Assignment 3

Group 54: Matthew Willaton (29658179), Rory Thompson, Muhammad Ali Aamir Raja, Yanting Chen

01/11/2020

Read in data

Part B: Multiple Linear Regression

```
n<-nrow(cat)
modelf <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data=cat)
tidyf <- tidy(modelf)
glancef <- glance(modelf)
augmentf <- augment(modelf)
```

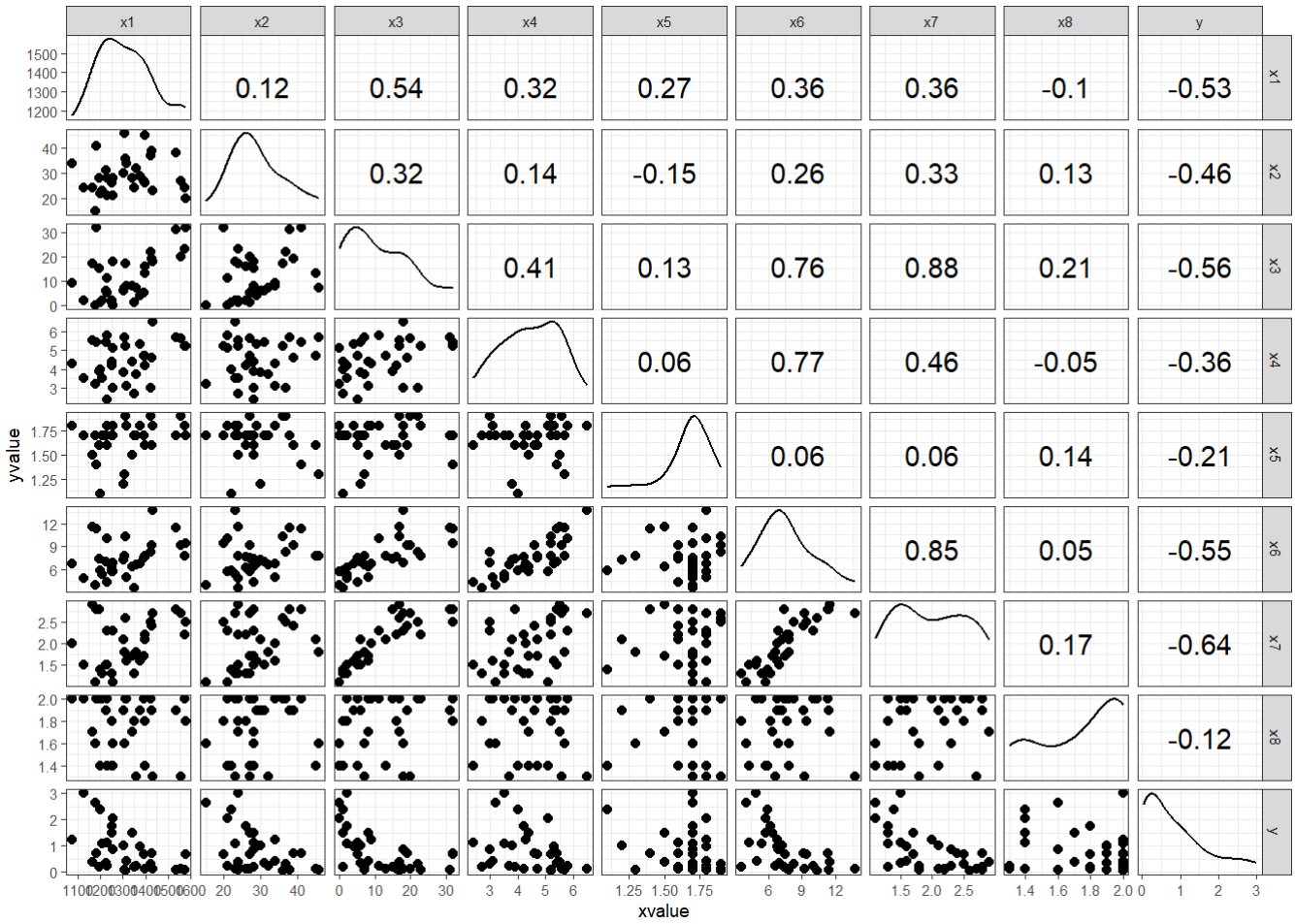
Q10. [10 marks]

The lower triangle in the matrix of scatterplots shows us the scatterplot of the each of the variables x_i against x_j for i != j and i,j is an element of {1,2,3,…,8}. This is useful for identifying any potential multicolinearity that may exist in our linear model if we run a regression. From this scatter plot we see that x6 appears to have a strong postive correlation with x4 and x3 and shares a weak positive correlation with x1 and x2, indicating that it might possibly be a variable that we ought to leave out if we choose to run the regression with x6 and those variables that it appears to have a correlation (where they would be the regressors). Also we can appreciate what appears to be ‘discrete’ behaviour from x5 when it is plotted against the other variables. This is strange because it indicates that x5 ought to have been a category this could be something we investigate further.

The leading diagonal is simply just the line graph of the distribution of the variables x_i, i is an element {1,2,3,…,8}. So basically it shows us the density of each variable. This is useful because it can give us a sense of how the values of that variables are distributed in our sample.

And the upper triangle of the matrix of scatterplots is the correlation coefficients R, of each of the variables x_i against x_j for i != j and i,j is an element of {1,2,3,…,8}. Again this is very useful for identifying any potential multicolinearity that may exist if we were to run a regression using these as regressors. And once again we would like to point out the very high correlation of x6 with the other regressors, indicating that it might not be a very good regressor to use if we choose to run a regression.

```
ggscatmat(cat, columns=c(1:9)) +
  theme(geom.text.size = 7, strip.text.x = element_text(size = 12)) +
  theme_bw(base_size = 7)
```



Q11. [5 marks]

So regressor 6 has a VIF above 10 which means that it is quite multicolinear with our other regressors. this suggests that by leaving in regressor 6 we will degrade our model, by making it less accurate (i.e beta_6 has a larger s.e which means that the true beta_6 may be a drastically different value then we actually have… This is not good because that will also means we aren't capturing the effect of the regressor properly.). And this aligns with what we saw in the previous question, thus we have a really good case for why we ought to leave x6 out of our regression model.

```
output %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

x1	x2	x3	x4	x5	x6	x7	x8
1.85	1.18	6.15	4.09	1.17	11.8	9.09	1.22

Q12. [10 marks]

The variance inflation factor is equal to the ratio between the variance of the overall model to the variance of a model with only that explanatory variable. This ratio looks at the amount of multicollinearity for a set of independent variables in a multiple linear regression. Multicollinearity is when two or more explanatory are highly correlated. This is a problem for our model since it increases the standard error of a regression coefficient and this will make the variable less likely to be significant as a result of arithmetic and means we are more likely to commit a Type 2 error for the individual significance of the regressor. (Not rejecting it's non-significance in favour of it's significance.). Now if a regressor has a VIF of > 10 then that means it is highly

collinear with other regressor(s) (in our case we found x_6 to have a VIF of 11.8). This would mean if we left the x_6 un-addressed it would make us more likely to commit a type 2 error. Hence it is important that we devise a way of dealing with the high collinearity of this regressor.

Q13. [5 marks]

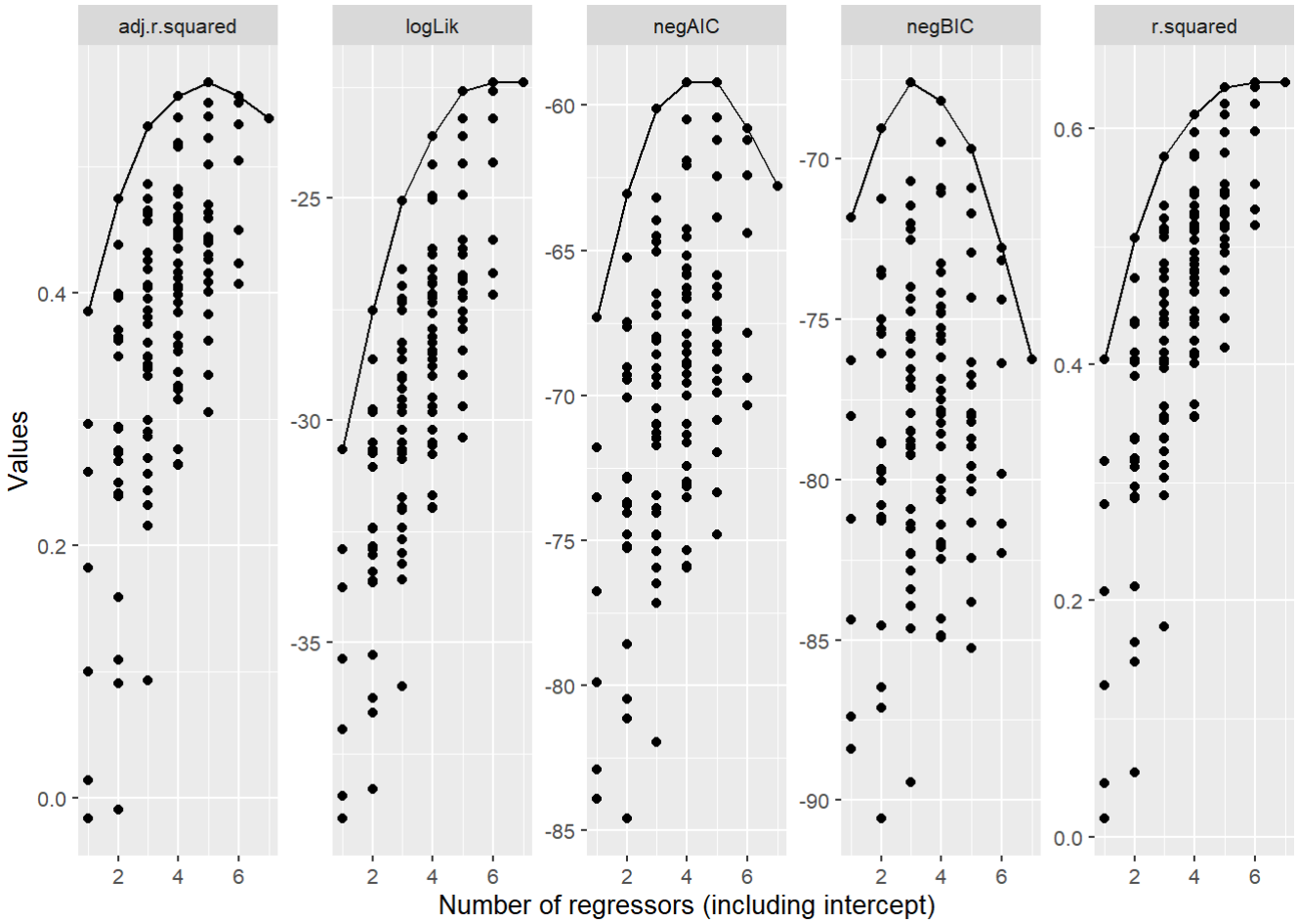
Well from the given code we observe that regressor 6 and the y column have been taken out from the original dataframe (and in fact that is what `cat[,-c(6,9)]` is doing) then simply the code creates an ensemble of models from each possible subset of {x1,x2,x3,x4,x5,x7,x8} (which has 7 elements as it is clearly visible) which we know is going to be 2^7 - 1 (minus 1 for the trivial case). Hence that is why it's 127. This method of creating ensemble models is a good way of just 'brute force' fitting all the possible models and then comparing the statistics of each model (i.e model comparison criterion to select the most appropriate model).

