Advanced Bayesian Assignment 1

16343261

02/07/2023

Before we begin this assignment, we will first load our necessary packages and our data.

```
setwd("C:/Users/matth/Documents/Advanced Bayesian")
library(readr)
library(bayesrules)
library(rstanarm)
library(bayesplot)
library(tidybayes)
library(broom.mixed)
library(BayesFactor)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(grid)
library(AER)

bikes <- read_csv("bikes.csv")

## Rows: 1416 Columns: 11
## -- Column specification ----------------------------------------------------
------
## Delimiter: ","
## chr  (1): Day
## dbl  (9): Hour, Clontarf, Grove_Road, Richmond_Street, rain, temp, wdsp,
vis...
## date (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Question 1. Is the average temperature in January and February greater than 5 degrees?

We will use simple Bayesian hypothesis testing to answer this question.

First we define H : θ > 5 and we assume that x1, . . . , xn | θ , σ2 iid ∼ N (θ,9) ; and priors are truncated normal distributions:

θ | H0 ∼ Nθ > θ0 (μ0 = 6, σ2/k0 = 9/0.5 = 18)

θ | H1 ∼ Nθ ≤ θ0 (μ0 = 4, σ2/k1 = 9/0.5 = 18)

Here we are using a truncated Normal distribution as we are focusing our inference on a specific range between 4 and 6.

Next we will fit our H0 and H1 functions using the base dnorm function in r. Subsequently, we will use the integrate functions to calculate p(data | H0) and p(data | H1) and calculate p_data_given_H1 / p_data_given_H0.

```
x <- bikes$temp
xbar <- mean(x); n <- length(x)
theta_0 <- 5; sigma2 <- 9
mu_0 <- 6; kappa_0 <- 1/2
mu_1 <- 4; kappa_1 <- 1/2
c_0 <- pnorm(theta_0, mean = mu_0, sd = sqrt(sigma2 / kappa_0))
c_1 <- pnorm(theta_0, mean = mu_1, sd = sqrt(sigma2 / kappa_1), lower.tail =
FALSE)
p_data_H0 <- function(theta) { # p(data, H0)
dnorm(xbar, mean = theta, sd = sqrt(sigma2 * (1/n + 1 / (kappa_0)))) / c_0 *
dnorm(theta, mean = mu_0, sd = sqrt(sigma2 / kappa_0)) }
p_data_H1 <- function(theta) { # p(data, H1)
dnorm(xbar, mean = theta, sd = sqrt(sigma2 * (1/n + 1 / (kappa_1)))) / c_1 *
dnorm(theta, mean = mu_1, sd = sqrt(sigma2 / kappa_1)) }
p_data_given_H0 <- integrate(p_data_H0, theta_0, Inf)$value # p(data | H0)
p_data_given_H1 <- integrate(p_data_H1, -Inf, theta_0)$value # p(data | H1)
BF_10 <- p_data_given_H1 / p_data_given_H0
BF_10

## [1] 0.6088382
```

The ratio p_data_given_H1 / p_data_given_H0 compares the likelihood of the data under H1 relative to the likelihood under H0. It can be thought of as the strength of evidence in favor of H1 compared to H0. Given a value of less than 1, there is not substantial evidence to think that our Null hypothesis is not true. Therefore, we accept our NULL hypothesis that the average temperature in January and February is greater than 5 degrees.

Question 2. Is there a relationship between the number of cyclists passing on different locations on the same day?

To begin answering the remaining questions for this report, we must first define our Bayesian Workflow process. This can used throughout the document for the basis of our question answering process. The steps are as follows.

1. We must begin by specifying a prior distribution.
2. We then fit our Bayesian GLM model using the stan_glm function. We provide the formula specifying the outcome variable and predictor variables, the data, the prior distribution, and specify the number of chains and iterations for the MCMC sampling.
3. Stating our priors, we then use the Update function to include our new priors into our model and simulate the posterior as necessary.

4. We then use density plots to either validate or invalidate our findings. 5.We then plot our PPC intervals and interpret our plots, that coupled with interpreting the Mean Average Error, we can ultimately answer our question.

I will be loading in my model outputs from saved files in my directory to save computation time for RMarkdown, this entire analysis is reproducible by removing the eval = False notation in the .Rmd file.

```
load(file="Clontarf_Model1.Rdata")
load(file="Clontarf_Model_prior1.Rdata")
load(file="Clontarf_Model.Rdata")
load(file="Clontarf_Model_prior.Rdata")
```

To answer this question, we will fit a Bayesian Poisson Glm with Clontarf as our response variable and Grove Road and Richmond Street as explanatory variables. By specifying family = poisson in the stan_glm function, the model is fitted using the Poisson likelihood function. The reason for using Poisson family is that this is commonly used for modelling Count Data, which hourly bicycle traffic volume is as this data represents discrete, non-negative counts of bicycle occurrences. We should test for overdispersion before continuing.

Furthermore as we have not been given any prior information for our model, we will be using weakly informative priors. We will be using the stan_glm function to find our adjusted priors and using these in our final model and interpreting. This process will be used for each of the 3 subsequent models in this report as we don't have any additional prior information on either of them.

Now let's fit our prior model:

```
Clontarf_Model_prior1 <- stan_glm(Clontarf ~ Grove_Road + Richmond_Street,
data = bikes,family = poisson,
# same as poisson(link = "log")
prior_intercept = normal(0, 2.5),
# prior for beta_0
prior = normal(0, 2.5, autoscale = TRUE),
# to tune other priors
chains = 4, iter = 5000*2, seed = 8566, prior_PD = TRUE)
```

Now lets fit our final model.

```
Clontarf_Model1 <- update(Clontarf_Model_prior1, prior_PD = FALSE)
```

Now we will test for overdispersion.

```
dispersiontest(Clontarf_Model1)

##
##  Overdispersion test
##
## data:  Clontarf_Model1
## z = 12.875, p-value < 2.2e-16
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   37.74529
```

From our dispersion test, we find our model is exhibiting over dispersion, meaning our variance of the counts is exceeding the mean. This violates the assumptions of the Poisson model. As over dispersion is present, an alternative family such as negative binomial is more appropriate. The negative binomial distribution allows for overdispersion by introducing an additional parameter that captures the extra variation in the data. This is fitted below:

```
Clontarf_Model_prior <- stan_glm(Clontarf ~ Grove_Road + Richmond_Street,
data = bikes,family = neg_binomial_2,
# same as poisson(link = "log")
prior_intercept = normal(0, 2.5),
# prior for beta_0
prior = normal(0, 2.5, autoscale = TRUE),
prior_aux = exponential(1, autoscale = TRUE),
# to tune other priors
chains = 4, iter = 5000*2, seed = 8566, prior_PD = TRUE)

Clontarf_Model_prior$prior.info$prior$adjusted_scale

## [1] 0.02131486 0.05089528
```

So, stan_glm gives the following adjusted priors:

- β1 ~ Normal(0, (0.0213)^2)
- β2 ~ Normal(0, (0.0509)^2)

