Bayesian Linear Regression

Bruna Wundervald

2018-10-08

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

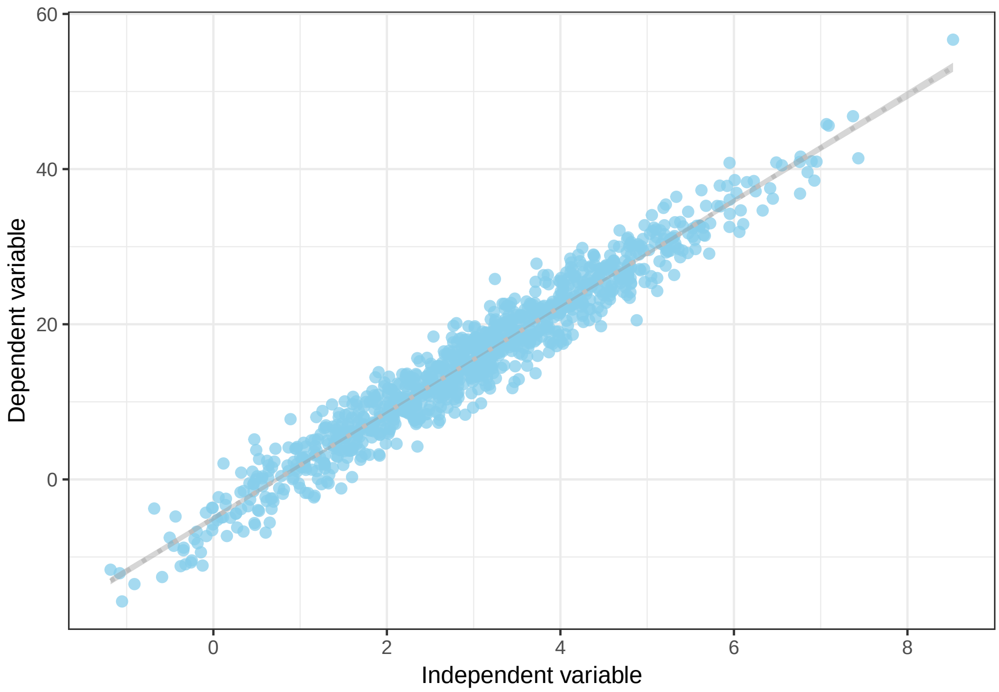
$$f(\mathbf{x}) = \sum_{i=1}^{k} \beta_i B_i(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X}$$

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))

# Priors of the parameters - intercept and slope
# Justify the priors when simulating!
betas <- rnorm(n = 2, mean = 0, sd = sqrt(10))

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

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

```r
# 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) -5.00041    0.17521  -28.54   <2e-16 ***
#> x            6.81798    0.05199  131.15   <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:  0.9452, Adjusted R-squared:  0.9451
#> F-statistic: 1.72e+04 on 1 and 998 DF,  p-value: < 2.2e-16

# Vanilla flavour:
```

```r
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] -5.000406  6.817976

# y_hat
y_hat <- mm %*% betas_hat

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

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

# Rs - Multiple correlation coefficients
(R <- sum((y_hat - mean(y))^2)/sum((y - mean(y))^2))
#> [1] 0.9451586
(R_adj <- 1 - (1 - R)*((n-1)/(n-k)))
#> [1] 0.9451037

# 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
2 * pt(-abs(t1), df = n - k) # ?
#> [1] 0
2 * pt(-abs(t2), df = n - k)
#> [1] 1.151247e-131
```

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.

