# CS109b Data Science 2: Midterm Exam

March, 2018

*Paul M. Washburn*

```
knitr::opts_chunk$set(echo=TRUE, message=FALSE, warning=FALSE)
library(factoextra)
library(cluster)
library(tidyverse)
library(tibble)
library(gam)
library(stringr)
library(splines)
library(tidyverse)
library(ggthemes)
library(ggplot2)
library(bayesplot)
library(reshape2)
library(rstan)

`%+%` <- function(a, b) paste0(as.character(a), as.character(b))
`%!in%` <- function(a, b) !(a %in% b)

base_dir <- 'C:/Users/paulm/Desktop/Harvard/CS109b/Midterm/'#'/Users/pmw/Documents/Harv

wine <- read.csv(base_dir %+% 'winequality-red.csv')
predictor_cols <- names(wine)[names(wine) != 'quality']
print(predictor_cols)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"
```

```
summary(wine)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
```

```
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median :0.07900   Median :14.00       Median : 38.00
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00
##     density             pH          sulphates         alcohol
##  Min.   :0.9901   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :0.9968   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9967   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0037   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.636
##  3rd Qu.:6.000
##  Max.   :8.000
```

## Please include at the top of your exam whether you are a 109b/121b student, or a 209b student

This exam involves exploring a data set on red wine quality, and how quality relates to physio-chemical features of wine. A description of the study can be downloaded from the **publisher's web site.** The following questions will focus on a subset of data consisting of 1599 red wines. The data are contained in the file `winequality-red.csv`. For each bottle of wine, the following features are measured.

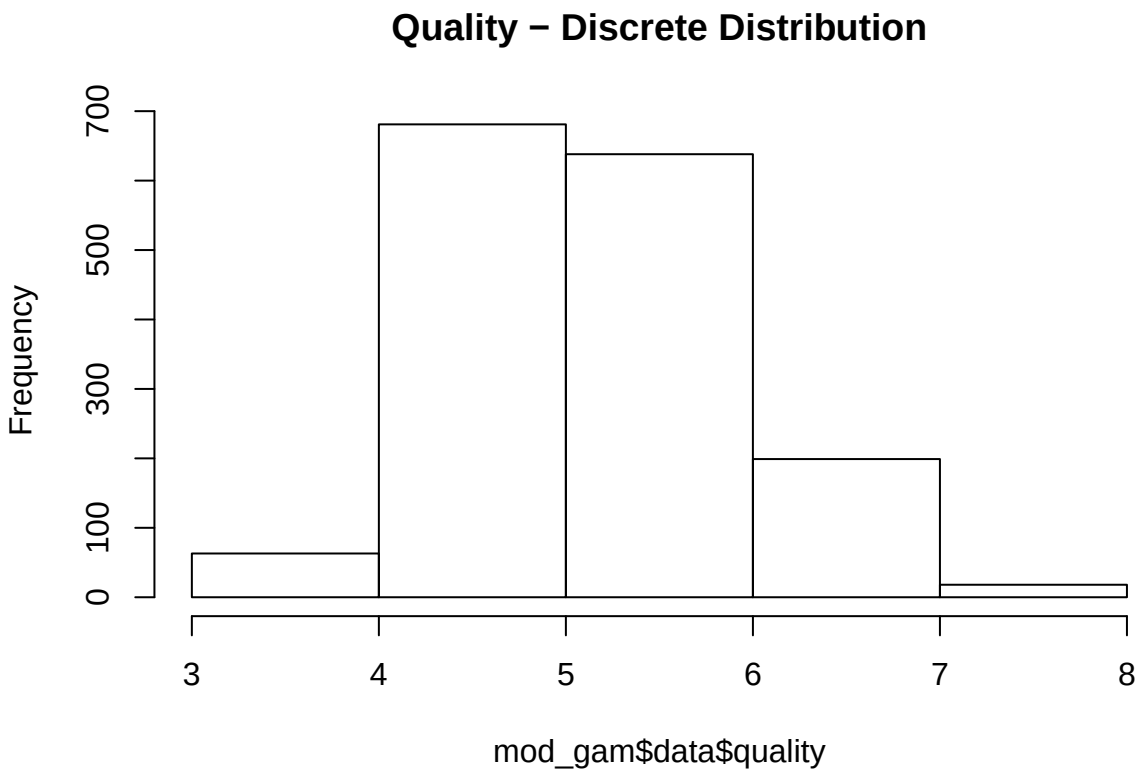
1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol
12. quality (score between 0 and 10)

The main goal of this analysis is to predict red wine quality from the physio-chemical features.

## Problem 1 [30 points]

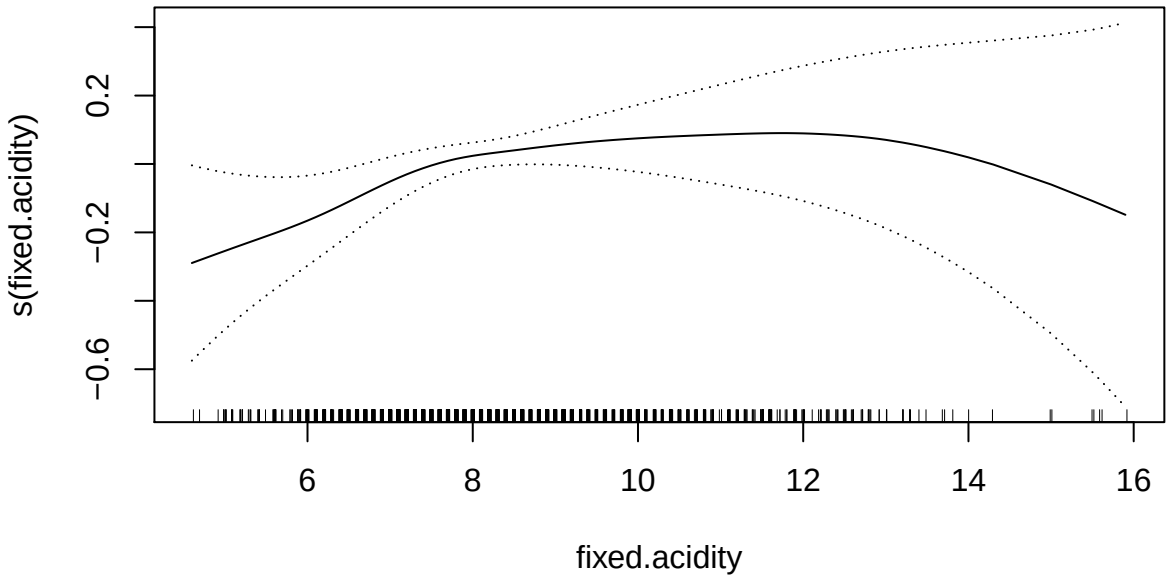
Fit an additive model of quality on the physio-chemical variables on all 1599 wines in the data set. Use smoothing splines to fit each predictor variable. No need to explicitly perform cross-validation - please use the default smoothing selections.

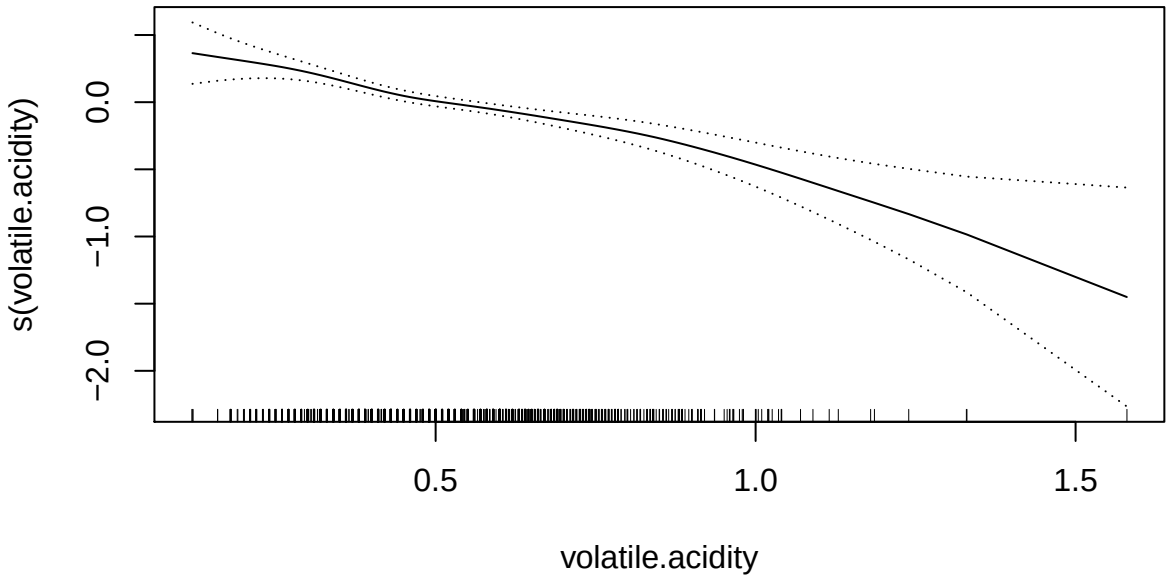
```
## GOAL: PREDICT RED WINE QUALITY FROM PHYSIO-CHEMICAL FEATURES

rsq <- function(y, y_pred) {
  # derives r-squared without the model # Input:
  # y: actual labels
  # predict: predicted labels
  # Output:
  # r_squared: R-squared
  tss <- sum((y - mean(y))^2)
  rss <- sum((y - y_pred)^2)
  rsq <- 1 - rss / tss
  rsq <- max(0, rsq)
  return(rsq)
}

# get periods out of names
mod_gam <- gam(quality ~ s(fixed.acidity) + s(volatile.acidity) +
                 s(citric.acid) + s(residual.sugar) + s(chlorides) +
                 s(free.sulfur.dioxide) + s(total.sulfur.dioxide) +
                 s(density) + s(pH) + s(sulphates) + s(alcohol), data=wine)

hist(mod_gam$data$quality,
     main='Quality - Discrete Distribution',
     breaks=6)
```

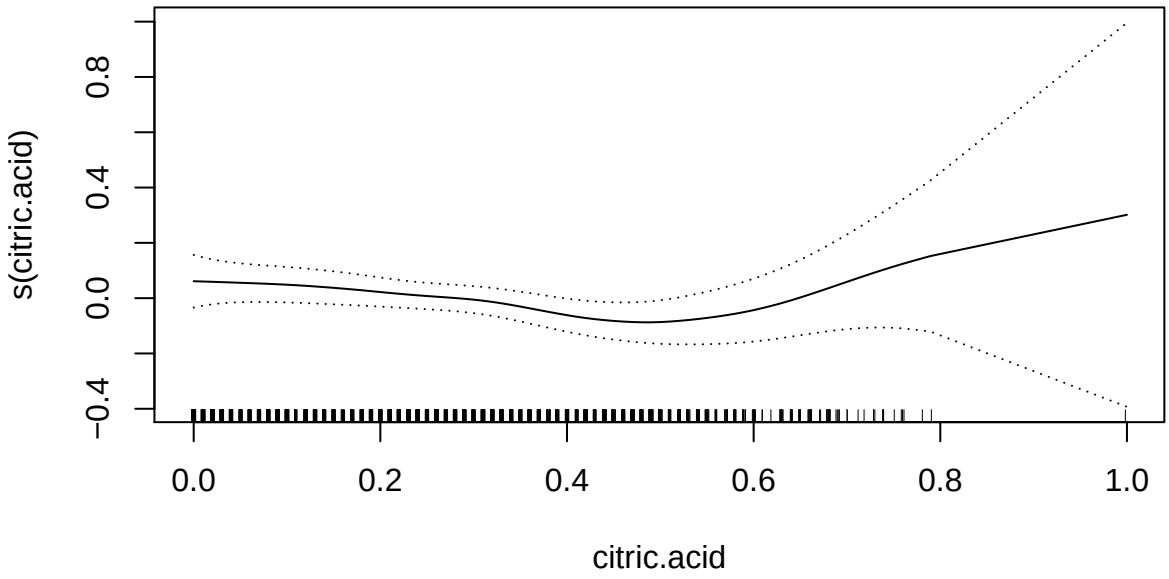


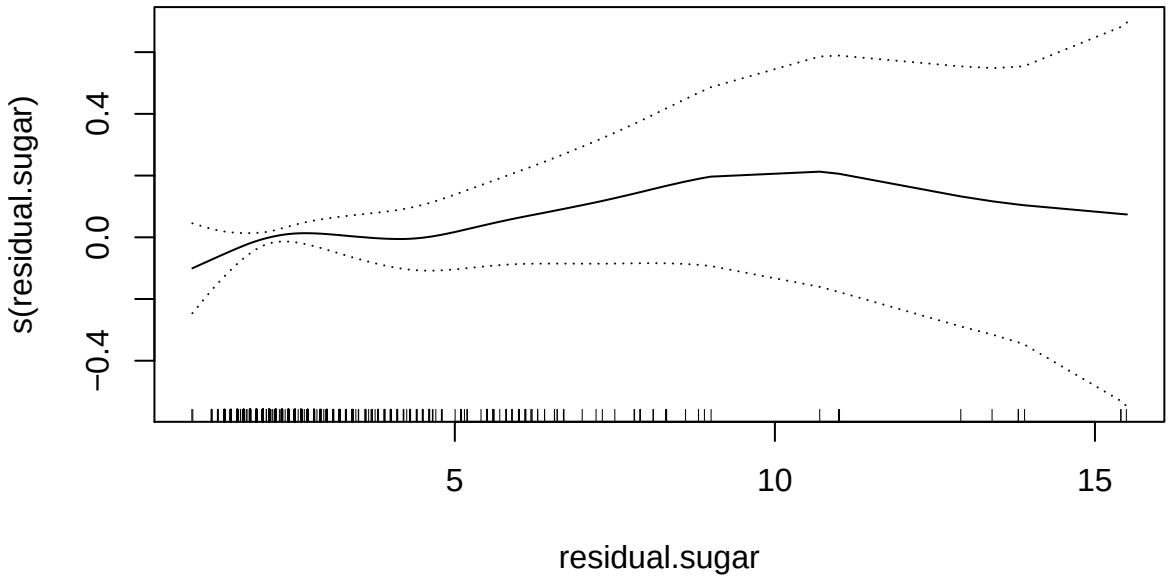

(a) [5 points] Plot the smooth of each predictor variable with standard error bands. Which variables seem to have a non-linear contribution to mean quality?

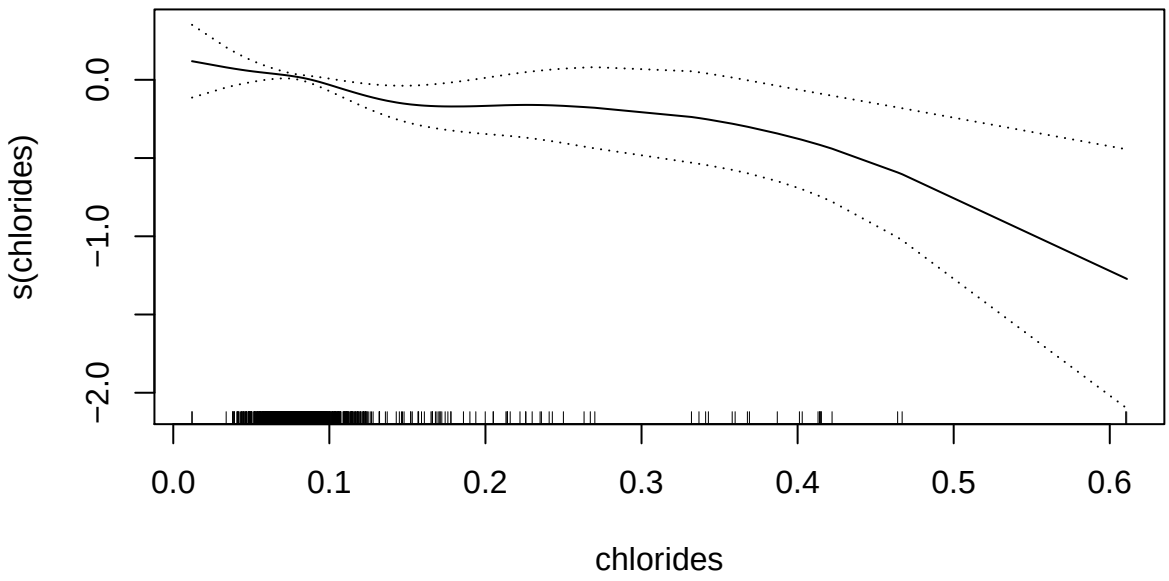
```
plot(mod_gam, rug=TRUE, se=TRUE)
```

