Code ▾

BDA - project

1 Introduction

It is generally known by railway traffic managers (and the public) that bad weather is one major reason for widespread rail traffic disruption. While several different weather situations affects to rail traffic, especially long lasting snow storms are known to be very problematic for the rail traffic.

This project tries to analyse rail traffic disruptions caused by bad weather. Only few previous similar attemps exists so far but the results are encouringing. Ludvigsen & Klæboe have studied affects of winter weather to the rail traffic in Norway, Sweden, Finland, Switcherland and Poland [1]. They have found that cold temperature together with heavy snow precipitation explains from 60 to 80 percent of all delays. Oneton & co. have created a train delay prediction caused by bad weather in Northern Italy [2]. Their method was based on Random Forest Regression (RFR) and features consists of air temperature, humidity, wind direction and spped, precipitation amount, pressure and sun radiation.

Train network is naturally very interconnected system and delay of one train often affects to several others. Here, we anyway consider only effects of weather and leave these dependecies out. This keeps things much more simplier but of course cause some ‘noise’ to the data. First some necessary initialisation tasks

Hide

```
require(lubridate)
library(anytime)
library(ggplot2)
library(brms)
theme_set(theme_minimal())
```

2 Description of the data, and the analysis problem

For the task we need train delay information among with weather data. Train delay information is provided by Finnish Transport Agency (FTA). Online data is available at https://digitrafic.com (https://digitrafic.com). In our case, we use archive data got directly from FTA. Our dataset has been pre-processed so that train information is reported with one hour interval. The dataset contains trains delays reported on every hour on every station. Delays are reported as sum of delays of all trains arrived to the station during the particular hour. The dataset contain additional delays between train stations (‘delay’) and cumulated delays at the end station (‘total_delay’). Here, we use additional delays.

Finnish Meteorological Institute (FMI) provides weather observations. The data is available online (see https://en.ilmatieteenlaitos.fi/open-data (https://en.ilmatieteenlaitos.fi/open-data) for more details). Our data fetched from the online system and stored separately. The data contains aggregation of weather observations withing 100 kilometers and particular hour from all trains stations and times when trains have arrived to the station.

Our archive contain data from 2010 to 2018. Train delay data and weather data have been concatted to the one single dataframe. The whole dataset contains over 27 million rows which is beyond a scope of this project. Thus we use only subset of the data. For training purposes, we use only Helsinki Railway station data For validation purposes, we use 10 days period from 1st Jan to 10th Jan 2011.

More detailed description of the parameters are listed below

Hide

```
time:                  start time of one hour interval
trainstation:          train station short code
train_type:            K -> inter city
                       L -> commuter
                       T -> cargo
                       M -> other
train_count:           amount of trains passed the station during the particular hour
total_delay:           amount of delay at the end station of the train
delay:                 amount of delay between previous and current station
name:                  observation station name
lat:                   train station latitude
lon:                   train station longitude
pressure:              air pressure (hPa)
max_temperature:       maximum temperature during the time interval (C)
min_temperature:       minimum temperature (C)
mean_temperature:      mean temperature (C)
mean_dewpoint:         mean dew point (C)
mean_humidity:         mean humidity (percents)
mean_winddirection:    mean wind direction (degrees)
mean_windspeedms:      mean wind speed (m/s),
max_windgust:          maximum wind gust (m/s)
max_precipitation1h:   1 hour precipitation accumulation (mm)
max_snowdepth:         maximum snow depth (cm)
max_n:                 maximum cloudiness (1/8)
min_vis:               minimum visibility (m)
min_clhb:              minimum cloud base height (m)
max_precipitation3h:   3 hour precipitation accumulation (mm)
max_precipitation6h:   6 hour precipitation accumulation (mm)
flashcount:            flash count with 30 km radius from the train station

99 stands for missing value

Data license is CC4BY.
```

We load data from csv file following:

Hide

```
data_file <- "hki_all.csv"
val_data_file <- "validation.csv"
df <- read.csv(data_file)
df$datetime <- anytime::anytime(df$time)
```

Loaded dataframe is arranged by time. It looks like following. Now there’s 99014 rows of data.

Hide

```
head(df)
```

	X	time	trainstation	train_type	train_count	total_de
	<int>	<fctr>	<fctr>	<fctr>	<int>	<i
1	25058	2015-07-04 18:00:00 UTC	HKI	L	13	
2	37542	2010-01-02 21:00:00 UTC	HKI	L	4	
3	37551	2010-07-27 09:00:00 UTC	HKI	L	17	
4	37557	2010-01-27 21:00:00 UTC	HKI	L	3	
5	37559	2010-07-27 11:00:00 UTC	HKI	L	6	
6	37576	2010-01-09 23:00:00 UTC	HKI	K	2	

6 rows | 1-8 of 28 columns

Hide

```
tail(df)
```

	X	time	trainstation	train_type	train_count
	<int>	<fctr>	<fctr>	<fctr>	<int>
99009	1538934	2012-02-28 20:00:00 UTC	HKI	K	4
99010	1538963	2012-01-20 13:00:00 UTC	HKI	L	22
99011	1538975	2010-03-30 21:00:00 UTC	HKI	L	6
99012	1539005	2014-02-05 01:00:00 UTC	HKI	L	4
99013	1539025	2011-03-07 20:00:00 UTC	HKI	M	1
99014	1539031	2013-02-01 09:00:00 UTC	HKI	K	1

6 rows | 1-7 of 28 columns

Hide

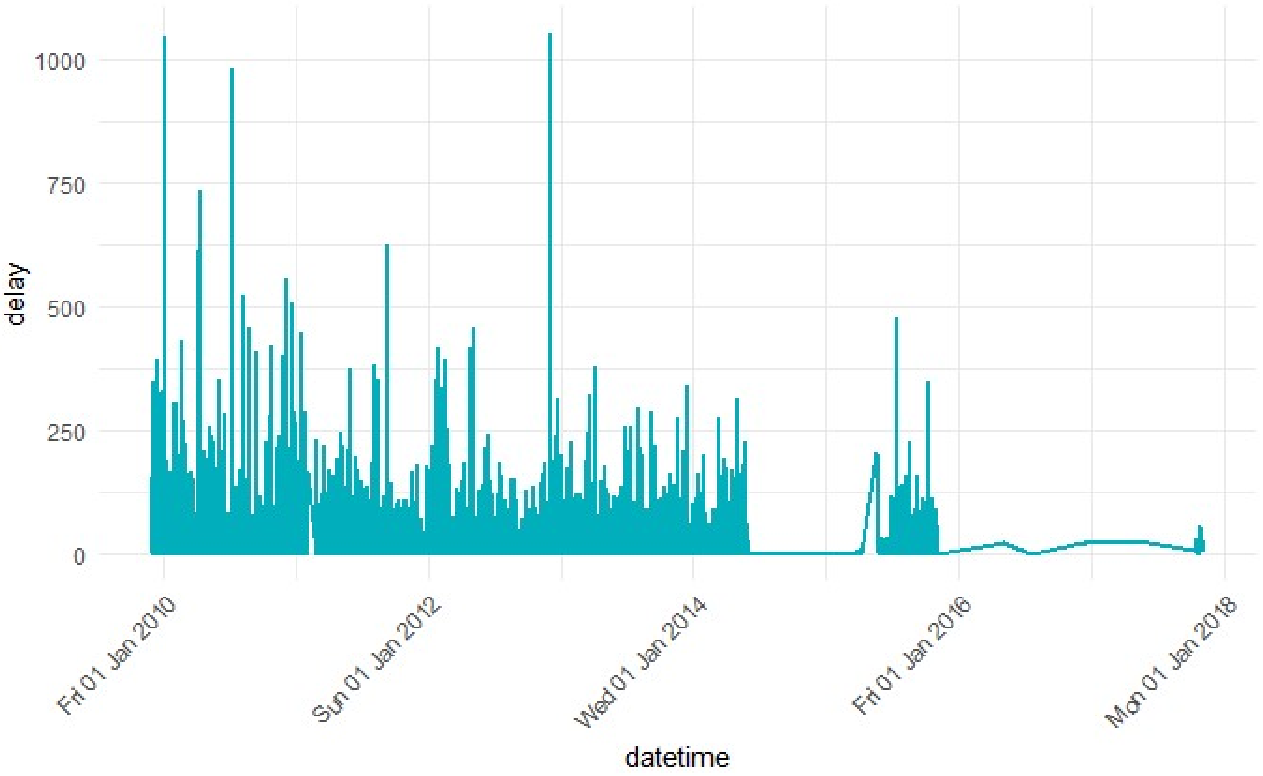
```
dim(df)
```

```
[1] 99014    28
```

Timeseries of the delay is plotted below. One can see, that the timeseries is very spiky which is expected. Normally trains run relatively well on schedule and when bad delays happen, disruptions are typically widespread accross the rail network. One may also wonder high base value of the delays (roughly 100 minutes). It’s good to remember that values are sum of delays of all trains arrived to all stations. The plot also shows that dataset contains lots of missing data.