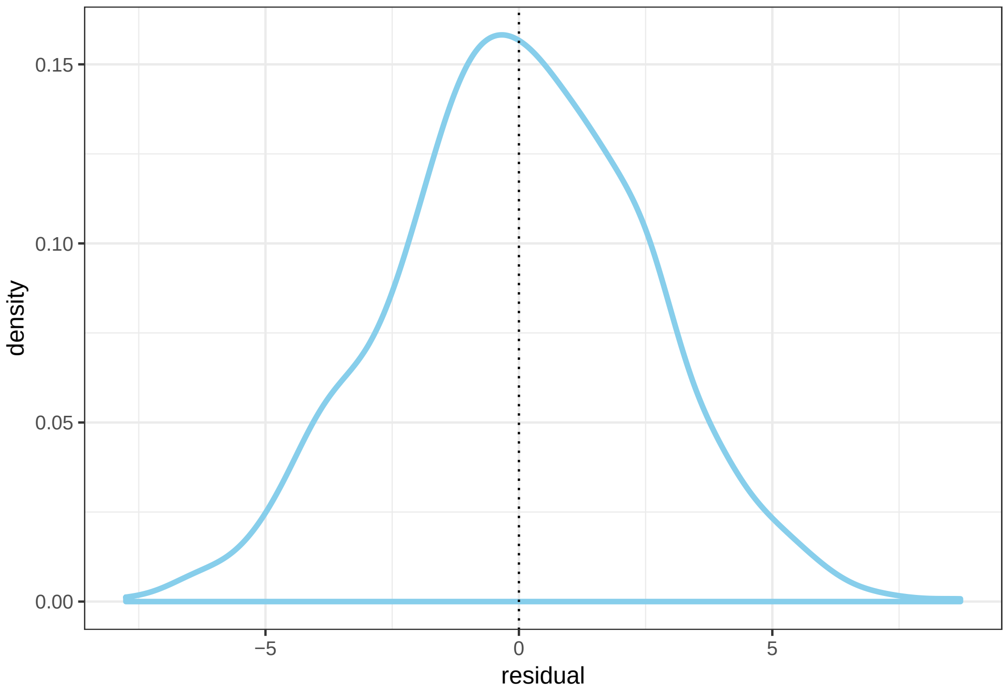
1. We find thta the conjugate choice of (joint) prior for

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

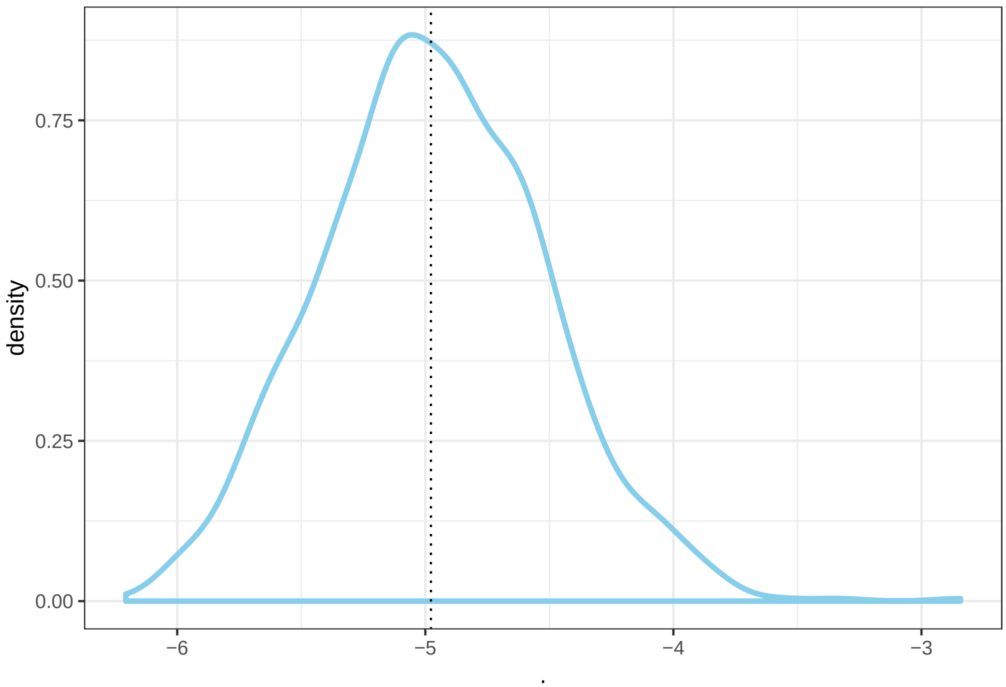
$$p(\beta, \sigma^2) = p(\beta|\sigma^2)p(\sigma^2)$$
$$p(\beta, \boldsymbol{\sigma^2}) = N(\mathbf{m}, \sigma^2\mathbf{V}) \times IG(a, b)$$