```
Clontarf_Model <- update(Clontarf_Model_prior, prior_PD = FALSE)
```

Now that we have have fitted our model, we will now analyse our findings.

Furthermore we will perform a hypothesis test with:

H0: βi /= 0 H1: βi = 0

for i = 1,2

We will now use the tidy function to simulate our posterior.

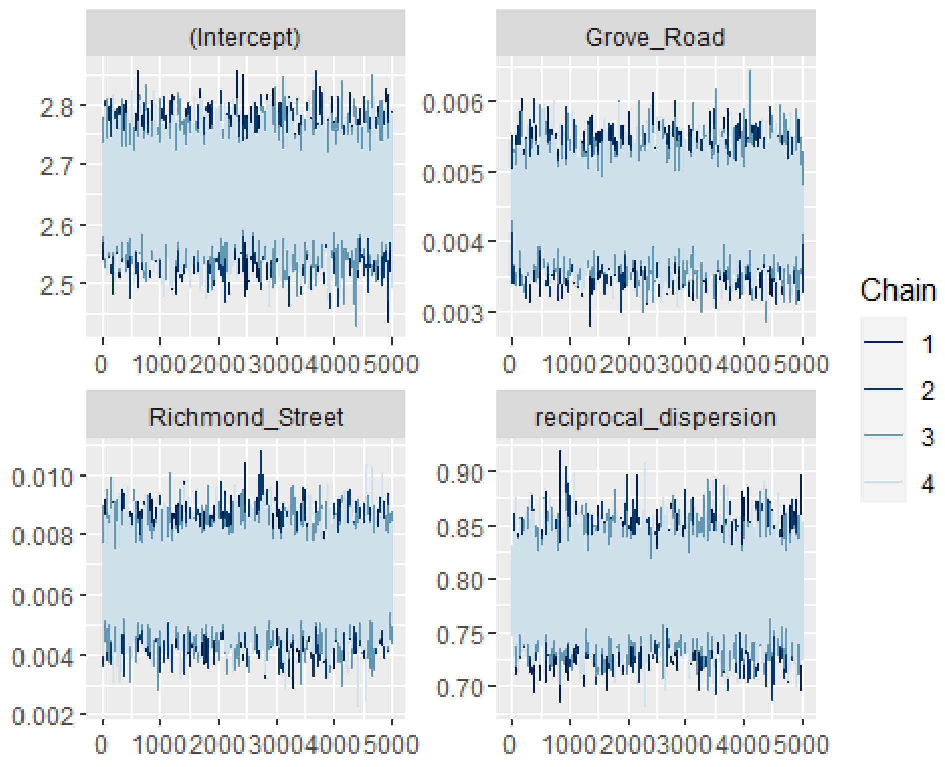
```
tidy(Clontarf_Model, conf.int = TRUE, conf.level = 0.95)

## # A tibble: 3 x 5
##   term            estimate std.error conf.low conf.high
##   <chr>              <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)      2.65      0.0563    2.55      2.76
## 2 Grove_Road       0.00445   0.000450  0.00360   0.00536
## 3 Richmond_Street  0.00648   0.00103   0.00451   0.00845
```

From our table above, our model finds that there is a relationship between the number of cyclists passing on the different locations on the same day, albeit not a large one. The average increase for Grove Road and Richmond Street is 0.00445 and 0.00648 respectively for each cyclist.

Using our confidence intervals defined, we find we are 95% confident that for each person cycling through Grove Road and/or Richmond Street there is a positive increase in the number of people cycling through Clontarf on average.

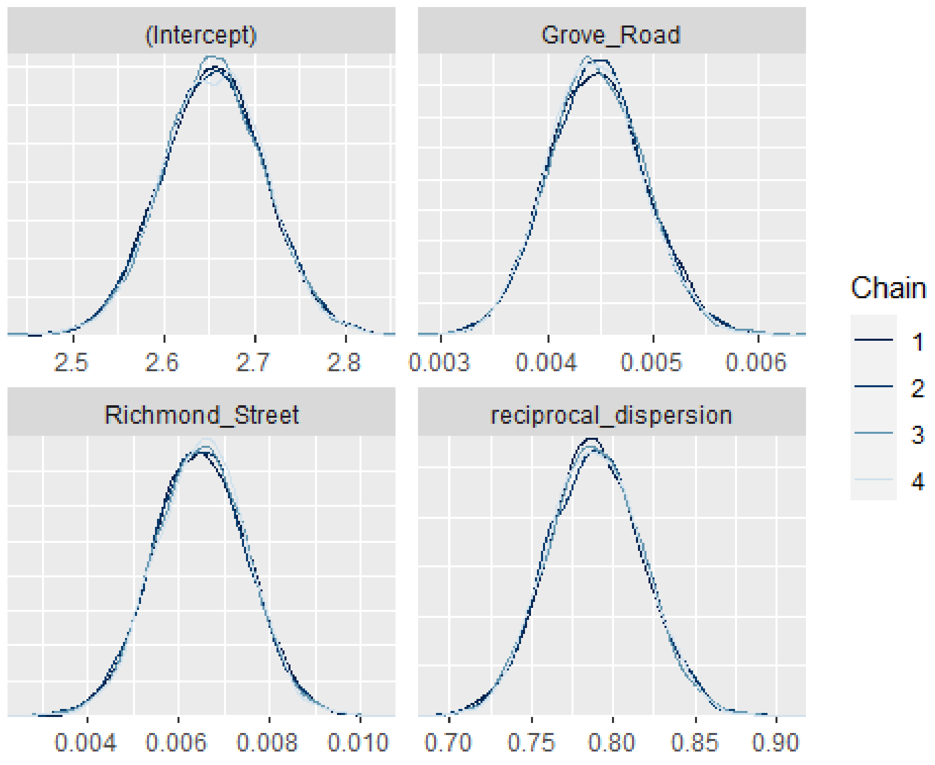
Before we validate our analysis, we must first check our MCMC trace graph.