```
summary(all_mod)%>%
  head() %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

	df	logL	AIC	BIC	R2	adjR2	n	model
m1	3	-33.8	-73.5	-78.0	0.281	0.258	33	1
m2	3	-35.4	-76.7	-81.2	0.207	0.182	33	2
m3	4	-29.7	-67.5	-73.5	0.437	0.399	33	3
m4	3	-32.9	-71.8	-76.3	0.318	0.296	33	4
m5	4	-31.0	-70.1	-76.1	0.391	0.350	33	5
m6	4	-30.7	-69.5	-75.5	0.402	0.362	33	6

Q14. [20 marks]

```
nmod <-127
all_mod_s <- all_mod %>%
  map_df(glance) %>%
  mutate(model = nmod) %>%
  mutate(negBIC = -1*BIC, negAIC = -1*AIC)

label <- NULL
for (i in nmod) {
  l <- as.character(summary(all_mod[[i]])$call)[2]
  label <- c(label,
    substr(l, 5, str_length(l)))
}

all_mod_s_long <- all_mod_s %>%
  gather(fit_stat, val, adj.r.squared, negAIC,
         negBIC, logLik, r.squared) %>%
  group_by(fit_stat, df) %>%
  mutate(rank = min_rank(desc(val)))

p1 <- ggplot(all_mod_s_long, aes(df, val)) +
  geom_point() +
  geom_line(data=filter(all_mod_s_long, rank == 1)) +
  facet_wrap(~fit_stat, ncol = 5, scales = "free_y") +
  xlab("Number of regressors (including intercept)") +
  ylab("Values") +
  theme_grey(base_size = 10)

p1
```



```
print("Adjusted R-squared")
```

```
## [1] "Adjusted R-squared"
```

```
indexadjRsq<-c(1:nmod)[all_mod_s$adj.r.squared==max(all_mod_s$adj.r.squared)]
indexadjRsq
```

```
## [1] 55
```

```
max_adjRsq <- all_mod[[indexadjRsq]]
max_adjRsq
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x5 + x7, data = data, model = FALSE)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x5           x7
##      8.1696      -0.0026      -0.0361       0.0396      -0.6923      -1.1078
```

```
print("log-Likelihood")
```

```
## [1] "log-Likelihood"
```

```
indexlogLik<-c(1:nmod)[all_mod_s$logLik==max(all_mod_s$logLik)]
indexlogLik
```

```
## [1] 127
```

```
max_logLik <- all_mod[[indexlogLik]]
max_logLik
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x7 + x8, data = data,
##     model = FALSE)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4           x5
##     8.68330     -0.00275     -0.03530      0.04252     -0.01268     -0.62963
##          x7           x8
##    -1.11429     -0.23308
```

```
print("Negative AIC")
```

```
## [1] "Negative AIC"
```

```
indexAIC<-c(1:nmod)[all_mod_s$negAIC==max(all_mod_s$negAIC)]
indexAIC
```

```
## [1] 55
```

```
max_AIC <- all_mod[[indexAIC]]
max_AIC
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x5 + x7, data = data, model = FALSE)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x5           x7
##      8.1696      -0.0026      -0.0361       0.0396      -0.6923      -1.1078
```

```
print("Negative BIC")
```

```
## [1] "Negative BIC"
```

```
indexBIC<-c(1:nmod)[all_mod_s$negBIC==max(all_mod_s$negBIC)]
indexBIC
```

```
## [1] 35
```

```
max_BIC <- all_mod[[indexBIC]]
max_BIC
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x7, data = data, model = FALSE)
##
## Coefficients:
## (Intercept)           x1           x2           x7
##     5.71117     -0.00215     -0.03058     -0.59857
```

```
print("R-squared")
```

```
## [1] "R-squared"
```

```
indexRsq<-c(1:nmod)[all_mod_s$r.squared==max(all_mod_s$r.squared)]
indexRsq
```

```
## [1] 127
```

```
max_Rsq <- all_mod[[indexRsq]]
max_Rsq
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x7 + x8, data = data,
##     model = FALSE)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4           x5
##     8.68330     -0.00275     -0.03530      0.04252     -0.01268     -0.62963
##          x7           x8
##    -1.11429     -0.23308
```

```
#df <- c(1:nmod)[all_mod_s$df==min(all_mod_s$df)]
```

Well now we see that different model criterion are directing us to different linear regression models. So it is up to us now to decide which one is the best out of the following models

- adj_r_sq model: lm(formula = y ~ x1 + x2 + x3 + x5 + x7, data = cat, model = FALSE)
- log_lik_hood model: lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x7 + x8, data = cat, model = FALSE)
- neg_aic model: lm(formula = y ~ x1 + x2 + x3 + x5 + x7, data = cat, model = FALSE)
- neg_bic model: lm(formula = y ~ x1 + x2 + x7, data = cat, model = FALSE)
- r_sq model: lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x7 + x8, data = cat, model = FALSE)

If we observe these models and their graphs we can appreciate that `log_lik_hood` and `r_sq` , always are just blindly increasing with the number of regressors as they go up. And this is because of how they are defined in terms of their computation, since these statistics can never go down regardless of whatever regressor we add on (i.e we may choose to add on a completely useless regressor and the statistic will at worst not increase), and if we appreciate the upward trend in the graphs of these models we see they stop increasing in a meaningful way after 4 regressors and at that point it is only ever increasing little by little, this is why we believe the r_sq and log_lik_hood are not the best statistics to consider and we will overlook them.

That leaves us with adj_r_sq, neg_aic, and neg_bic to consider. Of which the adj_r_sq model and neg_aic model are the same. So really we can think of being left with just 2 models to consider which are the following.