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 $+\infty$.

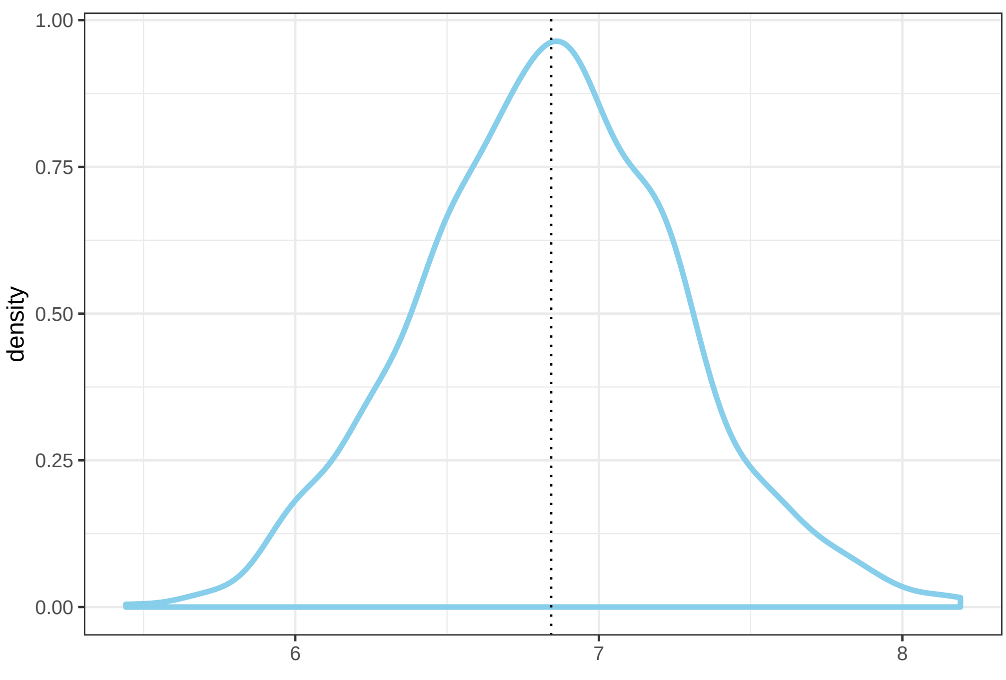
```
# Bayesian model ----------------------------------------------------

# Priors were already assigned
# Bs ~ Normal
# Sigma ~ IG

# The same as before
mm <- model.matrix(~x, data = da)
k <-  ncol(mm)
n <- nrow(mm)
v <- solve(t(mm) %*% mm)
betas_hat <- v %*% t(mm) %*% y

# Posterior distribution:
# Betas | Sigma, y ~ N(beta_hat, \sigma^2 * V_betas)
# Sigma | y ~ IG((n-k)/2, (n-k)*s^2_hat/2)

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')
```




```
# Estimated s2 (of sigma^2)
(s2 <- (t(da$res) %*% da$res)/(n-k))
#>          [,1]
#> [1,] 6.261173

# Simulations ------------------------------------------------------------
sim <- 10000
gamma <- rgamma(sim, shape = (n-k)/2,
                rate = (n-k)*s2/2)

# For the variance
sigma <- 1/gamma

err <- sigma * MASS::mvrnorm(n = sim, mu = c(0, 0), v)
beta_sim <- rep(c(betas), each = 1000) + c(err[,1], err[,2])


beta_sim %>%
  as.data.frame() %>%
  slice(1:1000) %>%
  ggplot(aes(.)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  geom_vline(xintercept = betas[1], linetype = 'dotted') +
  theme_bw()
```




```
beta_sim %>%
  as.data.frame() %>%
  slice(1001:2000) %>%
  ggplot(aes(.)) +
  geom_density(colour = 'skyblue', size = 1.2, alpha = 0.75) +
  geom_vline(xintercept = betas[2], linetype = 'dotted') +
  theme_bw()
```