```
mcmc_trace(Clontarf_Model)
```



As can be seen below, we have the proper trace for all of our variables. Next we will check if all our chains are create the correct/ similar density plots.

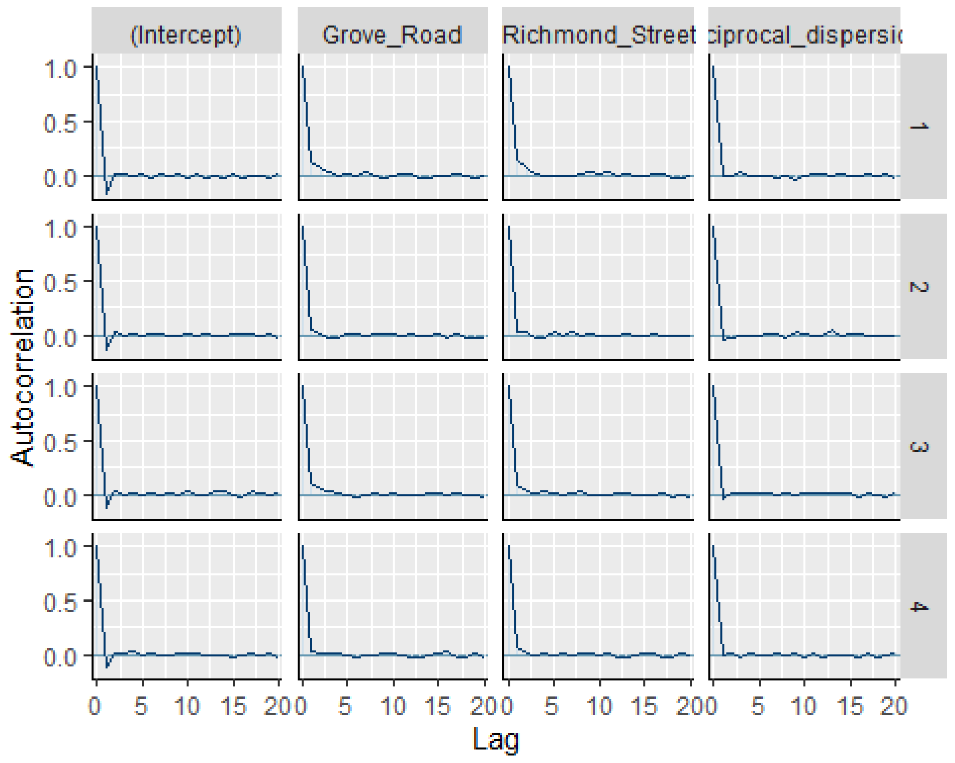
```
mcmc_dens_overlay(Clontarf_Model)
```



As we can see above, they are all quite similar, which means we can take interpret as non-suspect.

Lastly we will check our Autocorrelation plot.

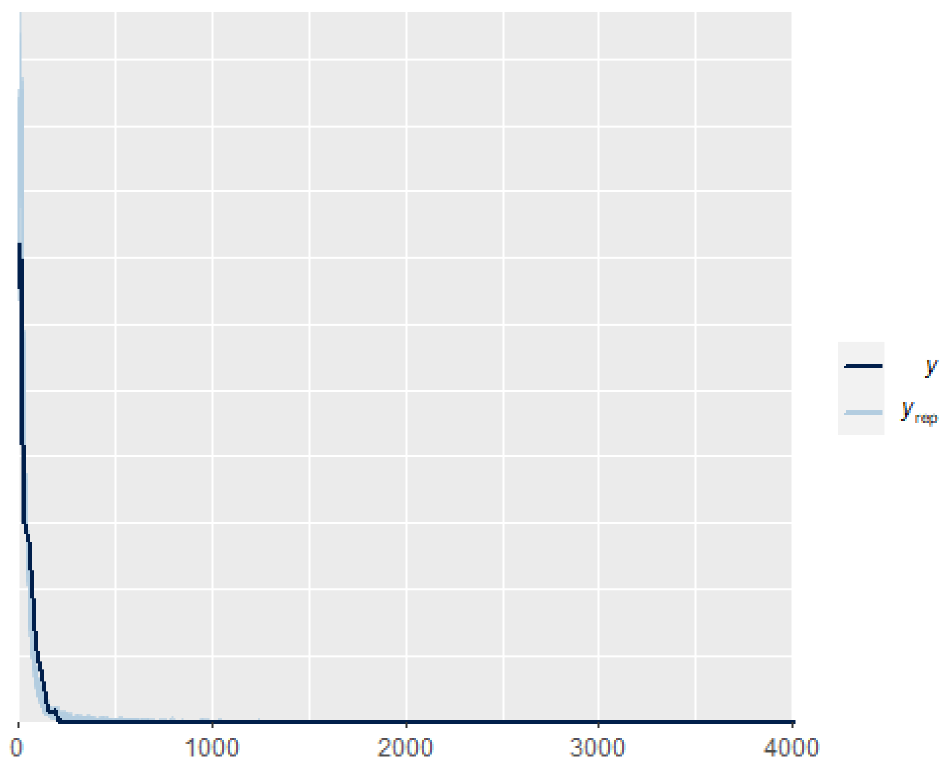
```
mcmc_acf(Clontarf_Model)
```



As our autocorrelation decreases very quickly, we can accept our computation as valid.

Next we will perform a Posterior Predictive Check on our dataset.

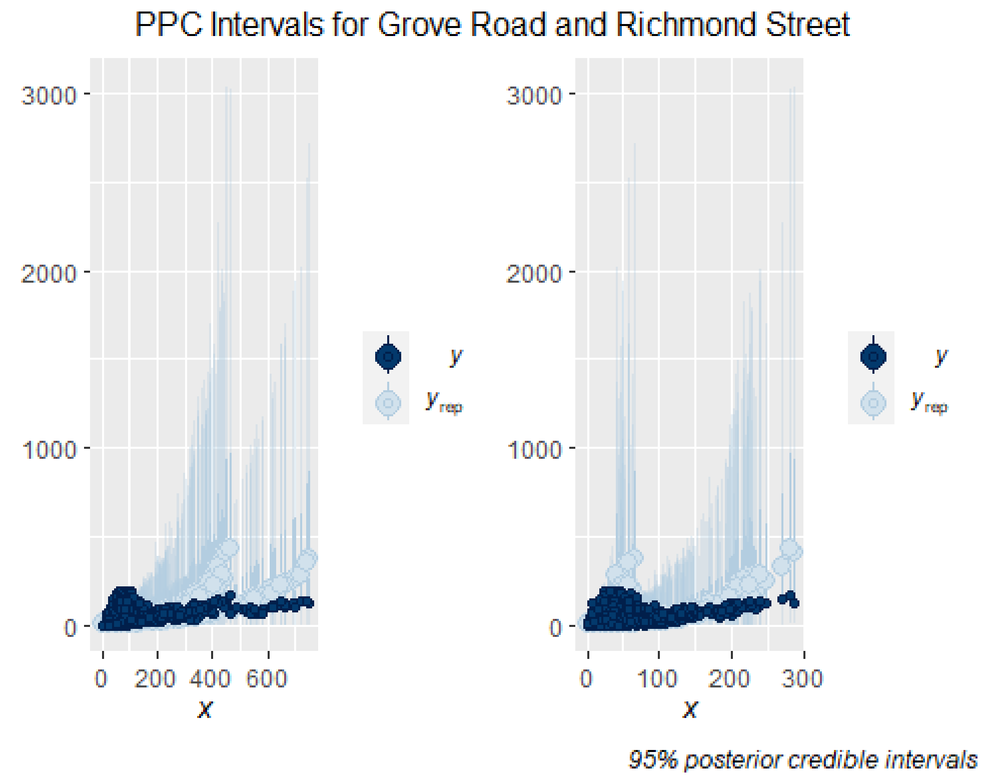
```
pp_check(Clontarf_Model)
```



Here we find our simulated dataset is very close to the one which we are observing.

Next we will simulate posterior predictive models and plot the posterior predictive models for each variable.

```
negbin_predictions_Clontarf <- posterior_predict(Clontarf_Model,
                                                  newdata = bikes)

p1 <- ppc_intervals(bikes$Clontarf,
                    yrep = negbin_predictions_Clontarf,
                    x = bikes$Grove_Road,
                    prob = 0.5, prob_outer = 0.95,
                    # 50% and 95% posterior credible intervals
                    facet_args = list(scales = "fixed"))

p2 <- ppc_intervals(bikes$Clontarf,
                    yrep = negbin_predictions_Clontarf,
                    x = bikes$Richmond_Street,
                    prob = 0.5, prob_outer = 0.95,
                    # 50% and 95% posterior credible intervals
                    facet_args = list(scales = "fixed"))

load(file="p1.Rdata")
load(file="p2.Rdata")

plot <- grid.arrange(p1,
  p2,
  nrow = 1,
  top = "PPC Intervals for Grove Road and Richmond Street",
  bottom = textGrob(
    "95% posterior credible intervals",
    gp = gpar(fontface = 3, fontsize = 9),
    hjust = 1,
    x = 1
  )
)
```



```
plot

## TableGrob (3 x 2) "arrange": 4 grobs
##   z     cells    name                grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (2-2,2-2) arrange      gtable[layout]
## 3 3 (1-1,1-2) arrange text[GRID.text.1091]
## 4 4 (3-3,1-2) arrange text[GRID.text.1092]
```

From our plot, we can see that our finding fall in the credible intervals of our posterior.

Now we will evaluate the performance of our model:

```
mean(bikes$Clontarf)

## [1] 37.83263

prediction_summary(model = Clontarf_Model, data = bikes)
```

```
##        mae mae_scaled within_50 within_95
## 1 14.73702   0.591636 0.5720339 0.9512712
```

Here we find a Mean Absolute Error of 14.7918. However given a mean Clontarf cyclists of 37.83263,average error between the predictions and actuals in this dataset of 14.7918 is not a great value.

It should be noted there may be slight variation in the numbers shown here, but interpretation should be the same.

Question 3. Is there a relationship between the number of cyclists and the weather?

To answer this question, we will employ the same process as in question 2.

