Preliminary ELSA Machine Learnings Tests

Jonathan Seiden, Thu Pham

8/08/2022

```
#There are some issues with NA and NaN in the observation data that will mess up our analysis. We will

getmode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

replace.na <- function(var){
  ifelse(is.na(var) | is.nan(var),
         ifelse(is.factor(var), getmode(var), mean(var, na.rm = TRUE)), var)
}
```

Data consolidation process

In this section, we process the data to get it into a format where each row is a child.

Child and Teacher Observation

First we input the child and teacher observations and process them.

For each child in the COP data, we calculate:

1) The average for the child for each of the indicators across sweeps
2) The class average for each indicator *omitting* the child her/himself
3) The class standard deviation for each indicator *omitting* the child her/himself (only including children with 10 or more sweeps)

We then calculate the class average of the TOP indicators for the adults in the class by averaging across sweeps, and merge this data (one to many) with the child-level data. This merged data set contains XXX children in XXX classes

```
#Input the Year One long child and teacher observation data
y1_child_obs_raw <- read.dta13("y1o_c_long.dta")
y1_teacher_obs_raw <- read.dta13("y1o_t_long.dta")
y1_coverpage_obs <- read.dta13("y1o_coverpage.dta")

#Re-format the child data so that it is one row per child

y1_child_obs <- y1_child_obs_raw %>%
  mutate(cid = ifelse(childid == "N/A", o_c_uniqueid, childid)) %>%
  group_by(cid, classid) %>%
  mutate(nsweeps = n()) %>%
  mutate_at(vars(o_c_verbal:o_c_focus), as.character) %>%
  dplyr::select(c(classid, nsweeps, o_c_verbal:o_c_focus, starts_with("c_m8"))) %>%
```

```
  dummy_cols(select_columns = c("o_c_verbal", "o_c_towhom", "o_c_schedule", "o_c_interaction", "o_c_type
             remove_selected_columns = TRUE) %>%
  group_by(classid, cid) %>%
  # replaced everything() with nsweeps:last_col()
  summarize(across(nsweeps:last_col(), ~ mean(.x, na.rm = TRUE))) %>%
  filter(nsweeps >= 10 ) %>% #THIS IS AN ARBITRARY PARAMETER
  group_by(classid) %>%
  mutate(nclass = n()) %>%
  # ifelse takes care of the case where there is only one student per class
  mutate(across(starts_with("o_c"), ~
                 (ifelse(get('nclass') == 1, .x, ((sum(.x, na.rm = TRUE) -.x) /
                                                    get('nclass')))), .names =
                 "{col}_classmean")) %>%
  mutate(across(starts_with("o_c") & !ends_with("classmean"), ~
                 (ifelse(get('nclass') == 1, 0, sqrt((sum((.x - get(str_c(cur_column(),
                                                                          '_classmean')))^2) -
                                                       (.x - get(str_c(cur_column(),
                                                                       '_classmean')))^2) /
                                                      get('nclass')))), .names =
                 "{col}_classsd")) %>%
  ungroup
```

```
## Adding missing grouping variables: `cid`
## `summarise()` has grouped output by 'classid'. You can override using the
## `.groups` argument.
```

```
#Re-format the teacher data so that it is one row per class
y1_teacher_obs <- y1_teacher_obs_raw %>%
  dummy_cols(select_columns = c("o_t_verbal_o", "o_t_whom_o", "o_t_schedule_o",
                                "o_t_task_o", "o_t_instruct", "o_t_focus_o",
                                "o_t_tone_o", "o_t_attention_o", "o_t_es_o"),
             remove_selected_columns = TRUE) %>%
    group_by(classid) %>%
  summarize(
    nsweeps = n(),
    nadult = length(unique(o_t_uniqueid)),
    across(starts_with(c("o_t_verbal_o", "o_t_whom_o", "o_t_schedule_o",
                         "o_t_task_o", "o_t_instruct", "o_t_focus_o",
                         "o_t_tone_o", "o_t_attention_o", "o_t_es_o", "m8")),
           ~ mean(.x, na.rm = TRUE))) %>%
  dplyr::select(-ends_with("_")) %>%
  ungroup

#Merge teacher and child observations
y1_obs <- left_join(y1_child_obs, y1_teacher_obs, by = "classid")
```

```
table(y1_teacher_obs$nadult)
```

```
##
##   1   2   3   4   5
## 131 376 135  25   5
```

```
#Input the Year One long child and teacher observation data
y2_child_obs_raw <- read.dta13("y2o_c_long.dta")
y2_teacher_obs_raw <- read.dta13("y2o_t_long.dta")
```

```
y2_coverpage_obs <- read.dta13("y2o_coverpage.dta")

mean(y2_coverpage_obs$classid %in% y1_coverpage_obs$classid)
```

```
## [1] 0.2031008
```

```
#Re-format the child data so that it is one row per child
y2_child_obs <- y2_child_obs_raw %>%
 mutate(cid = ifelse(cid == "N/A", o_c_uniqueid, cid)) %>%
  group_by(cid, classid) %>%
  mutate(nsweeps = n()) %>%
  mutate_at(vars(o_c_verbal:o_c_focus), as.character) %>%
  dplyr::select(c(classid, nsweeps, o_c_verbal:o_c_focus, starts_with("c_m8"))) %>%
  dummy_cols(select_columns = c("o_c_verbal", "o_c_towhom", "o_c_schedule", "o_c_interaction", "o_c_typ
             remove_selected_columns = TRUE) %>%
  group_by(classid, cid) %>%
  # replaced everything() with nsweeps:last_col()
  summarize(across(nsweeps:last_col(), ~ mean(.x, na.rm = TRUE))) %>%
  filter(nsweeps >= 10 ) %>% #THIS IS AN ARBITRARY PARAMETER
  group_by(classid) %>%
  mutate(nclass = n()) %>%
    # ifelse takes care of the case where there is only one student per class
  mutate(across(starts_with("o_c"), ~
                 (ifelse(get('nclass') == 1, .x, ((sum(.x, na.rm = TRUE) -.x) /
                                                    get('nclass')))), .names =
                 "{col}_classmean")) %>%
  mutate(across(starts_with("o_c") & !ends_with("classmean"), ~
                 (ifelse(get('nclass') == 1, 0, sqrt((sum((.x - get(str_c(cur_column(),
                                                                          '_classmean')))^2) -
                                                       (.x - get(str_c(cur_column(),
                                                                       '_classmean')))^2) /
                                                      get('nclass')))), .names =
                 "{col}_classsd")) %>%
  ungroup
```

```
## Adding missing grouping variables: `cid`
## `summarise()` has grouped output by 'classid'. You can override using the
## `.groups` argument.
```

```
#Re-format the teacher data so that it is one row per class
y2_teacher_obs <- y2_teacher_obs_raw %>%
  dummy_cols(select_columns = c("o_t_verbal_o", "o_t_whom_o", "o_t_schedule_o",
                                "o_t_task_o", "o_t_instruct", "o_t_focus_o",
                                "o_t_tone_o", "o_t_attention_o", "o_t_es_o"),
             remove_selected_columns = TRUE) %>%
    group_by(classid) %>%
  summarize(
    nsweeps = n(),
    nadult = length(unique(o_t_uniqueid)),
    across(starts_with(c("o_t_verbal_o", "o_t_whom_o", "o_t_schedule_o",
                         "o_t_task_o", "o_t_instruct", "o_t_focus_o",
                         "o_t_tone_o", "o_t_attention_o", "o_t_es_o", "m8")),
           ~ mean(.x, na.rm = TRUE))) %>%
  dplyr::select(-ends_with("_")) %>%
  ungroup
```

```
#Extract the caretype from the observation sheet
## ASK -- what is meant by this?

#Merge teacher and child observations
y2_obs <- left_join(y2_child_obs, y2_teacher_obs, by = "classid")
y2_obs <- y2_obs %>%
  rename_at(vars(everything()), ~str_replace_all(., "\\s+", ""))
```