Real Data

Data Fields

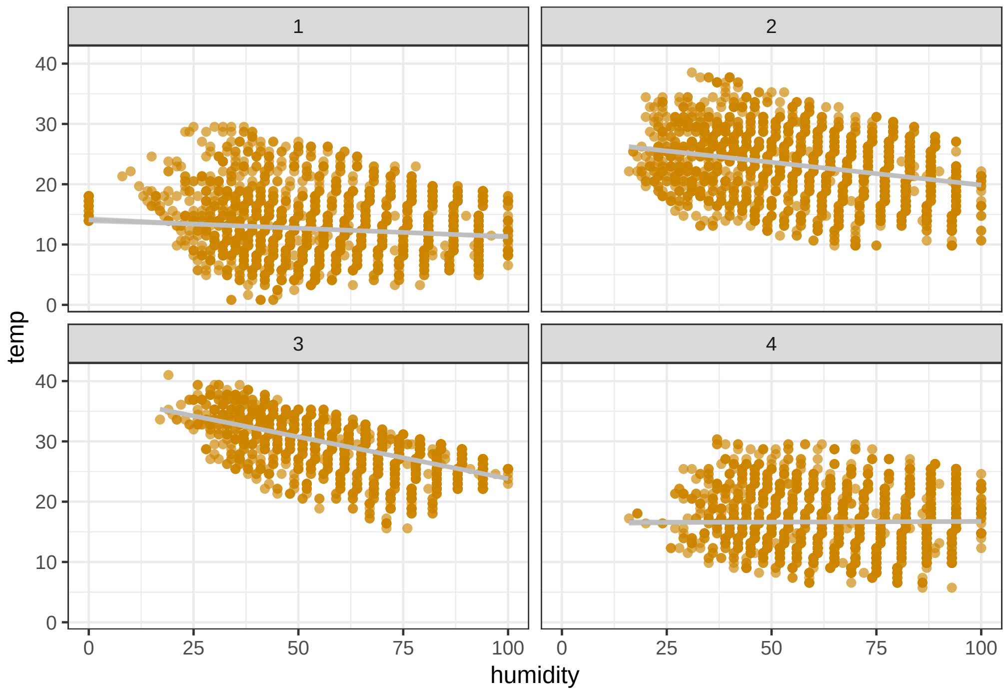
- datetime - hourly date + timestamp

- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday - whether the day is considered a holiday
- workingday - whether the day is neither a weekend nor holiday
- weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp - temperature in Celsius
- atemp - "feels like" temperature in Celsius
- humidity - relative humidity
- windspeed - wind speed
- casual - number of non-registered user rentals initiated
- registered - number of registered user rentals initiated
- count - number of total rentals

```
da <- read.csv('~/Maynooth University/Andrew Parnell - CDA_PhD/data/Kaggle/Bike sharing/train.csv') %>%
  select(temp, humidity, season) %>%
  mutate(season = as.factor(season))
head(da) %>% knitr::kable()
```