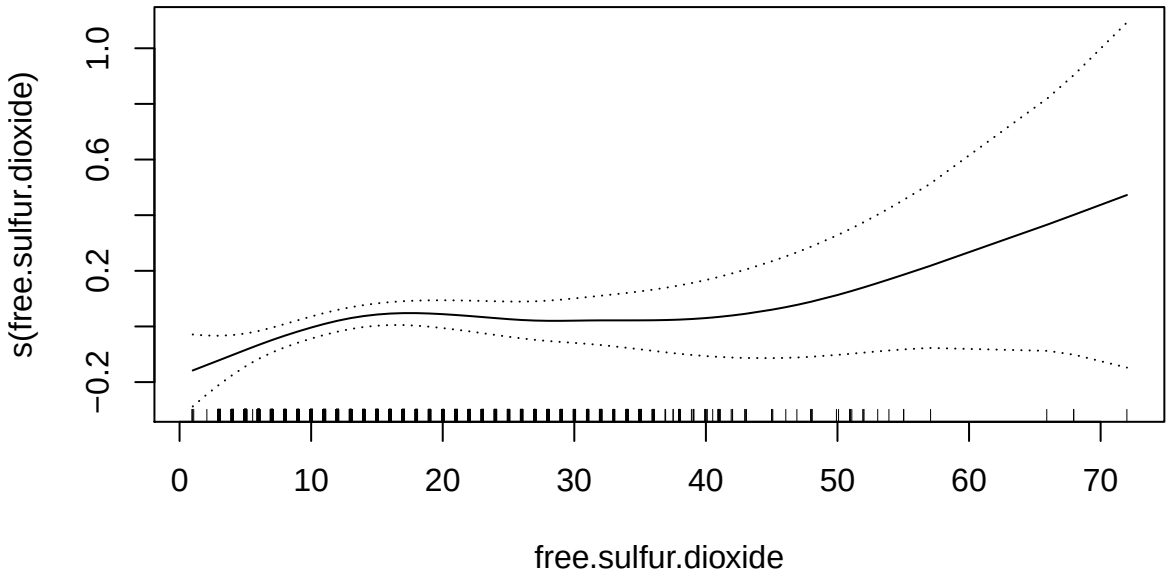

s(fixed.acidity)
0.2
−0.2
−0.6
6
8
10
12
14
16
fixed.acidity

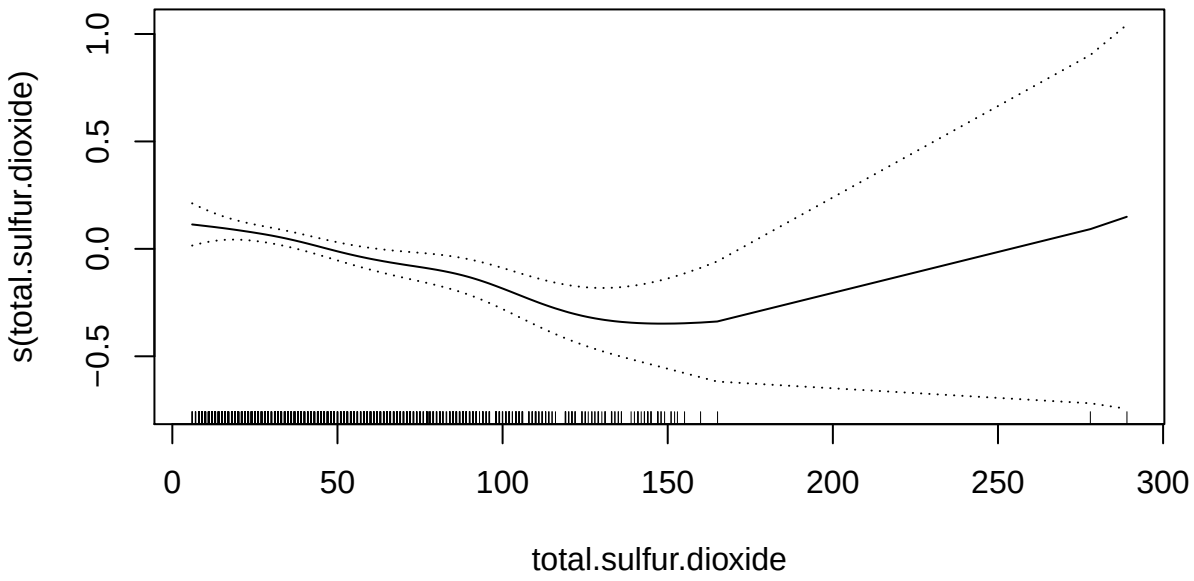

s(volatile.acidity)
0.0
−1.0
−2.0
0.5
1.0
1.5
volatile.acidity

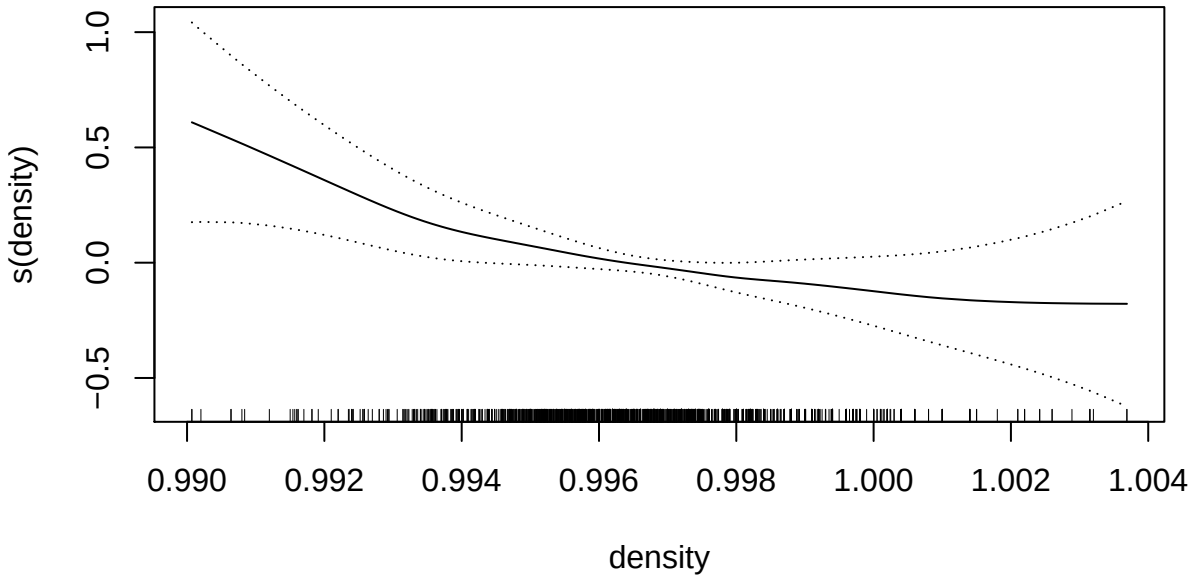

s(citric.acid)
0.8
0.4
0.0
−0.4
0.0
0.2
0.4
0.6
0.8
1.0
citric.acid

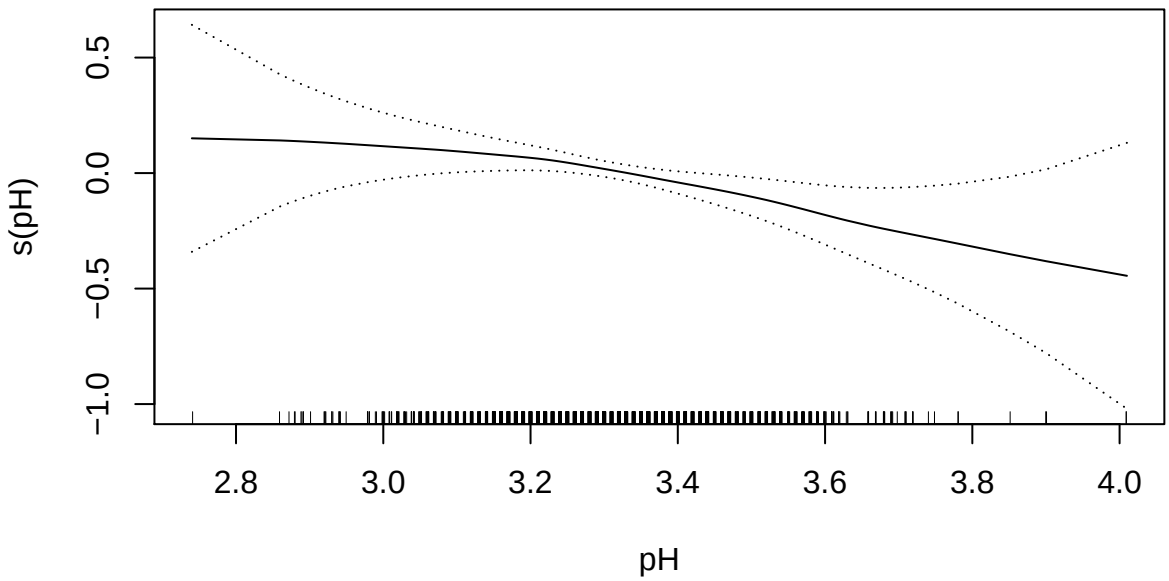

s(residual.sugar)
0.4
0.0
−0.4
5
10
15
residual.sugar

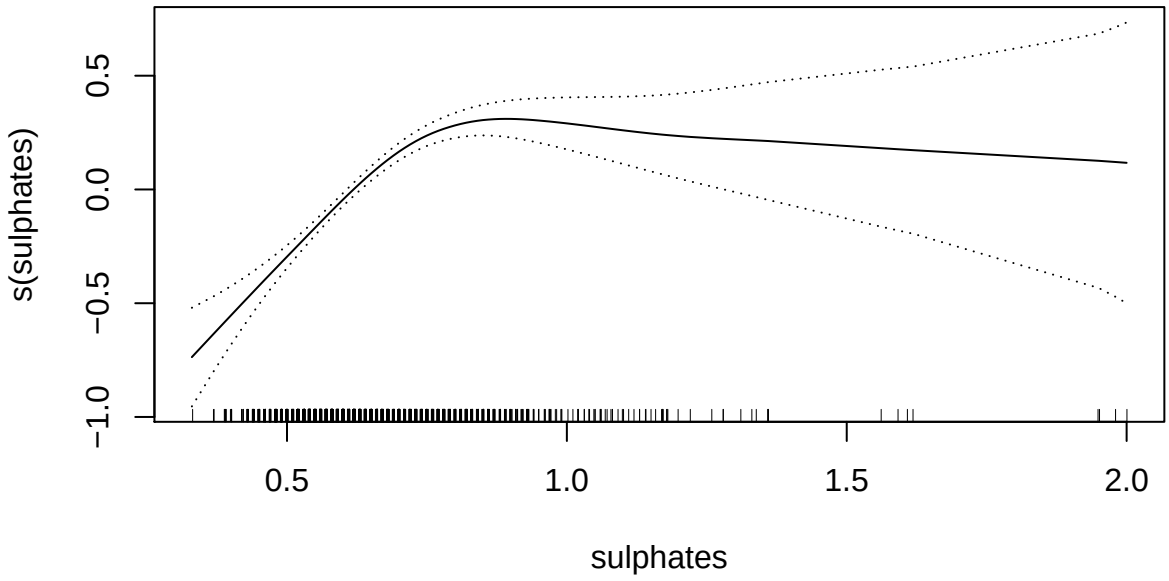

s(chlorides)
0.0
−1.0
−2.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides

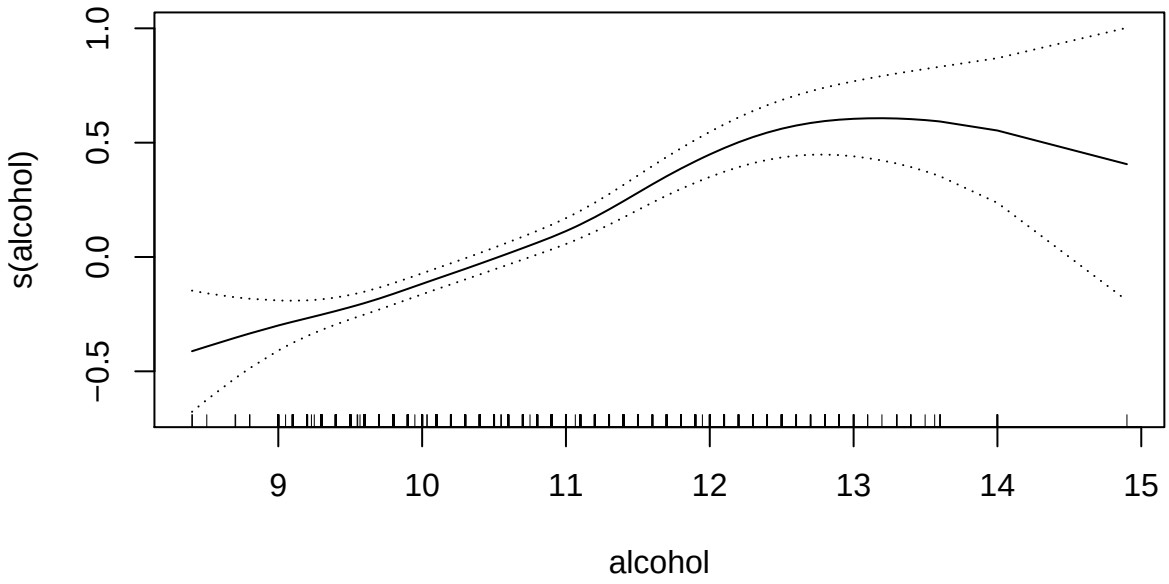

s(free.sulfur.dioxide)
1.0
0.6
0.2
−0.2
0
10
20
30
40
50
60
70
free.sulfur.dioxide


s(total.sulfur.dioxide)
1.0
0.5
0.0
−0.5
0
50
100
150
200
250
300
total.sulfur.dioxide


s(density)
1.0
0.5
0.0
−0.5
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004
density


s(pH)
0.5
0.0
−0.5
−1.0
2.8
3.0
3.2
3.4
3.6
3.8
4.0
pH


s(sulphates)
0.5
0.0
−0.5
−1.0
0.5
1.0
1.5
2.0
sulphates

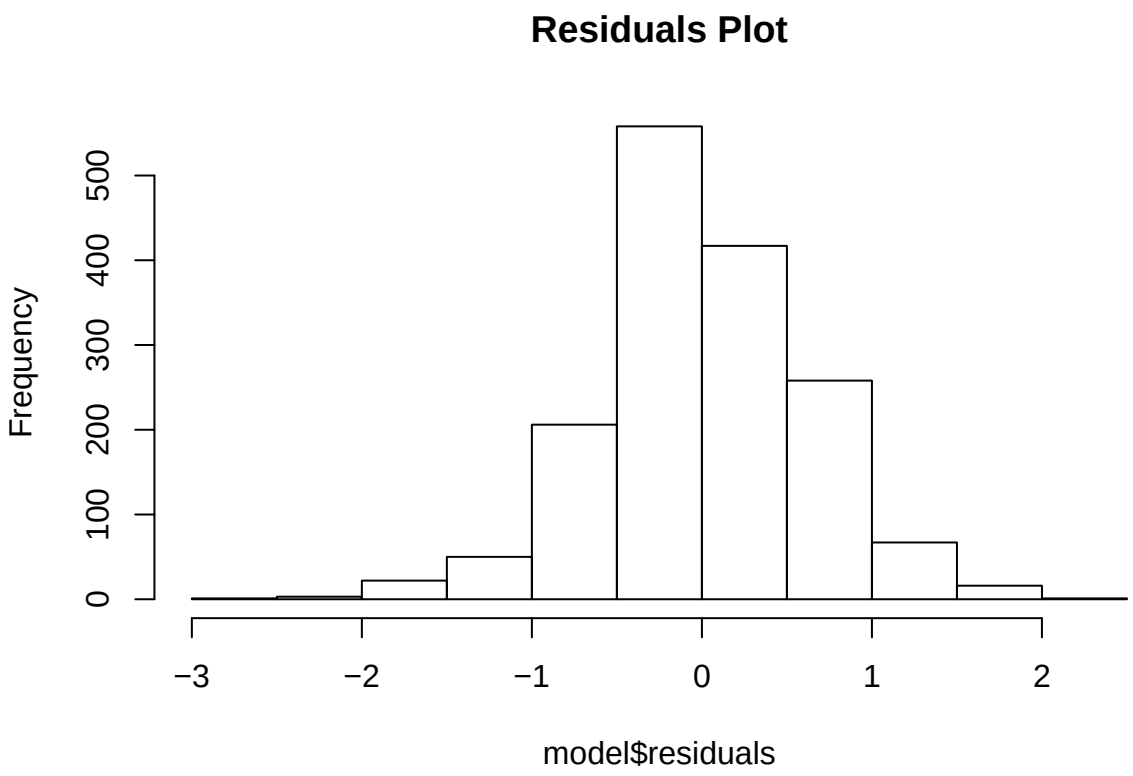



(b) [10 points] Is the overall non-linearity evidenced in the variable-specific smooths statistically significant? Justify your answer with a likelihood ratio test comparing the additive model to a model that includes the features linearly.

### Overall Non-Linearity

It appears that the non-linearity assumption is most true for `s(sulphates)` and `s(alcohol)`, both of which show significance beyond the 1% level.

```
describe_model <- function(model) {
  print(hist(model$residuals, main='Residuals Plot'))
  print(summary(model))
  print("R-squared")
  print(rsq(mod_gam$y, mod_gam$fitted.values))
}
describe_model(mod_gam)
```



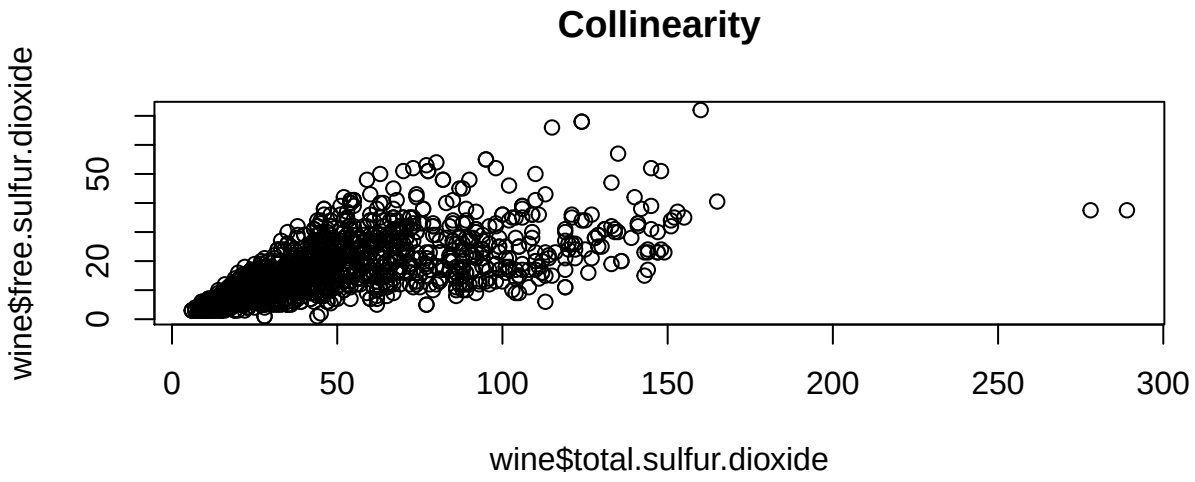

```
## $breaks
##  [1] -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5
##
## $counts
##  [1]   1   3  22  50 206 558 417 258  67  16   1
##
## $density
##  [1] 0.001250782 0.003752345 0.027517198 0.062539087 0.257661038
##  [6] 0.697936210 0.521575985 0.322701689 0.083802376 0.020012508
## [11] 0.001250782
##
## $mids
##  [1] -2.75 -2.25 -1.75 -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75  2.25
##
## $xname
## [1] "model$residuals"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
```

```
## [1] "histogram"
##
## Call: gam(formula = quality ~ s(fixed.acidity) + s(volatile.acidity) +
##     s(citric.acid) + s(residual.sugar) + s(chlorides) + s(free.sulfur.dioxide) +
##     s(total.sulfur.dioxide) + s(density) + s(pH) + s(sulphates) +
##     s(alcohol), data = wine)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.59210 -0.39451 -0.01445  0.41169  2.01386
##
## (Dispersion Parameter for gaussian family taken to be 0.3913)
##
##     Null Deviance: 1042.165 on 1598 degrees of freedom
## Residual Deviance: 608.1472 on 1554 degrees of freedom
## AIC: 3083.986
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                            Df Sum Sq Mean Sq  F value    Pr(>F)
## s(fixed.acidity)            1  11.98  11.977  30.6059 3.704e-08 ***
## s(volatile.acidity)         1 120.14 120.136 306.9848 < 2.2e-16 ***
## s(citric.acid)              1   0.00   0.002   0.0051  0.943113
## s(residual.sugar)           1   0.09   0.090   0.2299  0.631661
## s(chlorides)                1   6.66   6.663  17.0254 3.884e-05 ***
## s(free.sulfur.dioxide)      1   2.71   2.711   6.9287  0.008567 **
## s(total.sulfur.dioxide)     1  26.12  26.121  66.7468 6.328e-16 ***
## s(density)                  1  71.26  71.258 182.0868 < 2.2e-16 ***
## s(pH)                       1   2.92   2.915   7.4493  0.006418 **
## s(sulphates)                1  60.58  60.578 154.7958 < 2.2e-16 ***
## s(alcohol)                  1  33.92  33.919  86.6728 < 2.2e-16 ***
## Residuals                1554 608.15   0.391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                            Npar Df  Npar F     Pr(F)
## (Intercept)
## s(fixed.acidity)                3  3.7217  0.011048 *
## s(volatile.acidity)             3  1.9932  0.113083
## s(citric.acid)                  3  2.8406  0.036714 *
## s(residual.sugar)               3  1.6058  0.186125
## s(chlorides)                    3  2.3446  0.071258 .
## s(free.sulfur.dioxide)          3  3.0062  0.029351 *
## s(total.sulfur.dioxide)         3  3.2367  0.021460 *
```

```
## s(density)                    3  1.9323  0.122401
## s(pH)                         3  0.6279  0.597019
## s(sulphates)                  3 22.9579 1.532e-14 ***
## s(alcohol)                    3  4.8324  0.002369 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "R-squared"
## [1] 0.4164579
```

While both `s(sulphates)` and `s(alcohol)` are highly significant as predictors of `quality`, `s(free.sulfur.dioxide)`, `s(total.sulfur.dioxide)` are both significant at a 5% level of significance. Since these two variables measure similar phenomena (i.e. sulfur dioxide content) there may be some collinearity here (confirmed in the plot below). Finally, both `s(fixed.acidity)` and `s(citric.acid)` are significant at the 5% threshold as well.

```
plot(wine$total.sulfur.dioxide, wine$free.sulfur.dioxide,
     main='Collinearity')
```




Below a separate `lm` model is run using the same variables as the `gam` model. The two models are then compared using `anova`, and it is shown that the `mod_gam` is a significant improvement from over `lm`'s linear paradigm using the same variables.

```
mod_lm_compare <- lm(quality ~ fixed.acidity + volatile.acidity +
                       citric.acid + residual.sugar + chlorides +
                       free.sulfur.dioxide + total.sulfur.dioxide +
                       density + pH + sulphates + alcohol, data=wine)
summary(mod_lm_compare)
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + citric.acid +
```

```
##     residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68911 -0.36652 -0.04699  0.45202  2.02498
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.197e+01  2.119e+01   1.036   0.3002
## fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
## volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
## citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
## residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
## chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
## free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
## total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
## density              -1.788e+01  2.163e+01  -0.827   0.4086
## pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
## sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
## alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.648 on 1587 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561
## F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

```r
# chi square test
anova(mod_lm_compare, mod_gam, test='Chi')
```

```
## Analysis of Variance Table
##
## Model 1: quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
## Model 2: quality ~ s(fixed.acidity) + s(volatile.acidity) + s(citric.acid) +
##     s(residual.sugar) + s(chlorides) + s(free.sulfur.dioxide) +
##     s(total.sulfur.dioxide) + s(density) + s(pH) + s(sulphates) +
##     s(alcohol)
##   Res.Df    RSS Df Sum of Sq  Pr(>Chi)
## 1   1587 666.41
## 2   1554 608.15 33    58.263 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(c) [10 points] We now want to investigate how to produce the best expected wine quality based on the physio-chemical content.

- [109b/121b students only] Based on the additive model fit, how might you **approximately** optimize the physio-chemical composition to produce the highest expected wine quality? Use the results from part (a) to answer this question. What is the resulting estimated wine quality? *Hint: For the latter part, use the* `predict` *function.*

### Strategy for Optimizing Physico-Chemical Composition of Wine

Assuming there are certain aspects of the wine-making process that can be held constant we can fit a model using those "control" parameters. These parameters would ideally be chosen for both their statistical significance in determining `quality` as well as management's capability to influence the variable. Finally, the organization would need to make a strategic decision as to where they want the new wine to fall in relation to others already in their portfolio.

Say management wants to set the values for both `alcohol` and `sulphates` for their new wine to fill a space in the market that they currently do not serve Once values are chosen for both `alcohol` and `sulphates` managers could then simulate the other predictors based on the fitted `gam` model (or some other, yet-to-be formulated model).

Assuming a truncated normal distribution, each instance of the predictors can be simulated by assuming $x_i \in X \sim N(\mu, \sigma)$ using `rnorm(1, mu, sigma)` where $\mu$ and $\sigma$ are calculated from the data. To avoid negative values or erroneous extreme values the simulations are truncated at the 99-th percentile and the 1-st percentile (i.e. these will be the upper and lower bounds). Doing this limits any potential damage done by the potentially incorrect assumption of a normal distribution.