- `m1 = lm(formula = y ~ x1 + x2 + x3 + x5 + x7, data = cat, model = FALSE)`
- `m2 = lm(formula = y ~ x1 + x2 + x7, data = cat, model = FALSE)`

The only difference between these two is the inclusion of regressors x3 and x5. And if we look at the individual significance of these regressors we will find that they are both individually insignificant in the regression model.

term	estimate	std.error	statistic	p.value
(Intercept)	8.170	1.549	5.28	0.000
x1	-0.003	0.001	-2.84	0.009
x2	-0.036	0.014	-2.60	0.015
x3	0.040	0.024	1.68	0.104

term	estimate	std.error	statistic	p.value
x5	-0.692	0.531	-1.30	0.203
x7	-1.108	0.358	-3.09	0.005

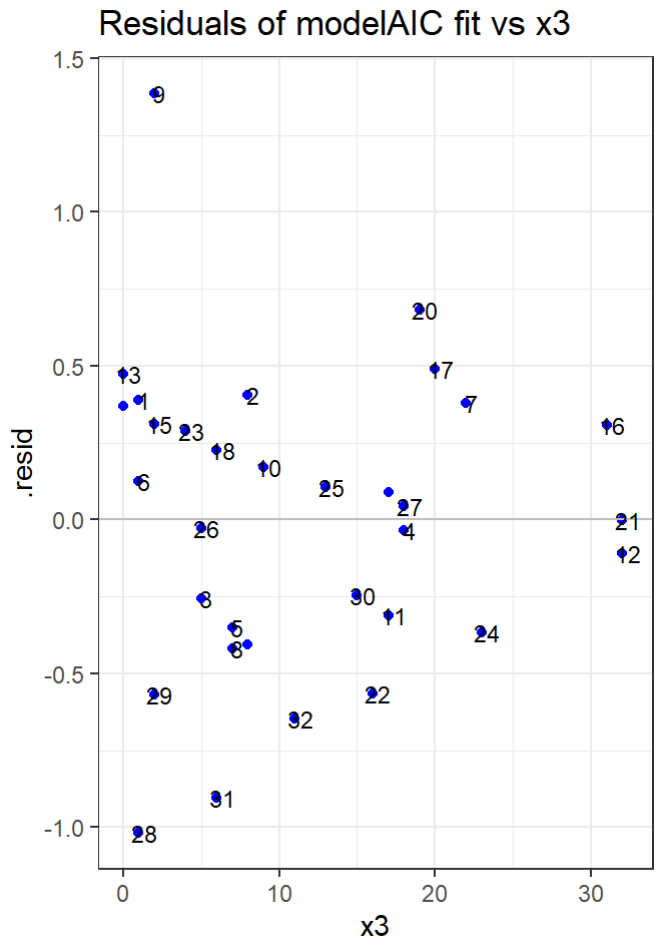
from this we appreciate the p-value for x3 is `0.104` this means it is individually insignificant and similarly if we consider x5 our p-value for x5 is `0.203` this means again that x5 is individually insignificant.

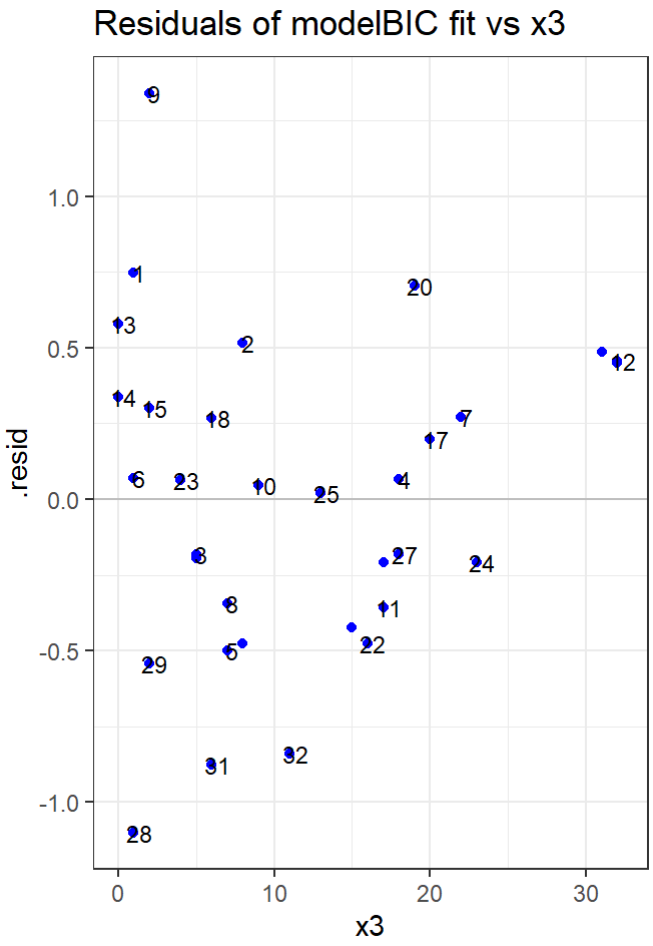
```
tidy(m2)%>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

term	estimate	std.error	statistic	p.value
(Intercept)	5.711	1.040	5.49	0.000
x1	-0.002	0.001	-2.65	0.013
x2	-0.031	0.014	-2.17	0.039
x7	-0.599	0.194	-3.09	0.004

In this model we can appreciate that no regressors are individually insignificant (that is that they all appear to be significant).

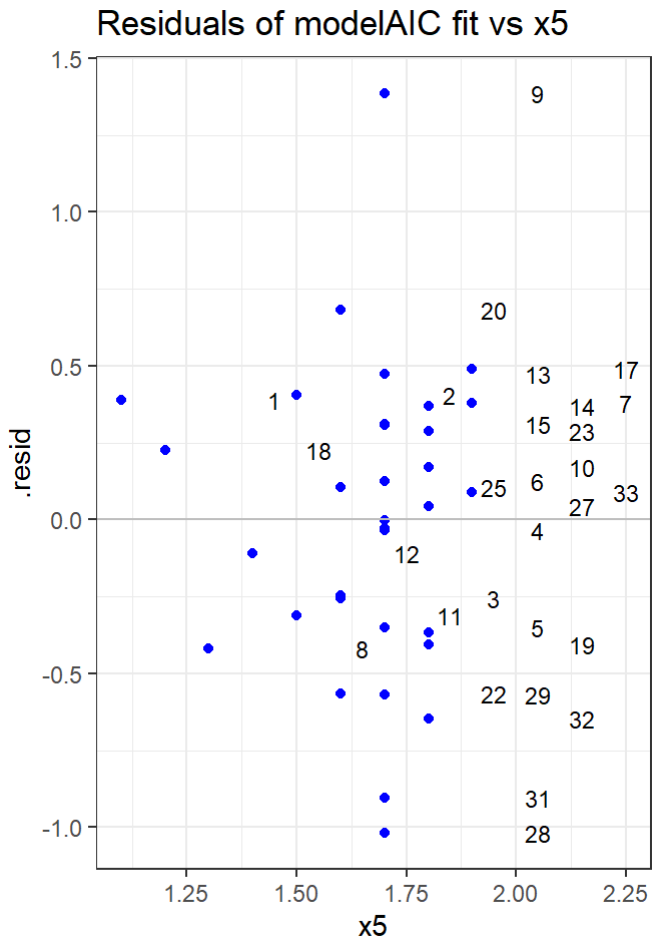
We should also consider looking at the residual graphs of x3 and x5 against the residual errors of m1 and m2 doing so will yield the following.

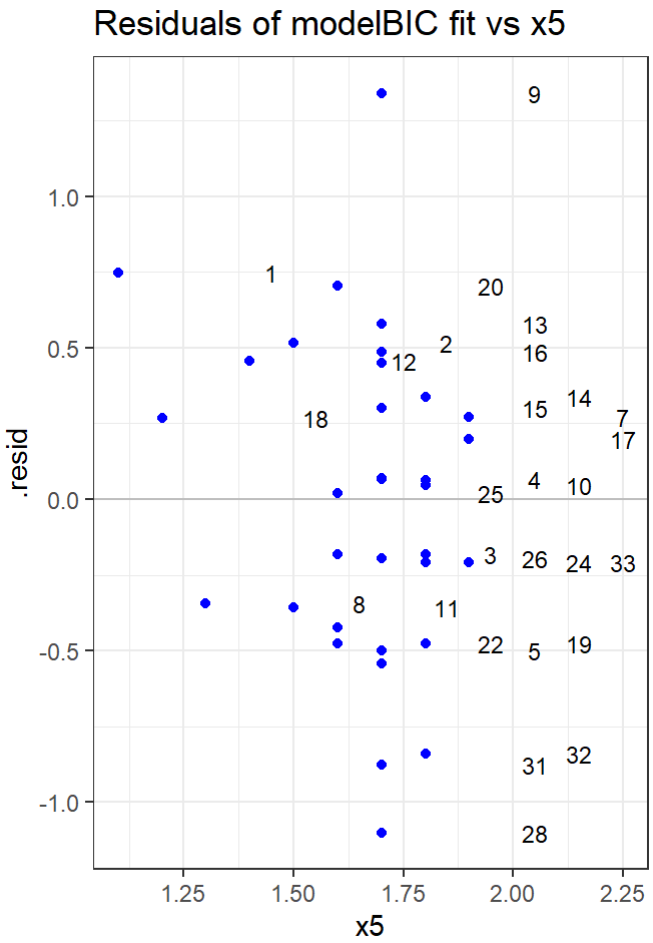




From this both model 1 and model 2 residuals when plotted against the x3 regressor seem to be randomly distributed with no apparent pattern in the residuals. And it is difficult to say 1 fits better then the other.