```
load(file="Cycling_Weather_Model.Rdata")
load(file="Cycling_Weather_Model_prior.Rdata")
```

To answer this question, we will fit a Bayesian Negative Binomial Glm with Clontarf as our response variable and Grove Road and Richmond Street as explanatory variables from our previous model and adding our additional weather variables rain, temp, wdsp, vis and clamt. By specifying family = neg_binomial_2 in the stan_glm function, the model is fitted using the negative binomial likelihood function.

we have not been given any prior information for our model, we will be using weakly informative priors. We will be using the stan_glm function to find our adjusted priors and using these in our final model and interpreting.

```
Cycling_Weather_Model_prior <- stan_glm(Clontarf ~ Grove_Road +
Richmond_Street + rain + temp + wdsp + vis + clamt,
data = bikes,family = neg_binomial_2,
# same as poisson(link = "log")
prior_intercept = normal(0, 3),
# prior for beta_0
prior = normal(0, 3, autoscale = TRUE),
prior_aux = exponential(1, autoscale = TRUE),
# to tune other priors
chains = 4, iter = 5000*2, seed = 8566, prior_PD = TRUE)

Cycling_Weather_Model_prior$prior.info$prior$adjusted_scale

## [1] 2.557783e-02 6.107434e-02 1.290414e+01 9.019023e-01 7.541681e-01
## [6] 3.646882e-04 1.442371e+00
```

So, stan_glm gives the following priors:

- β1 ~ Normal(0, (2.557783e-02)^2)
- β2 ~ Normal(0, (6.107434e-02)^2)
- β3 ~ Normal(0, (1.290414e+01)^2)
- β4 ~ Normal(0, (9.019023e-01)^2)
- β5 ~ Normal(0, (7.541681e-01)^2)

- β6 ~ Normal(0, (3.646882e-04)^2)
- β7 ~ Normal(0, (1.442371)^2)

```
Cycling_Weather_Model <- update(Cycling_Weather_Model_prior, prior_PD =
FALSE)
```

Now that we have have fitted our model, we will now analyse our findings. We will now use the tidy function to simulate our posterior.

```
tidy(Cycling_Weather_Model, conf.int = TRUE, conf.level = 0.95)

## # A tibble: 8 x 5
##   term              estimate  std.error   conf.low  conf.high
##   <chr>                <dbl>      <dbl>      <dbl>      <dbl>
## 1 (Intercept)      1.70      0.144       1.42       1.98
## 2 Grove_Road       0.00493   0.000428    0.00414    0.00582
## 3 Richmond_Street  0.00478   0.000966    0.00289    0.00664
## 4 rain            -0.477     0.140      -0.738     -0.183
## 5 temp             0.0467    0.0111      0.0245     0.0682
## 6 wdsp            -0.0340    0.00815    -0.0497    -0.0177
## 7 vis              0.0000298 0.00000306  0.0000238  0.0000359
## 8 clamt            0.0441    0.0157      0.0138     0.0747
```

As we have considerably more variables we will have to run a separate hypothesis test on each of our variables.

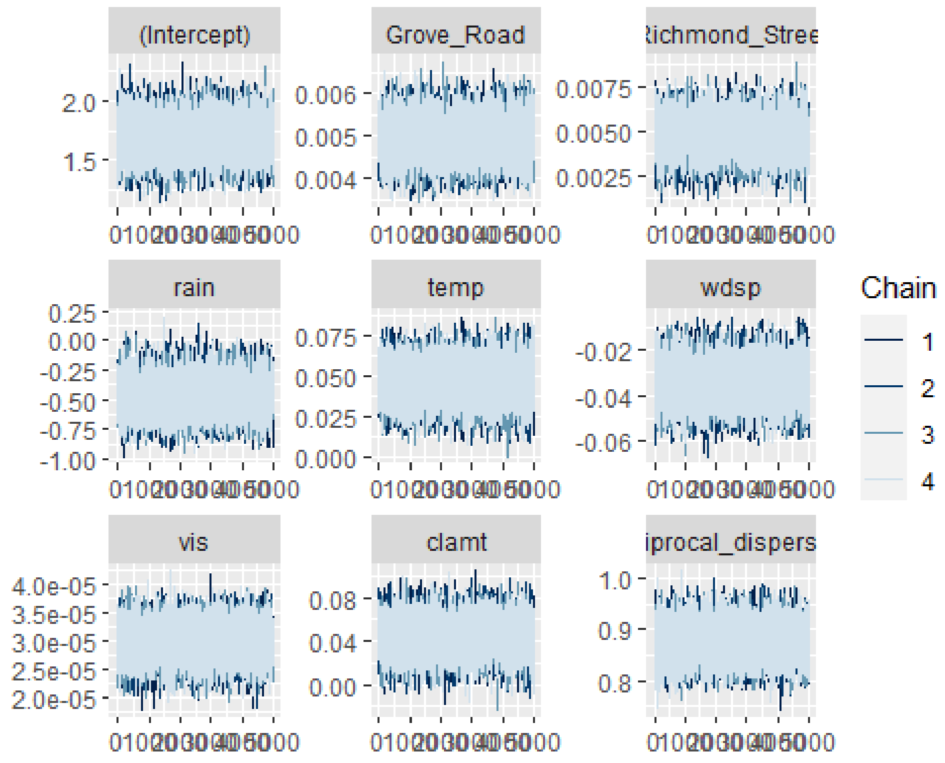
From our table above, our model finds that there is a relationship between the number of cyclists passing through Clontarf and all of our variables with all but two having a positive effect. This illustrates a major relationship between the number of the cyclists and the weather.

Our two negative variables are the most obvious, that being the precipitation amount and the windspeed. As expected, rain has the highest negative effect on the number of cyclists going through clontarf with negative 0.477 cyclists for each additional millimeter of rain. This effect is 10 times larger that any other variables in our model.

The positive variables we find are: Grove Road, Richmond Street, Temperature, Cloud Amount and a miniscule positive effect from visibility (this may be because visibility is rarely an issue for cyclists in Dublin due to the constant residential lighting and fog is not common in these areas. The highest positive effect from our model is temperature with 0.0467 more cyclists going through Clontarf on average for each additional degree of air temperature.

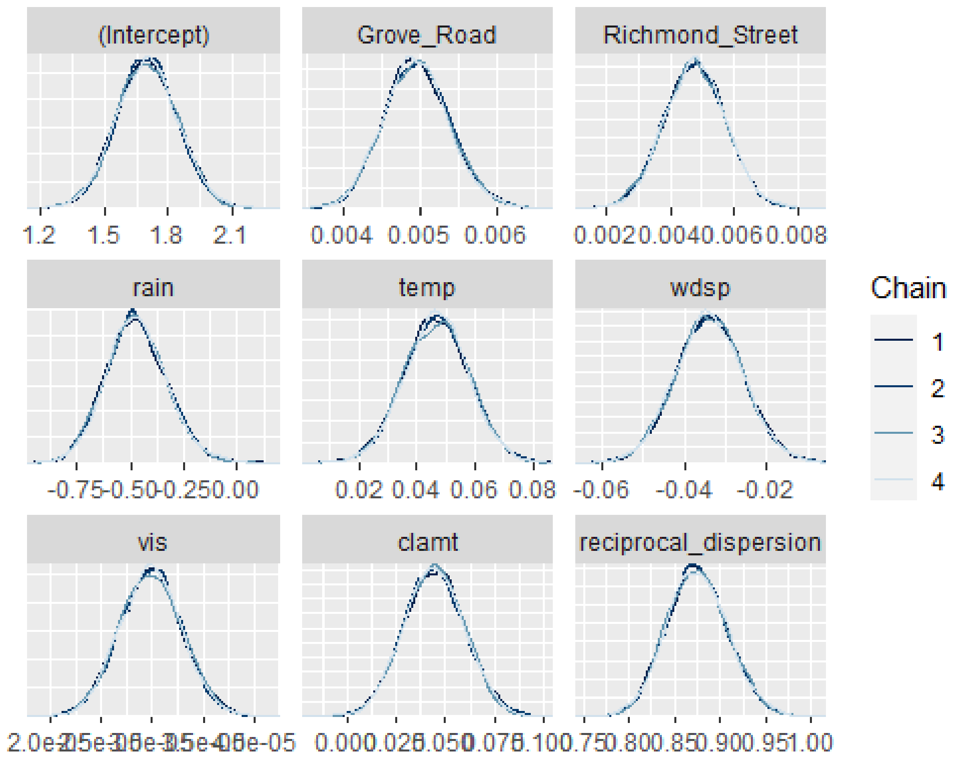
Before we validate our analysis, we must first check our MCMC trace graph.