Below we run the simulation 500 times, holding `alcohol=11.5` and `sulphates=0.55` constant with each run.

```
summary(wine[, c('alcohol', 'sulphates')])
```

```
##     alcohol        sulphates
##  Min.   : 8.40   Min.   :0.3300
##  1st Qu.: 9.50   1st Qu.:0.5500
##  Median :10.20   Median :0.6200
##  Mean   :10.42   Mean   :0.6581
##  3rd Qu.:11.10   3rd Qu.:0.7300
##  Max.   :14.90   Max.   :2.0000
```

```
generate_xrange <- function(y, model, hold_constant) {
  predictors <- names(model$data)[names(model$data) != y]
  predictors <- predictors[predictors %!in% names(hold_constant)]
  x_ranges <- c()
  for (p in predictors){
```

```
    # bound each distribution at its max and minimum
    xx = max(min(model$data[, p]),
             rnorm(1, mean(model$data[, p]), sd(model$data[, p])))
    xx = min(xx, max(model$data[, p]))
    x_ranges[p] <- xx
  }
  x_ranges <- data.frame(t(data.frame(append(x_ranges, hold_constant))))
  rownames(x_ranges) <- NULL
  return(x_ranges)
}

# simulate over 500 iterations
seeds <- 1:500
heuristic_df <- data.frame()
for (seed in seeds) {
  x_range <- generate_xrange('quality',
                             mod_gam,
                             hold_constant=c(alcohol=11.5, sulphates=0.55))
  y_pred <- predict(mod_gam, x_range)
  new_row <- data.frame(cbind(y_pred=y_pred, x_range))
  heuristic_df <- rbind(heuristic_df, new_row)
}

heuristic_df <- heuristic_df[order(heuristic_df$y_pred, decreasing=T), ]
rownames(heuristic_df) <- NULL

mu_std_simulation <- function(df) {
  mu_sim <- data.frame(mean.value=sapply(df, mean))
  sd_sim <- data.frame(sd.value=sapply(df, sd))
  return(cbind(mu_sim, sd_sim))
}

print('Mean of Top 25 quality Predictions of 500 Simulation Observations')
```

```
## [1] "Mean of Top 25 quality Predictions of 500 Simulation Observations"
```

```
mu_std_simulation(head(heuristic_df, 25))
```

```
##                       mean.value     sd.value
## y_pred                6.23809960  0.075751082
## fixed.acidity         9.44393481  1.553961582
## volatile.acidity      0.33872471  0.147322860
## citric.acid           0.18020903  0.166207559
## residual.sugar        3.05400341  1.305891567
## chlorides             0.05471989  0.050695345
```

```
## free.sulfur.dioxide  20.39913717 10.063003989
## total.sulfur.dioxide 23.80323970 24.035529416
## density               0.99558289  0.002133907
## pH                    3.24436854  0.147927706
## alcohol              11.50000000  0.000000000
## sulphates             0.55000000  0.000000000
```

```
print('Mean of Bottom 25 quality Predictions of 500 Simulation Observations')
```

```
## [1] "Mean of Bottom 25 quality Predictions of 500 Simulation Observations"
```

```
mu_std_simulation(tail(heuristic_df, 25))
```

```
##                       mean.value     sd.value
## y_pred                 5.1685656  0.077189832
## fixed.acidity          7.2089118  1.734001555
## volatile.acidity       0.6968211  0.154059966
## citric.acid            0.3145969  0.152242183
## residual.sugar         2.2111105  1.189105515
## chlorides              0.1157741  0.038583059
## free.sulfur.dioxide   11.0879956 11.267029644
## total.sulfur.dioxide 73.7688331 28.066552088
## density                0.9980125  0.001637037
## pH                     3.3865755  0.102266481
## alcohol               11.5000000  0.000000000
## sulphates              0.5500000  0.000000000
```

Shown above is a mean and standard deviation of all of the simulated predictors for both the top and bottom 25 simulated observations. It is interesting to note from this dichotomy that the top 25 summary might serve as a baseline guide for producing a wine with the desired `alcohol` and `sulphates` characteristics. The same process may be used to hold other variables constant.

This simulation would probably benefit from more observations.

(d) [5 points] What might be a concern or limitation with optimizing the physio-chemical composition of wine based on the additive model fit? Your answer should be connected to the assumptions underlying additive models.

### Concerns or Limitations Using Additive Fit

One of the main limitations is that predictions for unosbserved $X$ values that are out of sample (or near the edges) relative to the training data are likely to swing more wildly. This is especially true when a neighborhood is under-trained by one or more $X$ vectors in the training data.

Also, not all of the nonlinear predictors showed significance in predicting `quality` using the

`gam::s()` smoothing spline fit. The overall training $R^2$ was just 0.4165, only marginally improving on the linear model which had a training $R^2$ of 0.3561. This suggests another model may be more appropriate.

Finally, the additive model was not tuned on its paramters `df` or `spar` which may yield improvements to the model.

## Problem 2 [40 points]

Rather than fit a single model to all of the wines, we will fit different models to different subsets of the data (in Problem 3). In preparation, this problem will involve partitioning the data into different clusters/subsets.

(a) [5 points] Explain a reason we might expect different relationships between quality and physio-chemical wine composition by different subsets of the data identified in Problem 1.

### Hypothesized Reason for Heterogeneity Between Subsets of Data

There are many different species of grape that serve as raw inputs into wine production. For one species a given profile might indicate a high `quality` red wine (i.e. for that grape/locale/climate), yet for another species that same profile may be indicative of low `quality`. Variables such as operational excellence, geography and climate also play a role in determining `quality`.

While data on these phenomena, e.g. the qualitative grape species names and the geography/climate features, is unrepresented in the `wine` dataset it is safe to assume that some sort of natural separation will occur (when applying clustering algorithms to the data) manifesting from these known inherent-yet-unmeasured differences that characterize red wines.

(b) [5 points] Prior to performing clustering, you will center each column, and also scale each column so that each transformed feature has a standard deviation of 1.0. Briefly justify the decision to scale the data in this manner. Be specific to the context of this problem.

### Scaling Prior to Clustering

Clustering algorithms are reliant on some notion of distance between observations in the data. When dealing with absolute values that are of different orders of magnitude (e.g. pH and light years) it will be vital that the data be scaled prior to clustering. Doing such clustering ahead of time will level the playing field so that distances between features that are measured on different orders of magnitude can be sensed more readily.

```
quick_summary <- function(df) {
  data.frame(mean.value=sapply(df, mean),
             sd.value=sapply(df, sd),
             max.value=sapply(df, max),
             min.value=sapply(df, min))
}
quick_summary(wine[predictor_cols])
```

```
##                       mean.value     sd.value max.value min.value
## fixed.acidity          8.31963727  1.741096318  15.90000   4.60000
## volatile.acidity       0.52782051  0.179059704   1.58000   0.12000
## citric.acid            0.27097561  0.194801137   1.00000   0.00000
## residual.sugar         2.53880550  1.409928060  15.50000   0.90000
## chlorides              0.08746654  0.047065302   0.61100   0.01200
## free.sulfur.dioxide   15.87492183 10.460156970  72.00000   1.00000
## total.sulfur.dioxide 46.46779237 32.895324478 289.00000   6.00000
## density                0.99674668  0.001887334   1.00369   0.99007
## pH                     3.31111320  0.154386465   4.01000   2.74000
## sulphates              0.65814884  0.169506980   2.00000   0.33000
## alcohol               10.42298311  1.065667582  14.90000   8.40000
```

Above the `mean`, `sd`, `max` and `min` are taken of all the $X$ values. Notice how `total.sulfur.dioxide` is orders of magnitude larger than `chlorides`. This large difference would drown out any potential distance metric between these two vectors. Scaling the data takes care of this issue by centering all variables at 0 with a standard deviation of 1.

```
wine_scaled <- wine
wine_scaled[predictor_cols] <- scale(wine[predictor_cols])
quick_summary(wine_scaled[predictor_cols])
```

```
##                         mean.value sd.value max.value min.value
## fixed.acidity         3.518207e-16        1  4.353787 -2.136377
## volatile.acidity      1.841869e-16        1  5.876138 -2.277567
## citric.acid          -9.207575e-17        1  3.742403 -1.391037
## residual.sugar       -1.156003e-16        1  9.192806 -1.162333
## chlorides             8.613634e-17        1 11.123555 -1.603443
## free.sulfur.dioxide  -5.600528e-17        1  5.365606 -1.422055
## total.sulfur.dioxide  3.789652e-17        1  7.372847 -1.230199
## density               2.365840e-14        1  3.678904 -3.537625
## pH                   -2.245475e-17        1  4.526866 -3.699244
## sulphates             2.009076e-17        1  7.916200 -1.935902
## alcohol               8.786086e-17        1  4.201138 -1.898325
```

Note above the means are all approximately 0 and the standard deviatiosn are approximately 1. This supplies a level setting for clustering methods.

(c) [10 points] Suppose we decide to perform partitioning-around-medoids clustering of

the observations based only on the physio-chemical features but not using quality. To determine the best number of clusters, optimize based on the gap statistic in the following manner:

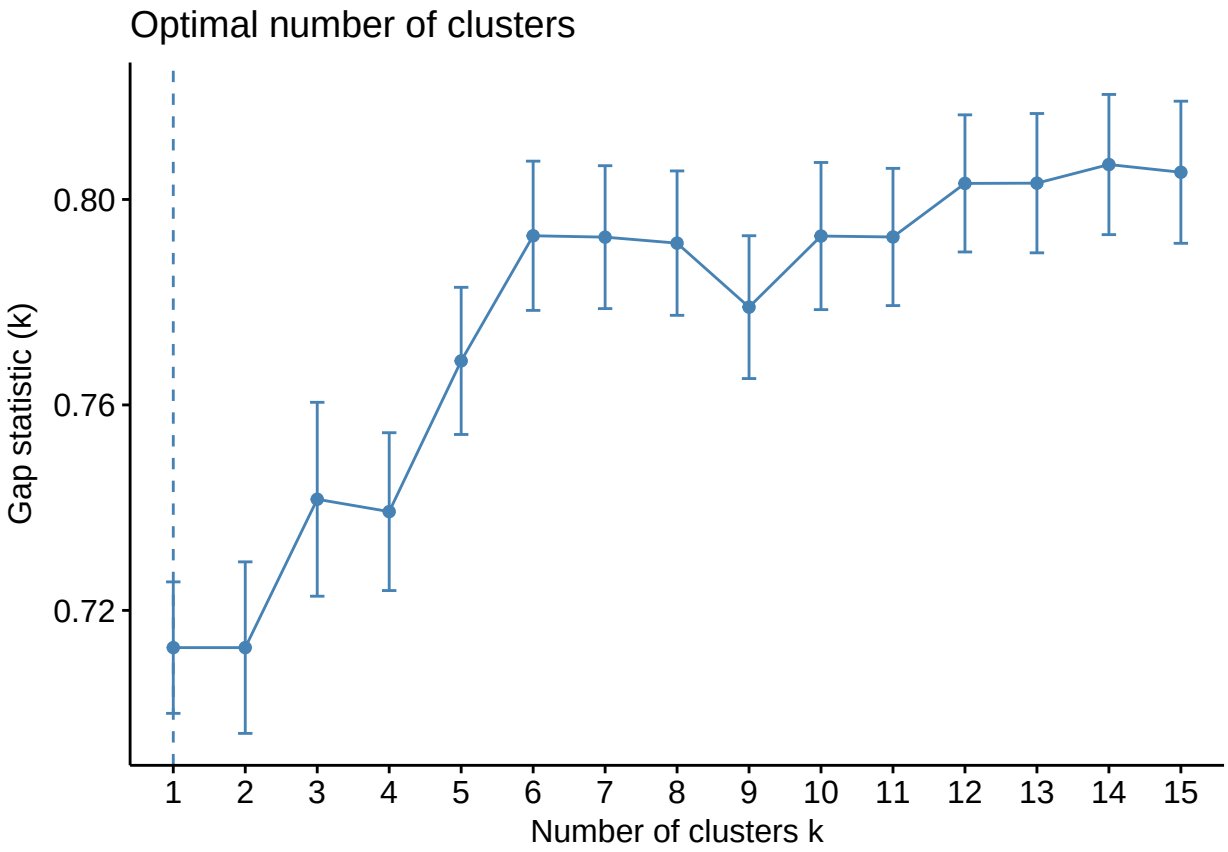
1. Set the random number seed to 123 (`set.seed(123)`). Now select a random sample of 200 wines (*hint: use the* `sample` *function*).
2. Set the random number seed to 321 (`set.seed(321)`). Optimize the gap statistic using the method described by Tibshirani (2001) based on the standard error rule, using `d.power=2`.

   Explain how 1 cluster is the optimal number of clusters according to Tibshirani's rule, even though 6 clusters would be chosen if one were to use the maximum gap statistic.

### Tibshirani's Gap Statistic

Below the `factoextra::fviz_nbclust()` function was employed using a subset of 200 random observations. Only the `predictor_cols` were sampled. Setting the parameter `FUNcluster` to `cluster::pam` ("partitioning around medoids") and the parameter `method='gap_stat'`, the algorithm is set to run from 1 to 15 mediods. A plot is then displayed showing the Tibshirani-method's optimal Gap Statistic.

```
set.seed(123)
sample_ix <- sample(rownames(wine_scaled), size=200)
wine_scaled_200 <- wine_scaled[sample_ix, predictor_cols]

set.seed(321)
kmeds_wine <- fviz_nbclust(wine_scaled_200,
                           FUNcluster=pam,
                           method='gap_stat',
                           k.max=15)
plot(kmeds_wine)
```




Tibshirani's Gap Statistic is a conservative approach to choosing a best number of clusters. Starting at the lowest number in a range (e.g. 1) the algorithm computes the statistic for each value in the range (e.g. 1 through 15) then highlights the best K. The Gap Statistic measures whether the data clustered into K groups is is significantly better than if they were generated at random. In other words, for a given choice of K clusters we can compare the **actual** total within cluster variation with the **expected** within cluster variation. This approach compares to a baseline wherein there is no obvious clustering.

Thus, a visual comparison of the Gap Statistics generated from $K_i$ to $K_{i+1}$ shows whether or not taking the step forward (increasing K medoids) yields a significant gain. In this case going from 1 cluster to 2 does not yield a considerable improvement in the Gap Statistic and the method points to $K = 1$ as the optimal number of clusters. This can be identified visually since the standard error bars for both $i = 1$ and $i = 2$ have considerable overlap, and the mean Gap Statistic barely changed at all.

(d) [10 points] Partition the full data into six clusters via partitioning-around-medoids on the scaled version of the data. Save the cluster identifiers as a new column in the original data frame (*Hint: the* `clustering` *component of the resulting cluster object contains the IDs*). Plot the first two principal components of the scaled data and visually show the cluster memberships. Show that the proportion of variance in the original data represented by the principal component plot is 45.7%. Use the output of `prcomp` to demonstrate this.

### Partitioning Data to 6 Clusters

Below a `pam` model is fit using `k=6` clusters.

```
# run pam model with k=6 on predictor_cols only
mod_pam <- pam(wine_scaled[, predictor_cols], k=6)

# save cluster memership in both dataframes
wine_scaled$medoid.member <- wine$medoid.member <- factor(mod_pam$clustering)

# medoid clusters (recall 6 chosen)
print('Six Medoids')
```

```
## [1] "Six Medoids"
```

```
mod_pam$medoids
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## [1,]    0.04615639        1.0453468  -0.3643491    -0.09844864 -0.2436305
## [2,]   -0.24101899        0.2914083  -0.2103458    -0.16937425 -0.2436305
## [3,]    0.85024746       -0.4904538   1.0216798    -0.02752304 -0.5198424
## [4,]   -0.64306452        0.2355610  -1.1343651    -0.16937425 -0.1373951
## [5,]    0.16102655       -0.2112173   1.2270174    -0.38215106  7.1078575
## [6,]   -0.58562945       -1.0489267   0.3029982    -0.02752304 -0.4985954
##      free.sulfur.dioxide total.sulfur.dioxide     density           pH
## [1,]         -0.56164758           -0.2574163  0.31966829 -0.007210449
## [2,]          0.68116360            1.4449533  0.06004281 -0.654935624
## [3,]         -0.65724844           -0.5614109  0.71705425 -0.460618072
## [4,]          0.01195758           -0.3182152 -0.58637168  0.446197173
## [5,]          0.01195758            0.4721707  0.61108466 -1.820840939
## [6,]          0.48996188           -0.2878157 -0.81950477  0.251879621
##        sulphates    alcohol
## [1,] -0.10706841 -0.9599458
## [2,] -0.46103614 -0.8661079
## [3,] -0.04807379  0.3537847
## [4,] -0.34304689 -0.1154048
## [5,]  3.01964650 -1.3352974
## [6,]  0.65986166  1.5736773
```

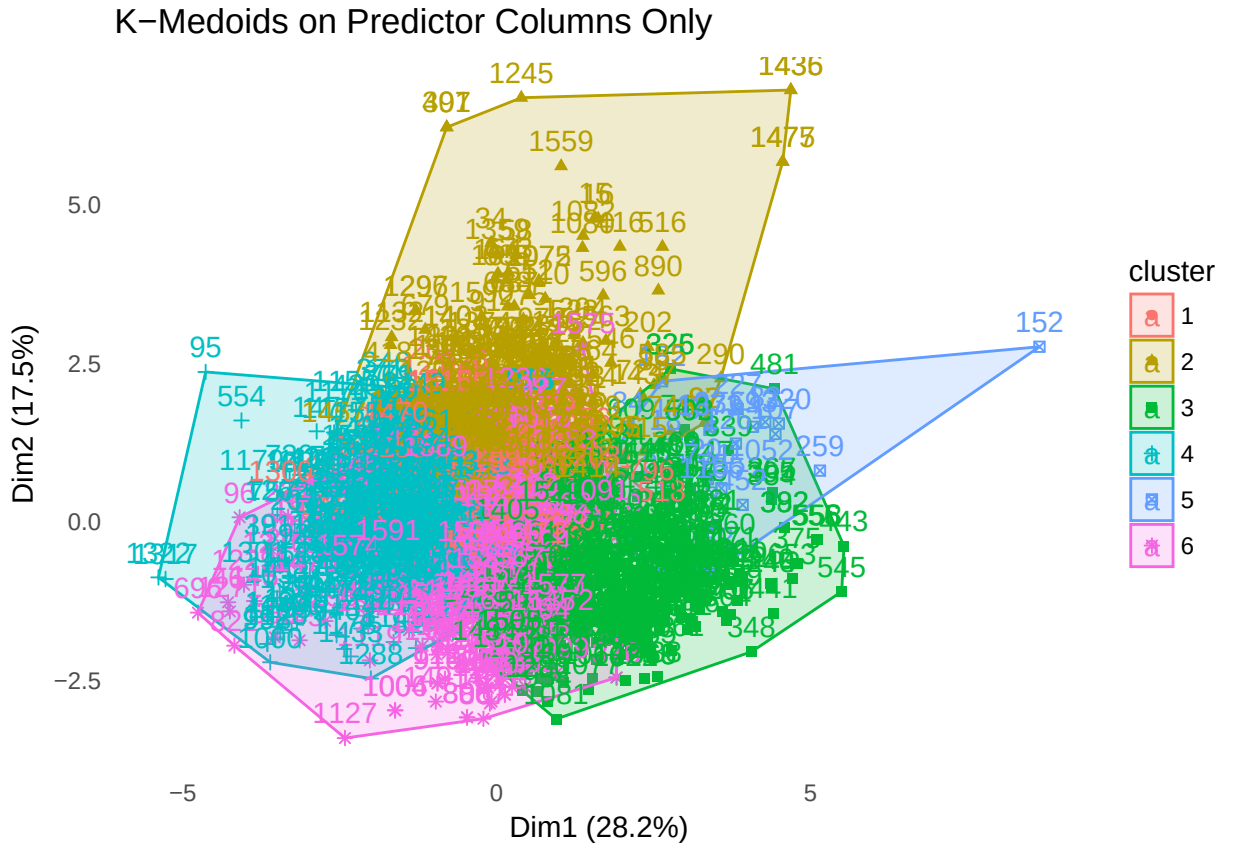
### Principal Components

It is shown below using R's base `prcomp.summary` function that the cumulative proportion explained by the first two principal components is 45.68%.

```
wine_pca <- prcomp(wine_scaled[, predictor_cols])
summary(wine_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7604 1.3878 1.2452 1.1015 0.97943 0.81216 0.76406
## Proportion of Variance 0.2817 0.1751 0.1410 0.1103 0.08721 0.05996 0.05307
## Cumulative Proportion  0.2817 0.4568 0.5978 0.7081 0.79528 0.85525 0.90832
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.65035 0.58706 0.42583 0.24405
## Proportion of Variance 0.03845 0.03133 0.01648 0.00541
## Cumulative Proportion  0.94677 0.97810 0.99459 1.00000
```

```
# run fviz_cluster to visualize medoids
fviz_cluster(mod_pam,
             data=wine_scaled[, predictor_cols],
             main='K-Medoids on Predictor Columns Only') +
  theme_minimal()
```

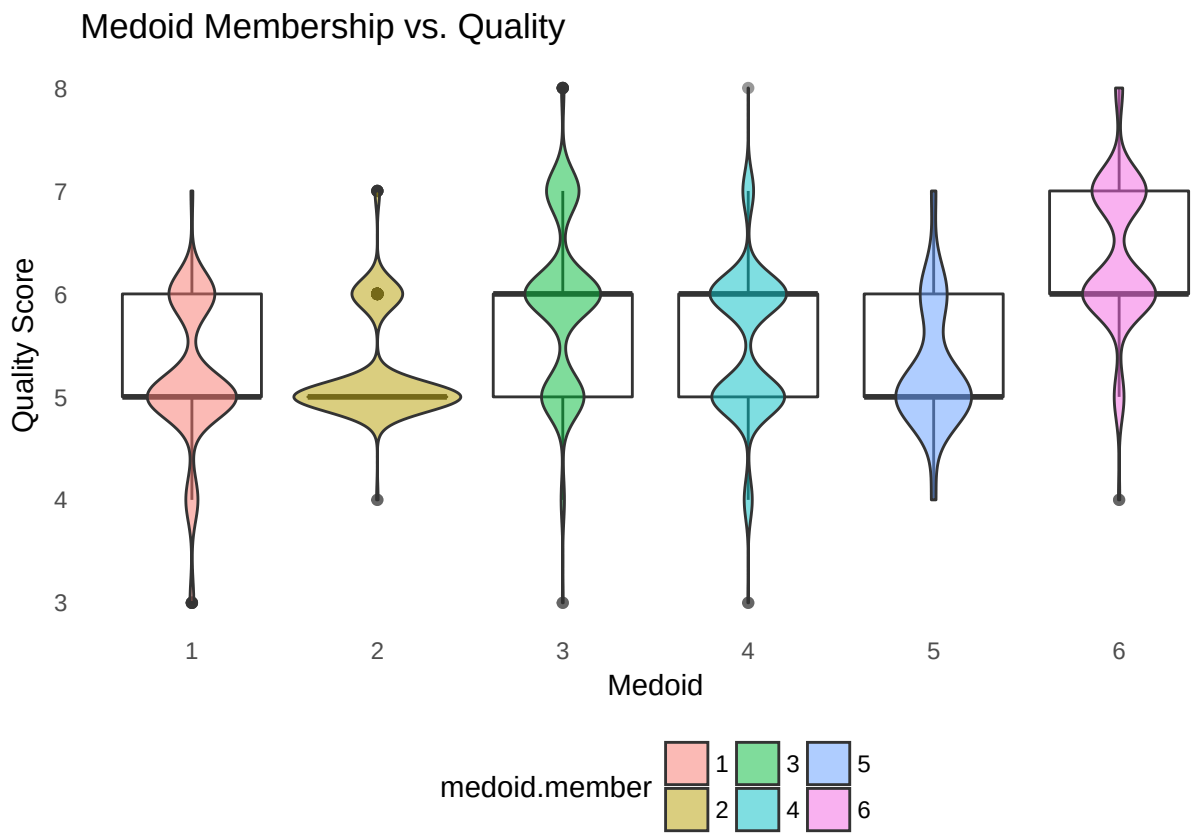



(e) [10 points] Create a side-by-side boxplot of quality scores by cluster (*Hint: If using* `ggplot` *you should use the* `geom_boxplot` *function – do not forget to make the cluster ID variable a factor in R*). Does the distribution of quality scores differ visually by the clusters you determined? Would you have expected the distribution of quality scores to differ?