Below we now input the child-level outcome data. We focus on the outcomes that Emily suggested, and extract the year 1 and year 2 values for each child and then merge to create a single dataset.

```
#Get Year 1 and Year 2 child data
y1_child_outcomes_raw <- read.dta13("y1c.dta")
y2_child_outcomes_raw <- read.dta13("y2c.dta")

#Rename all y1 variables and y2 variables so we don't lose them when merging
y1_child_outcomes <-  y1_child_outcomes_raw %>%
  dplyr::select(cid, c_mefs_str, c_pt_pcorrect, c_ltr_cogsoc_comp, c_ltr_emo_comp,
         c_pra_total, c_pbsa_total, c_quils_total_raw, c_wjlw_str, c_wjap_str) %>%
  rename_all( ~ paste0("y1_", .x)) %>%
  mutate(cid = as.character(y1_cid))

y2_child_outcomes <-  y2_child_outcomes_raw %>%
  dplyr::select(cid, c_mefs_str, c_pt_pcorrect, c_ltr_cogsoc_comp, c_ltr_emo_comp,
         c_pbsa_allgrades_total, c_pra_allgrades_total, c_quils_total_raw,
         c_wjlw_str, c_wjap_str, c_age_cat_test, c_age_test) %>%
  rename(c_pra_total = c_pra_allgrades_total,
         c_pbsa_total = c_pbsa_allgrades_total) %>%
  rename_all( ~ paste0("y2_", .x)) %>%
  mutate(cid = as.character(y2_cid))

#Merge Y2 and Y1 data together and calculate the gain score for each of the outcomes
child_outcomes <- merge(y1_child_outcomes, y2_child_outcomes, by = "cid") %>%
  mutate(gain_c_mefs_str = y2_c_mefs_str - y1_c_mefs_str,
         gain_c_pt_pcorrect = y2_c_pt_pcorrect - y1_c_pt_pcorrect,
         gain_c_ltr_cogsoc_comp = y2_c_ltr_cogsoc_comp - y1_c_ltr_cogsoc_comp,
         gain_c_ltr_emo_comp = y2_c_ltr_emo_comp - y1_c_ltr_emo_comp,
         gain_c_pra_total = y2_c_pra_total - y1_c_pra_total,
         gain_c_pbsa_total = y2_c_pbsa_total - y1_c_pbsa_total,
         gain_c_quils_total_raw = y2_c_quils_total_raw - y1_c_quils_total_raw,
         gain_c_wjlw_str = y2_c_wjlw_str - y1_c_wjlw_str,
         gain_c_wjap_str = y2_c_wjap_str - y1_c_wjap_str,
         cid = as.numeric(cid))
```

Finally, we merge together the Year 1 observation data with the Year 1 & 2 child outcome data and add in care type. We omit observations that have no classroom observation resulting in a total analytic dataframe of 1169 observations of 64 variables.

```
#Merge in the outcomes data
y1_obs <- y1_obs %>%
  filter(!is.na(cid)) %>%
  mutate(cid = as.numeric(cid))
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
outcomes_and_obs_y1 <- left_join(child_outcomes, y1_obs, by = "cid") %>%
  mutate(cid = as.character(cid))

#Add in the care type
caretype <- read.dta13("y1caretype.dta") %>%
  mutate(cid = as.character(cid))

#Remove observations that have no care type or no classroom observation
outcomes_and_obs_full_y1 <- left_join(outcomes_and_obs_y1, caretype, by = "cid") %>%
  mutate(hasobservation = is.na(classid)) %>%
  filter(!is.na(caretype)) %>%
  filter(!is.na(classid))

# outcomes_and_obs_full_y1 <- outcomes_and_obs %>%
#   filter(!is.na(classid))

#Remove Y1 and Y2 data for cleanliness
outcomes_and_obs_full_y1 <- outcomes_and_obs_full_y1 %>%
  dplyr::select(-starts_with(c("y1", "y2")))

#Remove some irrelevant variables and rename columns with
#illegal spaces
outcomes_and_obs_full_y1 <- outcomes_and_obs_full_y1 %>%
  dplyr::select(-c(famid, dob:dob_uncertain, actual_fcc:hasobservation)) %>%
  mutate(caretype = as.factor(caretype),
         actualtype = as.factor(actualtype)) %>%
  rename_at(vars(everything()), ~str_replace_all(., "\\s+", ""))
```

```
# names(outcomes_and_obs_full_y1)
outcomes_and_obs_full_y1 <- outcomes_and_obs_full_y1 %>%
  mutate_at(vars(c_m8_goal_1:actualtype), replace.na)

# what is o_t_sched_t supposed to be for? replacing with actualtype for now just to run analysis

# replacing all NaNs in hopes that it will fix the model matrix issue
# originally was o_c_verbal_talk:o_t_sched_t

dim(outcomes_and_obs_full_y1)
```

```
## [1] 1169  293
```

Analysis

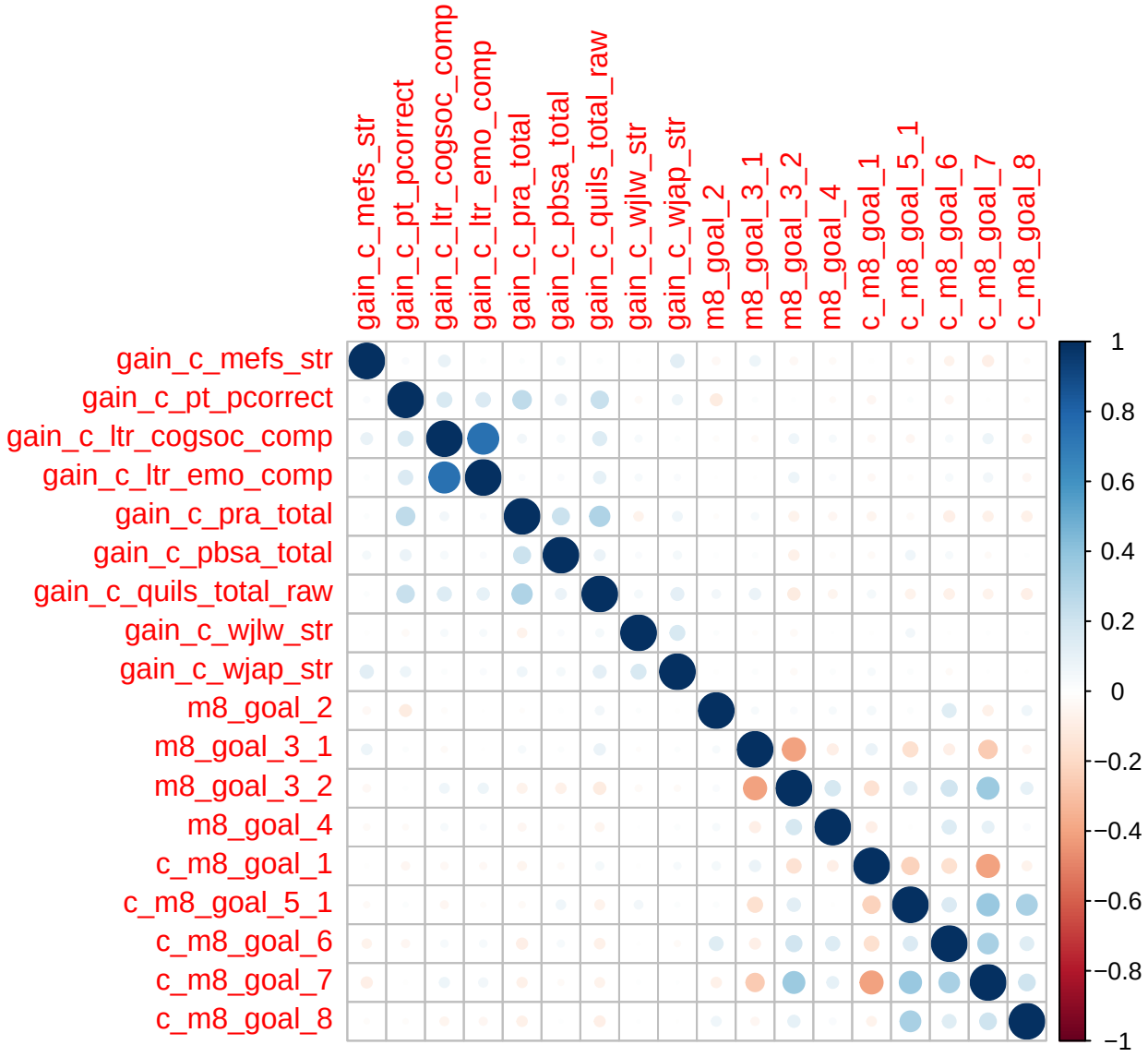
Before anything, let's just see how the "Magic 8" variables concord predict gains?

```
m8 <- outcomes_and_obs_full_y1 %>%
  dplyr::select(starts_with(c("gain", "m8", "c_m8")))

library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
m8cor <- cor(m8,
    use = "pairwise.complete.obs")
```

```
corrplot(m8cor)
```