```
p2.modelAIC <-  aug_modelAIC %>% ggplot(aes(x=x5, y=.resid)) +
  geom_point(colour="blue") +
  geom_hline(yintercept=0,colour="grey") +
  ggtitle("Residuals of modelAIC fit vs x5") +
  theme(plot.title = element_text(size = 8)) +
  theme_bw()  +
  geom_text(label=aug_modelAIC$rowid, nudge_x = 0.35, colour="black",
            size=3, check_overlap = T)

p2.modelBIC <- aug_modelBIC %>% ggplot(aes(x=x5, y=.resid)) +
  geom_point(colour="blue") +
  geom_hline(yintercept=0,colour="grey") +
  ggtitle("Residuals of modelBIC fit vs x5") +
  theme(plot.title = element_text(size = 8)) +
  theme_bw() +
  geom_text(label=aug_modelBIC$rowid, nudge_x = 0.35, colour="black",
            size=3, check_overlap = T)


grid.arrange(p2.modelAIC, p2.modelBIC, ncol=2)
```

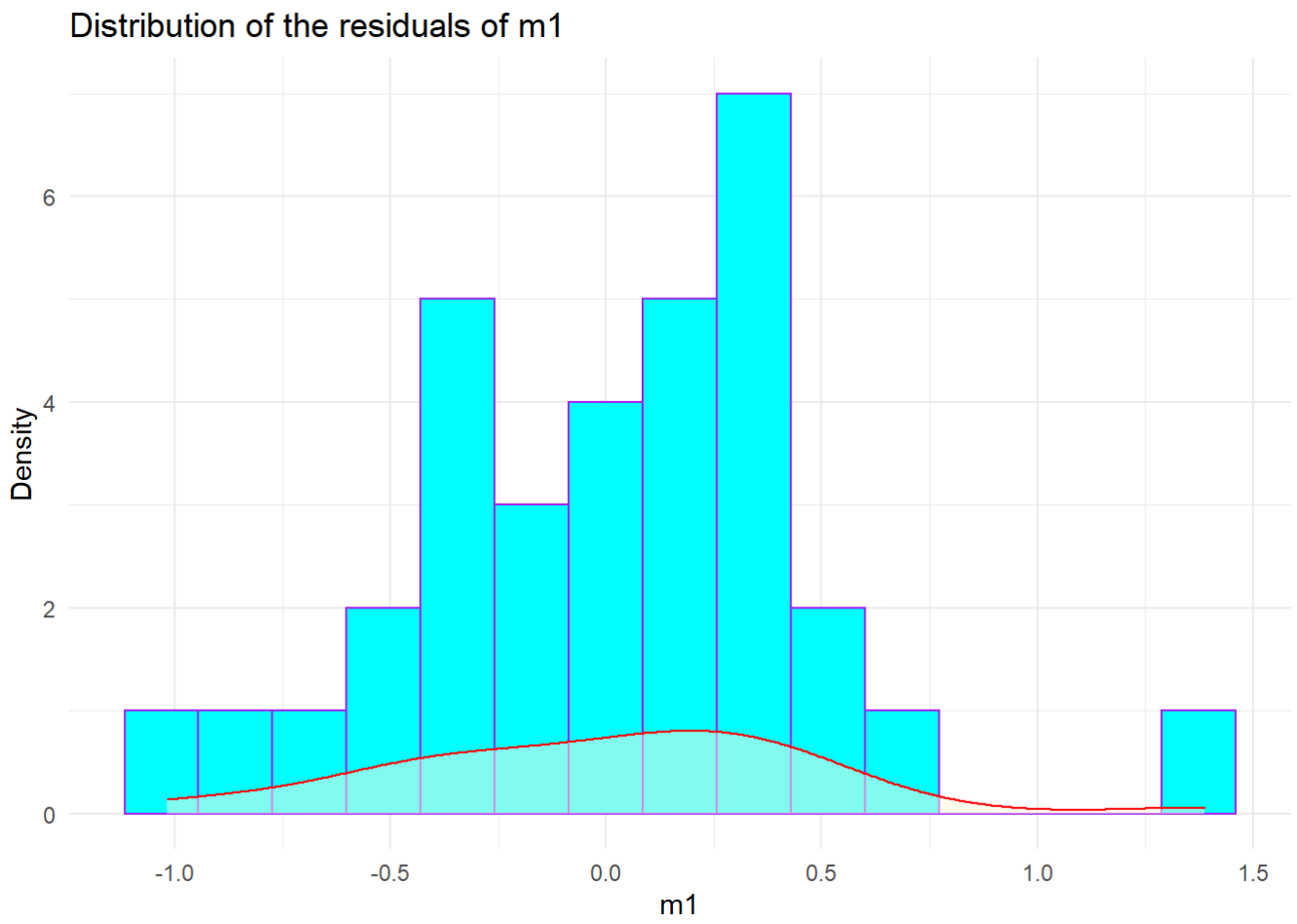




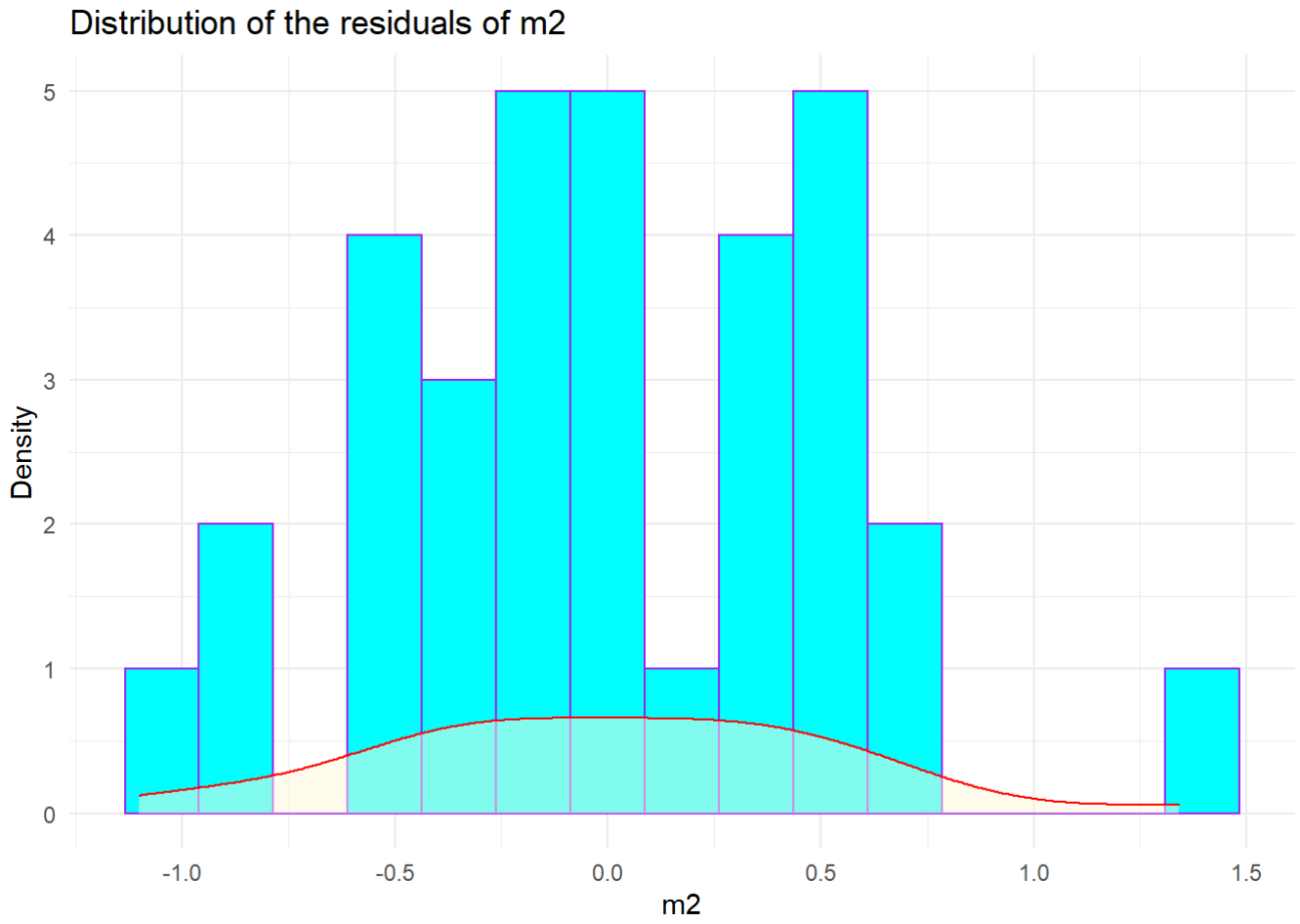
Now in this residual graph we can appreciate that x5 is a poorly recorded regressor. And upon further investigation of the data it appears x5 is actually `orientation of the area (from 1 if southbound to 2 otherwise)` this has been poorly recorded because many values are behaving like they are discrete (all the values that this regressor may take are 1,1.1,1.2,1.3,...,1.9,2), and because of this clear patterns have formed on both the residual graphs. So this is a good case for not considering x5 as it ought to have been a dummy variable and not recorded the way that it has been. This also explains why perhaps x5 has the highest p-value !, because it ought to have been broken down into dummy variables that we could add onto our model.

Now let us also compare the distribution of the residuals of the two models against each other. In a histogram, and QQplot to see which more closely fits a normal distribution, and that would be an indication of which errors are better distributed.