Hide

```
library(scales)
options(repr.plot.width=10, repr.plot.height=4)
ggplot(data = df, aes(x = datetime, y = delay))+
  geom_line(color = "#00AFBB", size = 1) +
  scale_x_datetime(date_labels = "%a %d %b %Y") +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



As weather observation network is relatively sparse (around 200 weather stations) and most stations measure only subset of all used variables, the data contain lots of missing values (encoded with -99). In this case we want to exclude them in the future processing. After this we get 41 255 rows of data which is used for fitting the model.

Hide

```
df[df == -99] <- NA
df <- df[complete.cases(df),]
```

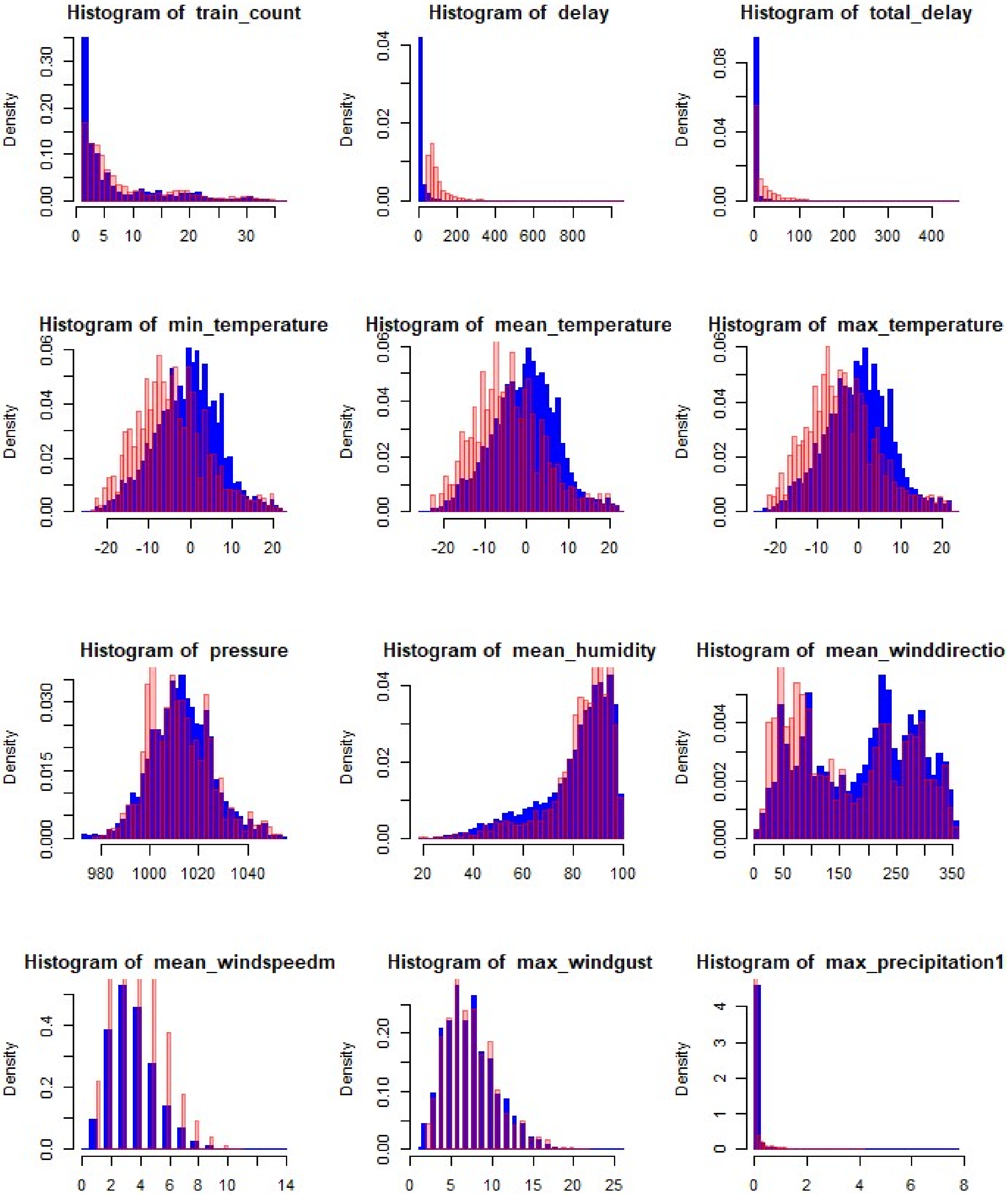
To get sense of data, we can print histograms of measurements. We can also draw histograms of parameters for all rows where delay between stations have been significant, let’s say over 50 minutes. This way we can compare cases where rail have travelled in time and where bad delays have occured.

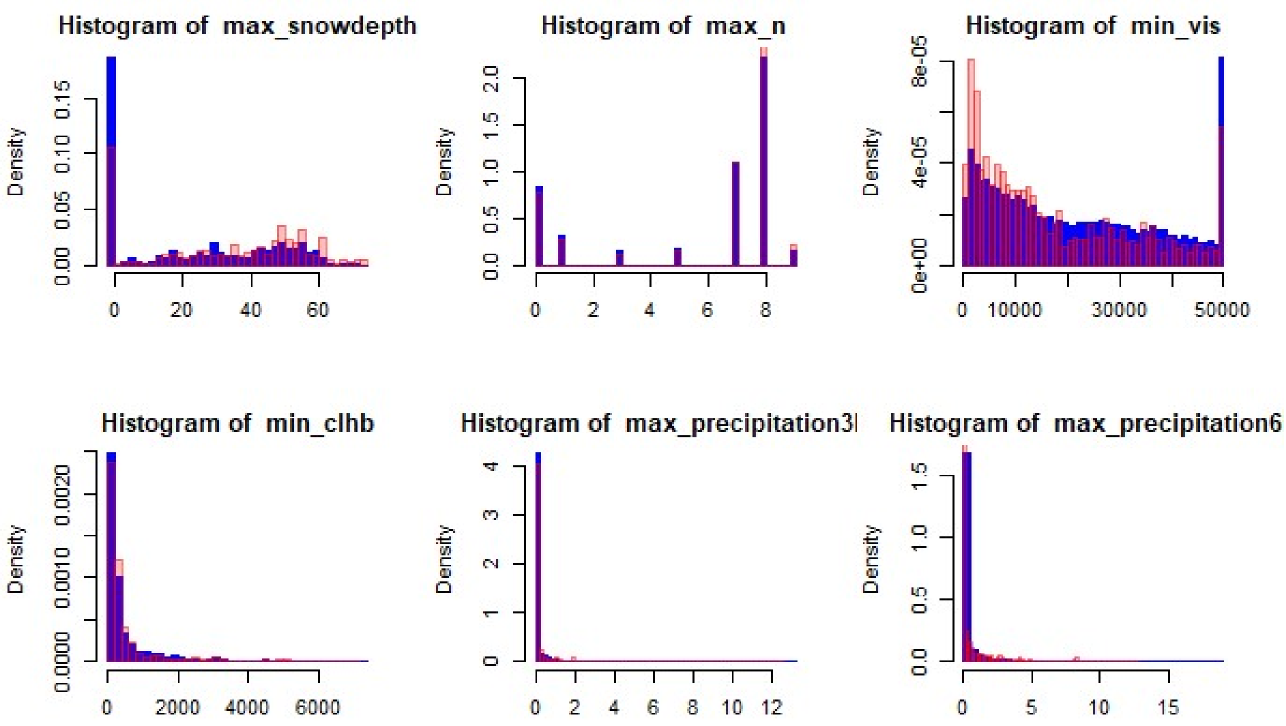
Resulting graph is plotted below. Blue bars stands for all rows and red columns for rows with significant delays. Most histograms looks roughly following normal distribution. Histogram of delays follows the same theme: most of trains run in time but the distribution has a very long tail. Total cloudiness (‘max_n’) can get discrete values from 0 to 8. 9 stands for ‘undetermined’.

From the plot we can see that at least temperature, snow depth, visibility and wind direction has been significantly different when severe rail traffic disruptions has been occurred. So we use these parameters to fit our model. We also add wind gust as it’s assumed to cause train delays (specially with drifting snow).

Hide

```
library(reshape2)
cols = c('train_count', 'delay', 'total_delay',
         'min_temperature', 'mean_temperature',
         'max_temperature', 'pressure', 'mean_humidity',
         'mean_winddirection', 'mean_windspeedms', 'max_windgust',
         'max_precipitation1h', 'max_snowdepth', 'max_n',
         'min_vis', 'min_clhb', 'max_precipitation3h',
         'max_precipitation6h')
df_delay <- df[df$delay > 50,]
par(mfrow=c(2,3))
options(repr.plot.width=10, repr.plot.height=6)
for (col in cols){
    p1 <- hist(df[,col],
          main=paste("Histogram of ",col),
          xlab='', col=rgb(0,0,1), border=rgb(0,0,1),
          breaks=50, freq=FALSE)
    p2 <- hist(df_delay[,col],
          main=paste("Histogram of ",col),
          xlab='', col=rgb(1,0,0,1/4), border=rgb(1,0,0,1/2),
          breaks=50, freq=FALSE, add=T)
}
```


Histogram of train_count
Density
0.00
0.10
0.20
0.30
0
5
10
20
30
Histogram of delay
Density
0.00
0.02
0.04
0
200
400
600
800
Histogram of total_delay
Density
0.00
0.04
0.08
0
100
200
300
400
Histogram of min_temperature
Density
0.00
0.02
0.04
0.06
-20
-10
0
10
20
Histogram of mean_temperature
Density
0.00
0.02
0.04
0.06
-20
-10
0
10
20
Histogram of max_temperature
Density
0.00
0.02
0.04
0.06
-20
-10
0
10
20
Histogram of pressure
Density
0.000
0.015
0.030
980
1000
1020
1040
Histogram of mean_humidity
Density
0.00
0.02
0.04
20
40
60
80
100
Histogram of mean_winddirectio
Density
0.000
0.002
0.004
0
50
150
250
350
Histogram of mean_windspeedm
Density
0.0
0.2
0.4
0
2
4
6
8
10
14
Histogram of max_windgust
Density
0.00
0.10
0.20
0
5
10
15
20
25
Histogram of max_precipitation1
Density
0
1
2
3
4
0
2
4
6
8



3 Description of the model

We decide to use BRMS to fit bayesian linear regression to the data, with target variable “delay” and predictors are: min_temperature, max_snowdepth, max_windgust, mean_winddirection ,min_vis

The selection of the predictors is based on histogram above and also based on their plor agaisnt the target variable below

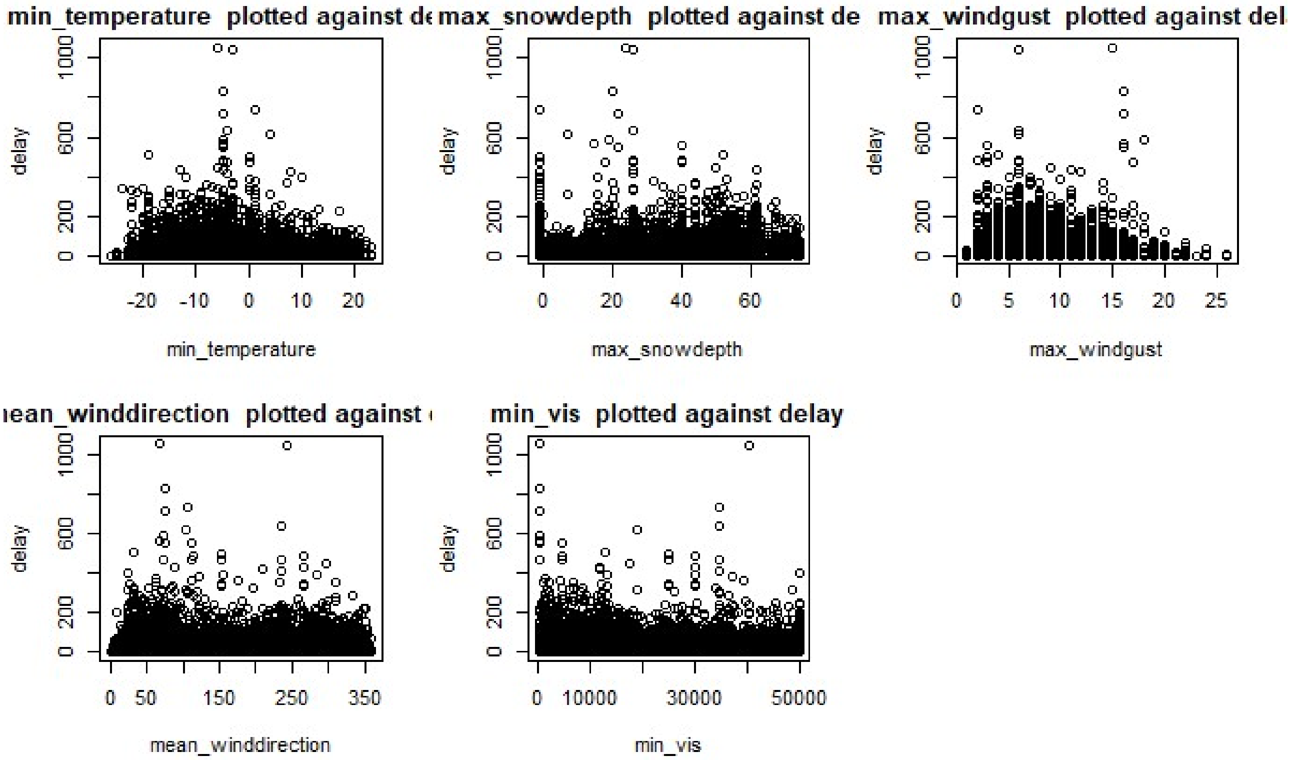
Hide