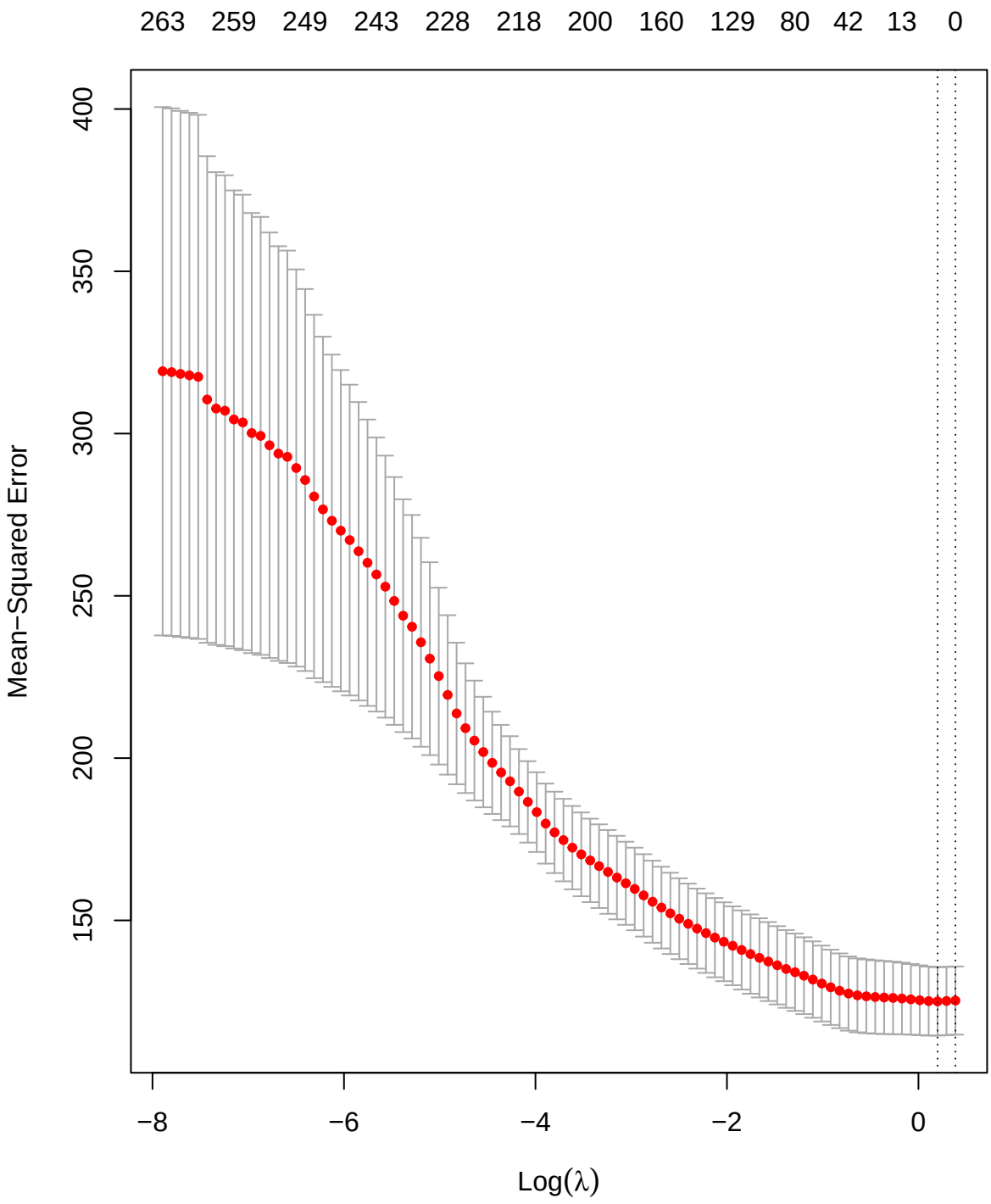
We will first try a cross-validated LASSO, which will aggressively remove variables that do little to improve the predictive accuracy of the model.

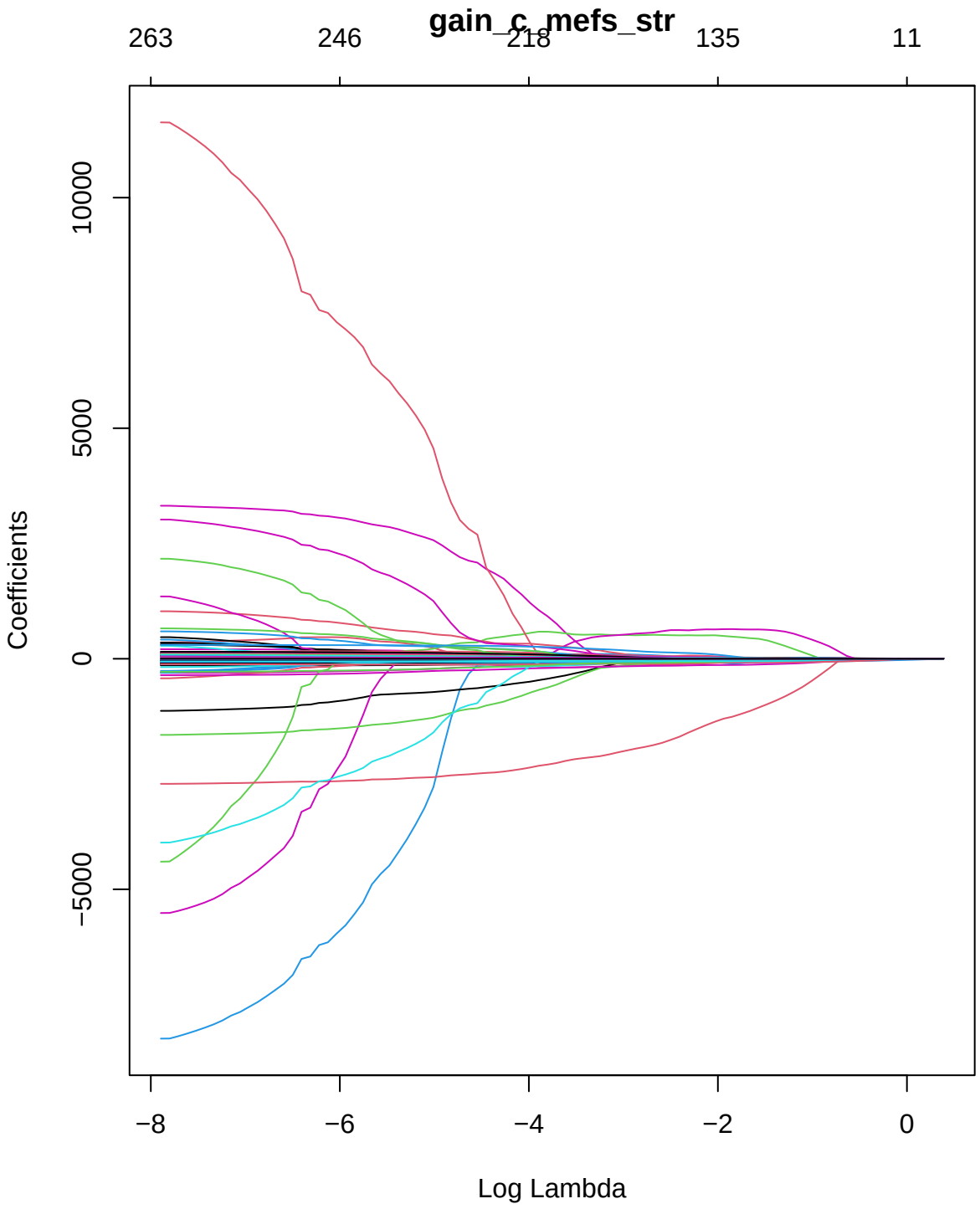
```
# looping through gain variables

# initiate two lists to store results of each model (for the graph and the coefficients)

# https://stackoverflow.com/questions/9332417/whats-a-good-way-to-store-multiple-models-in-an-r-data-st
set.seed(4224)

gain_ind <- which(startsWith(colnames(outcomes_and_obs_full_y1), "gain"))
models_y1 <- list()
coefs_y1 <- list()

# double check that the number of rows matches up with number of non-NAs
for (i in gain_ind) {
```

```
  name <- names(outcomes_and_obs_full_y1)[i]
  df_analysis <- outcomes_and_obs_full_y1 %>%
    filter(!is.na(outcomes_and_obs_full_y1[[name]])) %>%
    dplyr::select(c(name, c_m8_goal_1:actualtype))
  allSd <- apply(df_analysis[, -i], 2, sd)
  print(name)
  print(sum(!is.na(outcomes_and_obs_full_y1[[name]])))
  x = model.matrix(as.formula(paste(name, "~ .")), data = df_analysis)

  y = df_analysis[[name]]
  print(dim(x))
  x = x[, -1]


  # call cv.glmnet()
  model_lasso <- cv.glmnet(x = x, y = y, alpha = 1)
  plot(model_lasso)
  plot(model_lasso$glmnet.fit, "lambda", main=name)

  models_y1[[name]] <- model_lasso

  cc = coef(model_lasso, s = model_lasso$lambda.min)

  # print out the model coefficients and store in a list.
  # exclude the intercept
  cc = cc[cc[,1]!=0,1][-1]
  # remove backticks for ease of standardizing
  names(cc)<- gsub("`","", names(cc))
  coefs_y1[[name]] <- cc * allSd[names(cc)]
  # print(cc)

}
```