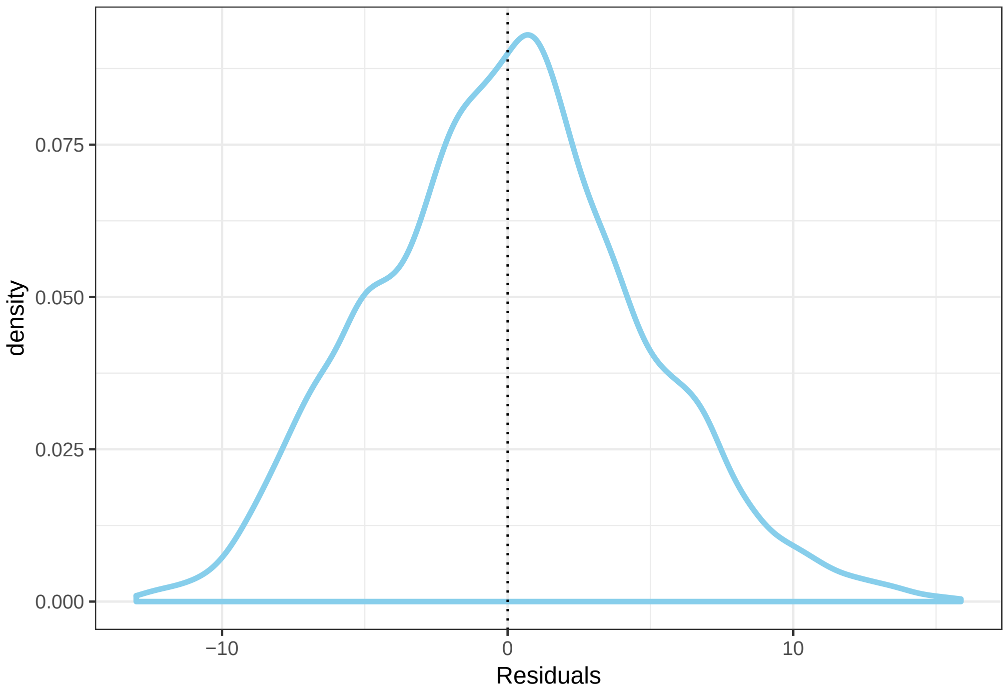
temp	humidity	season
9.84	81	1
9.02	80	1
9.02	80	1
9.84	75	1
9.84	75	1
9.84	75	1

```
da %>%
  ggplot(aes(x = humidity, y = temp)) +
  geom_point(colour = 'orange3', alpha = 0.65, size = 1.5) +
  geom_smooth(method = 'lm', colour = 'grey') +
  facet_wrap(~season) +
  theme_bw()
```