```
summary(fit4)
```

```
 Family: gaussian 
  Links: mu = identity; sigma = identity 
Formula: delay ~ 1 + min_temperature + max_snowdepth + max_windgust + mean_winddirecti
on + min_vis 
   Data: df (Number of observations: 41255) 
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects: 
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept             13.20      0.53    12.11    14.22       2139 1.00
min_temperature       -0.41      0.03    -0.46    -0.36       1359 1.00
max_snowdepth          0.05      0.01     0.04     0.07       1195 1.00
max_windgust           0.30      0.05     0.21     0.39       2066 1.00
mean_winddirection    -0.02      0.00    -0.02    -0.01       1530 1.01
min_vis               -0.00      0.00    -0.00    -0.00       3666 1.00

Family Specific Parameters: 
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma    29.26      0.10    29.06    29.46       4633 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample 
is a crude measure of effective sample size, and Rhat is the potential 
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Hide

```
cols = c(
         'min_temperature','max_snowdepth',
          'max_windgust',
         'mean_winddirection',
         'min_vis')
par(mfrow=c(2,3))
options(repr.plot.width=10, repr.plot.height=6)
for (col in cols){
    p1 <- plot(df[,col],df[,"delay"],
          main=paste(col, " plotted against delay"),
          xlab=paste(col), ylab = "delay")
}
```



Here is the code fitting the model (1 represent the intercept)

Hide

```
options(mc.cores = parallel::detectCores())
fit1 <- brm(
    delay ~ 1 + min_temperature + max_snowdepth +  max_windgust + mean_winddirection
+ min_vis,
    data = df
)
```

Hide

```
summary(fit1)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: delay ~ 1 + min_temperature + max_snowdepth + max_windgust + mean_winddirecti
on + min_vis
   Data: df (Number of observations: 41255)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept             13.19      0.53    12.18    14.24       2452 1.00
min_temperature       -0.41      0.02    -0.45    -0.36       1469 1.00
max_snowdepth          0.05      0.01     0.04     0.07       1595 1.00
max_windgust           0.29      0.05     0.20     0.38       2645 1.00
mean_winddirection    -0.02      0.00    -0.02    -0.01       1585 1.00
min_vis               -0.00      0.00    -0.00    -0.00       3970 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma    29.26      0.10    29.06    29.45       4741 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

It can be seen that most of the coefficient is near zero , indicating that linear regression might not be a good choice # Description of the prior choices

Here inthe model, we don’t specify any prior choice. The default prior is an non-informative flat prior over the reals. In this case we decide to let the data speak for itself

Hide

```
prior_summary(fit1)
```

prior	class	coef	gr…	r…	d…	nlpar	bo…
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
	b						
	b	max_snowdepth					
	b	max_windgust					
	b	mean_winddirection					
	b	min_temperature					
	b	min_vis					

prior	class	coef	gr…	r…	d…	nlpar	bo…
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
student_t(3, 3, 10)	Intercept						
student_t(3, 0, 10)	sigma						

8 rows

4 Stan

Stan code is generated by BRMS package. It can be printed following

Hide

```
fit1$model
```

```
// generated with brms 2.6.0
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;  // centered version of X
  vector[K - 1] means_X;  // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b;  // population-level effects
  real temp_Intercept;  // temporary intercept
  real<lower=0> sigma;  // residual SD
}
transformed parameters {
}
model {
  vector[N] mu = temp_Intercept + Xc * b;
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, 3, 10);
  target += student_t_lpdf(sigma | 3, 0, 10)
    - 1 * student_t_lccdf(0 | 3, 0, 10);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
}
```

5 How the Stan model run

We use default setting from brms with warmup = 1000, iter = 2000, chains = 4

Hide

```
starting worker pid=15200 on localhost:11373 at 10:36:12.977
starting worker pid=4136 on localhost:11373 at 10:36:13.517
starting worker pid=19624 on localhost:11373 at 10:36:13.871
starting worker pid=3020 on localhost:11373 at 10:36:14.306

SAMPLING FOR MODEL 'b88290d45df8be1ad9d5212af9bc766f' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.008 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 80 second
s.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)

SAMPLING FOR MODEL 'b88290d45df8be1ad9d5212af9bc766f' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.009 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 90 second
s.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)

SAMPLING FOR MODEL 'b88290d45df8be1ad9d5212af9bc766f' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.016 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 160 second
s.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)

SAMPLING FOR MODEL 'b88290d45df8be1ad9d5212af9bc766f' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.012 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 120 second
s.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 3547.36 seconds (Warm-up)
Chain 3:                1694.69 seconds (Sampling)
Chain 3:                5242.04 seconds (Total)
Chain 3: 
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 2943.15 seconds (Warm-up)
Chain 4:                2459.25 seconds (Sampling)
Chain 4:                5402.4 seconds (Total)
Chain 4: 
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2813.55 seconds (Warm-up)
Chain 2:                4081.32 seconds (Sampling)
Chain 2:                6894.87 seconds (Total)
Chain 2:
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2560.62 seconds (Warm-up)
Chain 1:                4546.7 seconds (Sampling)
Chain 1:                7107.31 seconds (Total)
Chain 1:
```

6 Convergence diagnostics

To analyse model convergence, we plot variables and check RHat values and effective sample size.

For weak prior, RHat stays under 1.05 so the model has converged well.

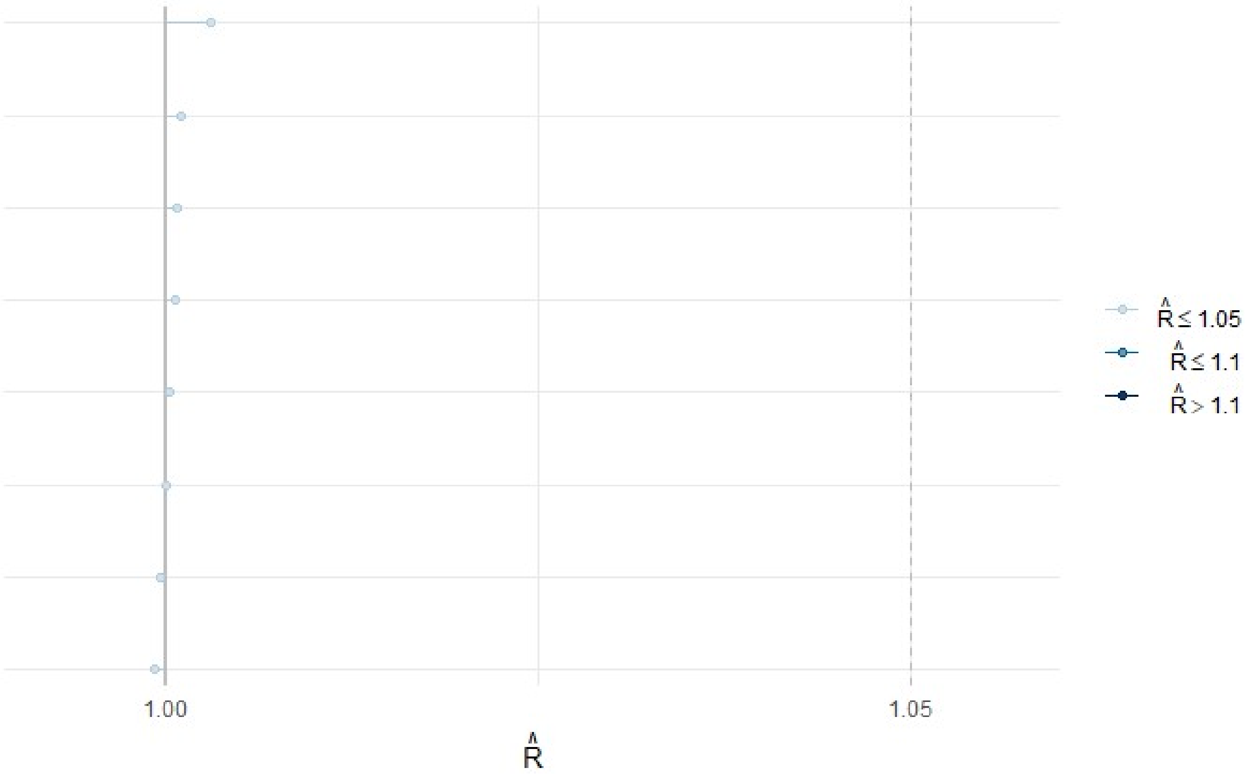
Hide