```
df_of_resids <- tibble(m1 = augment(m1)$.resid)

ggplot(data =df_of_resids,aes(x= m1))+
  geom_histogram(bins = 15,colour = 'purple',fill='cyan') +
  ggtitle('Distribution of the residuals of m1') +
  geom_density(fill = 'cornsilk',alpha = 0.5,colour = 'red') +
  ylab('Density') +
  theme_minimal()
```

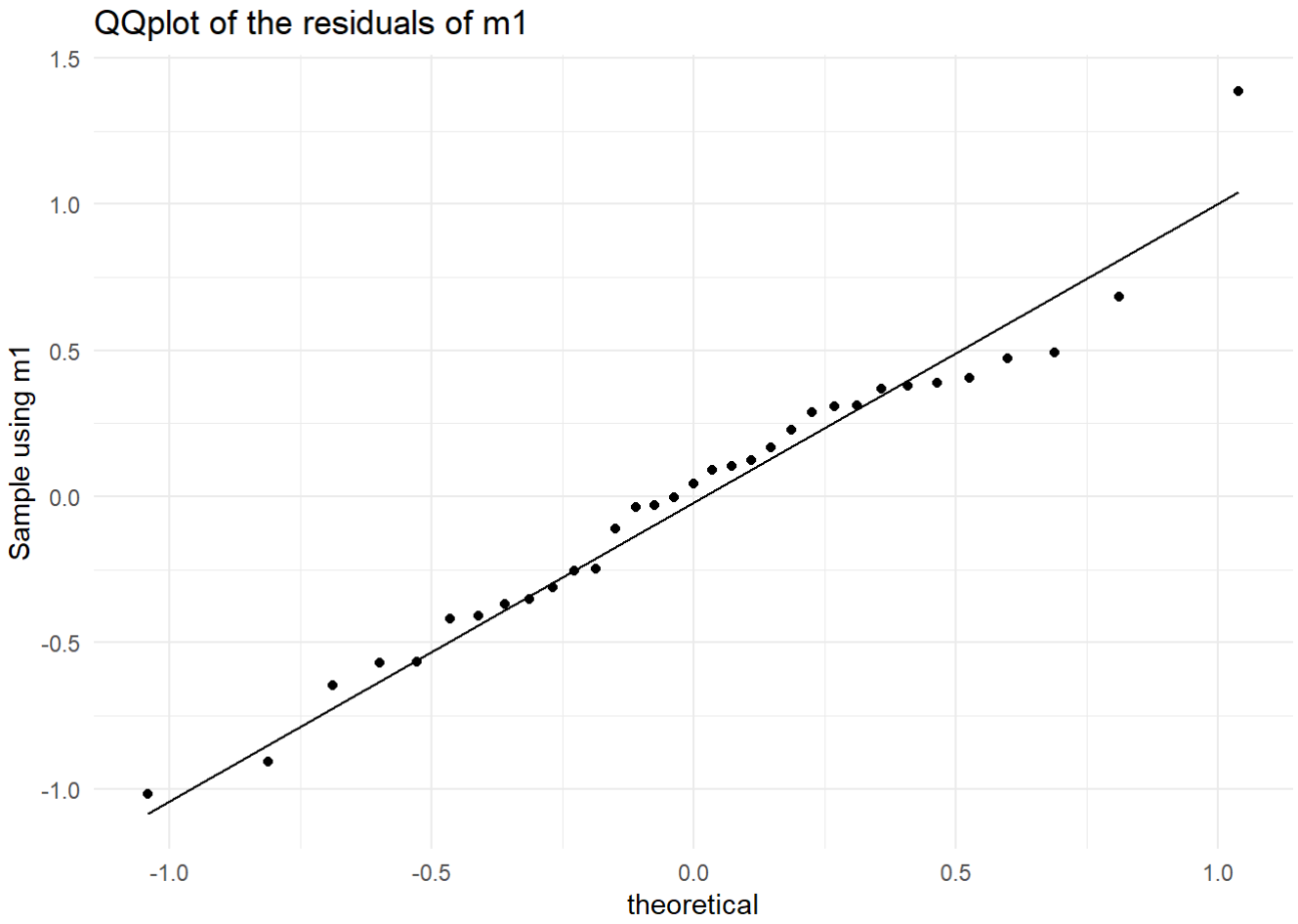



```
df_of_resids <- tibble(m2 = augment(m2)$.resid)

ggplot(data =df_of_resids,aes(x= m2))+
  geom_histogram(bins = 15,colour = 'purple',fill='cyan')+ ggtitle('Distribution of the resid
uals of m2') +
  geom_density(fill = 'cornsilk',alpha = 0.5,colour = 'red') +
  ylab('Density')+
  theme_minimal()
```

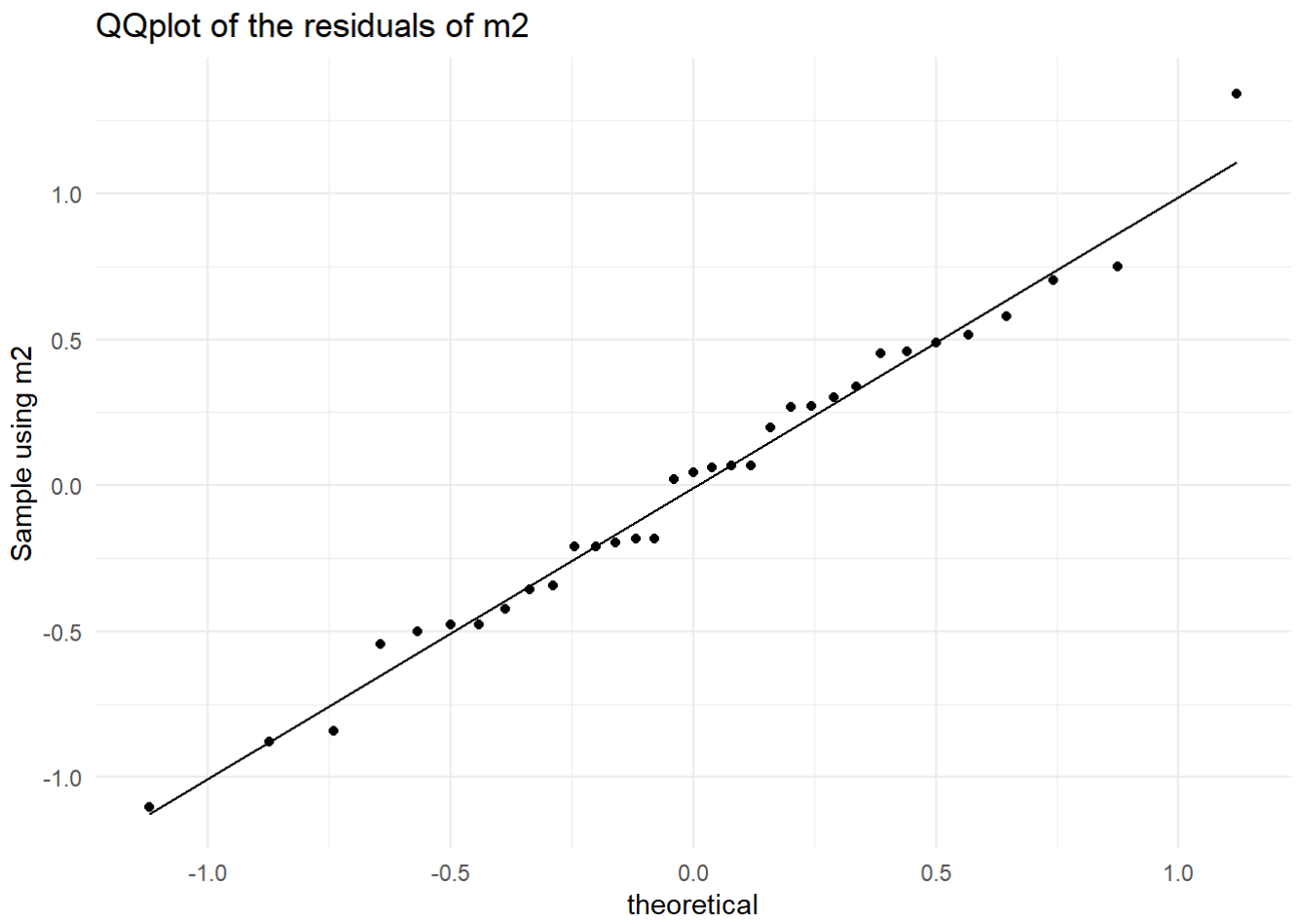


From these 2 histograms the histogram of the distribution of errors for m2 seems to ever so slightly better (more symmetric) where as m1 seems to be positively skewed. Though it is worth acknowledging that both are quite poor because of our tiny sample size.