### Variance in `quality` Between Clusters

The plot below shows the distribution of `quality` over `medoid.member`.

```
ggplot(data=wine_scaled,
       aes(x=medoid.member, y=quality, group=medoid.member)) +
  geom_boxplot(alpha=.5) +
  geom_violin(aes(fill=medoid.member), alpha=.5) +
  theme_minimal() +
  theme(legend.position='bottom') +
  labs(title='Medoid Membership vs. Quality',
       x='Medoid', y='Quality Score')
```




Since `medoid.member` was determined without `quality` in mind it is interesting to see that there is a bit of a pattern between the groups. The most stark contrasts are between `medoid.member=2` and `medoid.member=6`. The second medoid is strongly clustered lower (with fewer outliers than the others) than the sixth medoid. However the other distributions (i.e. between 3, 4 and 5) do not appear to be significantly different. the same goes for clusters 1 and 2. This indicates that we may have too many clusters.

Given that `quality` is somewhat of a subjective metric and that the clusters were generated based on physically observable phenomena, it is not surprising that there is considerable overlap in the distribution of `quality` between the medoids. This is made even more likely

by the fact that the clusters were generated on 11 features.

## Problem 3 [30 points]

We will now fit a normal hierarchical linear model for quality scores against the physio-chemical predictors nested in the formed clusters from the previous problem.

```
// midterm part 3.stan

data {
  int N; // number of observations
  int num_medoids; // number of distinct medoids
  int quality[N];  // response variable
  int medoid[N];   // medoid ID
  real x1_fixed_acidity[N];
  real x2_volatile_acidity[N];
  real x3_citric_acid[N];
  real x4_residual_sugar[N];
  real x5_chlorides[N];
  real x6_free_sulfur_dioxide[N];
  real x7_total_sulfur_dioxide[N];
  real x8_density[N];
  real x9_pH[N];
  real x10_sulphates[N];
  real x11_alcohol[N];
}

parameters {
  real a[num_medoids];   // intercept for each medoid
  real b1[num_medoids];  // fixed_acidity
  real b2[num_medoids];  // volatile_acidity
  real b3[num_medoids];  // citric_acid
  real b4[num_medoids];  // residual_sugar
  real b5[num_medoids];  // chlorides
  real b6[num_medoids];  // free_sulfur_dioxide
  real b7[num_medoids];  // total_sulfur_dioxide
  real b8[num_medoids];  // density
  real b9[num_medoids];  // pH
  real b10[num_medoids]; // sulphates
  real b11[num_medoids]; // alcohol
  real<lower=0> sigma_y ; //
  real<lower=0> sigma_a ; //
  real<lower=0> sigma_b1; //
  real<lower=0> sigma_b2; //
```

```
  real<lower=0> sigma_b3; //
  real<lower=0> sigma_b4; //
  real<lower=0> sigma_b5; //
  real<lower=0> sigma_b6; //
  real<lower=0> sigma_b7; //
  real<lower=0> sigma_b8; //
  real<lower=0> sigma_b9; //
  real<lower=0> sigma_b10; //
  real<lower=0> sigma_b11; //
}

transformed parameters {
  vector[N] y_hat;
  for (i in 1:N) {
    y_hat[i] = (
      a[medoid[i]] +
        b1[medoid[i]]  *  x1_fixed_acidity[i] +
        b2[medoid[i]]  *  x2_volatile_acidity[i] +
        b3[medoid[i]]  *  x3_citric_acid[i] +
        b4[medoid[i]]  *  x4_residual_sugar[i] +
        b5[medoid[i]]  *  x5_chlorides[i] +
        b6[medoid[i]]  *  x6_free_sulfur_dioxide[i] +
        b7[medoid[i]]  *  x7_total_sulfur_dioxide[i] +
        b8[medoid[i]]  *  x8_density[i] +
        b9[medoid[i]]  *  x9_pH[i] +
        b10[medoid[i]] *  x10_sulphates[i] +
        b11[medoid[i]] *  x11_alcohol[i]
           );
  }
}

model {
  // priors
  sigma_y   ~ inv_gamma(0.001, 0.001);
  sigma_a   ~ uniform(0, 100);
  sigma_b1  ~ uniform(0, 100);
  sigma_b2  ~ uniform(0, 100);
  sigma_b3  ~ uniform(0, 100);
  sigma_b4  ~ uniform(0, 100);
  sigma_b5  ~ uniform(0, 100);
  sigma_b6  ~ uniform(0, 100);
  sigma_b7  ~ uniform(0, 100);
  sigma_b8  ~ uniform(0, 100);
  sigma_b9  ~ uniform(0, 100);
  sigma_b10 ~ uniform(0, 100);
```

```
  sigma_b11 ~ uniform(0, 100);
  for (k in 1:num_medoids) {
    a[k]   ~ normal(5.636023, sigma_a);
    b1[k]  ~ normal(0, sigma_b1);
    b2[k]  ~ normal(0, sigma_b2);
    b3[k]  ~ normal(0, sigma_b3);
    b4[k]  ~ normal(0, sigma_b4);
    b5[k]  ~ normal(0, sigma_b5);
    b6[k]  ~ normal(0, sigma_b6);
    b7[k]  ~ normal(0, sigma_b7);
    b8[k]  ~ normal(0, sigma_b8);
    b9[k]  ~ normal(0, sigma_b9);
    b10[k] ~ normal(0, sigma_b10);
    b11[k] ~ normal(0, sigma_b11);
  }

  // likelihood
  for (i in 1:N){
    quality[i] ~ normal(y_hat[i], sigma_y);
  }
}
```

(a) [10 points] Implement a normal hierarchical linear model in Stan (called from R) to fit the model. Make sure you let all the linear model coefficients vary by cluster. *Hint: You may find the Stan code supplied with the lecture notes helpful.* You may assume that *the intercepts across the six clusters have a normal prior distribution* with a mean which is the average of the quality scores across the whole data set, and with an unknown standard deviation.

The 11 physio-chemical coefficients across the six clusters can be assumed to be normally distributed centered at 0 with different standard deviations. Finally, you may assume that all the standard deviation parameters have a prior uniform distribution with a minimum of 0 and maximum of 100 (which is sufficiently large).

### Hierarchical Linear Model

The figure below is used as a seed for the intercept coefficients `a` in the stan code.

```
mean(wine$quality)
```

```
## [1] 5.636023
```

Below the data `list` is specified to match the `.stan` file, then the code is executed for 4000 iterations and 4 chains (using all 8 cores of this machine).

```r
options(mc.cores = parallel::detectCores())

# specify model data
dat <- list()
dat$N                          <- nrow(wine_scaled)
dat$num_medoids                <- length(unique(wine_scaled$medoid.member))
dat$quality                    <- wine_scaled$quality
dat$medoid                     <- as.numeric(wine_scaled$medoid.member)
dat$x1_fixed_acidity           <- wine_scaled$fixed.acidity
dat$x2_volatile_acidity        <- wine_scaled$volatile.acidity
dat$x3_citric_acid             <- wine_scaled$citric.acid
dat$x4_residual_sugar          <- wine_scaled$residual.sugar
dat$x5_chlorides               <- wine_scaled$chlorides
dat$x6_free_sulfur_dioxide     <- wine_scaled$free.sulfur.dioxide
dat$x7_total_sulfur_dioxide    <- wine_scaled$total.sulfur.dioxide
dat$x8_density                 <- wine_scaled$density
dat$x9_pH                      <- wine_scaled$pH
dat$x10_sulphates              <- wine_scaled$sulphates
dat$x11_alcohol                <- wine_scaled$alcohol

# run Stan model using rstan package
mod_stan <- stan(file=base_dir %+% 'midterm part 3.stan',
                 warmup=1000,
                 data=dat,
                 iter=3000,
                 refresh=0,
                 chain=4,
                 seed=46)
```

```
## In file included from C:/Users/paulm/Documents/R/win-library/3.3/BH/include/boost/con
##                  from C:/Users/paulm/Documents/R/win-library/3.3/BH/include/boost/mat
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from file35183c7169b8.cpp:8:
## C:/Users/paulm/Documents/R/win-library/3.3/BH/include/boost/config/compiler/gcc.hpp:1
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
## In file included from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
```

```
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/
##                  from file35183c7169b8.cpp:8:
## C:/Users/paulm/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set
##      static void set_zero_all_adjoints() {
##                  ^
```

```
posterior <- as.array(mod_stan)
```

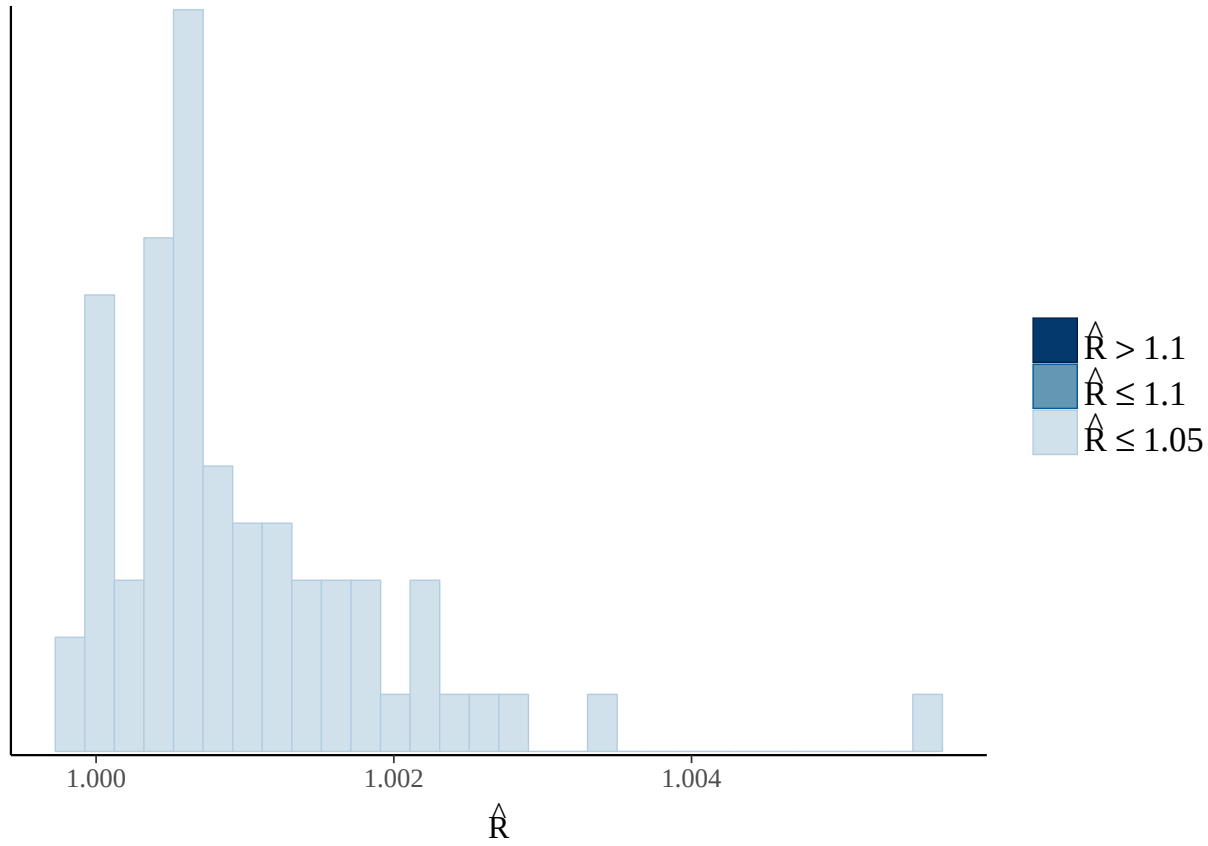
(b) [10 points] Briefly report on the details of your model implementation (number of iterations of burn-in and the number of iterations of saved parameters, number of parallel samplers, and any assurances that the sampler converged). (*Hint: If you saved the Stan fit of your model in the R object* `wine.fit`, *you can access the matrix of model summaries from* `summary(wine.fit)$summary`.) Do not be concerned about warnings of divergent transitions after warm-up if you have evidence that the sampler converged for the feature coefficients.

### Brief Summary of Model

A burn-in period of 1000 iterations was performed to ensure that the model was well into convergence territory. The parameter `chains` was set to 4 to ensure robustness of the simulation.

The $\hat{R}$ statistic measures equilibrium across chains, and if these values are at or near one then convergence has transpired. The figures and histogram below indicate that we do in fact have convergence.

```
mod_stan_summary <- summary(mod_stan)

mcmc_rhat_hist(mod_stan_summary$summary[1:66, 'Rhat'])
```




```
mod_stan_summary$summary[1:66, 'Rhat']
```

```
##      a[1]      a[2]      a[3]      a[4]      a[5]      a[6]     b1[1]
## 1.0014274 1.0004838 1.0005978 1.0023066 1.0011902 1.0006292 0.9998854
##     b1[2]     b1[3]     b1[4]     b1[5]     b1[6]     b2[1]     b2[2]
## 1.0005519 1.0014639 1.0017727 1.0005926 1.0021594 1.0009784 1.0000742
##     b2[3]     b2[4]     b2[5]     b2[6]     b3[1]     b3[2]     b3[3]
## 1.0004711 1.0005947 1.0001580 1.0003599 1.0012560 1.0016061 1.0003608
##     b3[4]     b3[5]     b3[6]     b4[1]     b4[2]     b4[3]     b4[4]
## 1.0027872 1.0001119 1.0008486 1.0000406 1.0021094 1.0002891 1.0004519
##     b4[5]     b4[6]     b5[1]     b5[2]     b5[3]     b5[4]     b5[5]
## 1.0009240 1.0000454 1.0010306 1.0021503 1.0003593 1.0007003 1.0006087
##     b5[6]     b6[1]     b6[2]     b6[3]     b6[4]     b6[5]     b6[6]
## 1.0016111 1.0006038 1.0006454 1.0024423 1.0026548 1.0011552 1.0034842
##     b7[1]     b7[2]     b7[3]     b7[4]     b7[5]     b7[6]     b8[1]
## 0.9999479 1.0000984 1.0008566 1.0010054 1.0004965 1.0009162 1.0000437
##     b8[2]     b8[3]     b8[4]     b8[5]     b8[6]     b9[1]     b9[2]
## 1.0015823 1.0011480 1.0005304 1.0017998 1.0055362 1.0007144 1.0008506
##     b9[3]     b9[4]     b9[5]     b9[6]    b10[1]    b10[2]    b10[3]
## 1.0005201 1.0000773 1.0003630 1.0019029 1.0005322 0.9997753 1.0002591
##    b10[4]    b10[5]    b10[6]
## 1.0008325 1.0013557 1.0005690
```

(c) [10 points] Create a visualization that demonstrates the variation of coefficients across clusters. One natural way would be to display side-by-side boxplots of the posterior simulated draws for the relevant coefficients. (*Hint: Use the* `extract` *function applied to the fitted Stan model to obtain simulated coefficient values.*) Based on these results, do you think that the hierarchical model by formed clusters was helpful in explaining the variation in quality scores? Briefly justify.

### Visualizing the Variation in Coefficients Across Clusters

Below a `data.frame` of beta names is created for easy access in plotting.

```
generate_beta_names <- function(beta, n){
  betas <- c()
  for(b in 1:n){
    betas[b] <- 'b' %+% beta %+% '[' %+% b %+% ']'
  }
  betas
}

beta_df <- data.frame()
for (b in 1:11){
  colname <- 'b' %+% b
  b_df <- data.frame(generate_beta_names(b, 6))
  colnames(b_df) <- colname
  beta_df <- rbind(beta_df, t(b_df))
}
beta_df <- data.frame(t(beta_df))
rownames(beta_df) <- NULL
beta_df
```

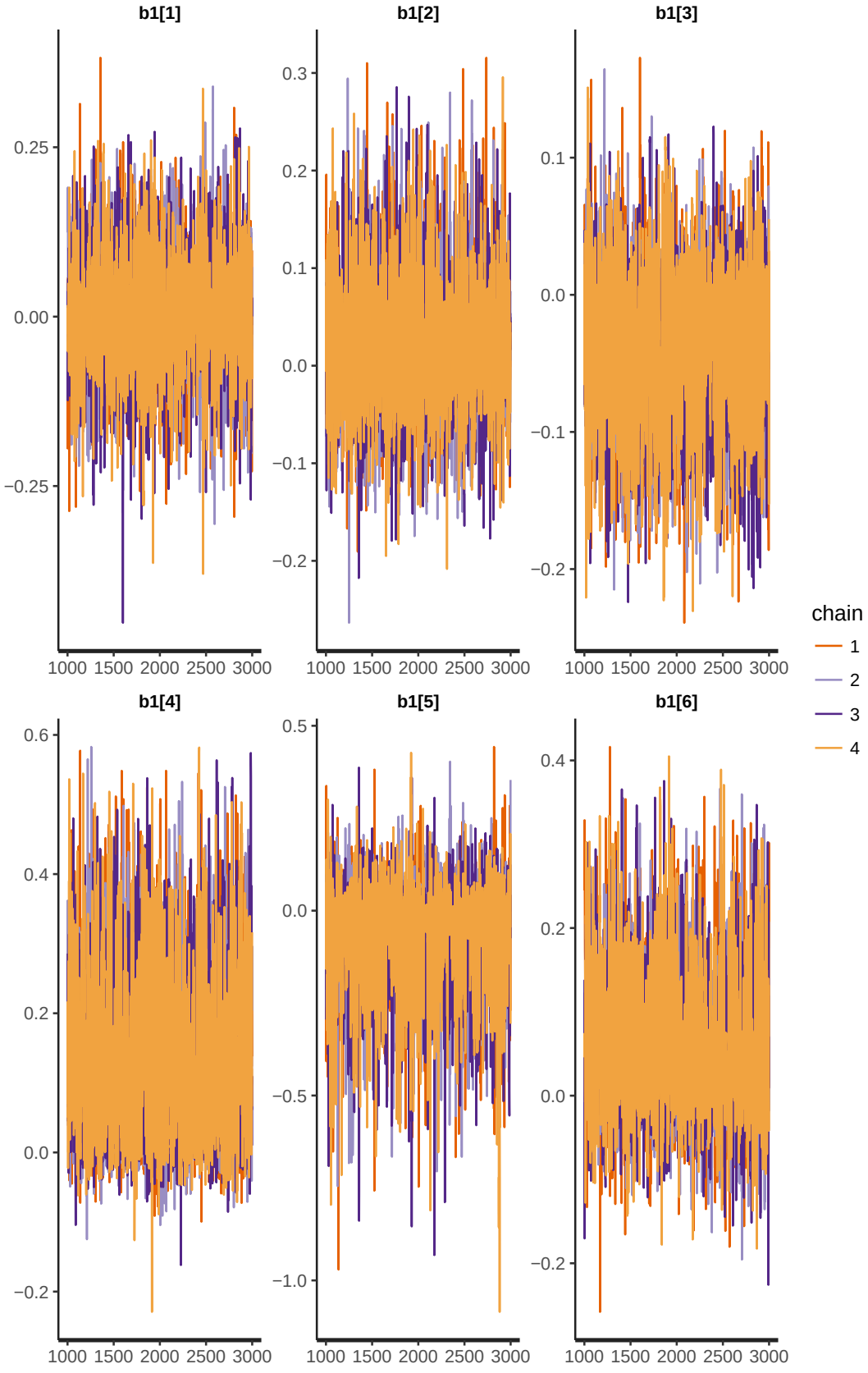
```
##       b1    b2    b3    b4    b5    b6    b7    b8    b9    b10    b11
## 1 b1[1] b2[1] b3[1] b4[1] b5[1] b6[1] b7[1] b8[1] b9[1] b10[1] b11[1]
## 2 b1[2] b2[2] b3[2] b4[2] b5[2] b6[2] b7[2] b8[2] b9[2] b10[2] b11[2]
## 3 b1[3] b2[3] b3[3] b4[3] b5[3] b6[3] b7[3] b8[3] b9[3] b10[3] b11[3]
## 4 b1[4] b2[4] b3[4] b4[4] b5[4] b6[4] b7[4] b8[4] b9[4] b10[4] b11[4]
## 5 b1[5] b2[5] b3[5] b4[5] b5[5] b6[5] b7[5] b8[5] b9[5] b10[5] b11[5]
## 6 b1[6] b2[6] b3[6] b4[6] b5[6] b6[6] b7[6] b8[6] b9[6] b10[6] b11[6]
```

A trace plot is generated below for each beta coefficient, over each cluster for that beta coefficient.