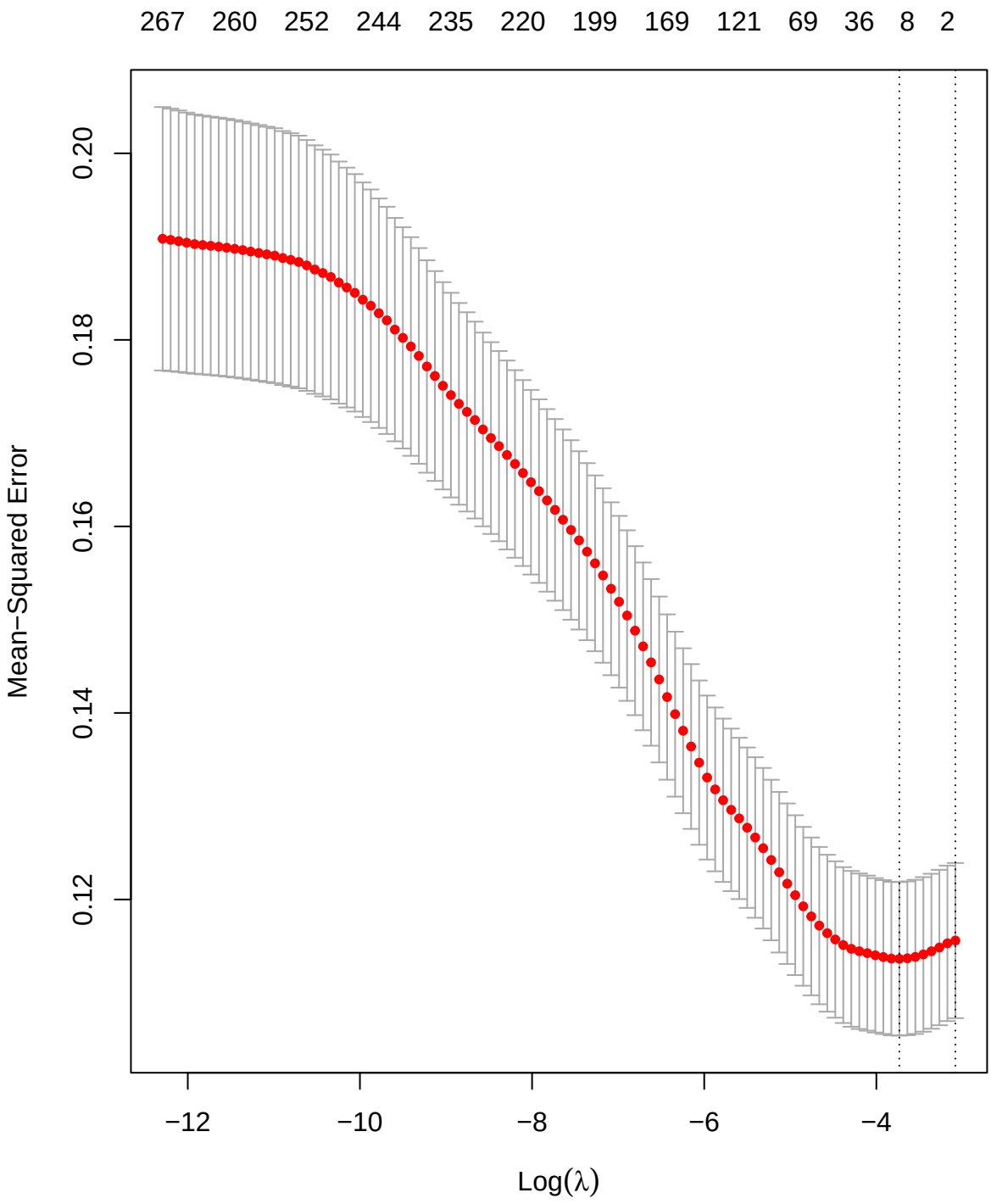
```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(name)` instead of `name` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

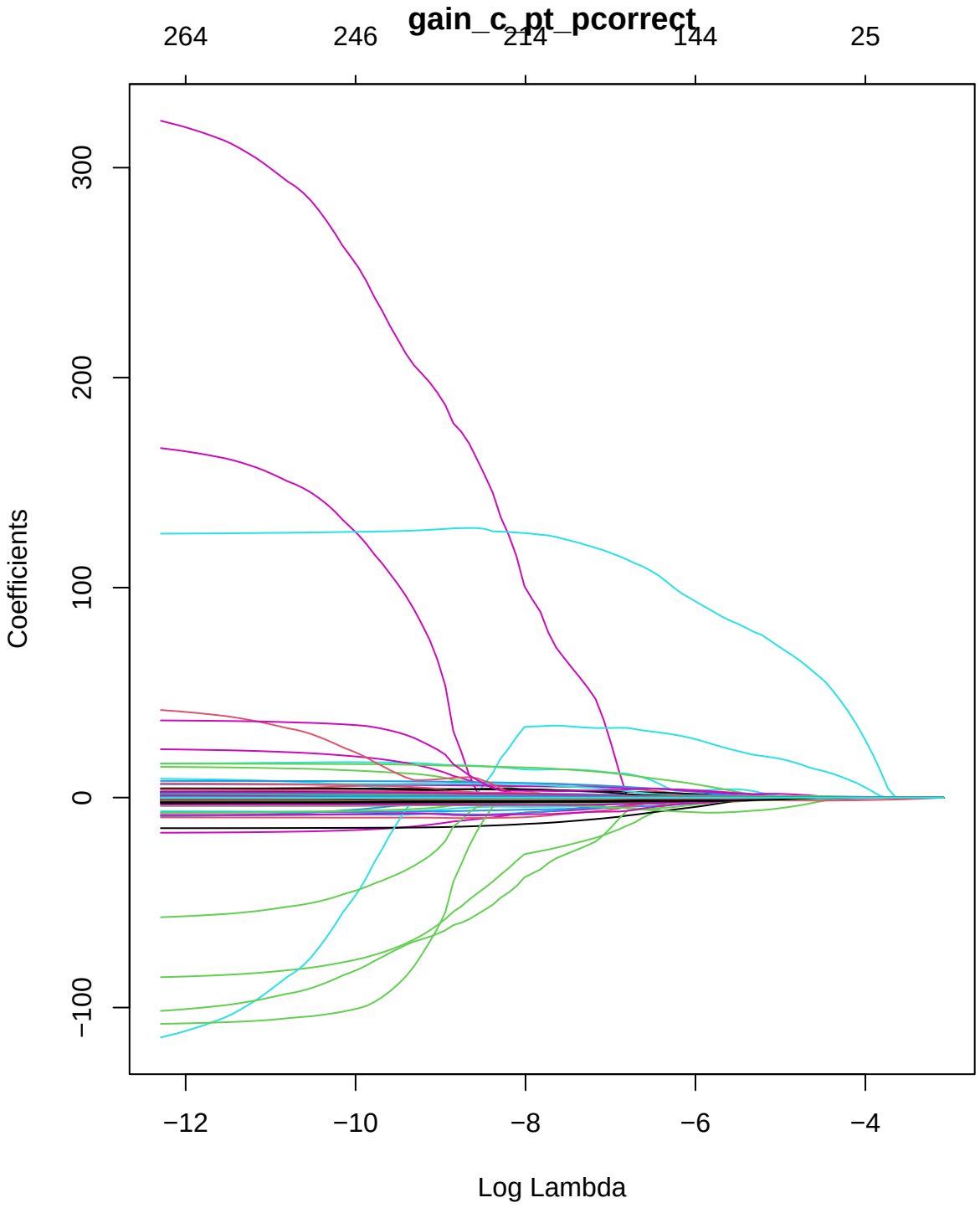
```