```
df_of_resids <- tibble(m1 = augment(m1)$.resid,m2 = augment(m2)$.resid)
normal_m1_resids <- tidy(fitdistr(df_of_resids$m1,"Normal"))
normal_m2_resids <- tidy(fitdistr(df_of_resids$m2,"Normal"))

params1 <- normal_m1_resids$estimate

ggplot(data = df_of_resids, aes(sample = m1)) + stat_qq(distribution = qnorm,dparams = params
1 )+
  stat_qq_line(distribution = qnorm, dparams = params1) +
  ggtitle('QQplot of the residuals of m1') +
  ylab('Sample using m1')+
  theme_minimal()
```




```
params2 <- normal_m2_resids$estimate

ggplot(data = df_of_resids, aes(sample = m2)) + stat_qq(distribution = qnorm,dparams = params
2 )+
  stat_qq_line(distribution = qnorm, dparams = params2) +
  ggtitle('QQplot of the residuals of m2') +
  theme_minimal()+
  ylab('Sample using m2')
```



Now from these two QQplots, they both appear to be quite decent fits to the normal distribution however we believe that the second QQplot (plot of the residuals of m2) is superior to the first QQplot (plot of the residuals of m1) and this is because if you consider the values of the theoretical quantiles, yes it the case that the QQplot of m1 fits better at smaller quantiles but we see that at really large quantiles their is gross inaccuracy where as the QQplot for the residuals for m2 remains consistent throughout. This paired with what we saw in the histograms gives us confidence in believing that m2 is superior to m1 thus far.

Now lastly we will do a F-test to test the joint insignificance of the regressors x3 and x5

H_0: x3 = x5 = 0 H_1: At least 1 of x3 or x5 != 0

```
tidy(anova(m2,m1))%>%
  kable() %>%
  kable_styling()
```

res.df	rss	df	sumsq	statistic	p.value
29	8.82	NA	NA	NA	NA
27	7.60	2	1.22	2.17	0.134

So with a reported p-value of 0.13 we cannot reject the null hypothesis which is to say that x3 and x5 are, as far as we can tell from this sample, jointly insignificant regressors.

So in conclusion it appears that x3 and x5 are both individually and jointly insignificant regressors, and that the residual plot against these 2 regressors indicates that x5 is a regressor that ought to have been encoded as a set of binary dummy variables. Thus we say our preferred model is model 2 which is `lm(y ~ x1 + x2 + x7)`.

```
modelp <- tidy(m2)
```

Question 15

The estimated final form of our preferred equation is

```
y_hat = 5.71 + -0.00215 * x1 + -0.03058 * x2 +  -0.59857    * x7
```

that means the estimated effect of the first regressor (Altitude in meters) on the log average nests of caterpillars per tree is -0.00215, which is to say for each 1 extra meter we climb in altitude we would anticipate that the expected log average of the number of nests per tree decreases by 0.03058.

Similarly for the second regressor (which is that the tree is on in degrees) for every 1 extra degree we would expect that the log average of the number of nests per tree decreases by 0.00215

and lastly for the 7th regressor (number of vegetation strata) for each extra vegetation strata that we climb up in (so each vertical strata we higher up in) the expected log average of the number of nests per tree decreases by a whopping 0.59857.

From these 3 estimates we can appreciate that vegetation strata has the highest effect on log average of the number of nests per tree. And this makes sense because food security would for caterpillars like any other creature that builds nests in trees be a big factor when deciding to nest.

and if we are at 0 altitude (x1 = 0) on perfectly flat ground (x2 = 0) and their is no vegetation strata (no vertical layering of vegetation, so most likely that means it's on ground level, which makes sense because we are perfectly flat ground) we would anticipate the expected number of log average of the number of nests per tree to be 5.71.

and the following is our estimated 95% confidence intervals for each of the beta's

```
b_0_ci_lower <- modelp$estimate[1] - (2.05 * modelp$std.error[1])
b_0_ci_upper <- modelp$estimate[1] + (2.05 * modelp$std.error[1])

b_1_ci_lower <- modelp$estimate[2] - (2.05 * modelp$std.error[2])
b_1_ci_upper <- modelp$estimate[2] + (2.05 * modelp$std.error[2])

b_2_ci_lower <- modelp$estimate[3] - (2.05 * modelp$std.error[3])
b_2_ci_upper <- modelp$estimate[3] + (2.05 * modelp$std.error[3])

b_3_ci_lower <- modelp$estimate[4] - (2.05 * modelp$std.error[4])
b_3_ci_upper <- modelp$estimate[4] + (2.05 * modelp$std.error[4])
c_i <- tibble(beta = c(0,1,2,7), lower = c(b_0_ci_lower,b_1_ci_lower,b_2_ci_lower,b_3_ci_lowe
r), upper = c(b_0_ci_upper,b_1_ci_upper,b_2_ci_upper,b_3_ci_upper ))

c_i%>%
  kable() %>%
  kable_styling()
```

beta	lower	upper
0	3.579	7.843
1	-0.004	0.000
2	-0.060	-0.002

beta	lower	upper
7	-0.996	-0.201

We are 95% confident that each of the true beta_0,beta_1, beta_2, and beta_7 coefficient values in the population will fall in between the above reported values. So the true effect of each of the beta's lies somewhere in those intervals.

```
resid_errors_tibble <- tibble("standard error of residuals" = augment(m2)$.std.resid)

resid_errors_tibble%>%
  kable() %>%
  kable_styling()
```

standard error of residuals
1.419
0.956
-0.341
0.123
-0.929
0.129
0.516
-0.717
2.574
0.090
-0.758
0.947
1.162
0.653
0.565
0.967
0.396
0.492
-0.898

standard error of residuals
1.349
0.938
-0.891
0.116
-0.419
0.043
-0.365
-0.358
-2.100
-1.033
-0.852
-1.647
-1.632
-0.398

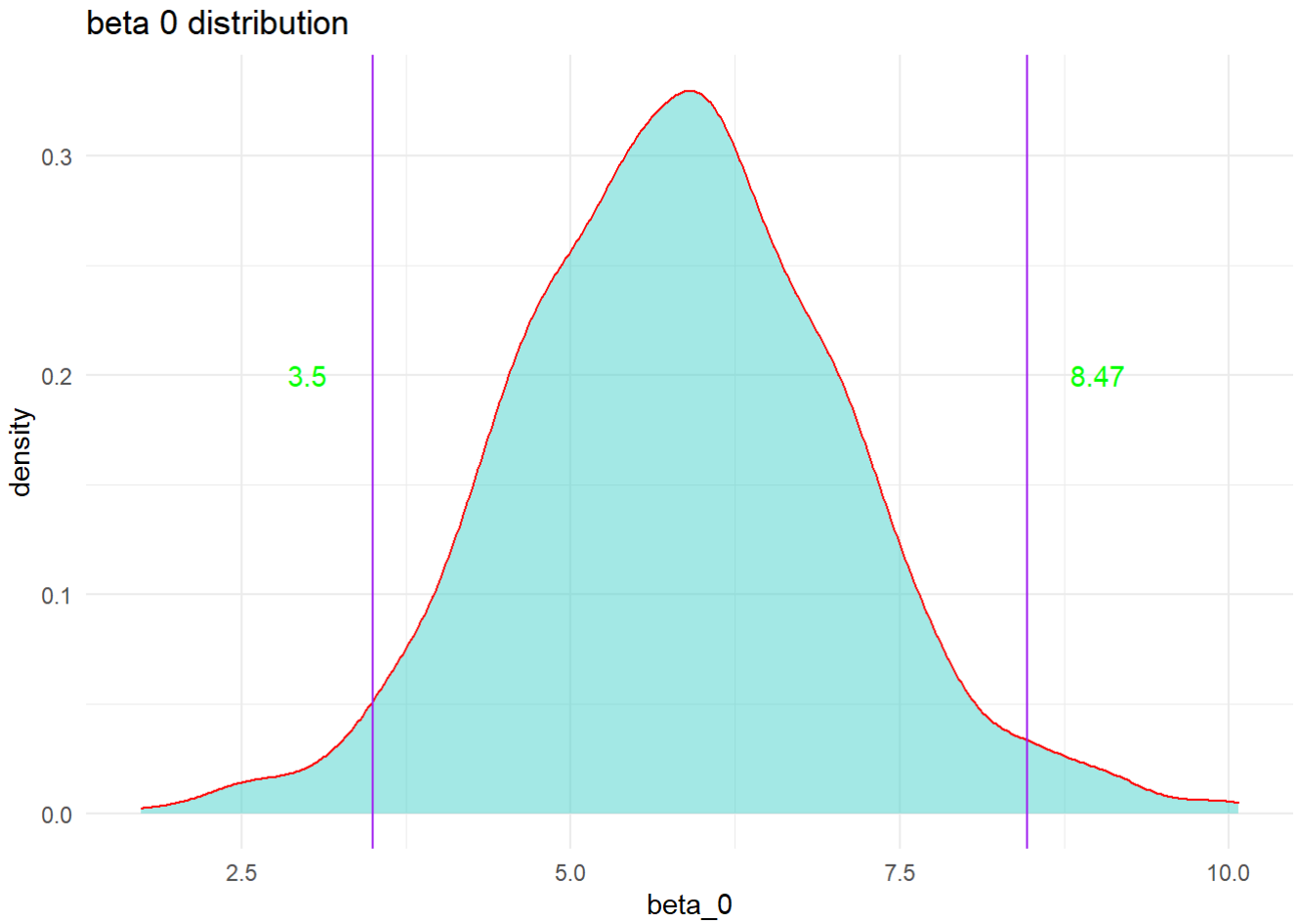
```
set.seed(1738)
modelp <- m2
R <- 1000
n <- nrow(cat)
R_coeffs <- tibble(b0 = rep(0, R), b1 = rep(0, R), b2 = rep(0,
R), b7 = rep(0, R))
set.seed(2020)
for (j in (1:R)) {
temp <- cat %>% slice_sample(n = n, replace = TRUE)
tempf <- lm(y ~ x1 + x2 +x7, data = temp)
tidyf <- tidy(tempf)
R_coeffs[j, ] <- t(tidyf$estimate)
}