```
mcmc_trace(Cycling_Weather_Model)
```



As can be seen below, we have the proper trace for all of our variables. Next we will check if all our chains are create the correct/ similar density plots.

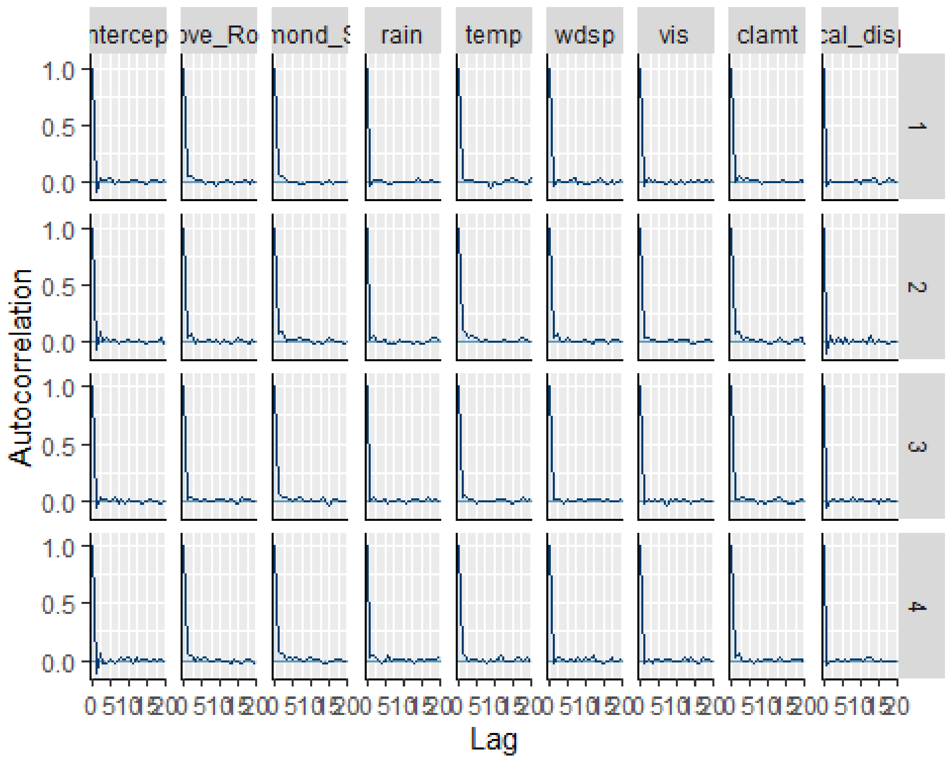
```
mcmc_dens_overlay(Cycling_Weather_Model)
```



As we can see above, they are all quite similar, which means we can take interpret as non-suspect.

Lastly we will check our Autocorrelation plot.

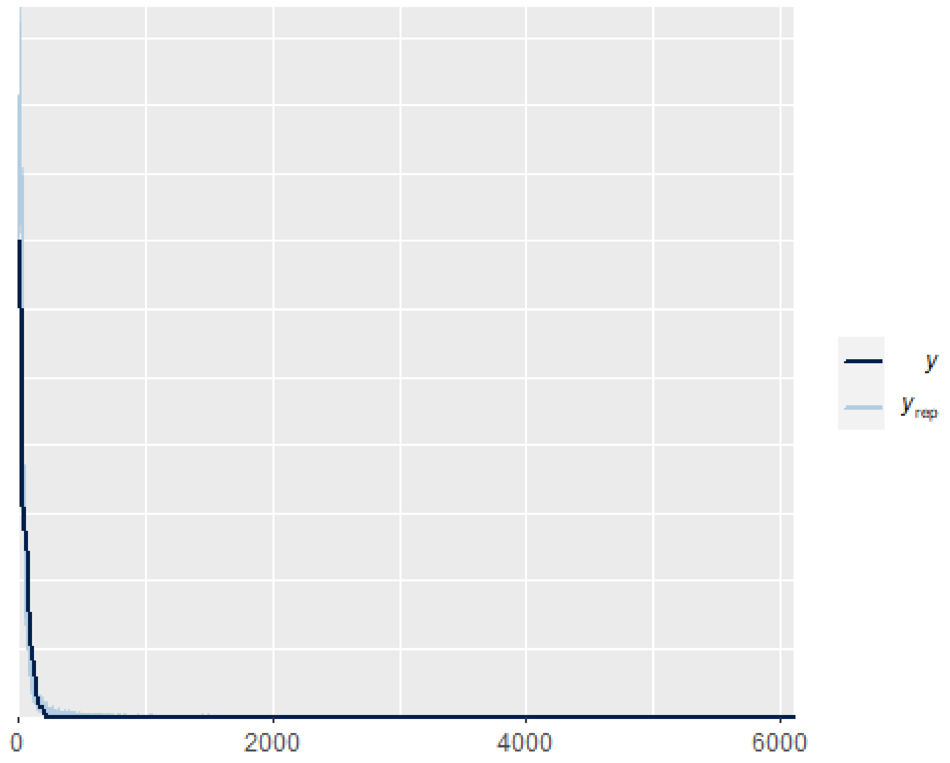
```
mcmc_acf(Cycling_Weather_Model)
```



As our autocorrelation decreases very quickly, we can accept our computation as valid.

Next we will perform a Posterior Predictive Check on our dataset.

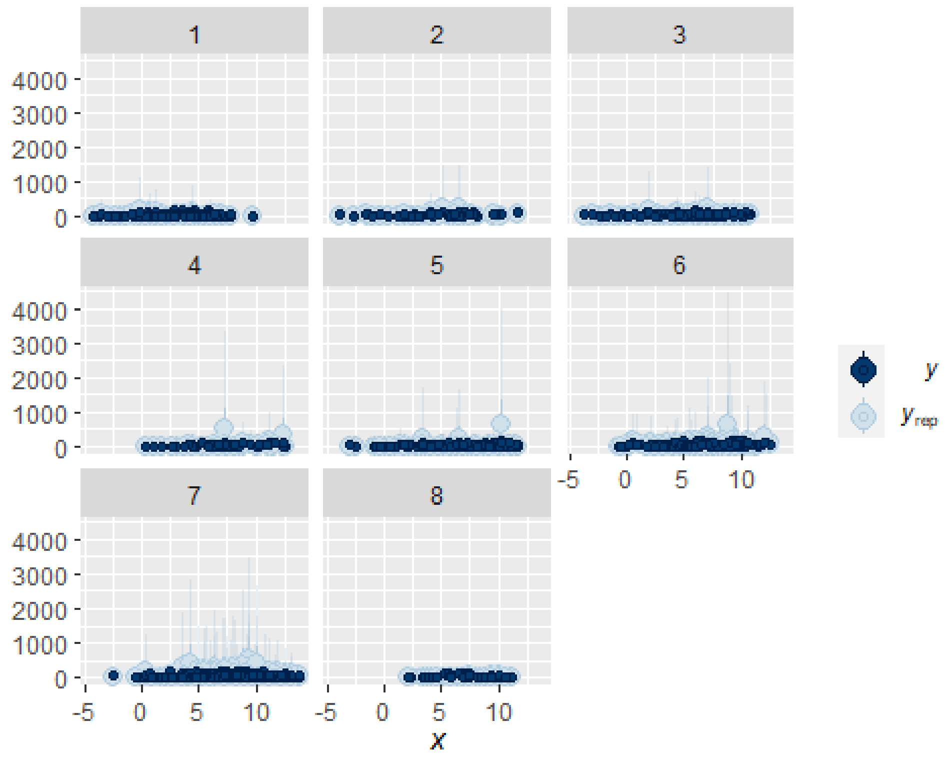
```
pp_check(Cycling_Weather_Model)
```



Here we find our simulated dataset is very close to the one which we are observing.

Next we will simulate posterior predictive models and plot the posterior predictive models for each variable.