## [1] "gain_c_mefs_str"
## [1] 726
## [1] 726 282
```


263 259 249 243 228 218 200 160 129 80 42 13 0
400
350
300
250
200
150
Mean−Squared Error
−8
−6
−4
−2
0
Log(λ)

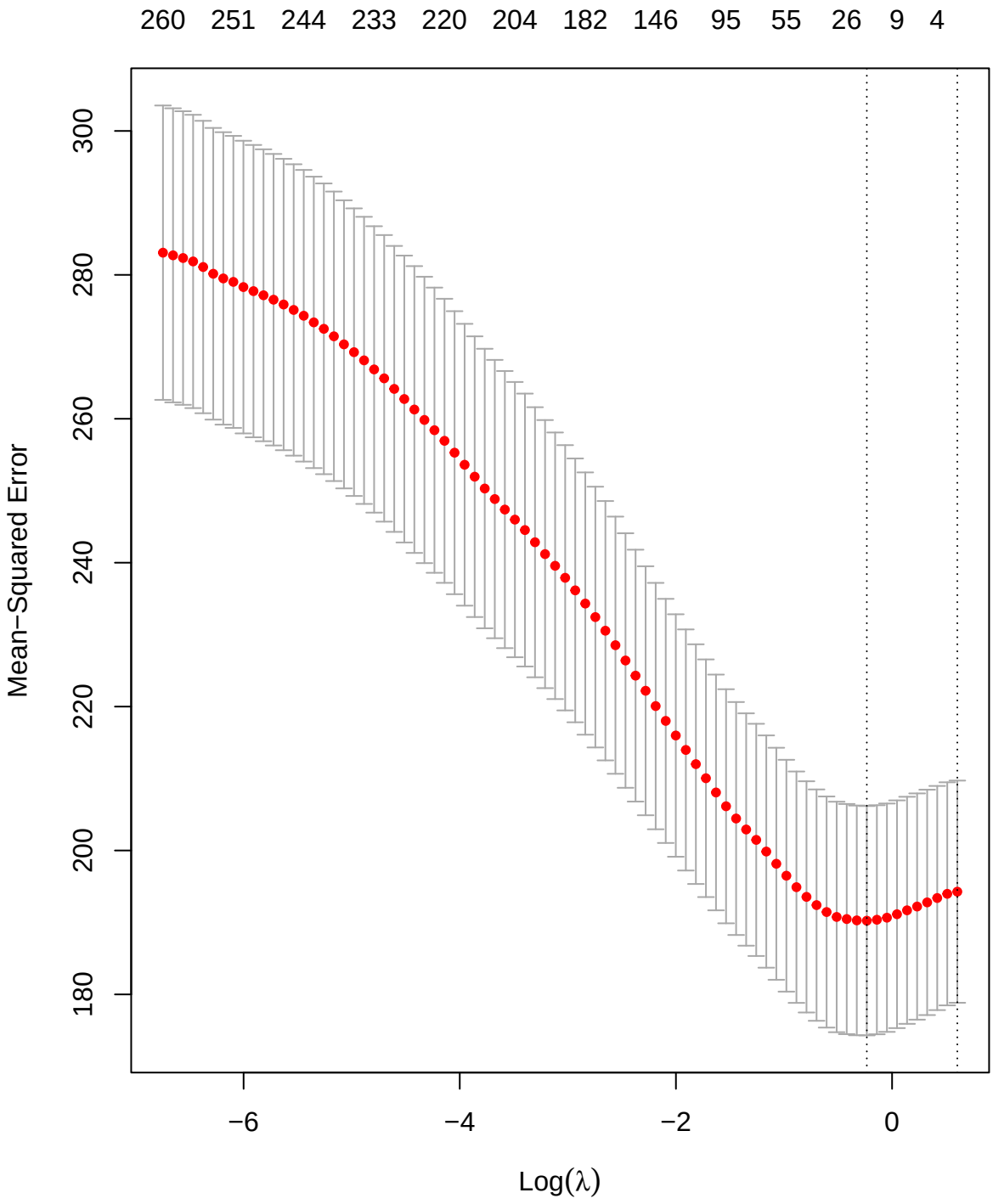


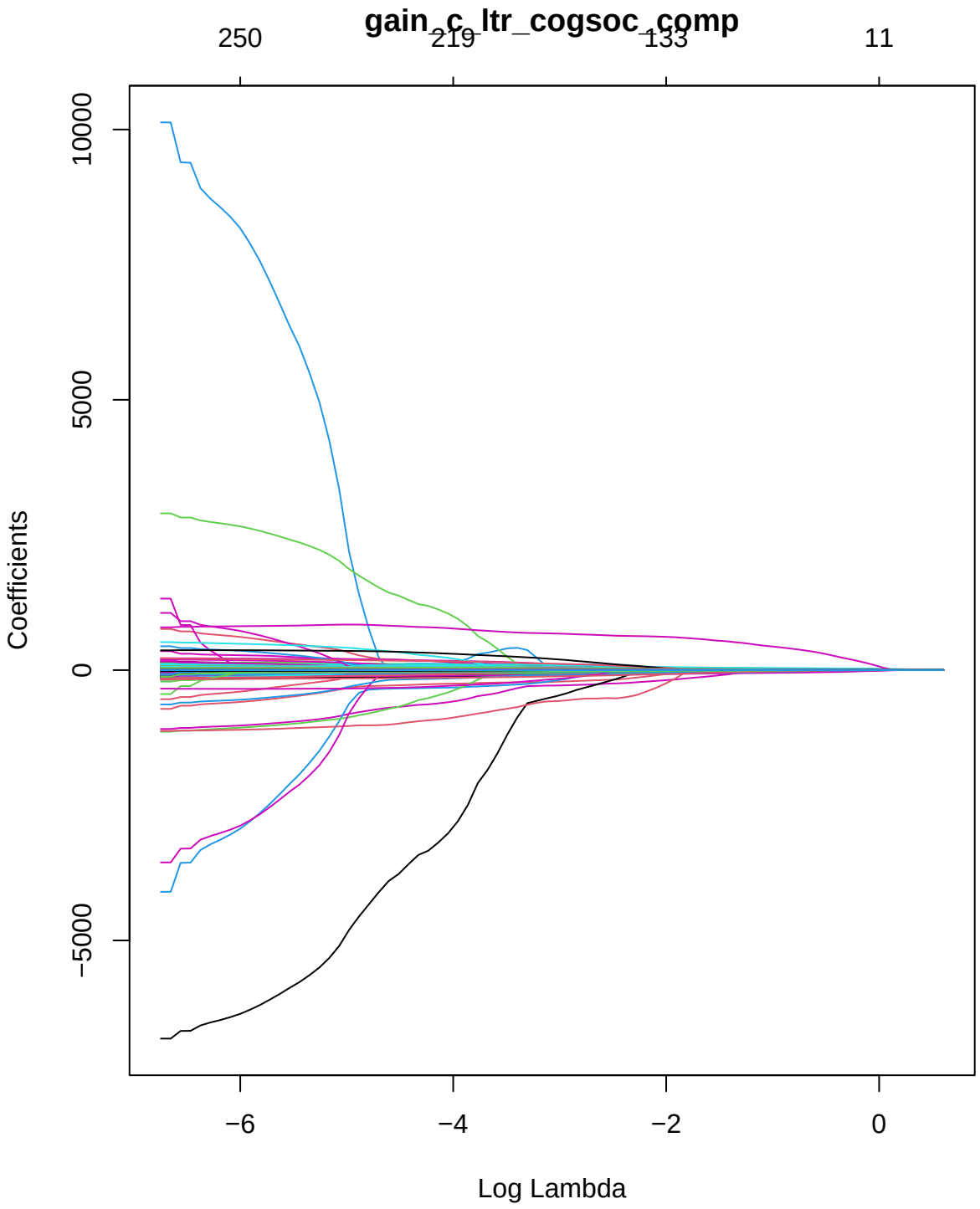