R_coeffs%>%
  head() %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

b0	b1	b2	b7
5.50	-0.002	-0.037	-0.415
7.03	-0.003	-0.052	-0.267

b0	b1	b2	b7
5.08	-0.002	-0.029	-0.711
5.80	-0.002	-0.040	-0.485
5.54	-0.002	-0.022	-0.596
6.18	-0.003	-0.016	-0.688

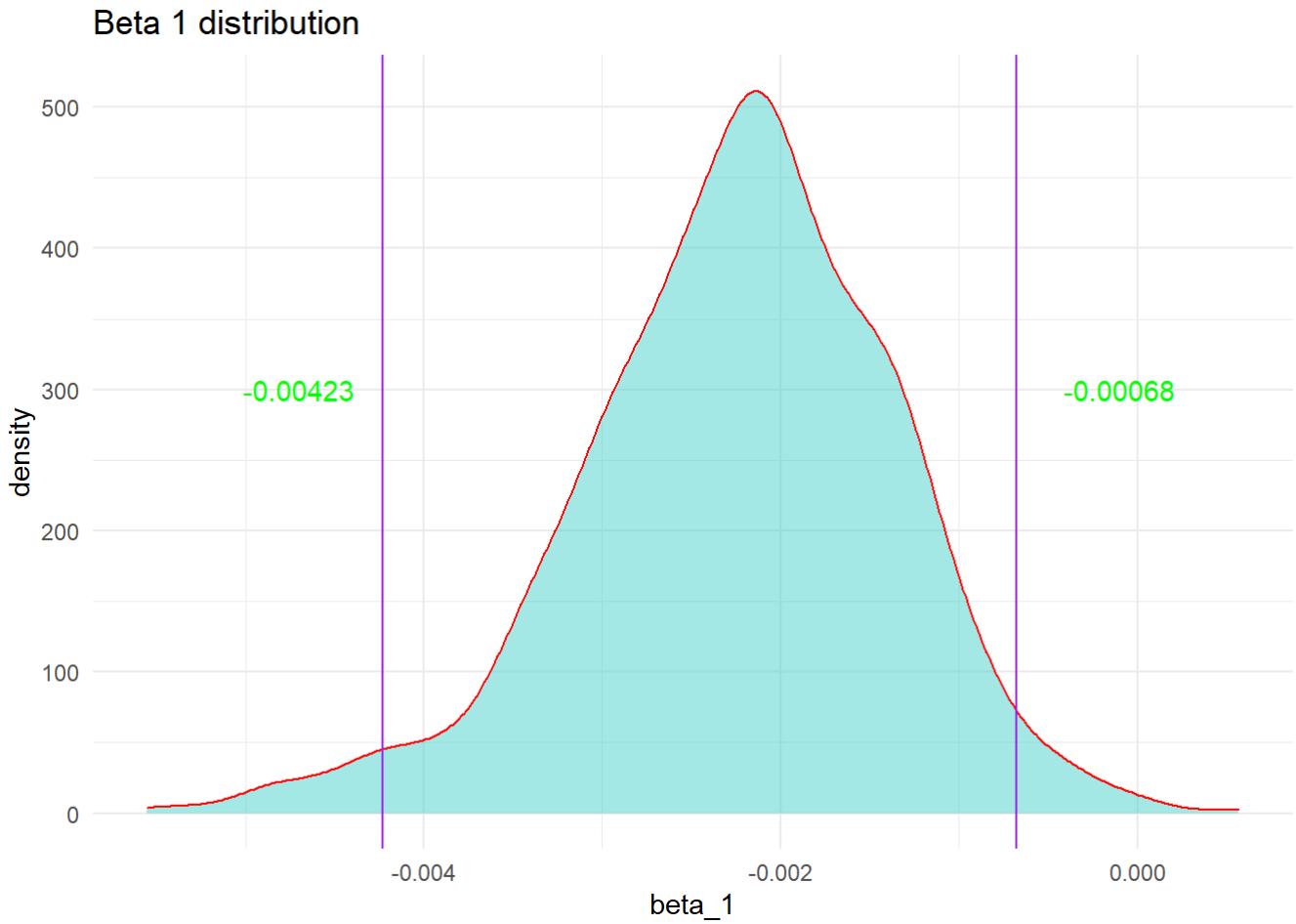
From the bootstrap sampling distribution of beta_0, we can say that we are 95% confident that the true population beta_0 will lie in between the interval `(3.5,8.47)`. Which is to say that when we are at 0 altitude (x1 = 0) on perfectly flat ground (x2 = 0) and their is no vegetation strata (no vertical layering of vegetation), the true number of log average number of nests per tree will be somewhere between 3.5 and 8.47.

```
b_0 <- tibble(beta_0 = R_coeffs$b0)

#quantile(R_coeffs$b0,c(0.025,0.975))

ggplot(data=b_0,aes(x=beta_0))+
  geom_density(colour = 'red', fill = "mediumturquoise", alpha =0.5)+
  geom_vline(xintercept = 3.5, colour = 'purple')+
  geom_vline(xintercept = 8.47, colour = 'purple')+
  ggtitle("beta 0 distribution")+annotate("text",
    label = 3.5, x = 3, y = 0.2,
    colour = "green")+
  annotate("text",
    label = 8.47, x = 9, y = 0.2,
    colour = "green")+
  theme_minimal()
```