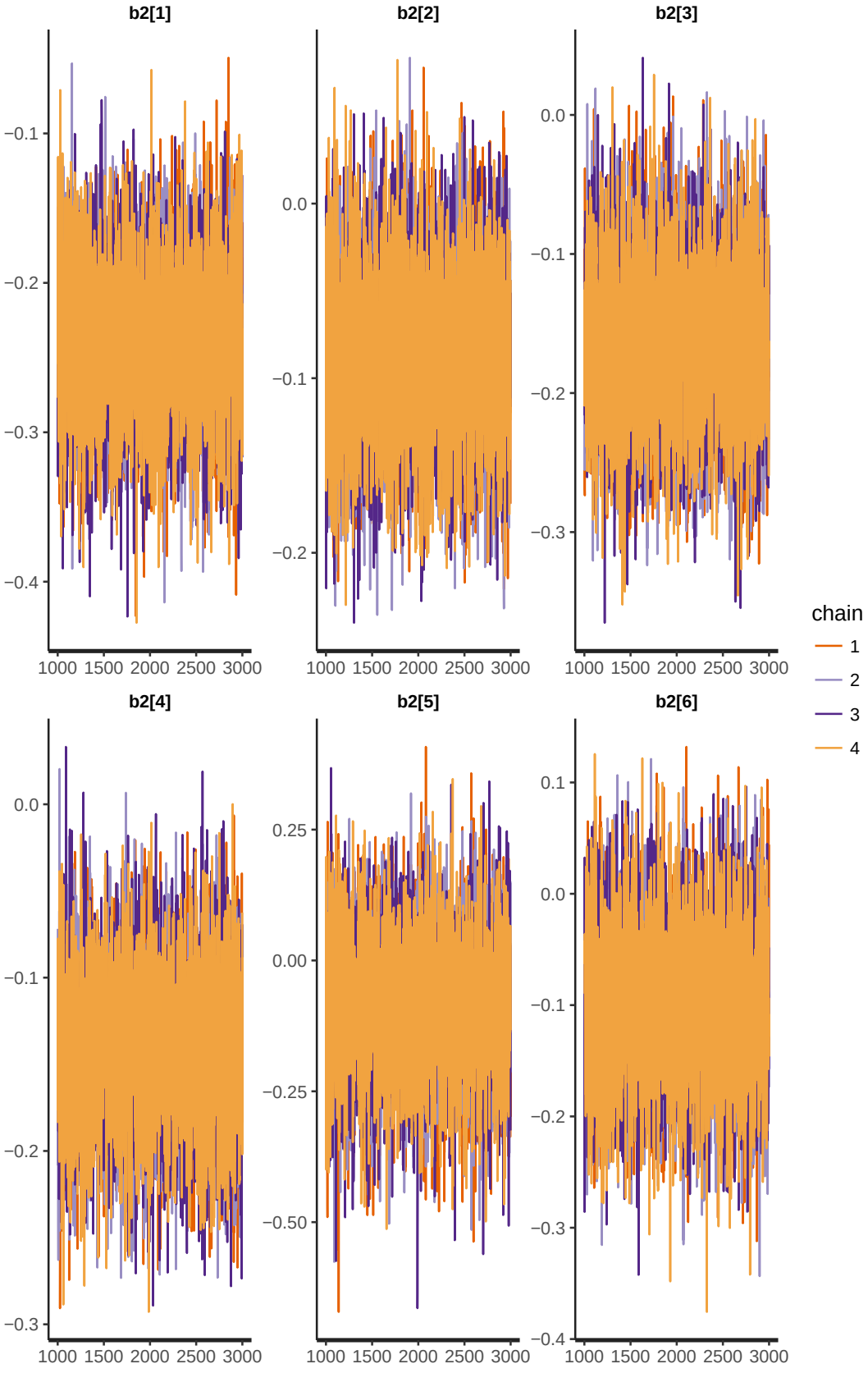
```
for (beta in names(beta_df)){
  msg <- 'Coefficients for ' %+% beta %+% ' Over 6 Clusters'
  print(msg)
  par(mfrow=c(2, 1))
```

```
  print(plot(mod_stan, plotfun='trace', par=beta_df[, beta]) )
}
```

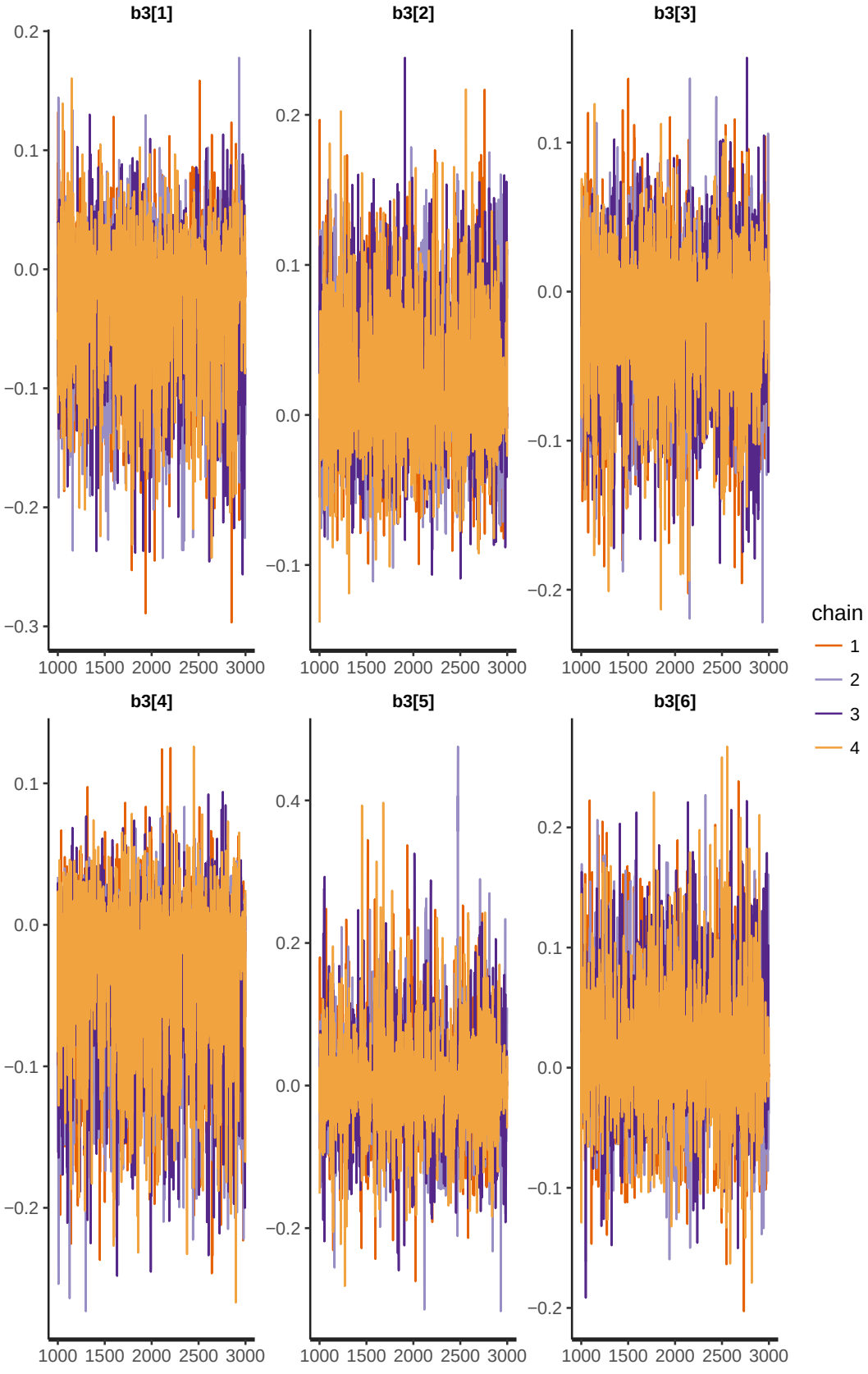
```
## [1] "Coefficients for b1 Over 6 Clusters"
```


b1[1]
b1[2]
b1[3]
b1[4]
b1[5]
b1[6]
0.25
0.00
−0.25
0.3
0.2
0.1
0.0
−0.1
−0.2
0.1
0.0
−0.1
−0.2
0.6
0.4
0.2
0.0
−0.2
0.5
0.0
−0.5
−1.0
0.4
0.2
0.0
−0.2
1000 1500 2000 2500 3000
chain
1
2
3
4

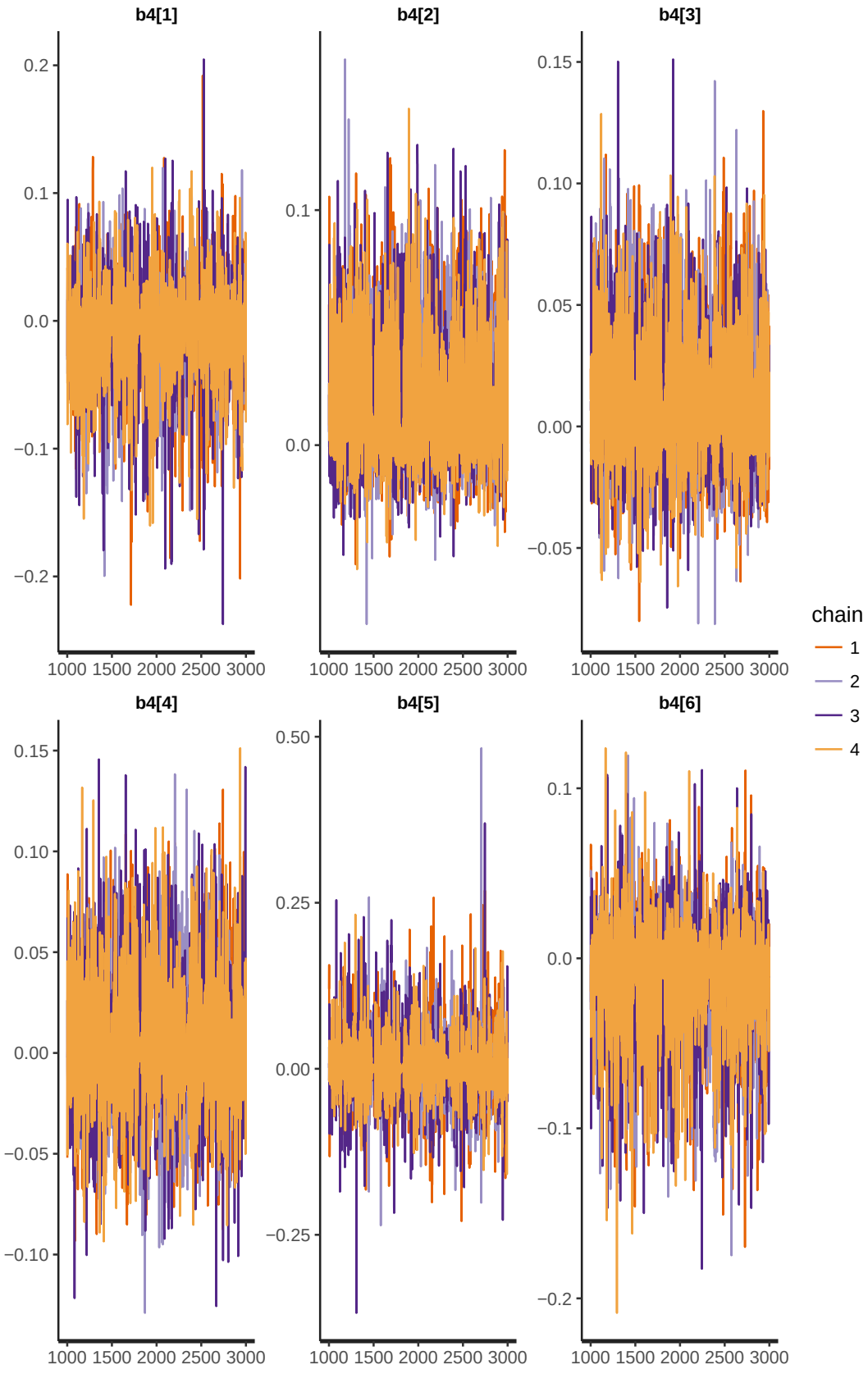
```
## [1] "Coefficients for b2 Over 6 Clusters"
```


b2[1]
b2[2]
b2[3]
b2[4]
b2[5]
b2[6]
−0.1
−0.2
−0.3
−0.4
0.0
−0.1
−0.2
0.0
−0.1
−0.2
−0.3
0.0
−0.1
−0.2
−0.3
0.25
0.00
−0.25
−0.50
0.1
0.0
−0.1
−0.2
−0.3
−0.4
1000 1500 2000 2500 3000
chain
1
2
3
4

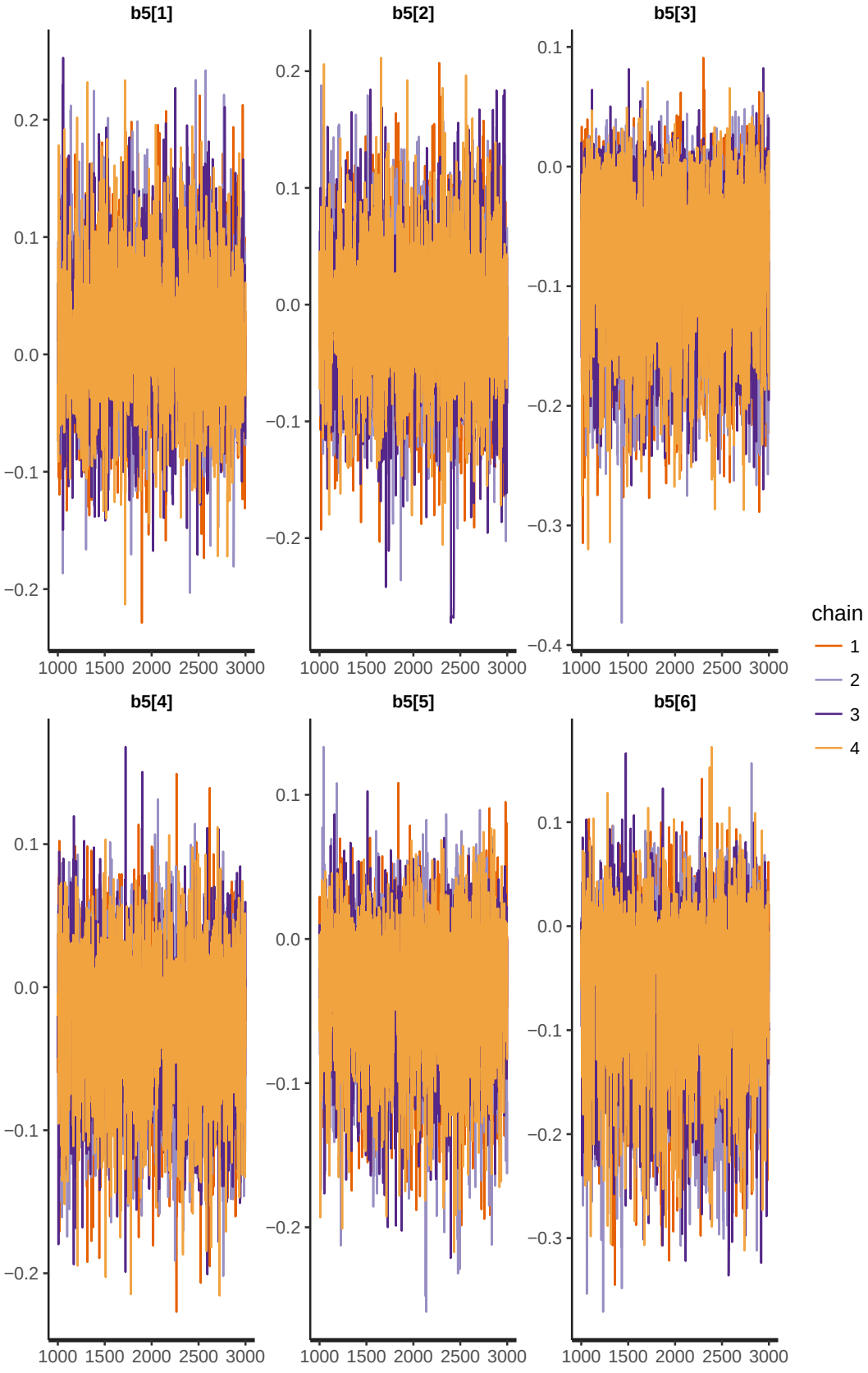
```
## [1] "Coefficients for b3 Over 6 Clusters"
```


b3[1]
b3[2]
b3[3]
b3[4]
b3[5]
b3[6]
chain
1
2
3
4

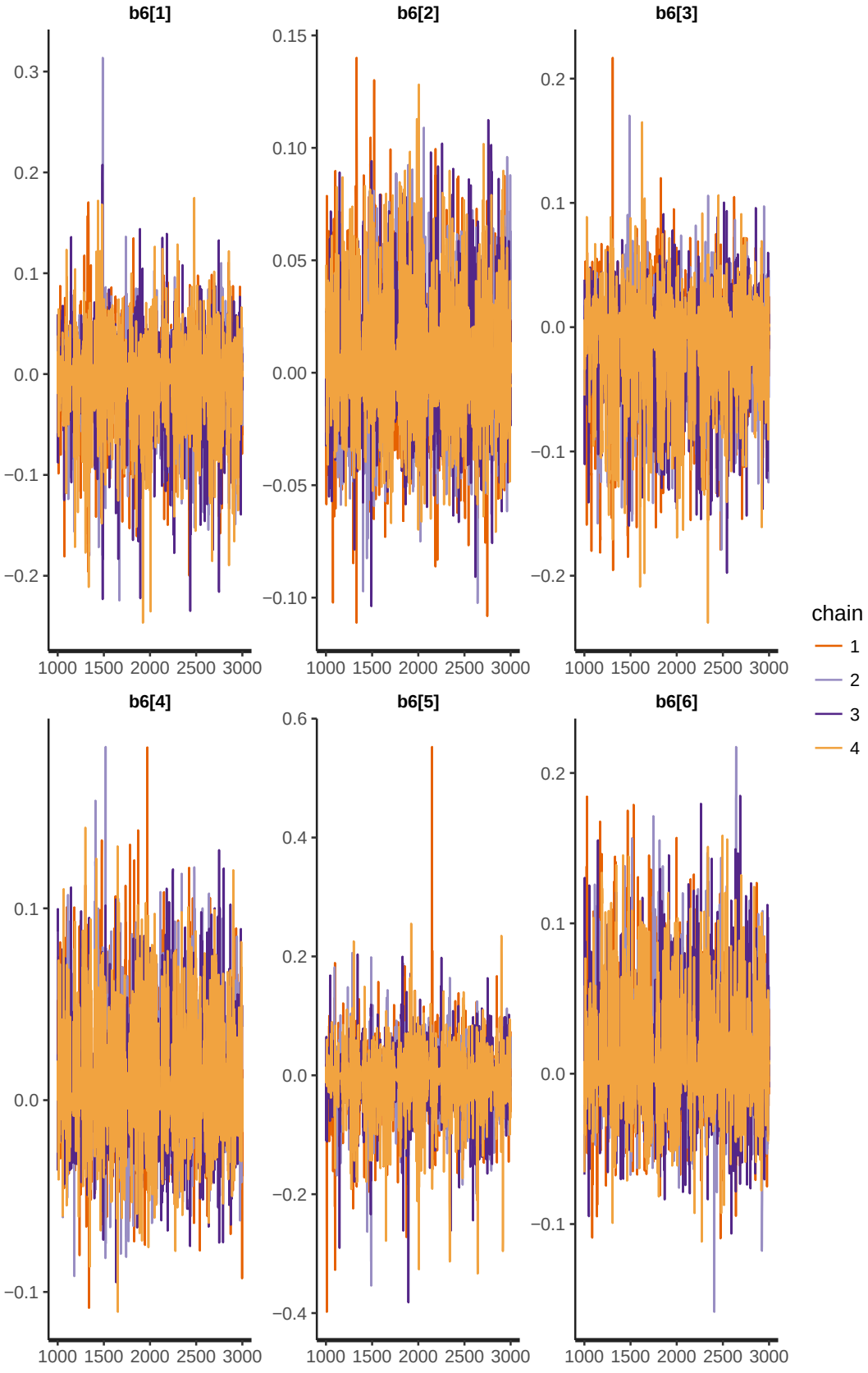
```
## [1] "Coefficients for b4 Over 6 Clusters"
```


b4[1]
b4[2]
b4[3]
b4[4]
b4[5]
b4[6]
0.2
0.1
0.0
−0.1
−0.2
0.15
0.10
0.05
0.00
−0.05
0.50
0.25
0.00
−0.25
−0.10
1000 1500 2000 2500 3000
chain
1
2
3
4