```
negbin_predictions_Cycling_Weather <-
posterior_predict(Cycling_Weather_Model, newdata = bikes)

ppc_intervals_grouped(bikes$Clontarf,
                    yrep = negbin_predictions_Cycling_Weather,
                    x = bikes$temp,
                    group = bikes$clamt,
                    prob = 0.5, prob_outer = 0.95,
                    # 50% and 95% posterior credible intervals
                    facet_args = list(scales = "fixed"))
```



From our plot, we can see that our finding fall in the credible intervals of our posterior.

Now we will evaluate the performance of our model:

```
mean(bikes$Clontarf)

## [1] 37.83263

prediction_summary(model = Cycling_Weather_Model, data = bikes)

##        mae mae_scaled within_50 within_95
## 1 13.95097  0.5984146 0.5685028 0.9682203
```

Here we find a Mean Absolute Error of 13.9092. However given a mean Clontarf cyclists of 37.83263,average error between the predictions and actuals in this dataset of 13.9092 is a better but still not great value.

Question 4. Other scientific question: Is there a relationship between the number of Cyclists and the time of day, is this consistant across the week?

```
load(file="Day_Hour_Model.Rdata")
load(file="Day_Hour_Model_prior.Rdata")
```

To answer this question, we will fit a Bayesian Negative Binomial Glm with Clontarf as our response variable and the Day and Hours variables as explanatory variables.

Once again, we have not been given any prior information for our model, we will be using weakly informative priors. We will be using the stan_glm function to find our adjusted priors and using these in our final model and interpreting.

```
Day_Hour_Model_prior <- stan_glm(Clontarf ~ Day + Hour,
data = bikes,family = neg_binomial_2,
# same as poisson(link = "log")
prior_intercept = normal(0, 3),
# prior for beta_0
prior = normal(0, 3, autoscale = TRUE),
prior_aux = exponential(1, autoscale = TRUE),
# to tune other priors
chains = 4, iter = 5000*2, seed = 8566, prior_PD = TRUE)

Day_Hour_Model_prior$prior.info$prior$adjusted_scale

## [1] 8.340913 8.759711 8.340913 8.759711 8.340913 8.759711 0.433236
```

So, stan_glm gives the following priors:

- β1 ~ Normal(0, (8.340913)^2)
- β2 ~ Normal(0, (8.759711)^2)
- β3 ~ Normal(0, (8.340913)^2)
- β4 ~ Normal(0, (8.759711)^2)
- β5 ~ Normal(0, (8.340913)^2)
- β6 ~ Normal(0, (8.759711)^2)
- β7 ~ Normal(0, (0.433236)^2)

```
Day_Hour_Model <- update(Day_Hour_Model_prior, prior_PD = FALSE)
```

Now that we have have fitted our model, we will now analyse our findings. We will now use the tidy function to simulate our posterior.

```
tidy(Day_Hour_Model, conf.int = TRUE, conf.level = 0.95)

## # A tibble: 8 x 5
##   term        estimate std.error conf.low conf.high
##   <chr>          <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)   2.28     0.127     2.04       2.53
## 2 DayMon        0.277    0.123     0.0383     0.519
## 3 DaySat        0.261    0.126     0.0147     0.504
## 4 DaySun        0.472    0.124     0.235      0.713
## 5 DayThu        0.192    0.127    -0.0534     0.439
## 6 DayTue        0.222    0.123    -0.0177     0.458
## 7 DayWed        0.111    0.126    -0.133      0.363
## 8 Hour          0.0912   0.00774   0.0762     0.106
```

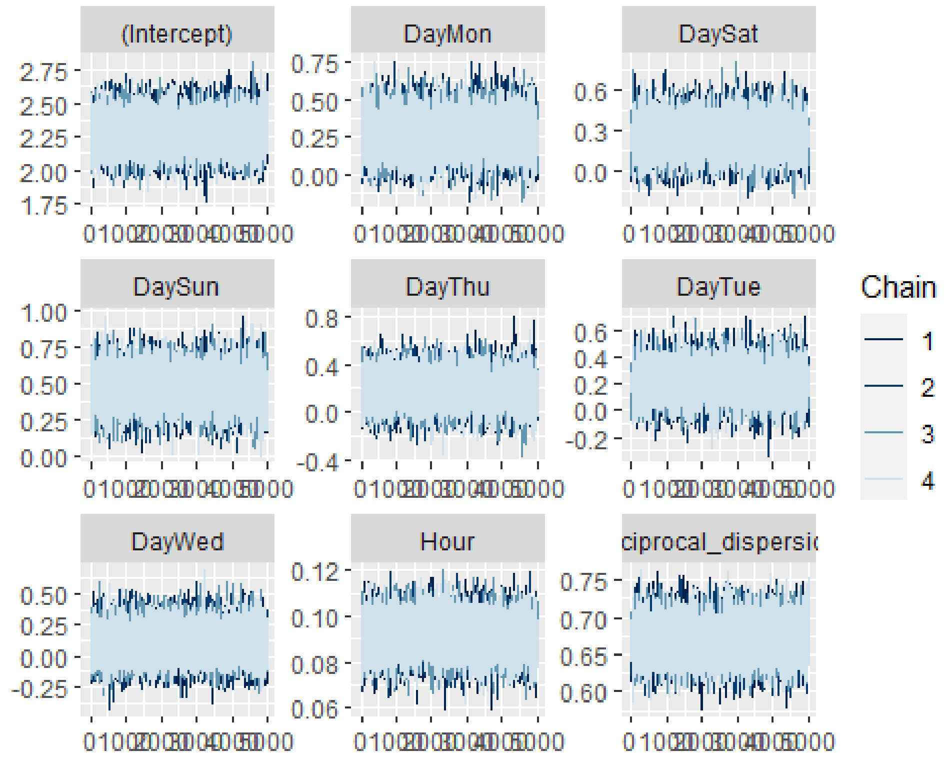
Here our model finds 3 days have no effect on the number of cyclists in Clontarf, these being Tuesday to Thursday. This indicates it is not a consistent relationship across the week. This makes sense as these days are often fall right in the middle of the work day and less people are going to be cycling. All other days in our model have a small positive effect on the average number of Cyclists passing through Clontarf. The day which has the highest positive affect on the number of cyclists passing through Clontarf is Sunday with 0.472 increase on average with each Sunday entry in our data. However, it is clear from our findings that the weekend has a considerable effect on the number of cyclists passing through Clontarf.

Lastly we can see the hour variable has a small positive effect on the number of cyclists. This is indicative of people cycling more later in the day.

Now let's plot our PPC intervals for hours grouped by days of the week. This will allow us to see the affect of the days on our data.

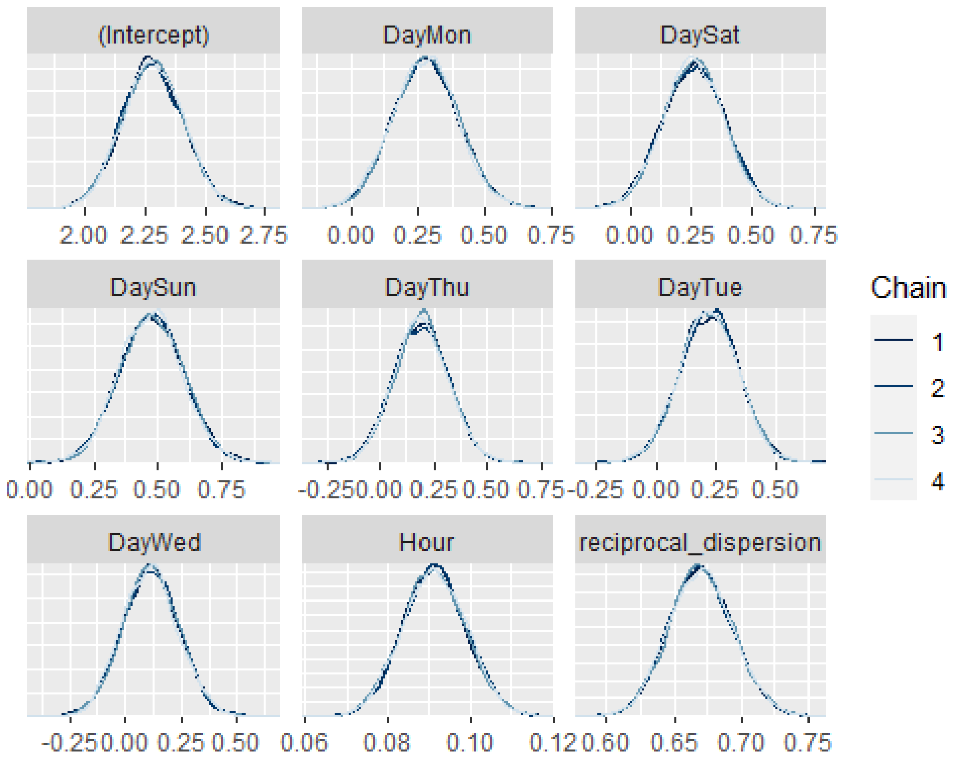
Before we validate our analysis, we must first check our MCMC trace graph.

