Y ~ Normal(0, v*sigma), which is our
# target distribution

err <- sqrt(sigma_sim) * MASS::mvrnorm(n = sim, mu = rep(0, 3), Sigma  = v_star)

# consider now that we are adding the random
# error ~ Normal(0, v*sigma) to a constant
# (the estimated mean for the betas), which will lead us to have
# beta ~ Normal(m_star, v*sigma), as we just added a
# constant to the location of the distribution

params <- data.frame(par = c(rep(c(betas_hat), each = sim) + as.vector(err),
                             sigma_sim))

params$groups <- as.factor(rep(c(1:4), each = sim))
params$groups_label <- factor(params$groups, labels =
                                c('beta[0]', 'beta[1]',
                                  'beta[2]', 'sigma^2'))

# Finding the 'optimal parameters' accordingly to
# a defined loss function - the values for the parameters that
# better represent the uncertainty after observing the data

# Loss for betas: chosen to be the mean squared residual
loss <- function(x, y) {
  mean((y - x)^2)
}
```

```
b0 <- optimize(function(x) loss(x, params$par[1:10000]),
               interval = c(-100, 100))$minimum
b1 <- optimize(function(x) loss(x, params$par[10001:20000]),
               interval = c(-100, 100))$minimum
b2 <- optimize(function(x) loss(x, params$par[20001:30000]),
               interval = c(0, 1))$minimum

# Loss for sigma: chosen to be the mean of the abs(residual)
loss_sigma <- function(x, y) {
  mean(abs(y - x))
}

s <- optimize(function(x) loss_sigma(x, params$par[30001:40000]),
               interval = c(0, 10))$minimum

params_est <- c(b0, b1, b2, s)

vline <- function(group){
  geom_vline(data = dplyr::filter(params,
                                  groups == group),
             aes(xintercept = params_est[group]), linetype = 'dotted')
}

params %>%
  ggplot(aes(par)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  1:4 %>% purrr::map(vline) +
  facet_wrap(~groups_label, scales = 'free',
             labeller = label_parsed) +
  theme_bw() +
  labs(x = 'Parameters', y = 'Density',
       title = 'Posterior densities of the parameters')
```


Posterior densities of the parameters
β0
β1
β2
σ2
Density
Parameters
0
1
2
3
10
20
30
40
5
15
1.5
2.0
0.06
0.08
0.10
0.12
−0.3
0.0
0.3
0.6
0.9
0.1
0.2
0.3