```
rhat(fit1)
```

```
        b_Intercept    b_min_temperature      b_max_snowdepth        b_max_windgust
          0.9996100            1.0007846            1.0006523             1.0009907
b_mean_winddirection            b_min_vis                sigma                  lp__
          0.9999388            0.9992266            1.0002893             1.0030001
```

Hide

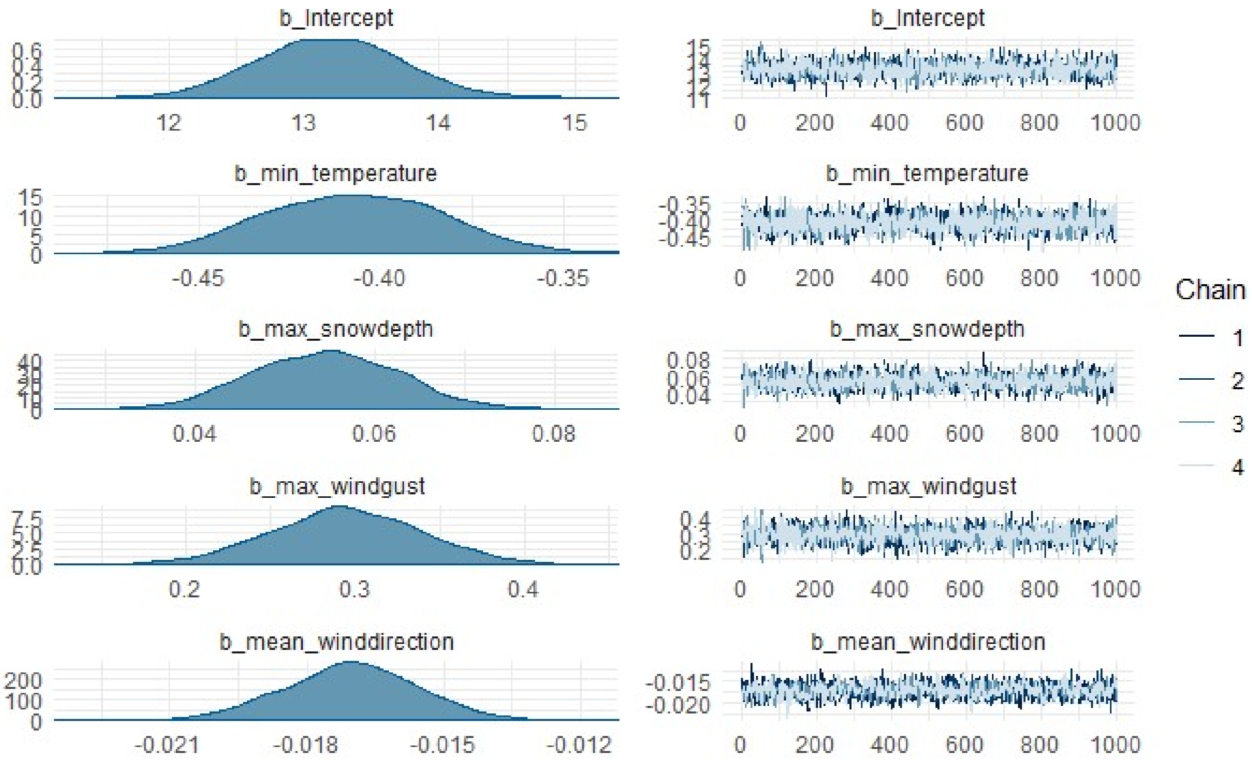
```
stanplot(fit1, type='rhat')
```

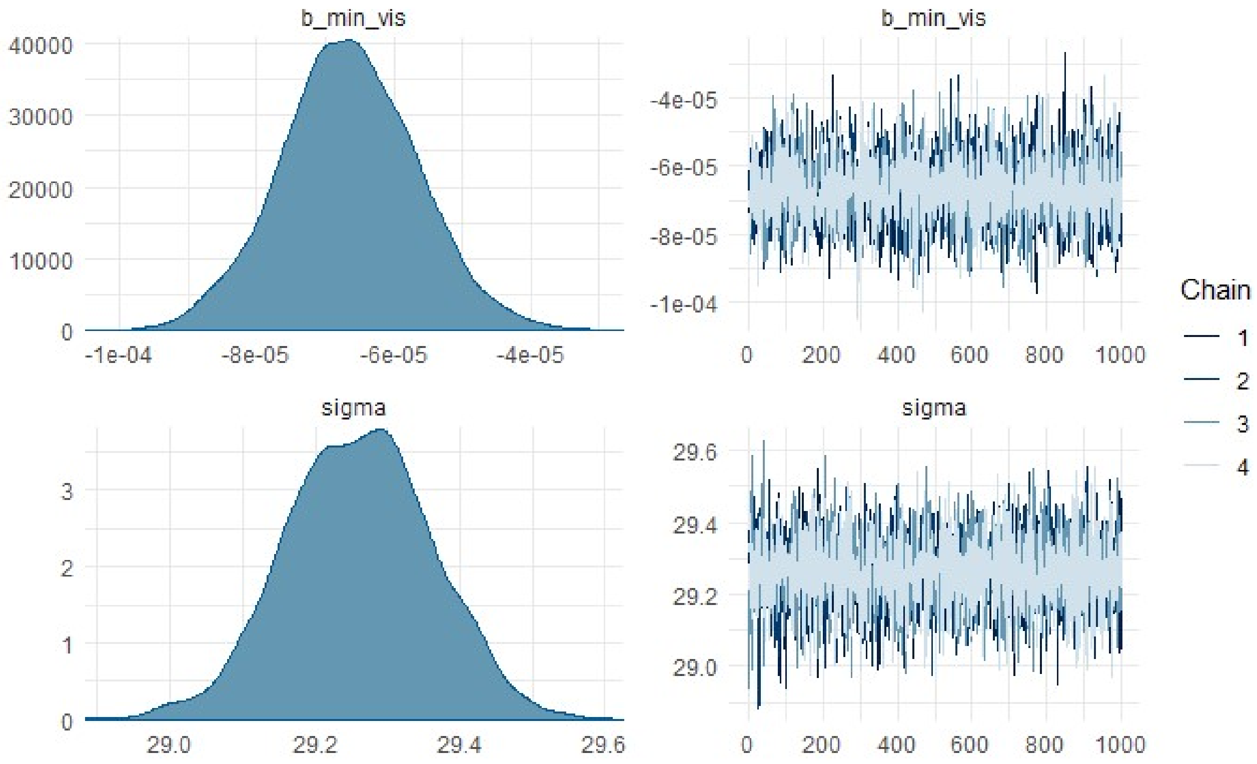


To check a convergence of different chains and posterior distribution, we can plot fitted models. We can see, that all chains have converged to similar values for all parameters. We can also see, that all parameters have quite similar posterior distribution (with different means of course).

Hide

```
plot(fit1)
```

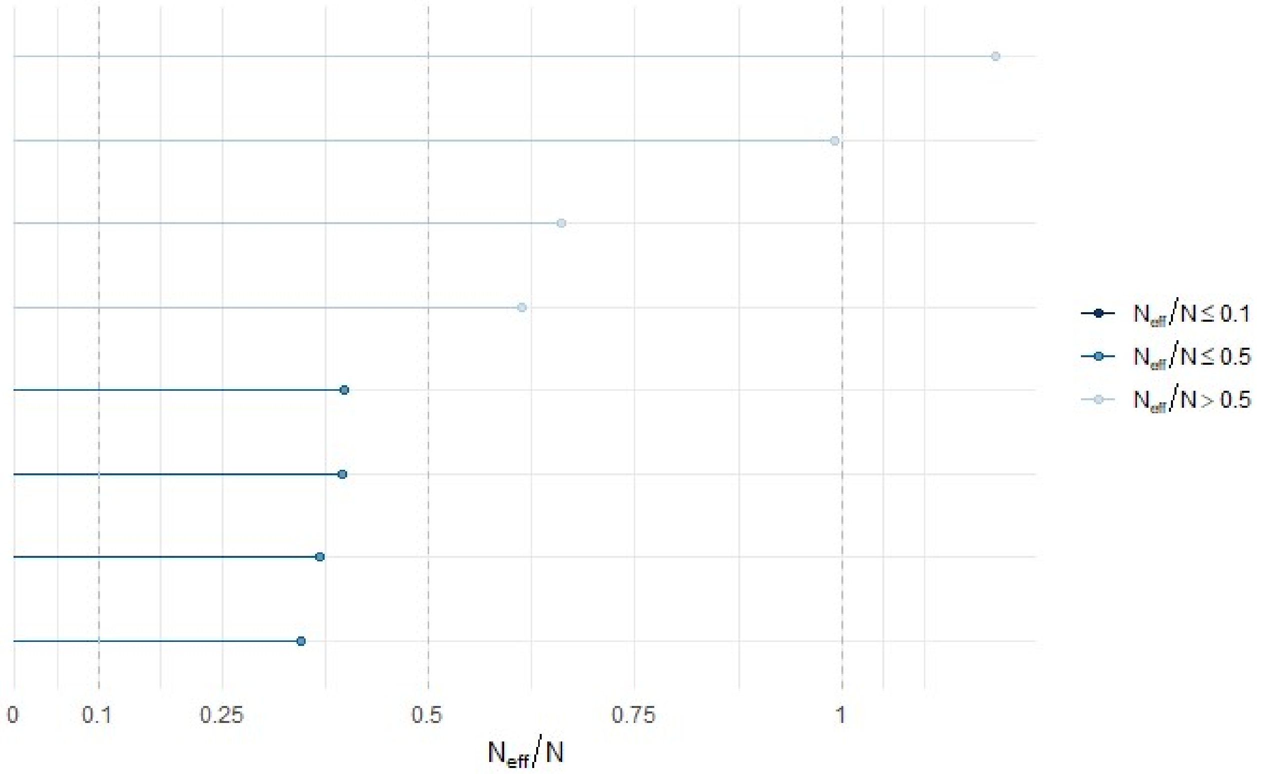

b_Intercept
0.6
0.4
0.2
0.0
12
13
14
15
b_Intercept
15
14
13
12
11
0
200
400
600
800
1000
b_min_temperature
15
10
5
0
-0.45
-0.40
-0.35
b_min_temperature
-0.35
-0.40
-0.45
0
200
400
600
800
1000
b_max_snowdepth
40
30
20
10
0
0.04
0.06
0.08
b_max_snowdepth
0.08
0.06
0.04
0
200
400
600
800
1000
b_max_windgust
7.5
5.0
2.5
0.0
0.2
0.3
0.4
b_max_windgust
0.4
0.3
0.2
0
200
400
600
800
1000
b_mean_winddirection
200
100
0
-0.021
-0.018
-0.015
-0.012
b_mean_winddirection
-0.015
-0.020
0
200
400
600
800
1000
Chain
1
2
3
4



To analyse effective sample size, we can plot ratio of effective samples and all samples. This means amount of draws being able to estimate true mean value of the parameter. We can see that none Neff/N values are below 0.1 and pretty large amount of them are over 0.5, which again verify that both models have converged well.

Hide