```
mcmc_trace(Day_Hour_Model)
```



As can be seen below, we have the proper trace for all of our variables. Next we will check if all our chains are create the correct/ similar density plots.

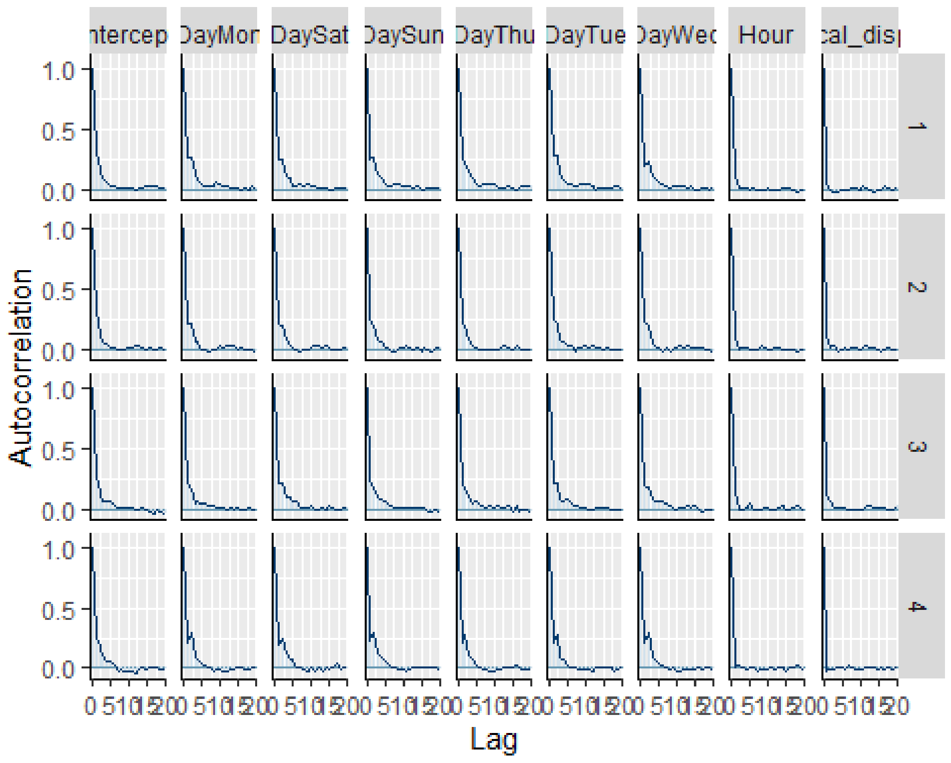
```
mcmc_dens_overlay(Day_Hour_Model)
```



As we can see above, they are all quite similar, which means we can take interpret as non-suspect. However there is some minor over fitting on the Thursday variable.

Lastly we will check our Autocorrelation plot.

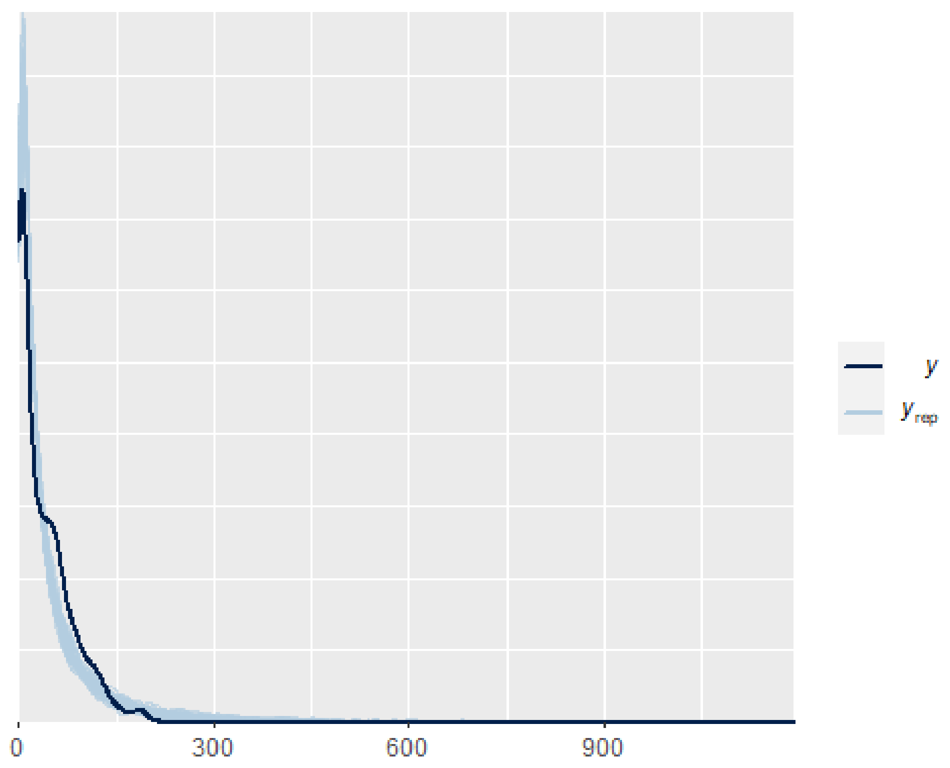
```
mcmc_acf(Day_Hour_Model)
```



Here we can see some of our autocorrelation plots take longer than our previous ones to drop. This could be due to the reduced data sample for all of these day variables. The most notable here are our Wednesday and Thursday variables. However, as most of our autocorrelation decreases very quickly, we can accept our computation as valid.

Next we will perform a Posterior Predictive Check on our dataset.

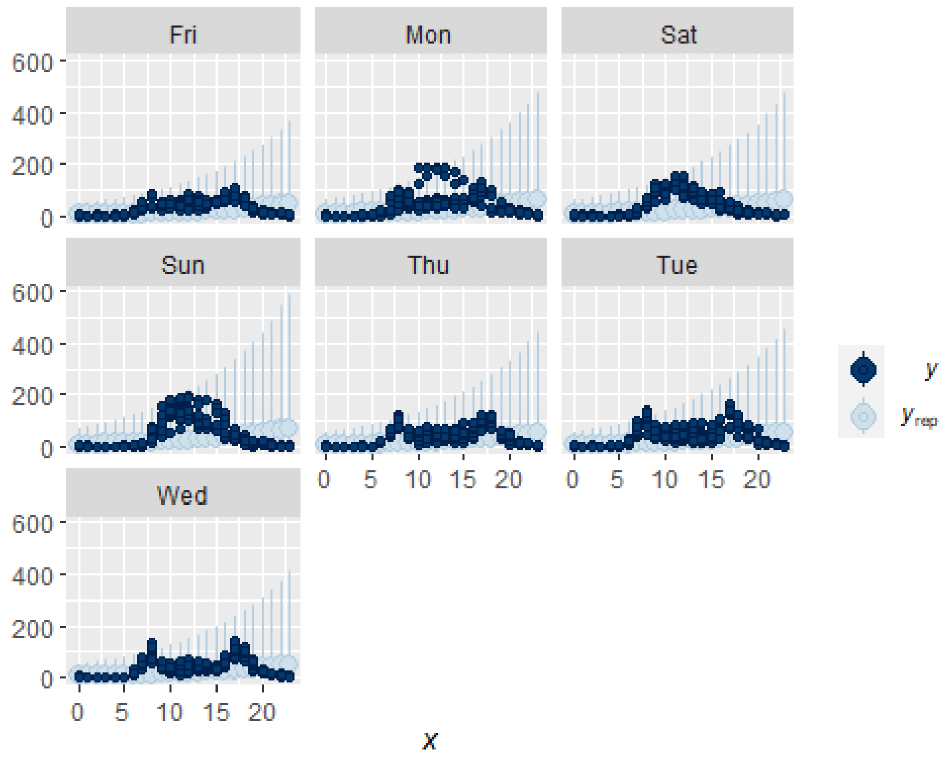
```
pp_check(Day_Hour_Model)
```



Here we find our simulated dataset is very close to the one which we are observing.

Next we will simulate posterior predictive models and plot the posterior predictive models for each variable.