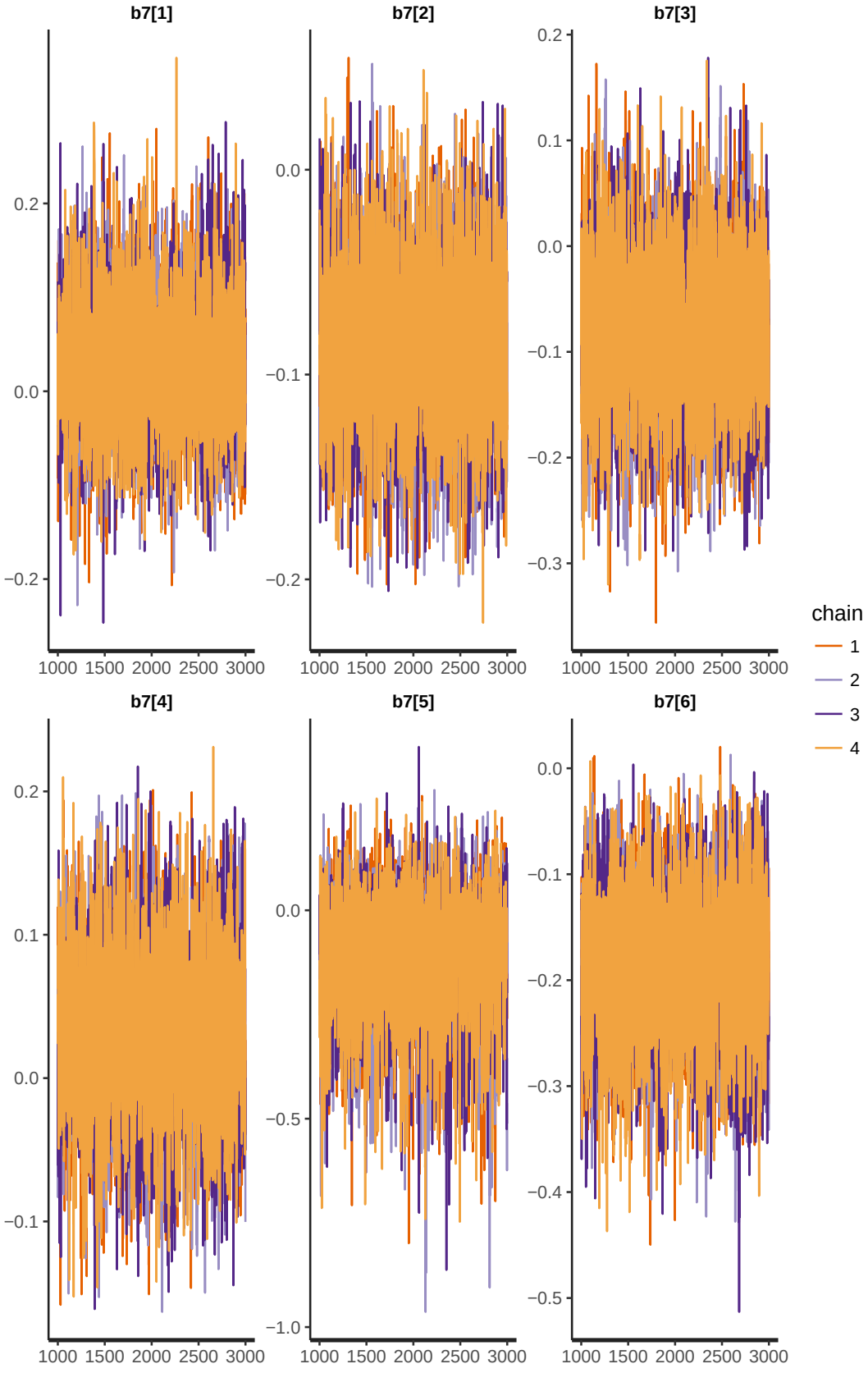
```
## [1] "Coefficients for b5 Over 6 Clusters"
```


b5[1]
b5[2]
b5[3]
b5[4]
b5[5]
b5[6]
1000 1500 2000 2500 3000
chain
1
2
3
4

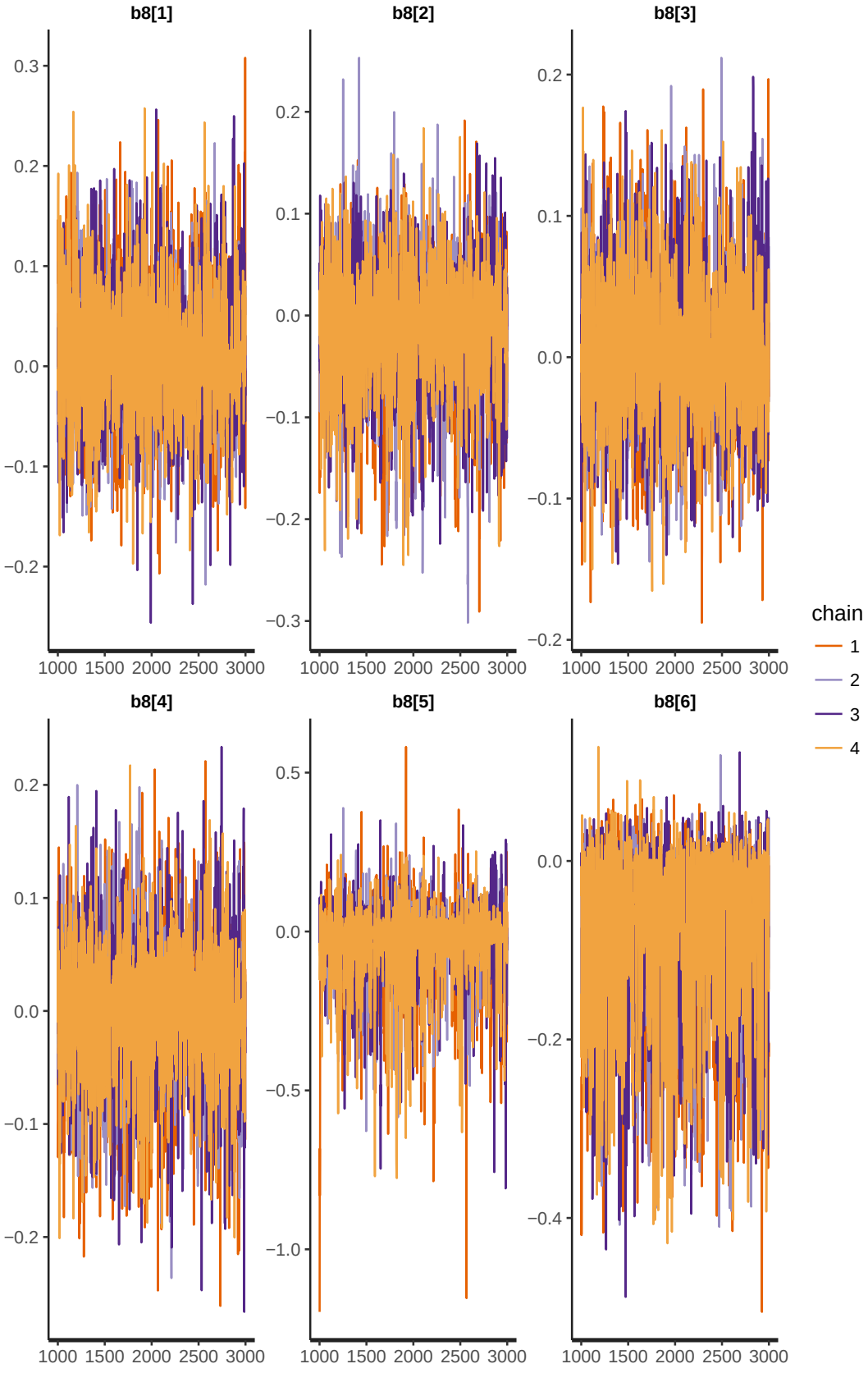
```
## [1] "Coefficients for b6 Over 6 Clusters"
```


b6[1]
b6[2]
b6[3]
b6[4]
b6[5]
b6[6]
chain
1
2
3
4

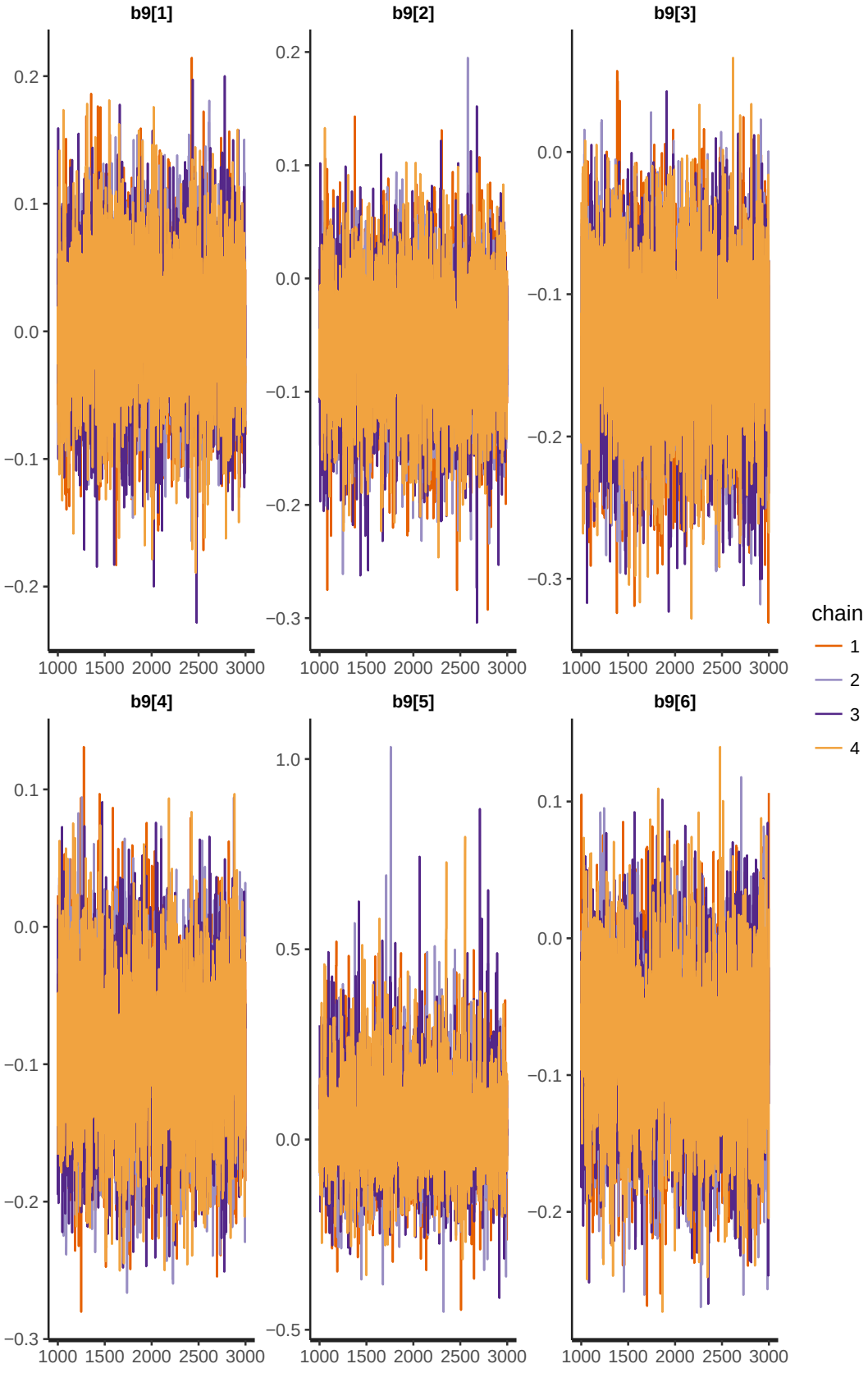
```
## [1] "Coefficients for b7 Over 6 Clusters"
```


b7[1]
b7[2]
b7[3]
b7[4]
b7[5]
b7[6]
0.2
0.0
−0.2
0.1
−0.1
−0.3
−0.5
−1.0
−0.4
1000 1500 2000 2500 3000
chain
1
2
3
4

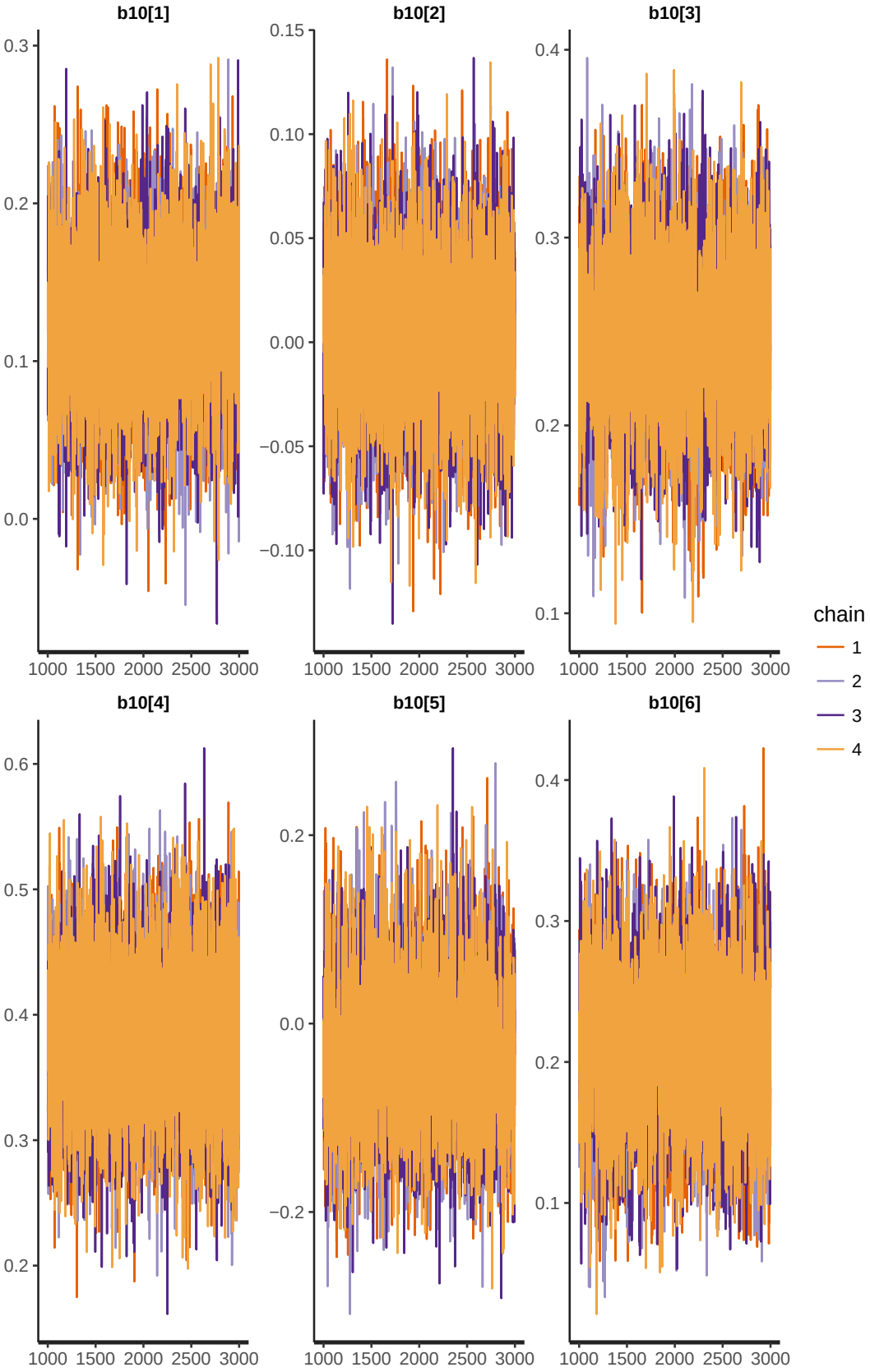
```
## [1] "Coefficients for b8 Over 6 Clusters"
```


b8[1]
b8[2]
b8[3]
b8[4]
b8[5]
b8[6]
chain
1
2
3
4

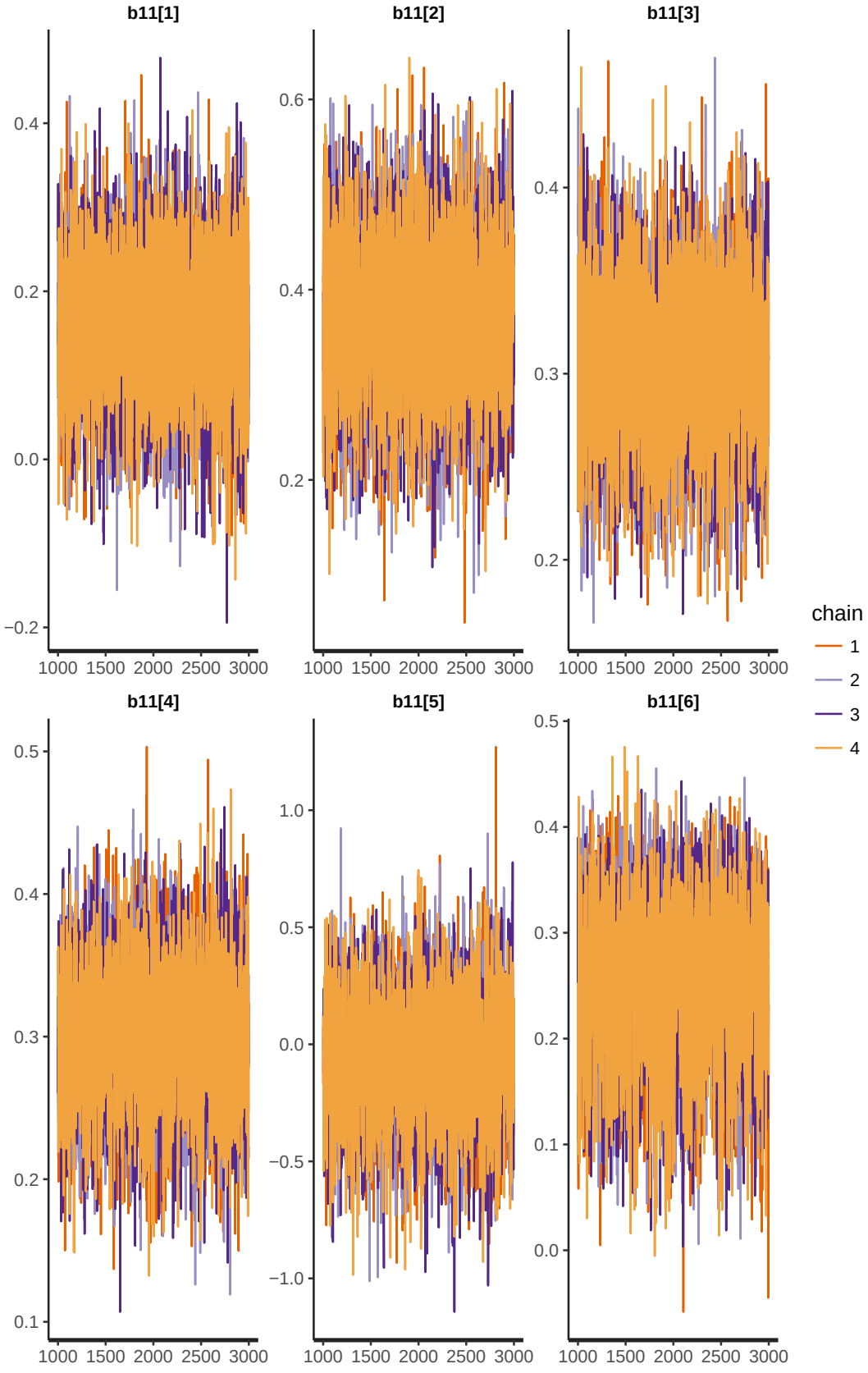
```
## [1] "Coefficients for b9 Over 6 Clusters"
```


b9[1]
b9[2]
b9[3]
b9[4]
b9[5]
b9[6]
chain
1
2
3
4
1000 1500 2000 2500 3000

```
## [1] "Coefficients for b10 Over 6 Clusters"
```

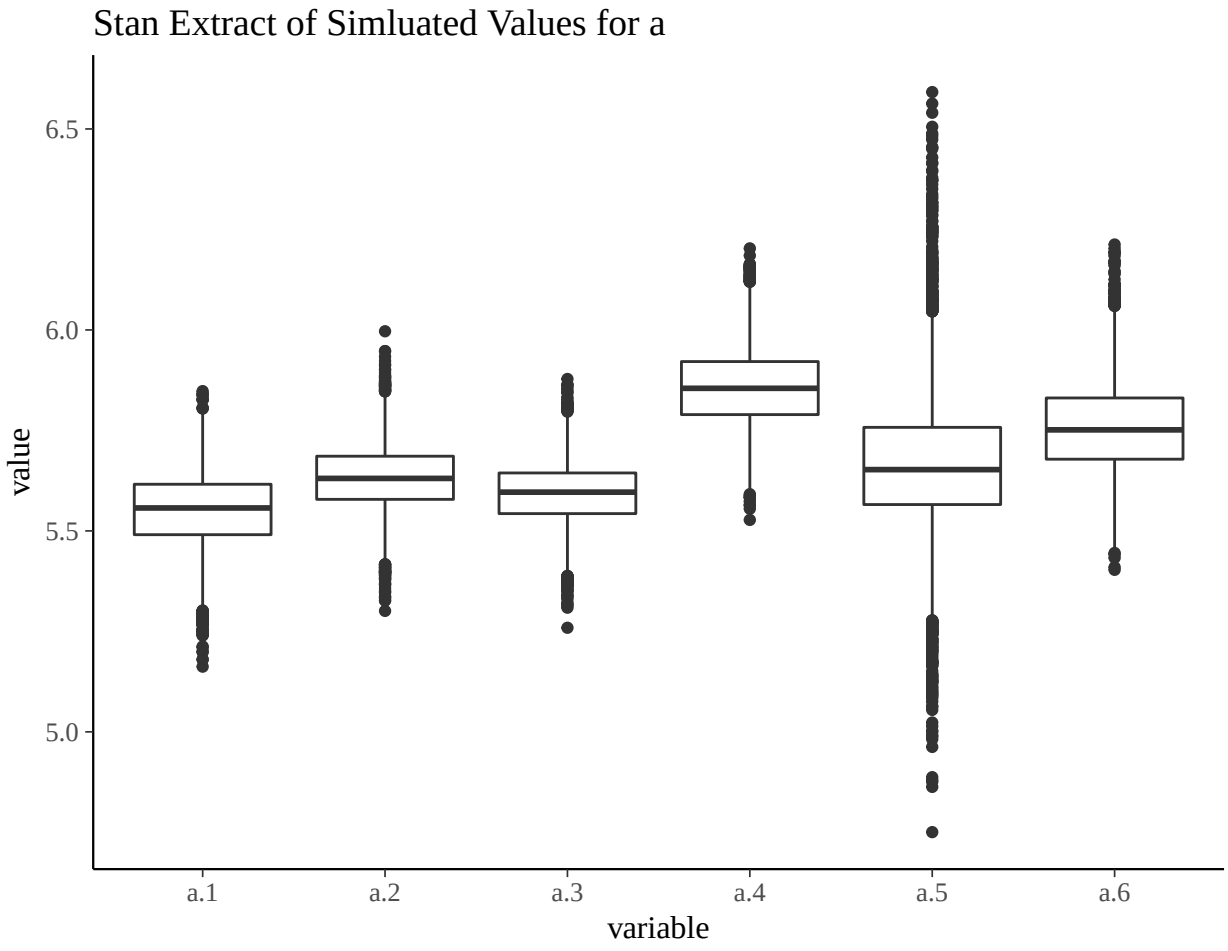

b10[1]
b10[2]
b10[3]
b10[4]
b10[5]
b10[6]
0.3
0.2
0.1
0.0
0.15
0.10
0.05
0.00
−0.05
−0.10
0.4
0.3
0.2
0.1
0.6
0.5
0.4
0.3
0.2
0.2
0.0
−0.2
0.4
0.3
0.2
0.1
1000 1500 2000 2500 3000
chain
1
2
3
4