```
stanplot(fit1, type='neff')
```



As conclusion, the models have converged well.

7 Posterior predictive checking

In posterior predictive checking one compares modelled and observed data. In other words, we take draws from the model and compare them with observed data. This way we can find systematic differenced which could indicate that modelling has failed somehow.

Hide

```
library("bayesplot")
yrep <- posterior_predict(fit1, draws = 500)
```

Hide

```
dim(yrep)
dim(df[complete.cases(df),])
```

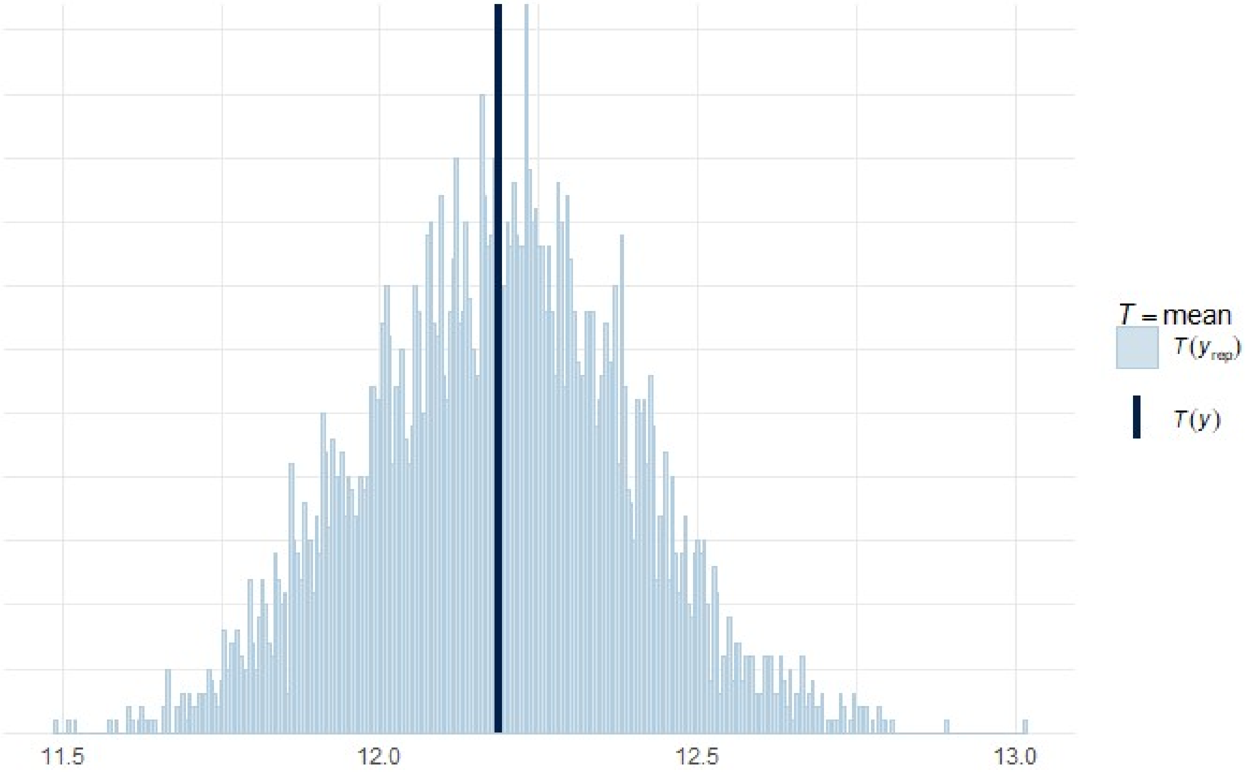
First we look plain mean value of the simulations (light blue) and the observed data (dark blue). Average of observed data is in the middle of the simulated distribution, so everything looks fine.

Hide

```
y <- df$delay
```

Hide

```
ppc_stat(y, yrep, binwidth=.005)
```



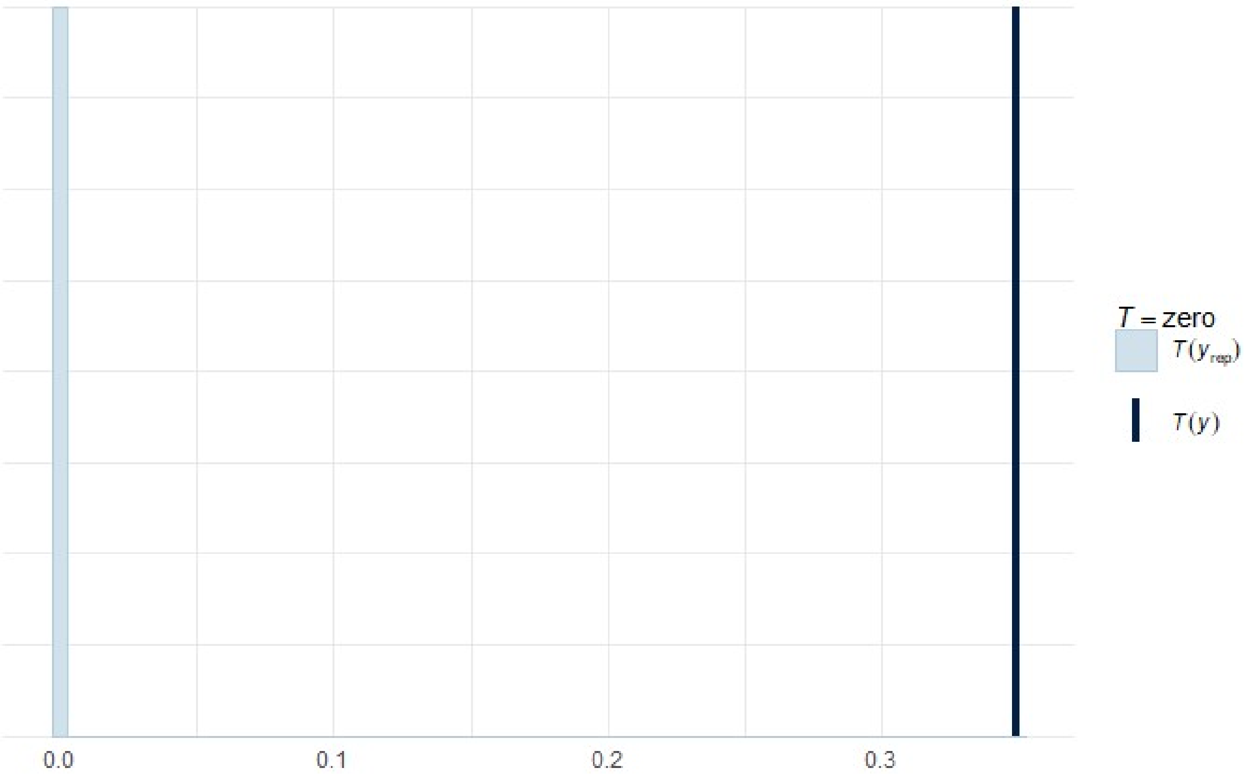
If we look probability that delay is 0 minutes in the simulations (light blue) and the observed data (dark blue), we can see a slight difference.

Hide