```
## [1] "gain_c_pt_pcorrect"
## [1] 845
## [1] 845 282
```


267 260 252 244 235 220 199 169 121 69 36 8 2
Mean−Squared Error
0.12
0.14
0.16
0.18
0.20
−12
−10
−8
−6
−4
Log(λ)

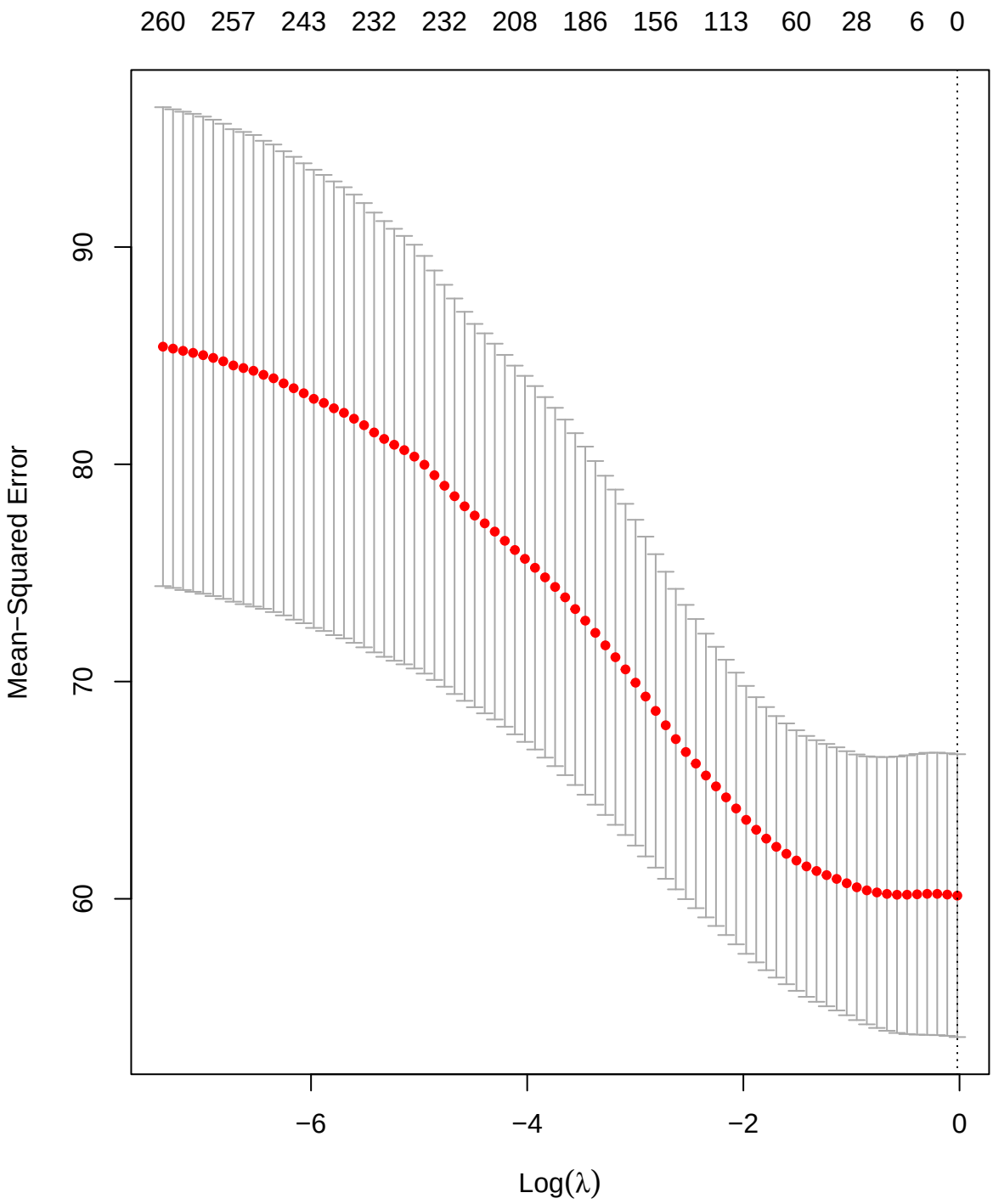


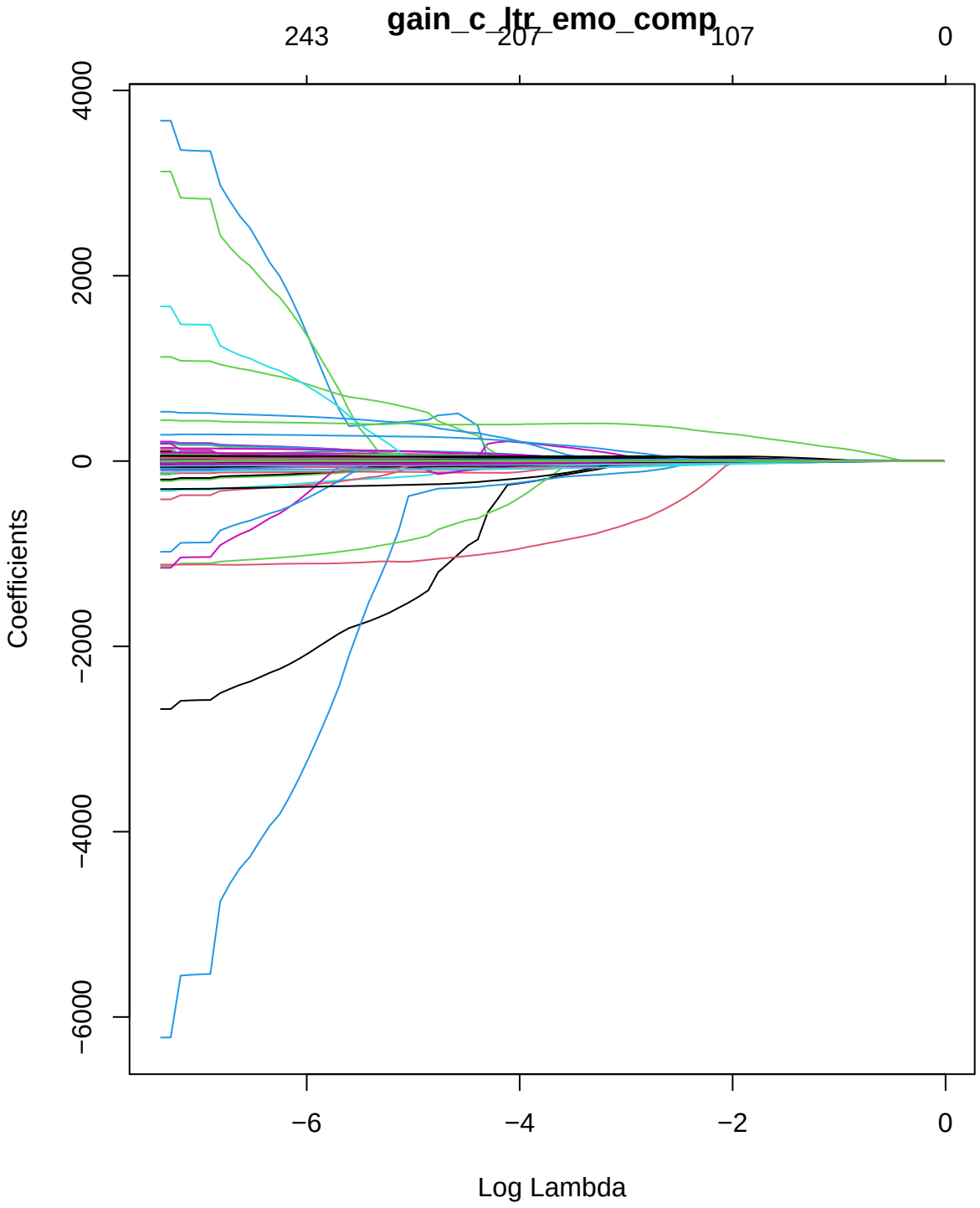