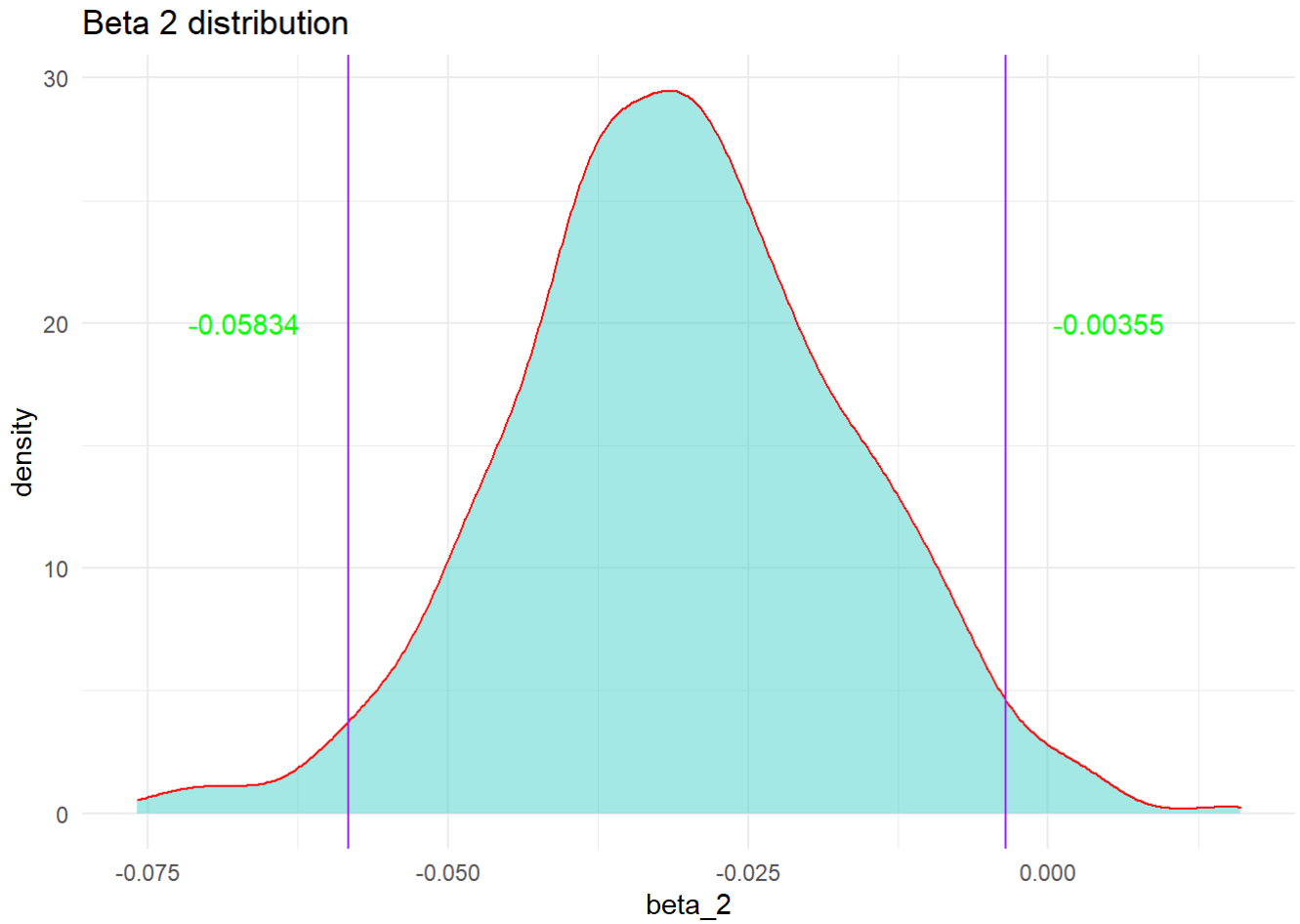
According to the bootstrap sampling distribution of the beta_1 values, we are 95% confident that the true beta_1 (effect of altitude on the log average number of nests per tree) will lie in between the interval `(-0.00423,-0.00068)` . Which is to say that the true effect that every extra meter of altitude has on the log average number of nests is going to make it decrease by within the interval `(-0.00423,-0.00068)` holding all else constant . And what this means then for the average number of nests per tree is that for every extra meter of altitude the average number of nests per tree is expected to decrease by -0.422, -0.068%, holding all else constant.

```
b_1 <- tibble(beta_1 = R_coeffs$b1)

#quantile(R_coeffs$b1,c(0.025,0.975))

ggplot(data=b_1,aes(x=beta_1))+

  geom_density(colour = 'red', fill = "mediumturquoise", alpha =0.5)+
  geom_vline(xintercept = -0.00423, colour = 'purple')+
  geom_vline(xintercept = -0.00068, colour ='purple')+
  ggtitle("Beta 1 distribution")+
  annotate("text",
    label = -0.00423, x = -0.0047, y = 300,
    colour = "green")+
  annotate("text",
    label = -0.00068, x = -0.0001, y = 300,
    colour = "green")+
  theme_minimal()
```



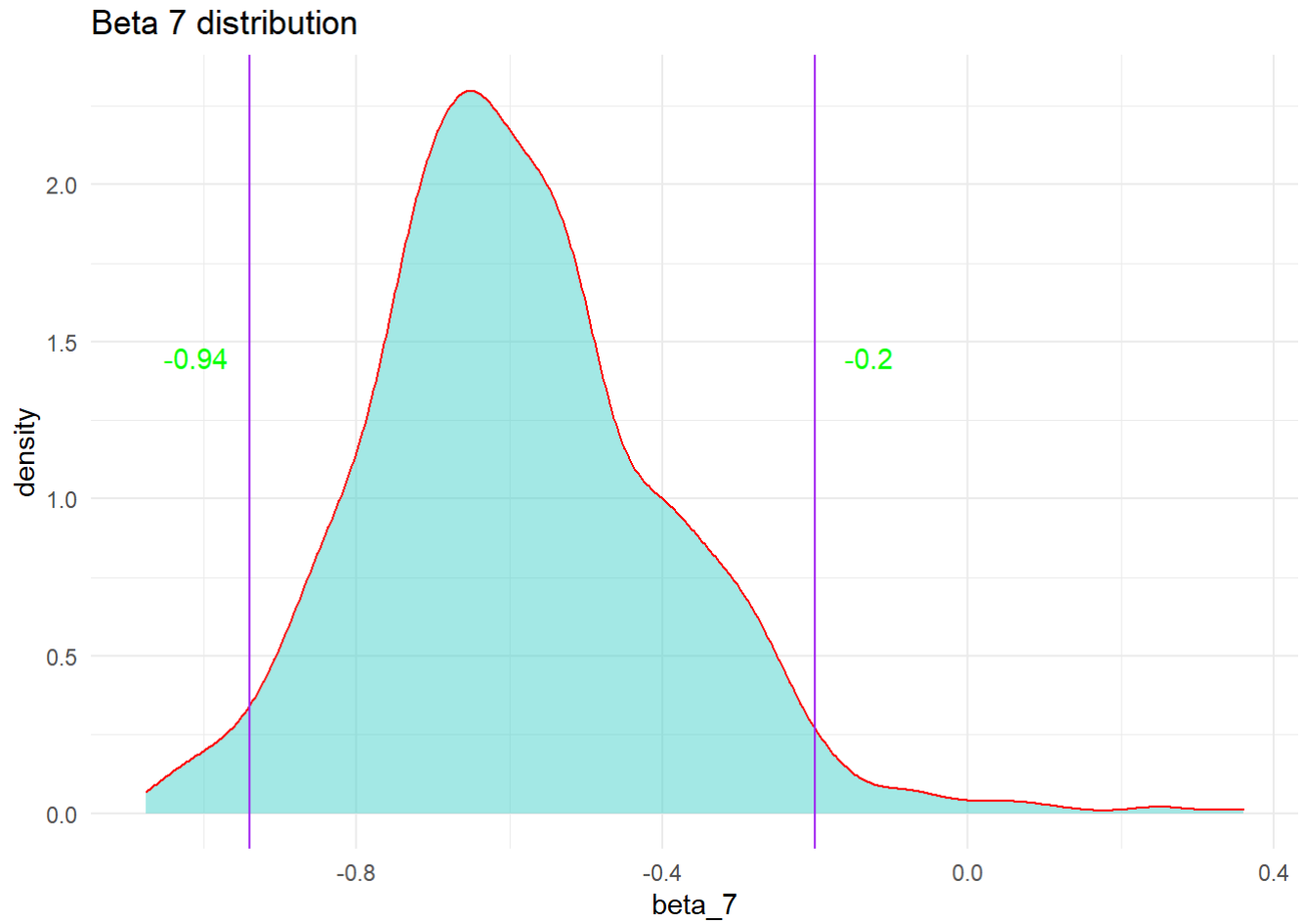
According to the bootstrap sampling distribution of the beta_2 values,we are 95% confident that the true beta_1 (effect of slope) will lie within the interval given by `(-0.05834,-0.00355)`. Which is to say that the true effect of the slope (in degrees) on the log average number of nests per tree will for every extra degree will decrease by some value contained within the interval `(-0.05834,-0.00355)` holding all else constant. And what this means for the average number of tree per 1 extra degree on the slope is that it will decrease by -5.667, -0.354% holding all else constant.

```
b_2 <- tibble(beta_2 = R_coeffs$b2)

#quantile(R_coeffs$b2,c(0.025,0.975))

ggplot(data=b_2,aes(x=beta_2))+

  geom_density(colour = 'red', fill="mediumturquoise", alpha = 0.5)+
  geom_vline(xintercept = -0.05834, colour = 'purple')+
  geom_vline(xintercept = -0.00355, colour = 'purple' )+
  ggtitle("Beta 2 distribution")+
  annotate("text",
    label = -0.05834, x = -0.067, y = 20,
    colour = "green")+
  annotate("text",
    label = -0.00355, x = 0.005, y = 20,
    colour = "green")+
  theme_minimal()
```



According to the bootstrap sampling distribution of beta_7, We are 95% confident that the true beta_7 (effect of the number of vegetation strata on the log average number of nests per tree) is contained within the interval given by `(-0.94,-0.2)`. Which is to say that for every extra vegetation strata the log average number of trees decreases by some number contained in the interval `(-0.94,-0.2)` holding all else constant. And what this means in general for average number of trees per nest is that for every 1 extra vegetation strata it will decrease by -60.937, -18.127% holding all else constant.

```
b_7 <- tibble(beta_7 = R_coeffs$b7)

#quantile(R_coeffs$b7,c(0.025,0.975))

ggplot(data=b_7,aes(x=beta_7))+

  geom_density(colour = 'red', fill = 'mediumturquoise', alpha = 0.5)+
  geom_vline(xintercept = -0.94, colour = 'purple')+
  geom_vline(xintercept = -0.20,colour = 'purple')+
  ggtitle('Beta 7 distribution')+
  annotate("text",
    label = -0.20, x = -0.2+0.07, y = 1.45,
    colour = "green")+
  annotate("text",
    label = -0.94, x = -0.94-0.07, y = 1.45,
    colour = "green")+
  theme_minimal()
```


Beta 7 distribution
density
2.0
1.5
1.0
0.5
0.0
-0.94
-0.2
-0.8
-0.4
0.0
0.4
beta_7

000