```
zero <- function(x) mean(x == 0)
```

Hide

```
ppc_stat(y, yrep, 'zero', binwidth=.005)
```



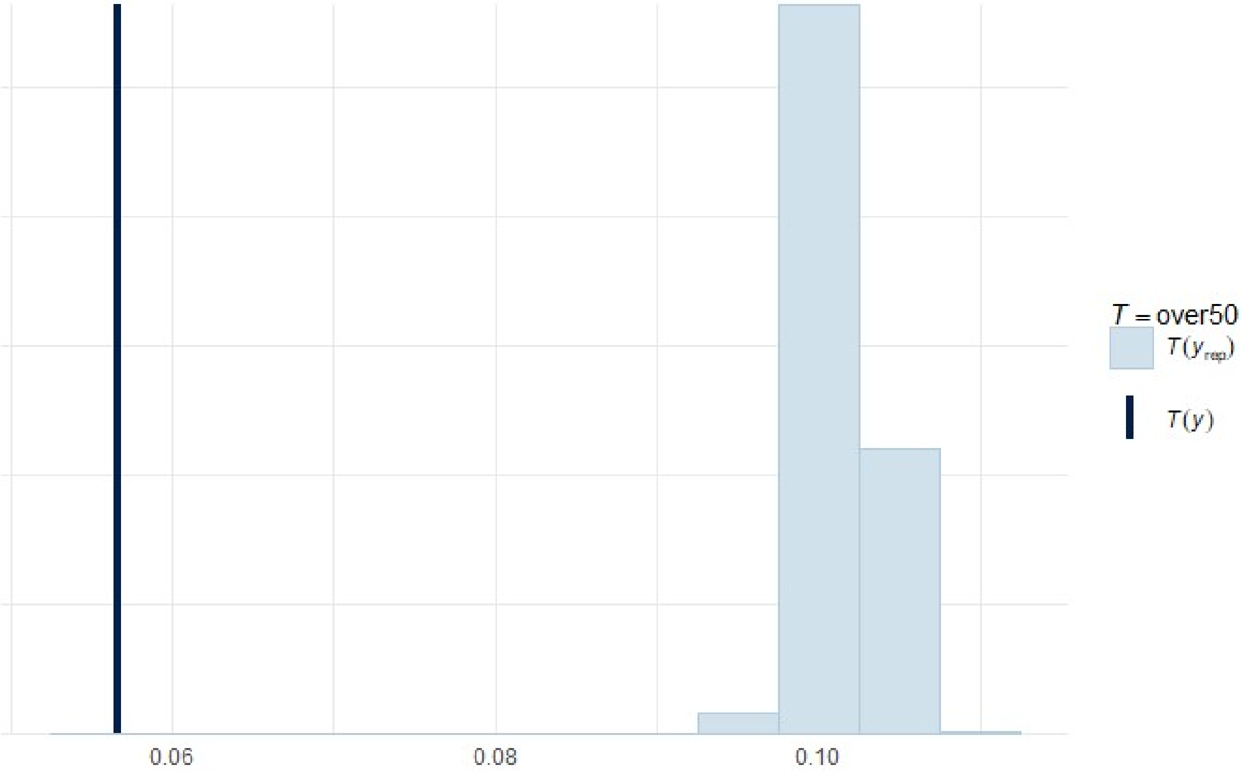
Similarly, if we look sever disrubtions, probability that delay is over 50 minutes, in the simulations (light blue) and the observed data (dark blue), we can see a slight difference. The difference is anyway smaller than in the probability of 0 minute delay.

Hide

```
over50 <- function(x) mean(x > 50)
```

Hide

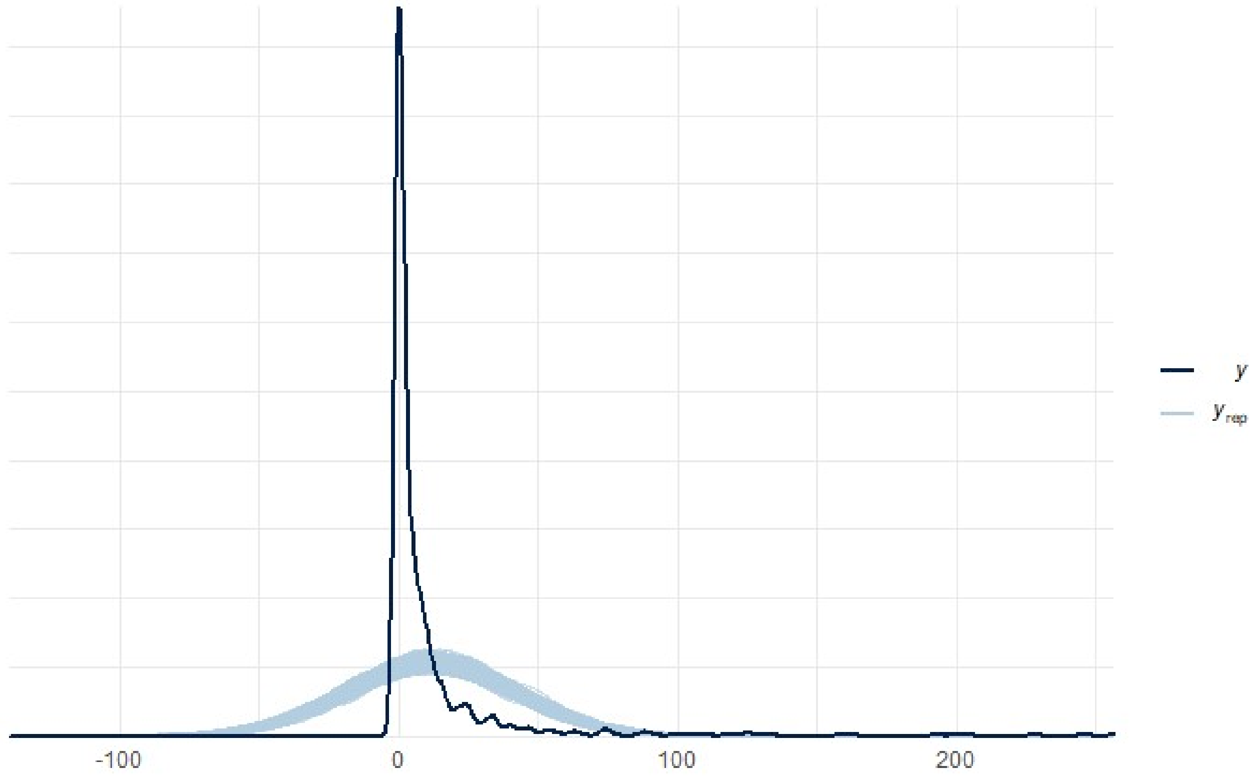
```
ppc_stat(y, yrep, 'over50', binwidth=.005)
```



We can analyse a distribution of simulated draws and observed data by plotting them into the same plot. We use first 1000 rows from the data to avoid running out of memory. Resulting image tells us that our models are not able to follow true shape of observations very well. They produce negative values and miss large amount of 0 values.

Hide

```
ppc_dens_overlay(y[0:1000], yrep[,0:1000])
```



8 Model comparison

We try model with fewer variables (predictors). Our second model includes only 3 predictors min_temperature , max_snowdepth, mean_winddirection

Hide

```
options(mc.cores = parallel::detectCores())
fit2 <- brm(
    delay ~ 1 + min_temperature + max_snowdepth + mean_winddirection,
    data = df
    )
```

Hide

```
summary(fit2)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: delay ~ 1 + min_temperature + max_snowdepth + mean_winddirection
   Data: df (Number of observations: 41255)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept             14.14      0.37    13.39    14.86       4151 1.00
min_temperature       -0.40      0.02    -0.45    -0.35       2122 1.00
max_snowdepth          0.06      0.01     0.05     0.08       2164 1.00
mean_winddirection    -0.02      0.00    -0.02    -0.02       4435 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma    29.29      0.10    29.08    29.48       5093 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

From the result, it can be seen that there is no big different from the posterior of model 1 and 2

Here we compare the 2 models using LOO

Hide

```
loo(fit1,fit2)
```

9 Predictive performance assessment

To do predictive performance assesment, we need to read test data which our model hasn’t seen yet. We use similar 10 days set from beginning of 2011.

Hide

```
validation <- read.csv(val_data_file)
validation$datetime <- anytime::anytime(validation$time)
validation <- validation[order(validation$datetime),]
```

Then we create a prediction of train delays based on the same parameters we have used to fit our model.

9.1 Model 1

Hide

```
val_data <- validation[,c('min_temperature', 'min_vis',  'max_windgust', 'mean_winddir
ection','max_snowdepth')]
pred <- predict(fit1, newdata = val_data, re_formula = NA)
head(pred, 3)
```

```
     Estimate Est.Error     Q2.5    Q97.5
[1,] 20.67313  29.85749 -37.05679 81.70831
[2,] 20.78134  30.62818 -38.56890 80.53615
[3,] 16.83574  29.55021 -40.68117 75.93480
```

We can plot values and corresponding estimated errors. We can see that model forecasts negative values which of course doesn’t make sense. Median values see plausible but estimated errors are large. From the plot we can also see, that predicted values are very discrete which is not good.

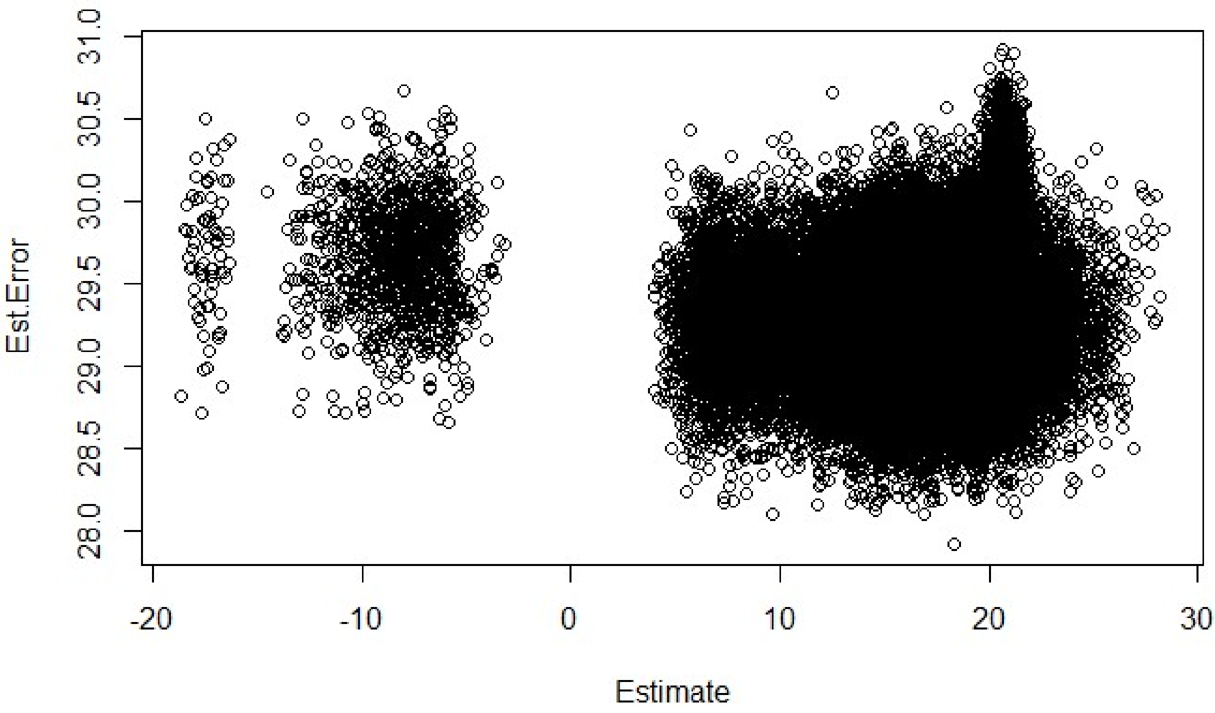
Hide

```
options(repr.plot.width=4, repr.plot.height=4)
summary(pred)
```

```
    Estimate        Est.Error          Q2.5            Q97.5
 Min.   :-18.64   Min.   :27.92   Min.   :-78.35   Min.   :37.07
 1st Qu.: 14.74   1st Qu.:29.08   1st Qu.:-42.81   1st Qu.:71.78
 Median : 17.08   Median :29.31   Median :-40.12   Median :74.47
 Mean   : 16.33   Mean   :29.33   Mean   :-41.09   Mean   :73.76
 3rd Qu.: 19.96   3rd Qu.:29.56   3rd Qu.:-37.75   3rd Qu.:77.32
 Max.   : 28.36   Max.   :30.92   Max.   :-27.35   Max.   :89.84
```

Hide

```
plot(pred)
```



To really visualise predictions, we plot predicted delays and true delays on the same plot. We select Helsinki Railwaystation for this illustration. From the plot we can see, that although the model has converged well, it has no real prediction power.