```
## [1] "gain_c_ltr_cogsoc_comp"
## [1] 992
## [1] 992 282
```


260 251 244 233 220 204 182 146 95 55 26 9 4
Mean−Squared Error
300
280
260
240
220
200
180
−6
−4
−2
0
Log(λ)

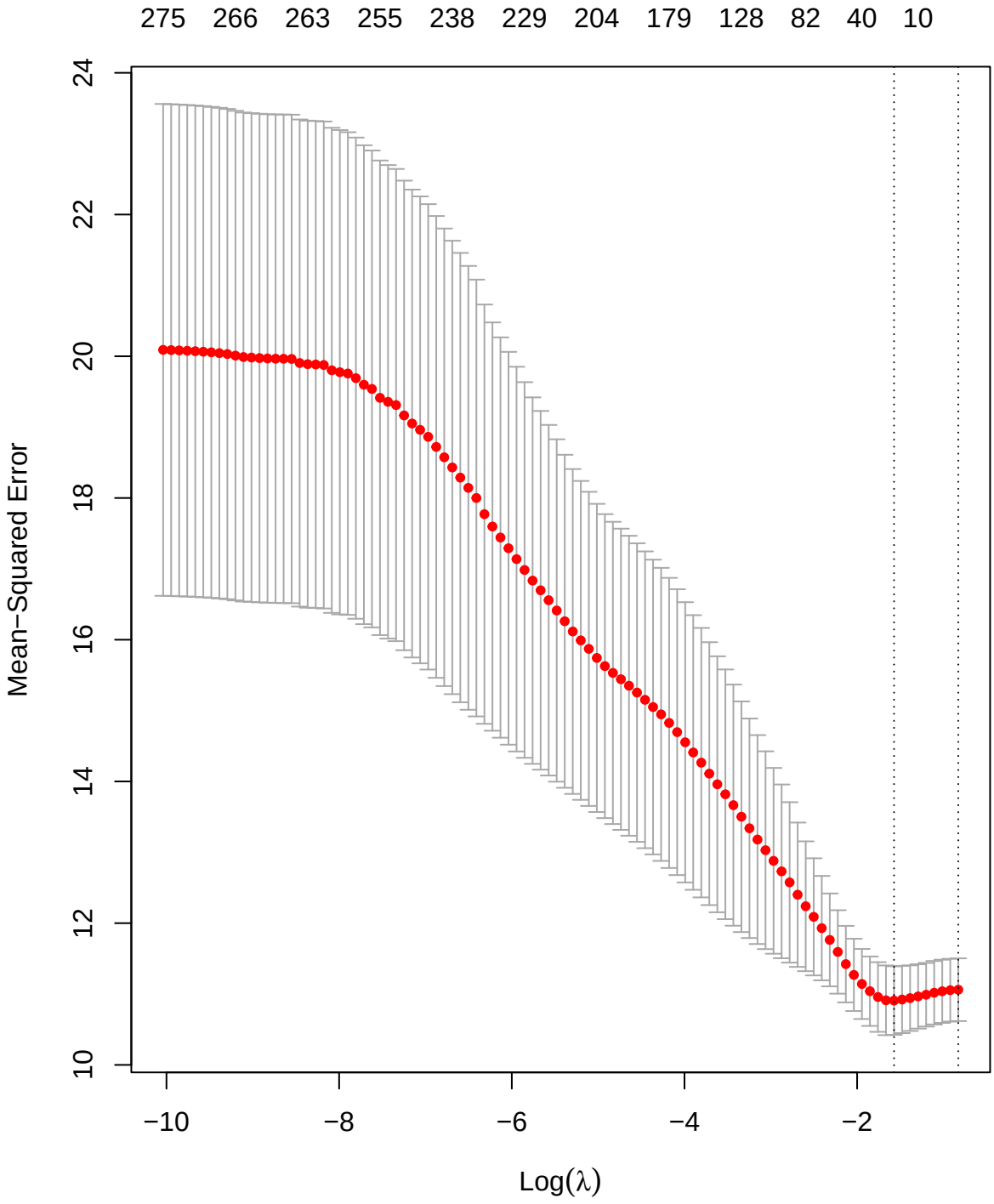


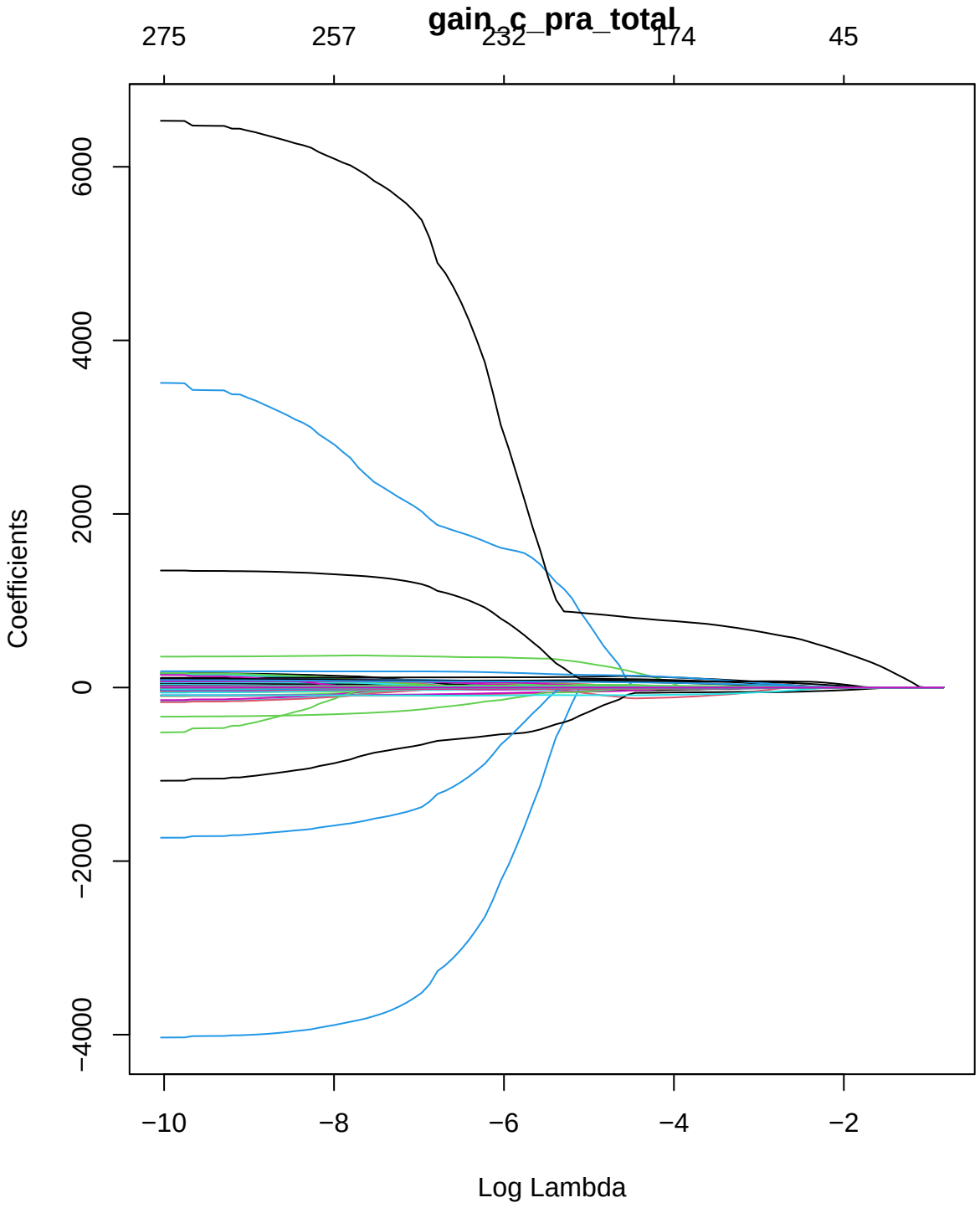