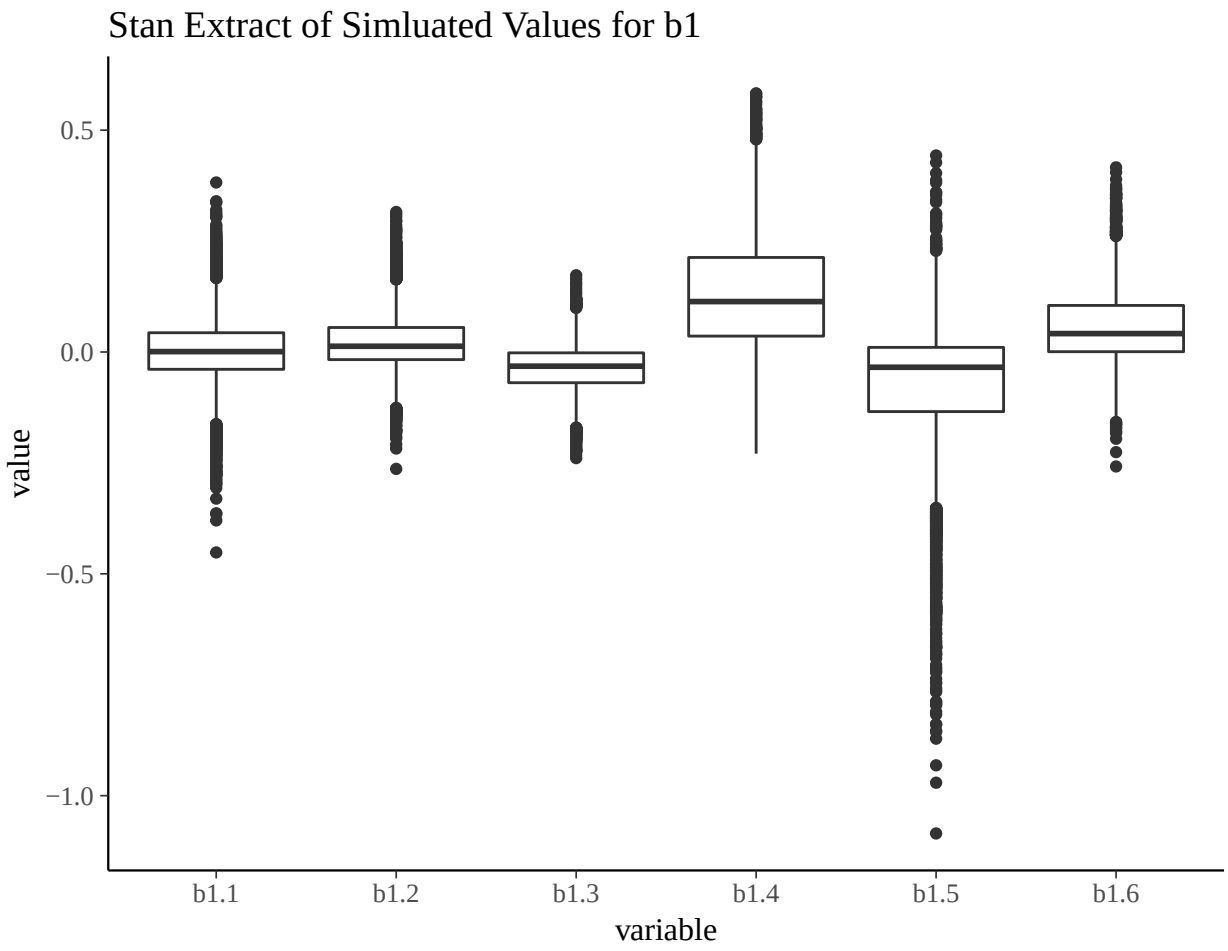
```
## [1] "Coefficients for b11 Over 6 Clusters"
```

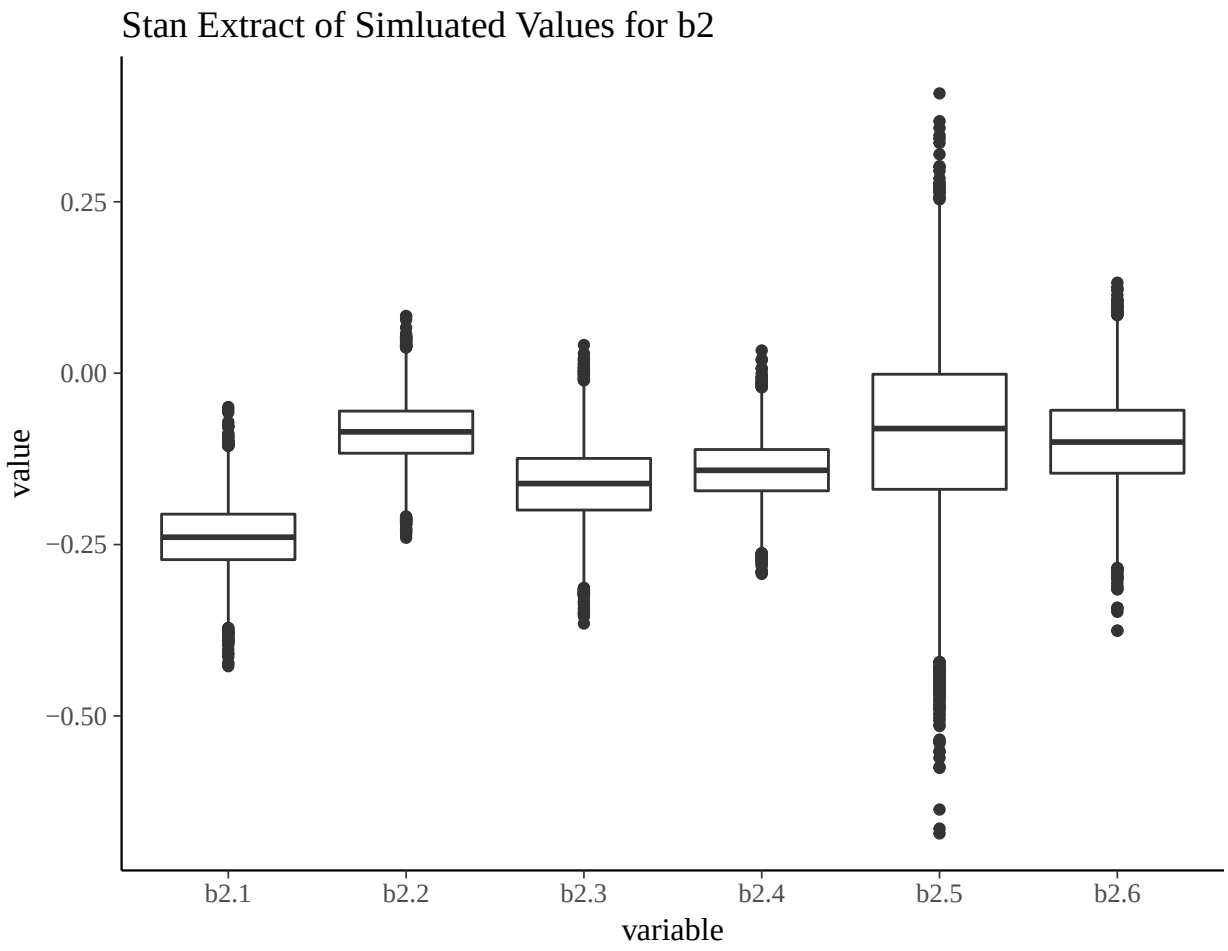

b11[1]
b11[2]
b11[3]
b11[4]
b11[5]
b11[6]
chain
1
2
3
4

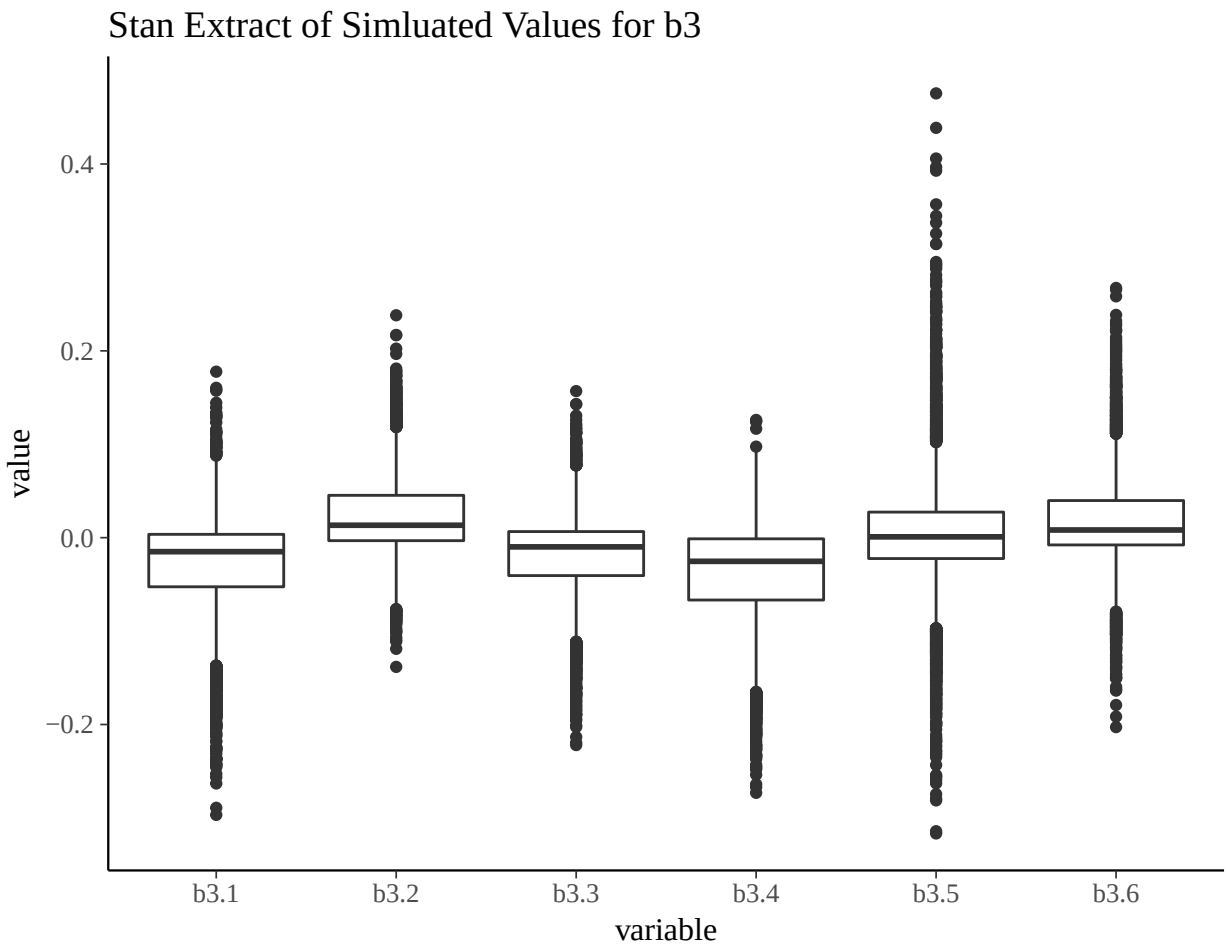
Finally boxplots are generated for each beta over the different clusters.

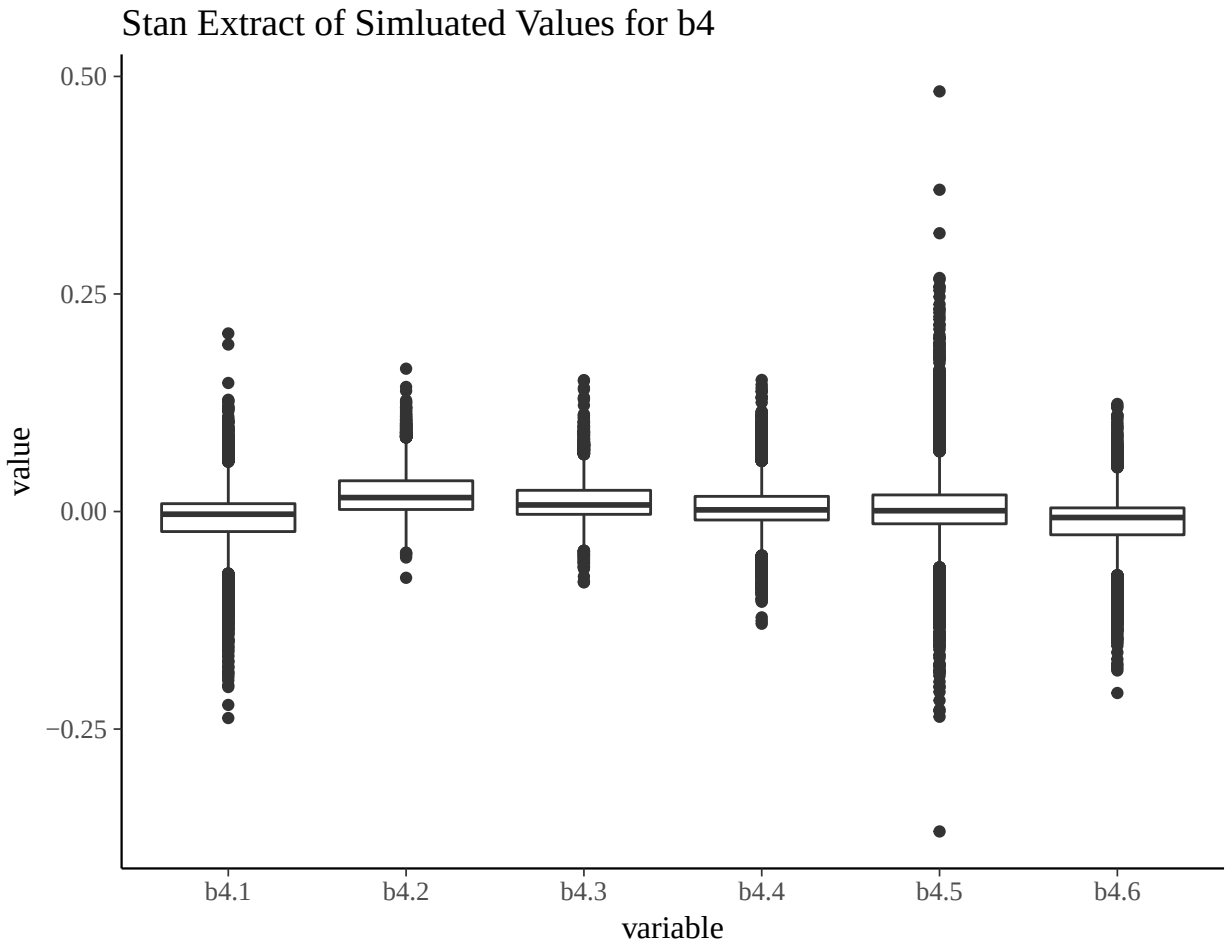
```
stan_extract <- rstan::extract(mod_stan)

for (coef in names(stan_extract)){
  df_temp <- data.frame(stan_extract[coef])
  df_temp_melt <- melt(df_temp)
  if (coef=='sigma_y'){
    break()
  }
  plt <- ggplot(data=df_temp_melt,
                aes(x=variable, y=value, group=variabe)) +
    geom_boxplot(aes(group=variable)) +
    labs(title='Stan Extract of Simluated Values for ' %+% coef)
  print(plt)
}
```

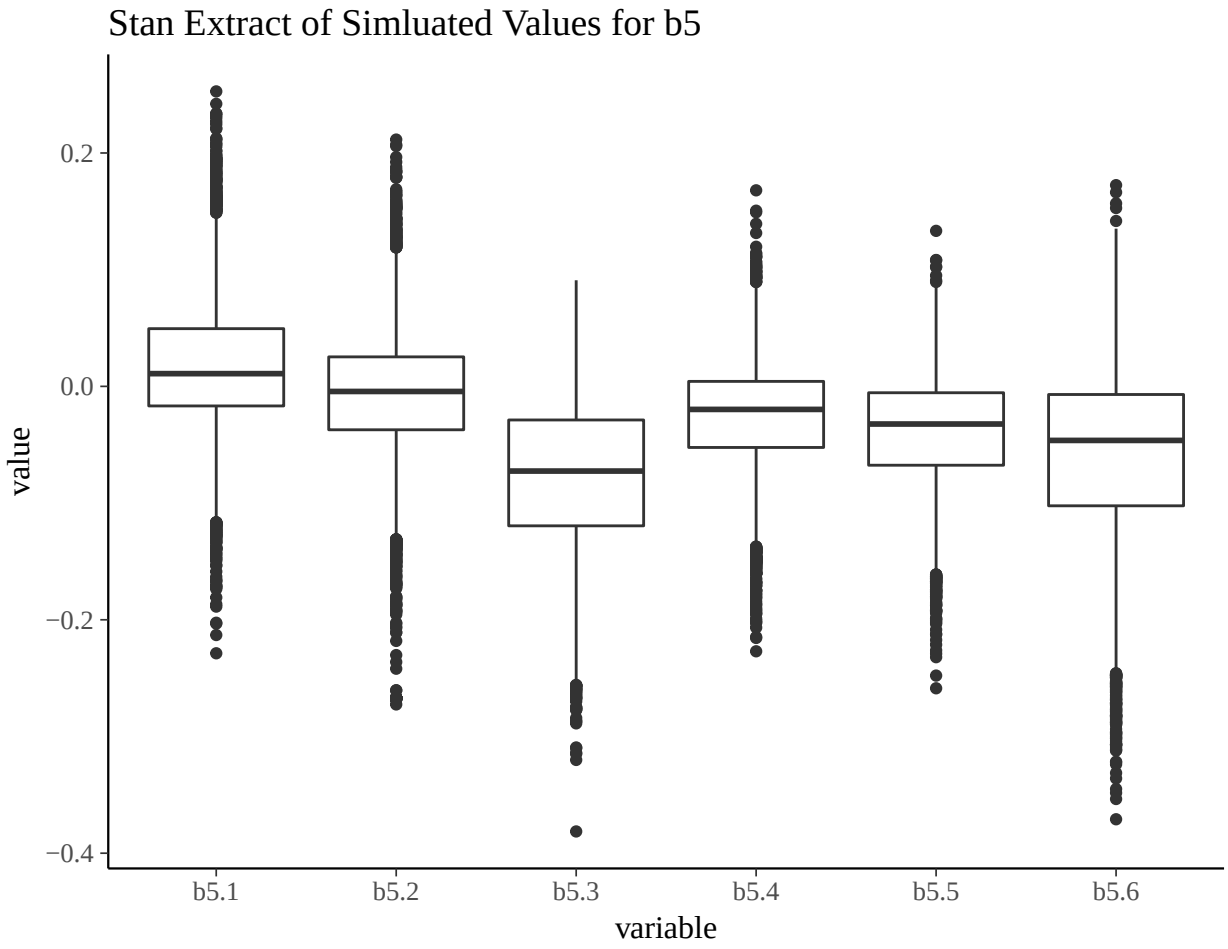


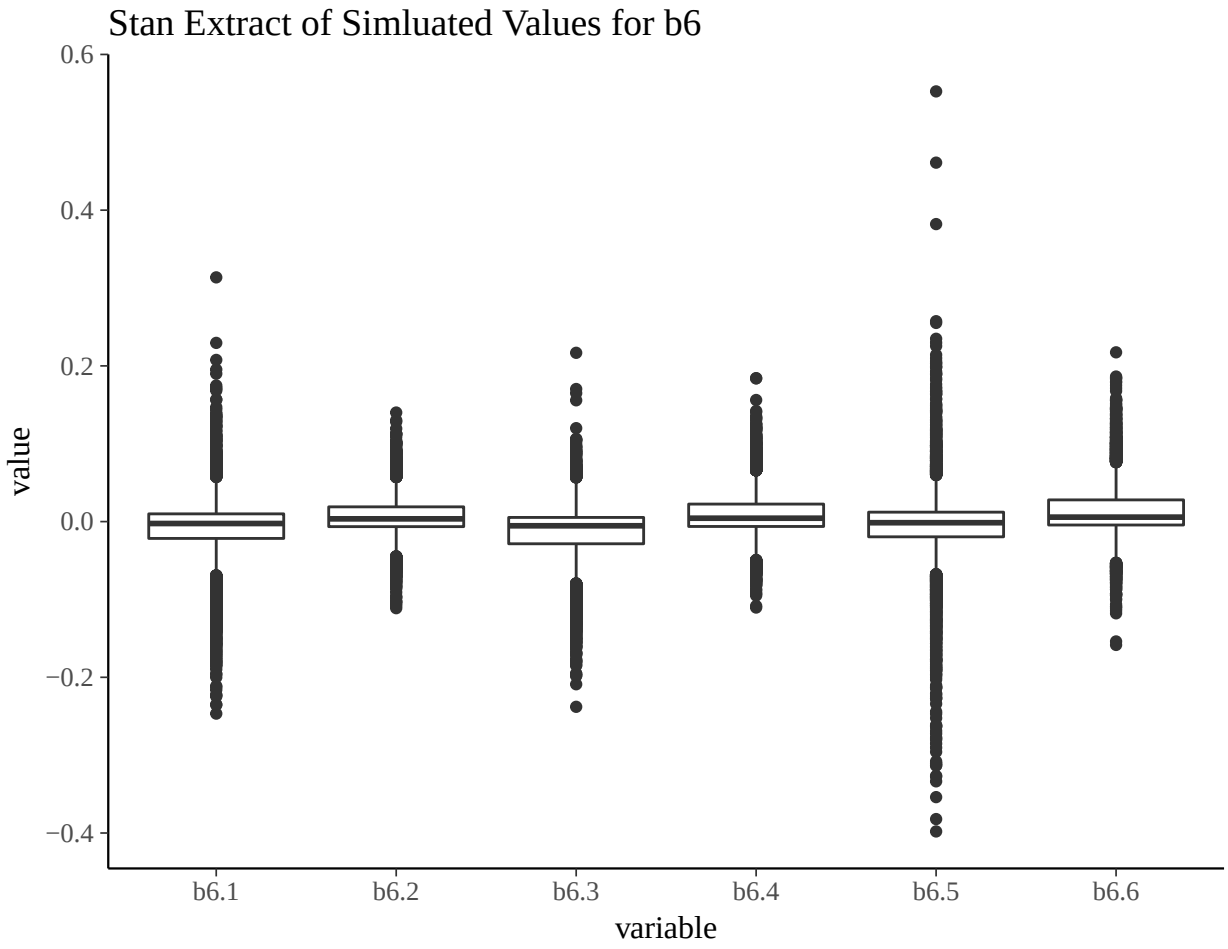

Stan Extract of Simluated Values for b1
0.5
0.0
−0.5
−1.0
value
b1.1
b1.2
b1.3
b1.4
b1.5
b1.6
variable