Hide

```
val_data_hki <- validation[validation$trainstation == 'HKI' ,c('min_temperature', 'min
_vis', 'mean_humidity', 'max_windgust', 'mean_winddirection', 'max_snowdepth')]
pred_hki <- predict(fit1, newdata = val_data_hki, re_formula = NA)
head(pred_hki, 3)
```

```
     Estimate Est.Error      Q2.5    Q97.5
[1,] 15.44574  29.39317 -41.91040 72.95315
[2,] 14.99603  28.63707 -40.97358 68.85918
[3,] 15.84407  29.10518 -41.30716 72.97717
```

Hide

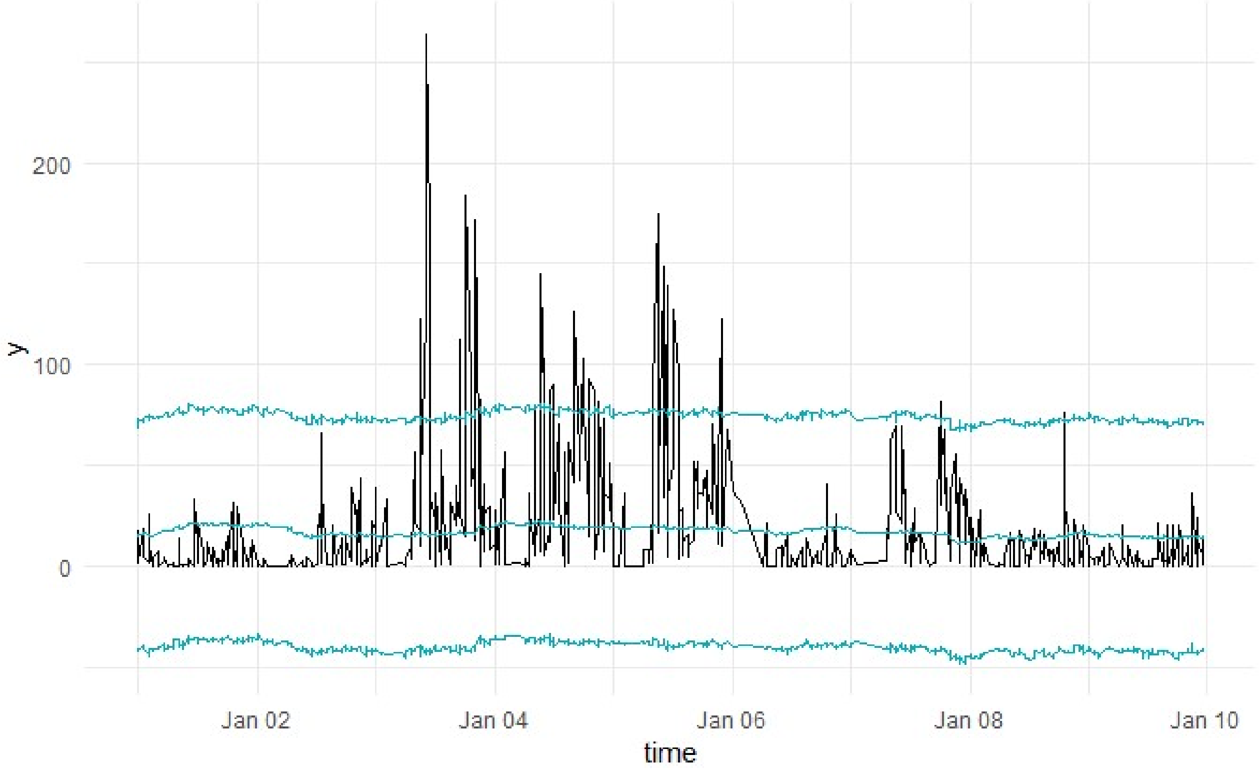
```
options(repr.plot.width=14, repr.plot.height=4)
val_times_hki <- validation[validation$trainstation == 'HKI', c('datetime','delay')]
fitval_hki <- data.frame(cbind(val_times_hki,pred_hki[,-2]))
names(fitval_hki) <- c("time", "delay", "estimate", "lower", "upper")
head(fitval_hki, 3)
```

	time <S3: POSIXct>	delay <int>	estimate <dbl>	lower <dbl>	upper <dbl>
35085	2011-01-01	18	15.44574	-41.91040	72.95315
52423	2011-01-01	17	14.99603	-40.97358	68.85918
52447	2011-01-01	2	15.84407	-41.30716	72.97717

3 rows

Hide

```
ggplot(data=fitval_hki, aes(x = time, y = 'delay')) +
    geom_line(data = fitval_hki, aes(y = delay), size = 0.5) +
    geom_line(data = fitval_hki, aes(y = lower), size = 0.11, col='#00AFBB') +
    geom_line(data = fitval_hki, aes(y = upper), size = 0.11, col='#00AFBB') +
    geom_line(data = fitval_hki, aes(y = estimate), size = 0.5, col='#00AFBB')
```



We can calculate model performance in terms of loss functions. In particular, we use RMSE and MAE to describe the performance.

Hide

```
rmse <- function(error)
{
    sqrt(mean(error^2))
}
mae <- function(error)
{
    mean(abs(error))
}
error <- validation[,'delay'] - fitval[,'estimate']
```

```
Error: object 'fitval' not found
```

9.2 Model 2

Hide

```
pred2 <- predict(fit2, newdata = val_data, re_formula = NA)
head(pred2, 3)
```

```
     Estimate Est.Error       Q2.5    Q97.5
[1,] 50.29093  29.52097  -7.076713 108.4649
[2,] 49.16018  29.11505  -8.693127 105.7865
[3,] 16.90615  29.03093 -39.843677  73.8284
```

We can plot values and corresponding estimated errors. We can see that model forecasts negative values which of course doesn’t make sense. Median values see plausible but estimated errors are large. From the plot we can also see, that predicted values are very discrete which is not good.

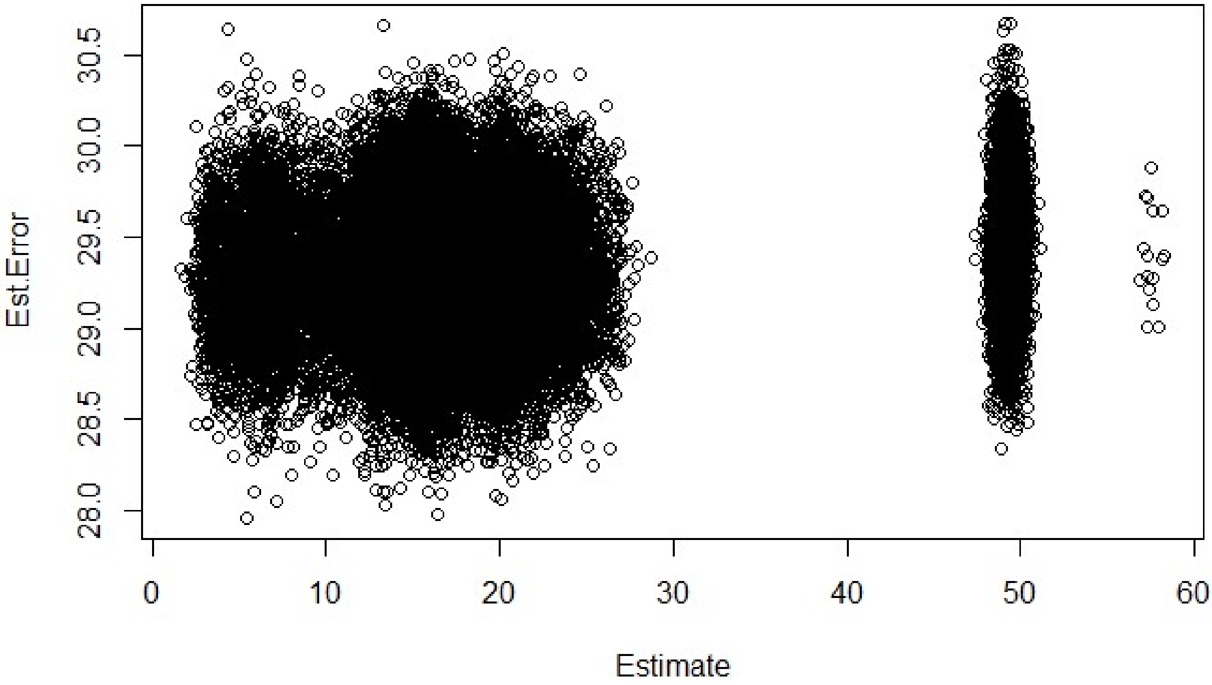
Hide

```
options(repr.plot.width=4, repr.plot.height=4)
summary(pred2)
```

```
    Estimate        Est.Error          Q2.5              Q97.5       
 Min.   : 1.678   Min.   :27.96   Min.   :-58.339   Min.   : 56.79  
 1st Qu.:14.539   1st Qu.:29.08   1st Qu.:-43.032   1st Qu.: 71.65  
 Median :16.624   Median :29.31   Median :-40.486   Median : 74.19  
 Mean   :19.841   Mean   :29.31   Mean   :-37.536   Mean   : 77.22  
 3rd Qu.:20.321   3rd Qu.:29.53   3rd Qu.:-36.824   3rd Qu.: 77.83  
 Max.   :58.271   Max.   :30.67   Max.   :  1.697   Max.   :117.30  
```

Hide

```
plot(pred2)
```



To really visualise predictions, we plot predicted delays and true delays on the same plot. We select Helsinki Railwaystation for this illustration. From the plot we can see, that although the model has converged well, it has no real prediction power.