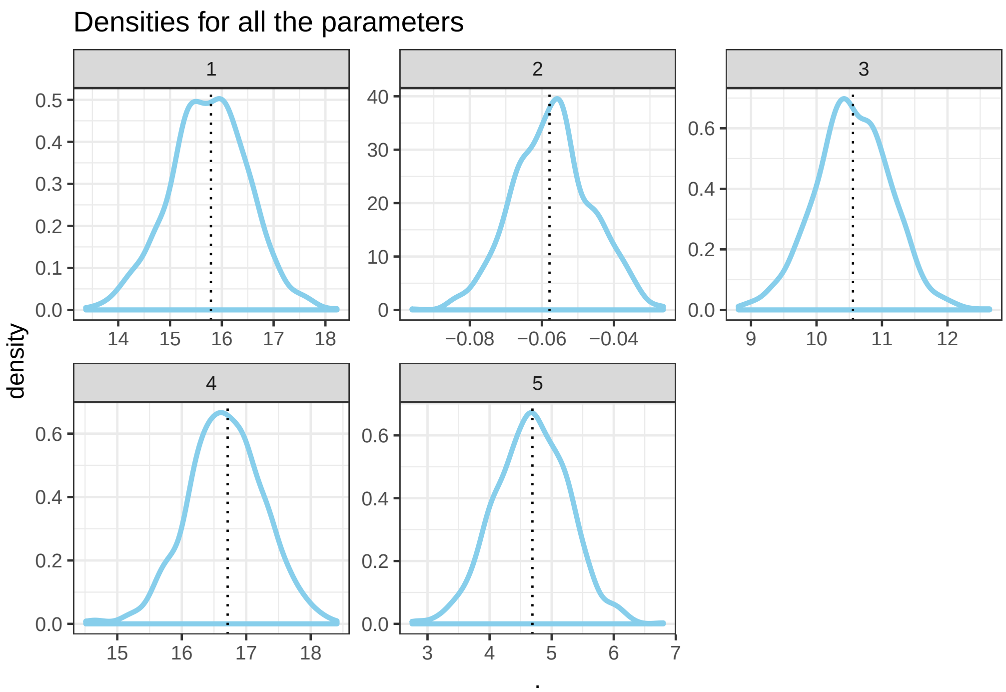



```
lm(temp ~ humidity +  factor(season), data = da) %>% summary()
#>
#> Call:
#> lm(formula = temp ~ humidity + factor(season), data = da)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -13.0011  -3.1032   0.0002   2.9715  15.8725
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     15.789079   0.160224   98.54   <2e-16 ***
#> humidity        -0.057881   0.002359  -24.54   <2e-16 ***
#> factor(season)2 10.556645   0.126703   83.32   <2e-16 ***
#> factor(season)3 16.711588   0.127589  130.98   <2e-16 ***
#> factor(season)4  4.690377   0.128367   36.54   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.647 on 10881 degrees of freedom
#> Multiple R-squared:  0.6445, Adjusted R-squared:  0.6444
#> F-statistic:  4931 on 4 and 10881 DF,  p-value: < 2.2e-16
lm(temp ~ humidity, data = da) %>% summary() # Explains nothing
#>
#> Call:
#> lm(formula = temp ~ humidity, data = da)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -20.1441  -6.4116   0.1718   6.4329  19.6414
#>
```

```r
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 21.858184   0.250977   87.09  < 2e-16 ***
#> humidity    -0.026295   0.003873   -6.79 1.18e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 7.775 on 10884 degrees of freedom
#> Multiple R-squared:  0.004218,	Adjusted R-squared:  0.004127
#> F-statistic: 46.11 on 1 and 10884 DF,  p-value: 1.178e-11

# The model -----------------------------------------------------------
mm <- model.matrix(~ humidity + season, data = da)
k <-  ncol(mm)
n <- nrow(mm)
v <- solve(t(mm) %*% mm)
betas_hat <- v %*% t(mm) %*% da$temp
betas_hat
#>                    [,1]
#> (Intercept) 15.78907941
#> humidity    -0.05788122
#> season2     10.55664484
#> season3     16.71158823
#> season4      4.69037681

y_hat <- mm %*% betas_hat
da$res <- da$temp - 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')
```




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

# Estimated s2
s2 <- (t(da$res) %*% da$res)/(n-k)

sim <- 1000
gamma <- rgamma(sim, shape = (n-k)/2,
                rate = (n-k)*s2/2)

sigma <- 1/gamma
err <- sigma * MASS::mvrnorm(n = 1000, mu = rep(0, 5), v)
beta_sim <- rep(c(betas_hat), each = 1000) + as.vector(err)
beta_sim <- beta_sim %>% as.data.frame() %>%
  mutate(groups = rep(1:5, each = 1000))

vline <- function(group){
  geom_vline(data = filter(beta_sim,
                           groups == group),
             aes(xintercept = betas_hat[group]), linetype = 'dotted')
}

beta_sim %>%
  ggplot(aes(.)) +
  geom_density(colour = 'skyblue', size = 1.1, alpha = 0.75) +
  1:5 %>% purrr::map(vline) +
  facet_wrap(~groups, scales = 'free') +
  theme_bw() +
  labs(title = 'Densities for all the parameters')
```



Credibility intervals