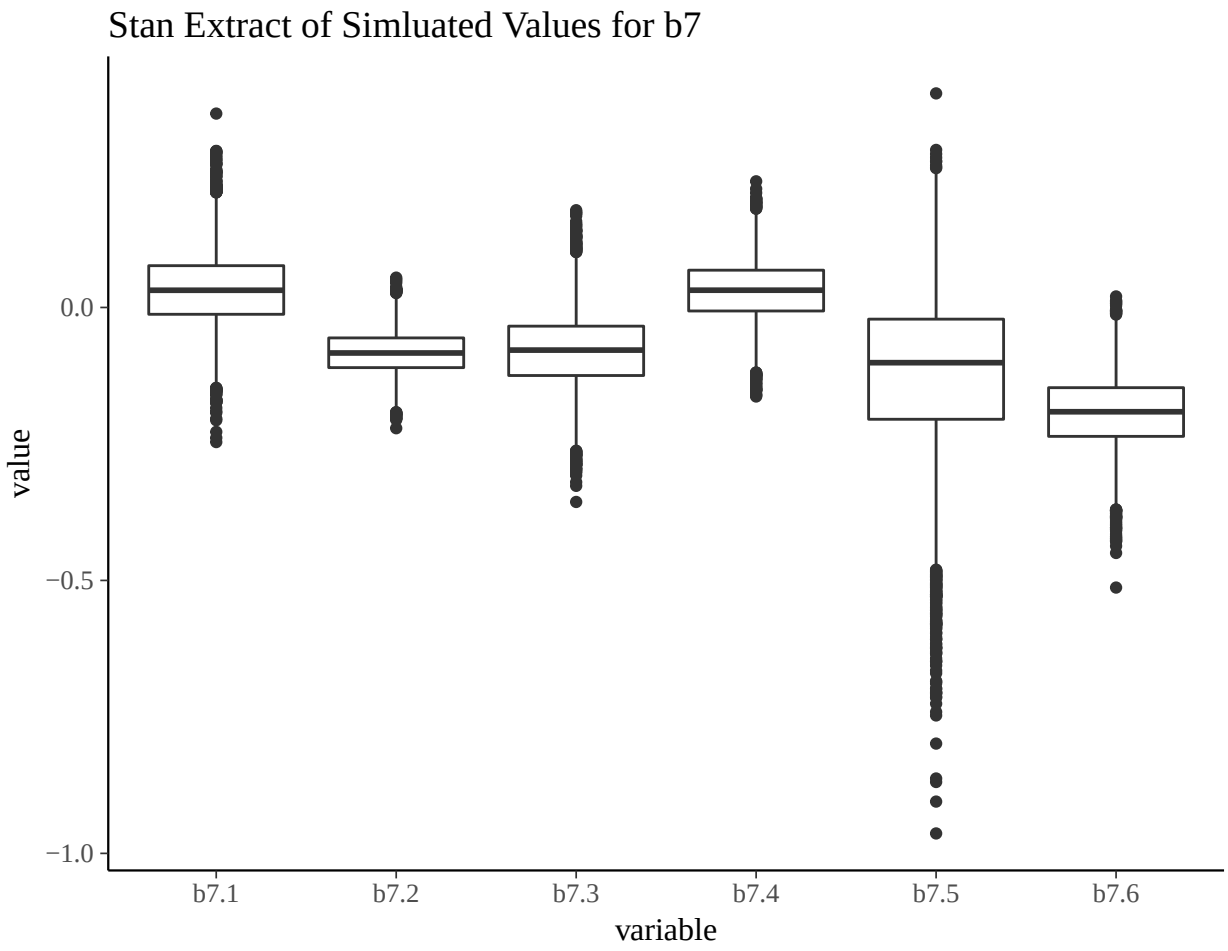

Stan Extract of Simluated Values for b2
value
0.25
0.00
−0.25
−0.50
b2.1
b2.2
b2.3
b2.4
b2.5
b2.6
variable

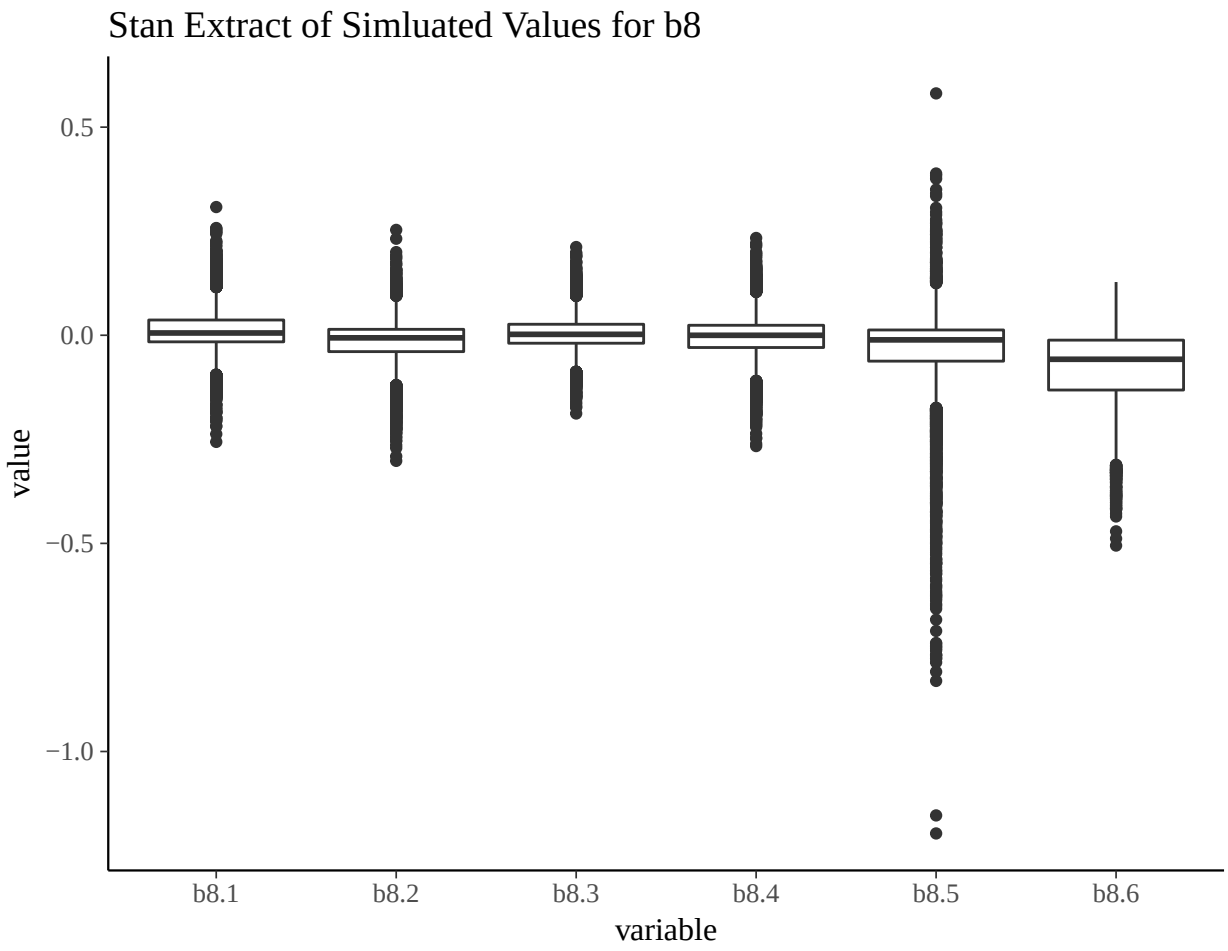

Stan Extract of Simluated Values for b3
value
0.4
0.2
0.0
−0.2
b3.1
b3.2
b3.3
b3.4
b3.5
b3.6
variable

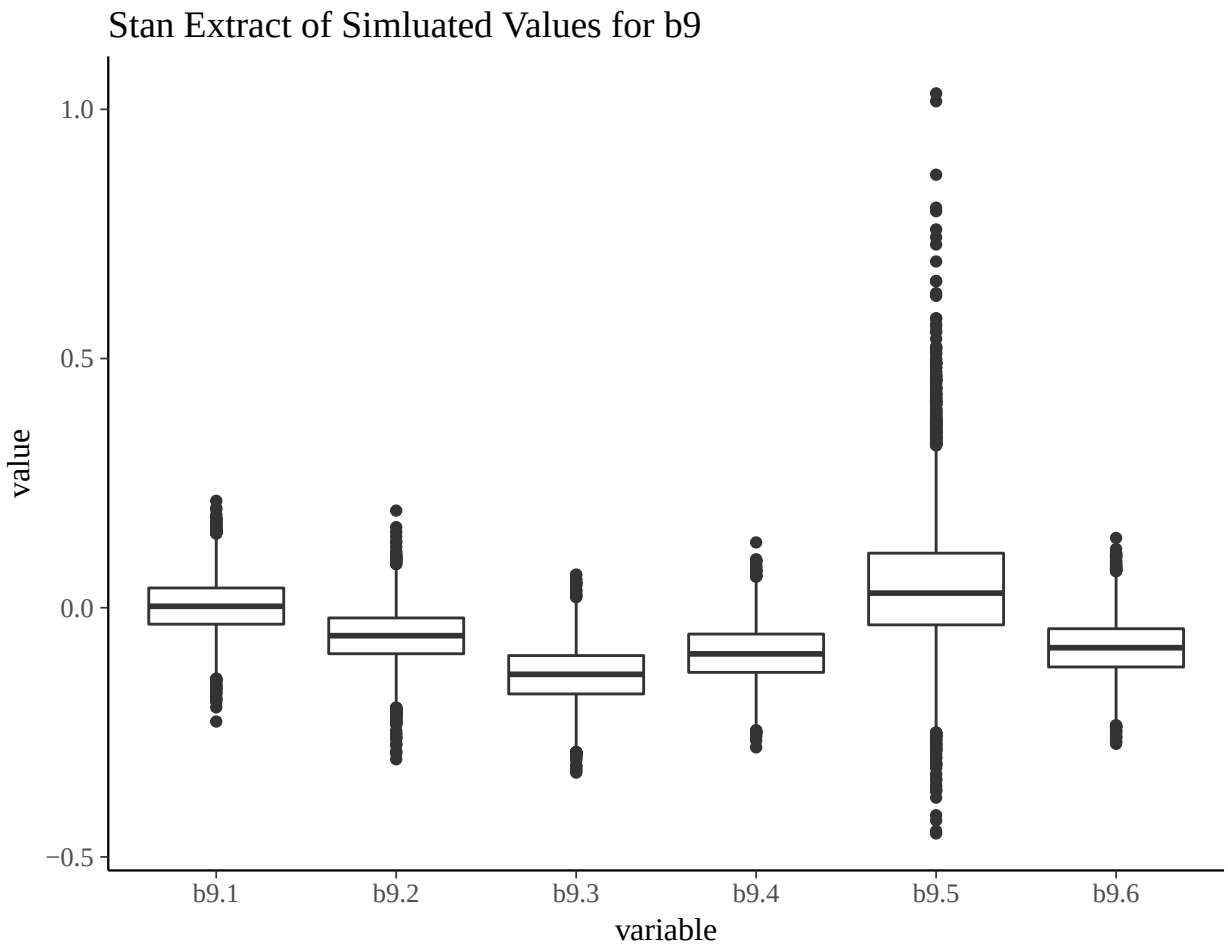

Stan Extract of Simluated Values for b4
0.50
0.25
0.00
−0.25
value
b4.1
b4.2
b4.3
b4.4
b4.5
b4.6
variable

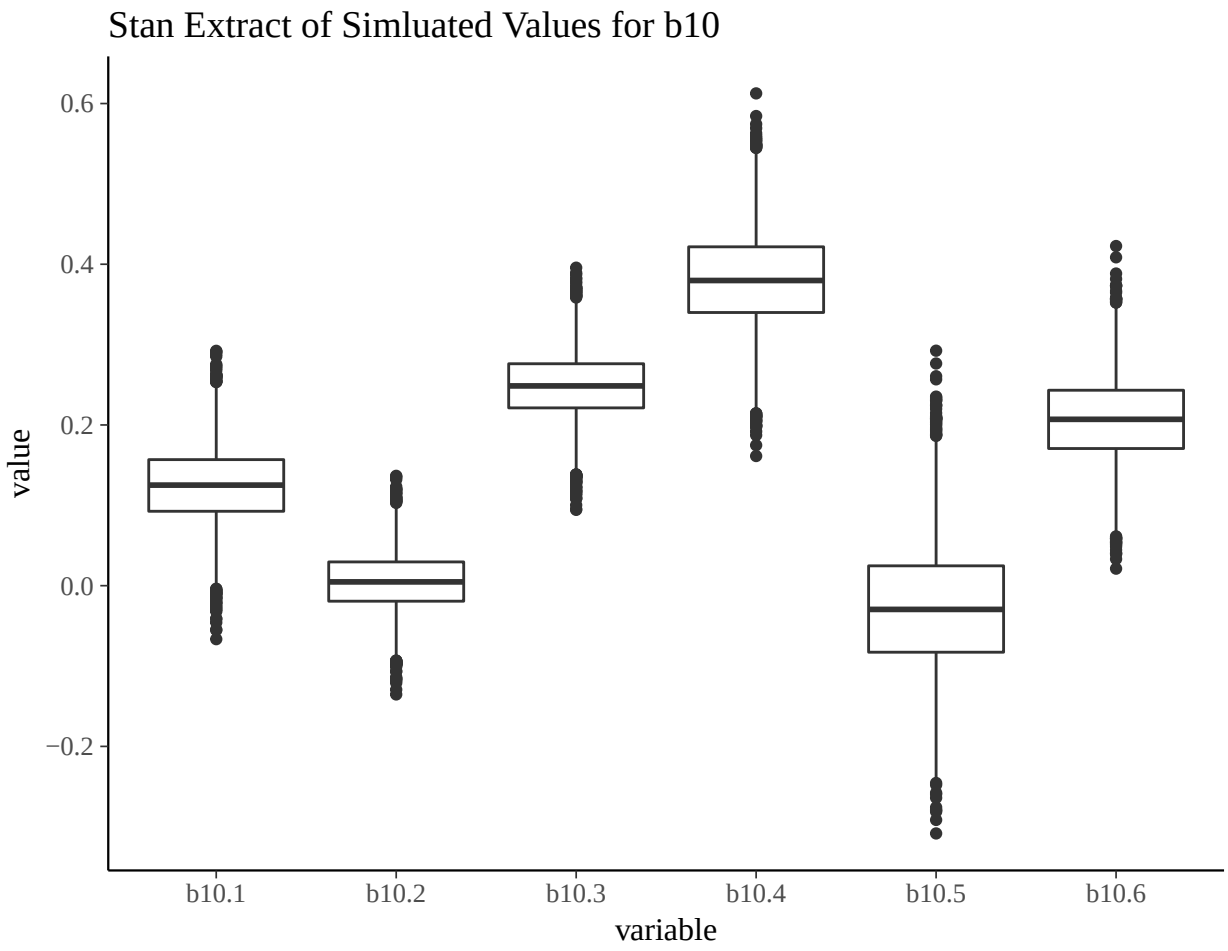

Stan Extract of Simluated Values for b5
0.2
0.0
−0.2
−0.4
value
b5.1
b5.2
b5.3
b5.4
b5.5
b5.6
variable

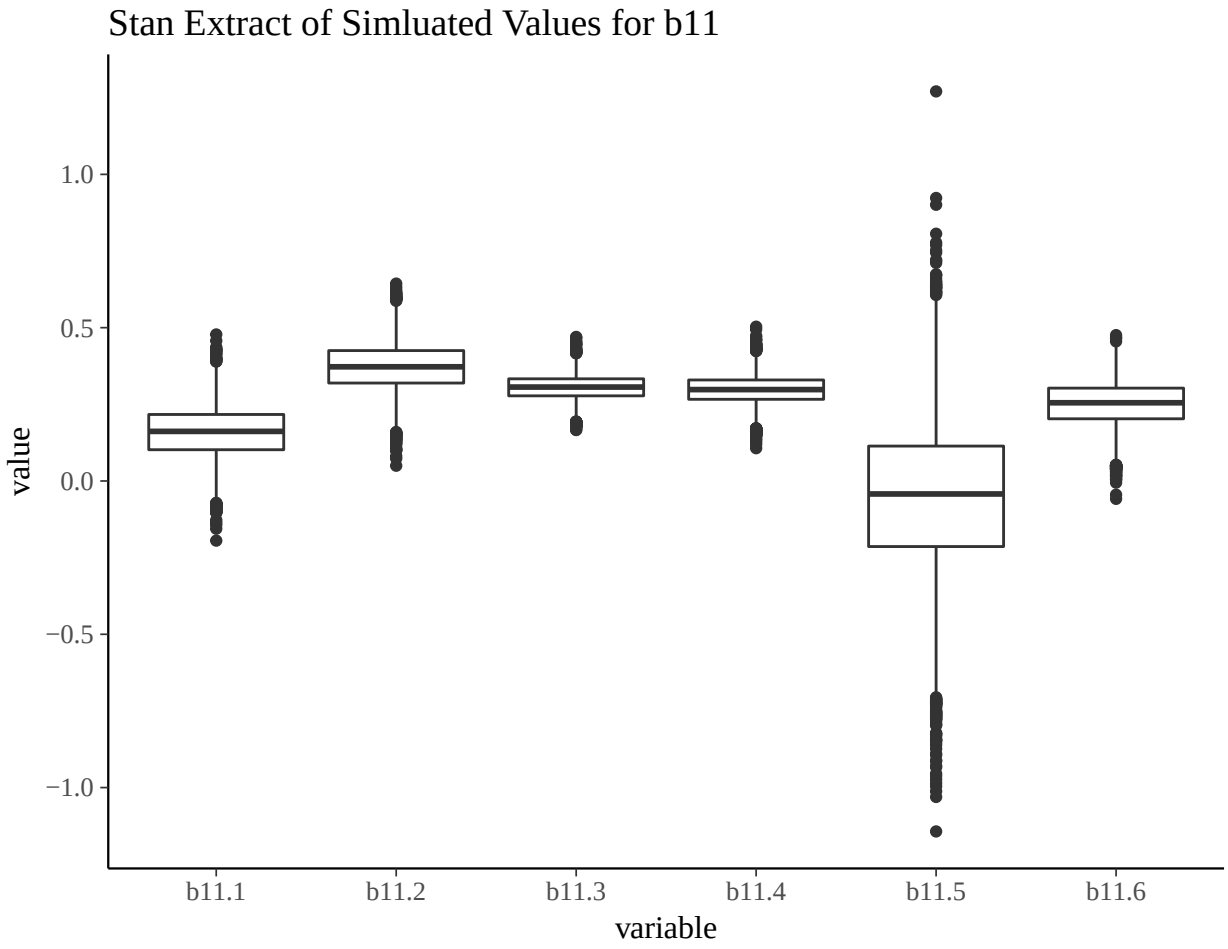

Stan Extract of Simluated Values for b6
0.6
0.4
0.2
0.0
−0.2
−0.4
value
b6.1
b6.2
b6.3
b6.4
b6.5
b6.6
variable


Stan Extract of Simluated Values for b7
value
0.0
−0.5
−1.0
b7.1
b7.2
b7.3
b7.4
b7.5
b7.6
variable


Stan Extract of Simluated Values for b8
0.5
0.0
−0.5
−1.0
value
b8.1
b8.2
b8.3
b8.4
b8.5
b8.6
variable


Stan Extract of Simluated Values for b9
1.0
0.5
0.0
−0.5
value
b9.1
b9.2
b9.3
b9.4
b9.5
b9.6
variable


Stan Extract of Simluated Values for b10
value
0.6
0.4
0.2
0.0
−0.2
b10.1
b10.2
b10.3
b10.4
b10.5
b10.6
variable




The hierarchical model does appear to do a reasonably good job of capturing variation in the same coefficients across the clusters that were generated from the `pam` model run in part 2. It is nto clear that every predictor should be varied over the cluster range. For example `b6` and `b8` do not appear to benefit at all by differentiating its affects by cluster. However other coefficients appear to greatly benefit from the hierarchical method (e.g. `b10`). It is also not clear that the number of clusters is appropriate.