Hide

```
pred2_hki <- predict(fit2, newdata = val_data_hki, re_formula = NA)
head(pred2_hki, 3)
```

```
     Estimate Est.Error      Q2.5    Q97.5
[1,] 14.30266  29.44232 -42.31943 72.81058
[2,] 14.10880  29.42835 -44.51562 72.33305
[3,] 14.40187  29.36632 -43.98446 71.83742
```

Hide

```
options(repr.plot.width=14, repr.plot.height=4)
fitval2_hki <- data.frame(cbind(val_times_hki,pred2_hki[,-2]))
names(fitval2_hki) <- c("time", "delay", "estimate", "lower", "upper")
head(fitval2_hki, 3)
```

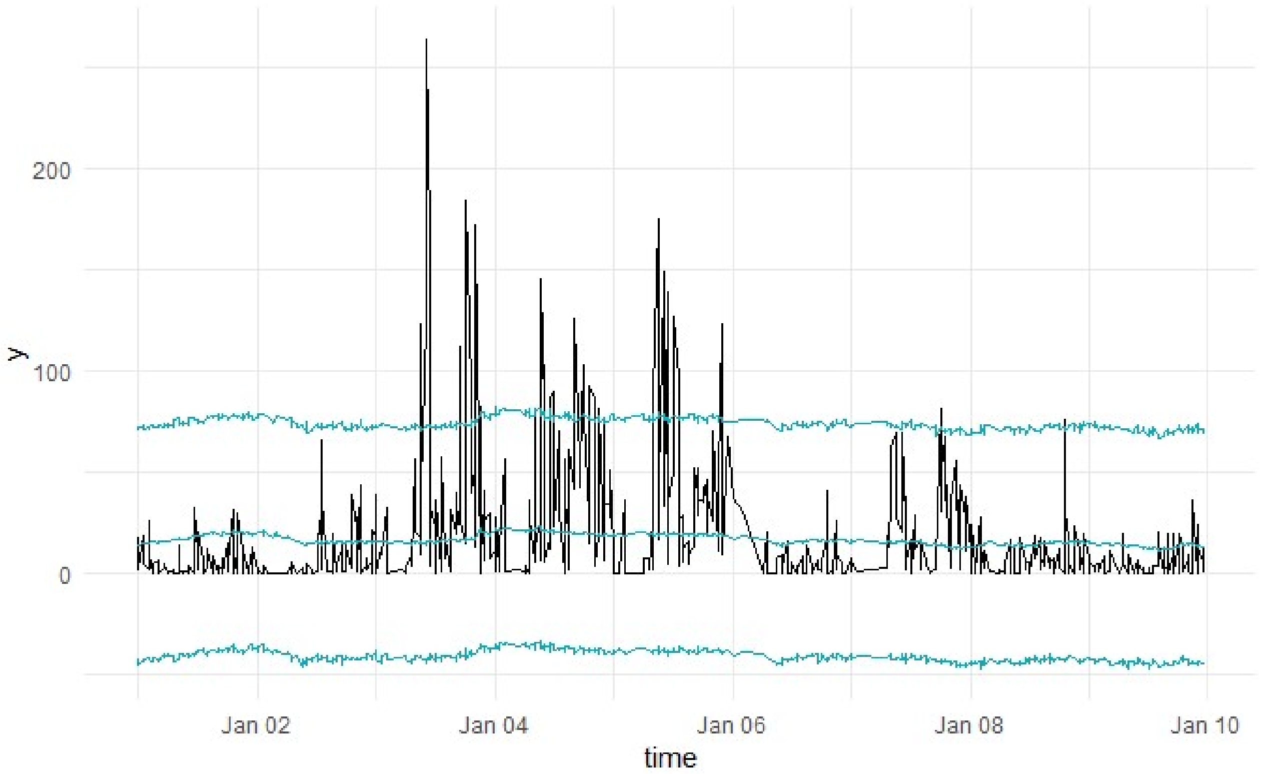
	time	delay	estimate	lower	upper
	<S3: POSIXct>	<int>	<dbl>	<dbl>	<dbl>
35085	2011-01-01	18	14.30266	-42.31943	72.81058

	time <S3: POSIXct>	delay <int>	estimate <dbl>	lower <dbl>	upper <dbl>
52423	2011-01-01	17	14.10880	-44.51562	72.33305
52447	2011-01-01	2	14.40187	-43.98446	71.83742

3 rows

Hide

```
ggplot(data=fitval2_hki, aes(x = time, y = 'delay')) +
    geom_line(data = fitval2_hki, aes(y = delay), size = 0.5) +
    geom_line(data = fitval2_hki, aes(y = lower), size = 0.11, col='#00AFBB') +
    geom_line(data = fitval2_hki, aes(y = upper), size = 0.11, col='#00AFBB') +
    geom_line(data = fitval2_hki, aes(y = estimate), size = 0.5, col='#00AFBB')
```



We can calculate model performance in terms of loss functions. In particular, we use RMSE and MAE to describe the performance.

Hide

```
rmse <- function(error)
{
    sqrt(mean(error^2))
}
mae <- function(error)
{
    mean(abs(error))
}
error <- validation[,'delay'] - fitval[,'estimate']
```

```
Error: object 'fitval' not found
```

10 Sensitivity analysis

In this section, we test with different informative prior to see how different prior can influence the final result

The first set of prior is selected based on the estimate we get from above analysis. Therefore this is informative prior. Here we use normal distribution for all of the prior choice

Hide

```
prior1 <- c(set_prior("normal(-0.4,0.1)", class = "b", coef="min_temperature"),
           set_prior("normal(0.1,0.1)", class = "b", coef="max_snowdepth"),
           set_prior("normal(0.3,0.1)", class = "b", coef="max_windgust"),
           set_prior("normal(-0.1,0.1)", class = "b", coef="mean_winddirection"),
           set_prior("normal(-0.1,0.1)", class = "b", coef="min_vis"),
           set_prior("normal(10,100)", class = "Intercept"))
```

Hide

```
options(mc.cores = parallel::detectCores())
fit3 <- brm(
    delay ~  1 + min_temperature + max_snowdepth + max_windgust + mean_winddirection
+ min_vis,
    data = df,
    prior = prior1
)
```

Hide

```
summary(fit3)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: delay ~ 1 + min_temperature + max_snowdepth + max_windgust + mean_winddirecti
on + min_vis
   Data: df (Number of observations: 41255)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept             13.21      0.52    12.21    14.25       2098 1.00
min_temperature       -0.41      0.02    -0.45    -0.36       1581 1.00
max_snowdepth          0.06      0.01     0.04     0.07       1267 1.00
max_windgust           0.29      0.04     0.21     0.37       1797 1.00
mean_winddirection    -0.02      0.00    -0.02    -0.01       1436 1.00
min_vis               -0.00      0.00    -0.00    -0.00       3815 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma    29.26      0.10    29.06    29.46       4200 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

We can see that there is no big difference in the posterior of model 3 and 1

The second model with informative prior - which we choose based on our understanding of the data. For example, we believe that min_vis (mininmum vision) has more influence than min_temprature in causing train delay

Hide

```
prior2 <- c(set_prior("normal(1,2)", class = "b", coef="min_temperature"),
           set_prior("normal(5,10)", class = "b", coef="max_snowdepth"),
           set_prior("normal(2,3)", class = "b", coef="max_windgust"),
           set_prior("normal(10,15)", class = "b", coef="mean_winddirection"),
           set_prior("normal(20,25)", class = "b", coef="min_vis"),
           set_prior("normal(10,100)", class = "Intercept"))
```

Hide

```
options(mc.cores = parallel::detectCores())
fit4 <- brm(
    delay ~  1 + min_temperature + max_snowdepth + max_windgust + mean_winddirection
+ min_vis,
    data = df,
    prior = prior2
)
```

Hide

```
summary(fit4)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: delay ~ 1 + min_temperature + max_snowdepth + max_windgust + mean_winddirecti
on + min_vis
   Data: df (Number of observations: 41255)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept             13.20      0.53    12.11    14.22       2139 1.00
min_temperature       -0.41      0.03    -0.46    -0.36       1359 1.00
max_snowdepth          0.05      0.01     0.04     0.07       1195 1.00
max_windgust           0.30      0.05     0.21     0.39       2066 1.00
mean_winddirection    -0.02      0.00    -0.02    -0.01       1530 1.01
min_vis               -0.00      0.00    -0.00    -0.00       3666 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma    29.26      0.10    29.06    29.46       4633 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Here we can see that even with highly informative prior, there is no big difference between the posterior of model 4 and 1, although the effective sample is somewhat smaller

Highly informative prior is far from the observed data –> highly influence the posterior results

11 Discussion of problems, and potential improvements

The model ignore the time-series structure of the data, treating them as thousands of independent observations The model is not good for predictive performance. In other previous research with same data, it show that tree model such that random forest performs better than linear regression.

Potential improvements maybe how to include Bayesian inference into complex model such that random forest

12 References

[1] Ludvigsen, J., & Klæboe, R. (2014). Extreme weather impacts on freight railways in Europe. Natural Hazards. https://doi.org/10.1007/s11069-013-0851-3 (https://doi.org/10.1007/s11069-013-0851-3)

[2] Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., … Anguita, D. (2016). Advanced analytics for train delay prediction systems by including exogenous weather data. In Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016 (pp. 458–467). https://doi.org/10.1109/DSAA.2016.57 (https://doi.org/10.1109/DSAA.2016.57)