```
negbin_predictions_Day_Hour <- posterior_predict(Day_Hour_Model, newdata =
bikes)

p3 <- ppc_intervals_grouped(bikes$Clontarf,
                      yrep = negbin_predictions_Day_Hour,
                      x = bikes$Hour,
                      group = bikes$Day,
                      prob = 0.5, prob_outer = 0.95,
                      # 50% and 95% posterior credible intervals
                      facet_args = list(scales = "fixed"))

load(file="p3.Rdata")
p3
```



From our plot, we can see that our finding fall in the credible intervals of our posterior except in our Monday plot where we find a number of our findings falling outside of our posterior predictive check.

All of these density plot results are indicative of a number of fitting issues with our Day-Hour model. This tells me that we should be skeptical of our results. It is not likely this is the best model of our previous 2.

Now we will evaluate the performance of our model:

```
mean(bikes$Clontarf)

## [1] 37.83263

prediction_summary(model = Day_Hour_Model, data = bikes)

##       mae mae_scaled within_50 within_95
## 1 20.5066  0.6725379  0.430791 0.9887006
```

Here we find a Mean Absolute Error of 20.45423. However given a mean Clontarf cyclists of 37.83263,average error between the predictions and actuals in this dataset of 20.45423 is a bad value. This model does not perform well.

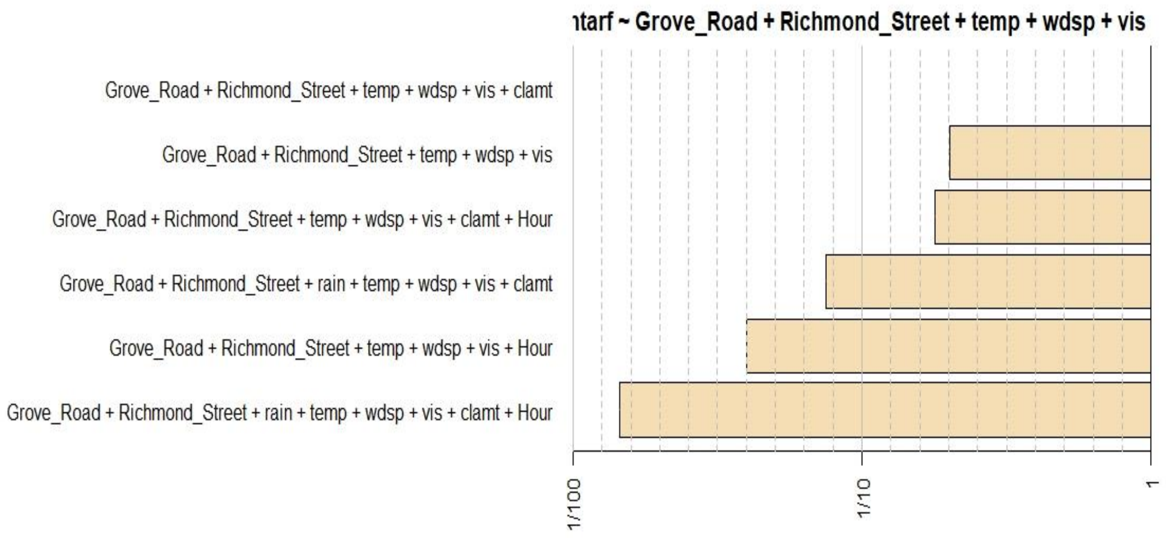
Question 5. Whats the best model between the ones proposed?

We will begin our comparisons by using Bayesian regression for all models vs the intercept-only model and see which of our models is the best performing.

```
bf <- regressionBF(Clontarf ~ Grove_Road + Richmond_Street +
                          rain + temp + wdsp + vis + clamt + Hour, data = bikes)
```

Bayesian regression and BF for all models vs the best model.

```
bf_vs_best <- head(bf) / max(bf)
plot(bf_vs_best)
```



Here we find the best performing model is very similar to Model 2, only leaving out the rain variable.

We can also perform model selection by using the function loo. This computes “leave-one-out” validation of the data, a type of cross-validation.

Then we evaluate the leave-one-out validation using loo, before comparing the models using loo_compare. Differences of less than 5 are quite small and differences bigger than ten are sizeable.

```
load(file="Comparison.Rdata")

loo_1 <- loo(Clontarf_Model)
loo_2 <- loo(Cycling_Weather_Model)
loo_3 <- loo(Day_Hour_Model)

Comparison <- loo_compare(loo_1, loo_2, loo_3)

Comparison

##                       elpd_diff se_diff
## Cycling_Weather_Model    0.0       0.0
## Clontarf_Model         -73.9      15.1
## Day_Hour_Model        -203.0      23.4
```

Here the difference between models is heavily in favor of Model 2 our Cycling Weather Model. Clearly using Leave One Out Validation, it is our best performing model, followed by Model 1 and lastly Model 3. Both of which have considerably worse performance than expected. This is especially true for Model 1, where it's Mean Absolute Error was not particularly far off that of Model 2. From this analysis, we can safely state that our Cycling_Weather_Model is our best model between the ones proposed.