```
## [1] "gain_c_ltr_emo_comp"
## [1] 992
## [1] 992 282
```


260 257 243 232 232 208 186 156 113 60 28 6 0
Mean−Squared Error
90
80
70
60
−6
−4
−2
0
Log(λ)

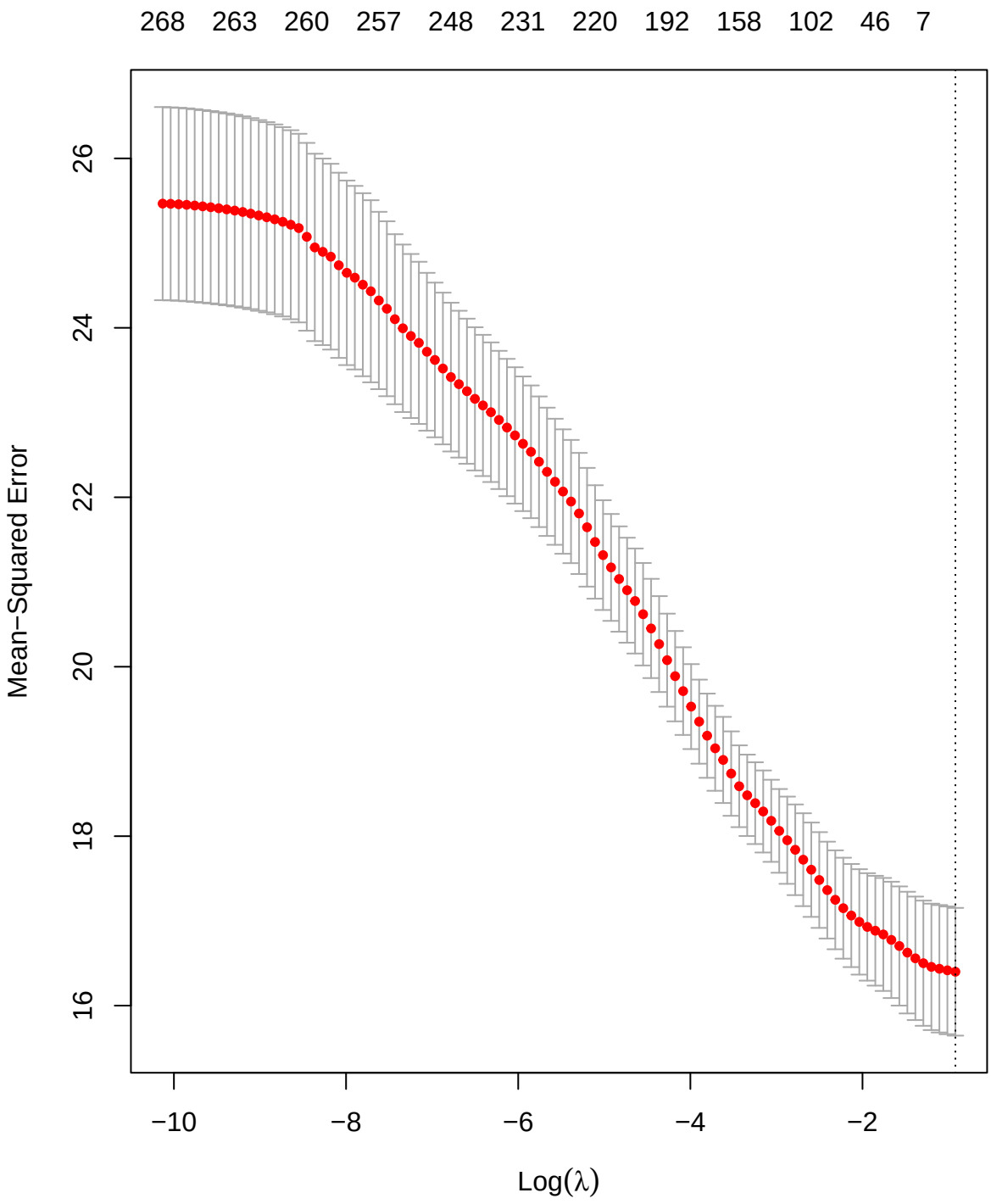


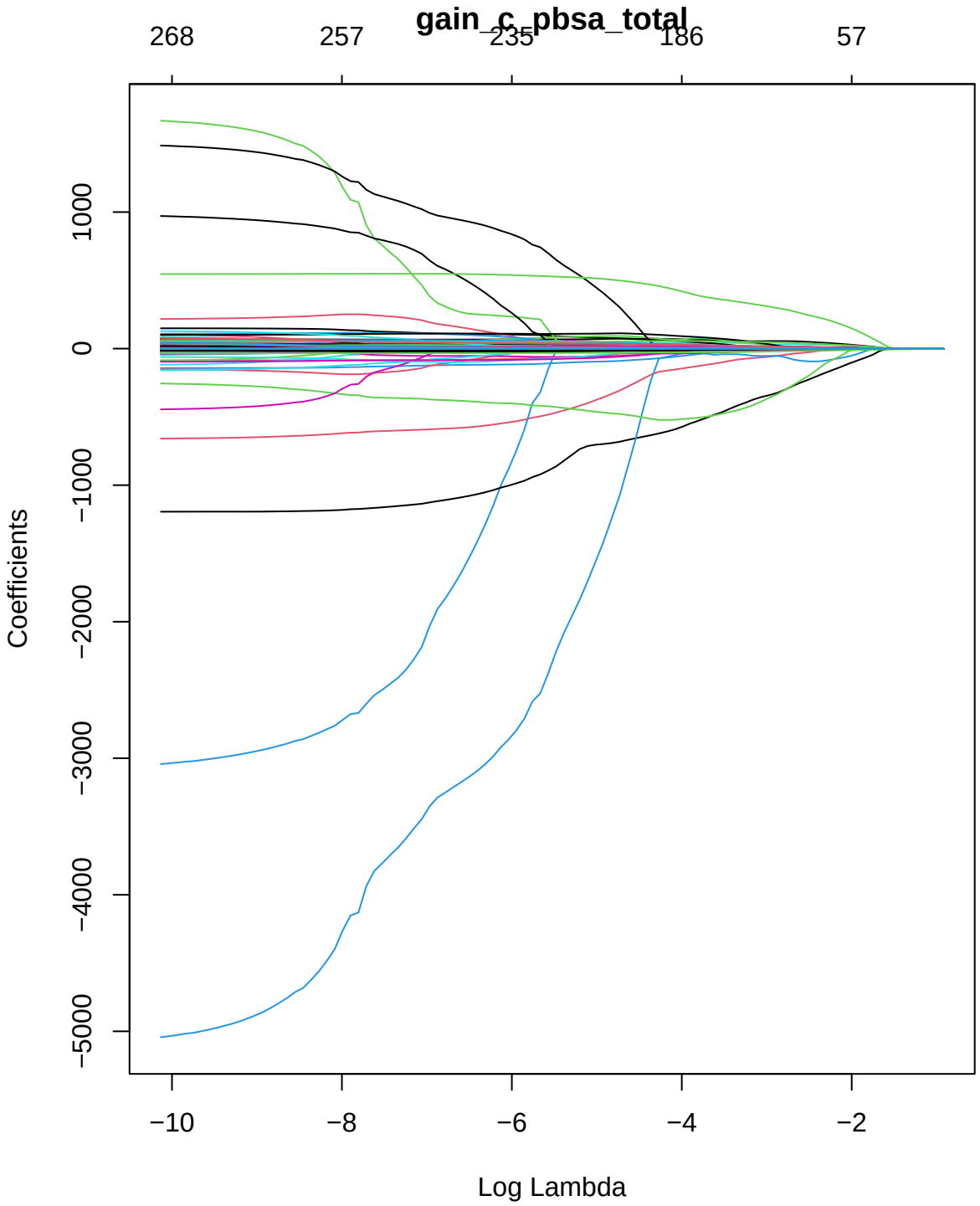

```
## [1] "gain_c_pra_total"
## [1] 873
## [1] 873 282
```


275 266 263 255 238 229 204 179 128 82 40 10
Mean−Squared Error
24
22
20
18
16
14
12
10
−10
−8
−6
−4
−2
Log(λ)

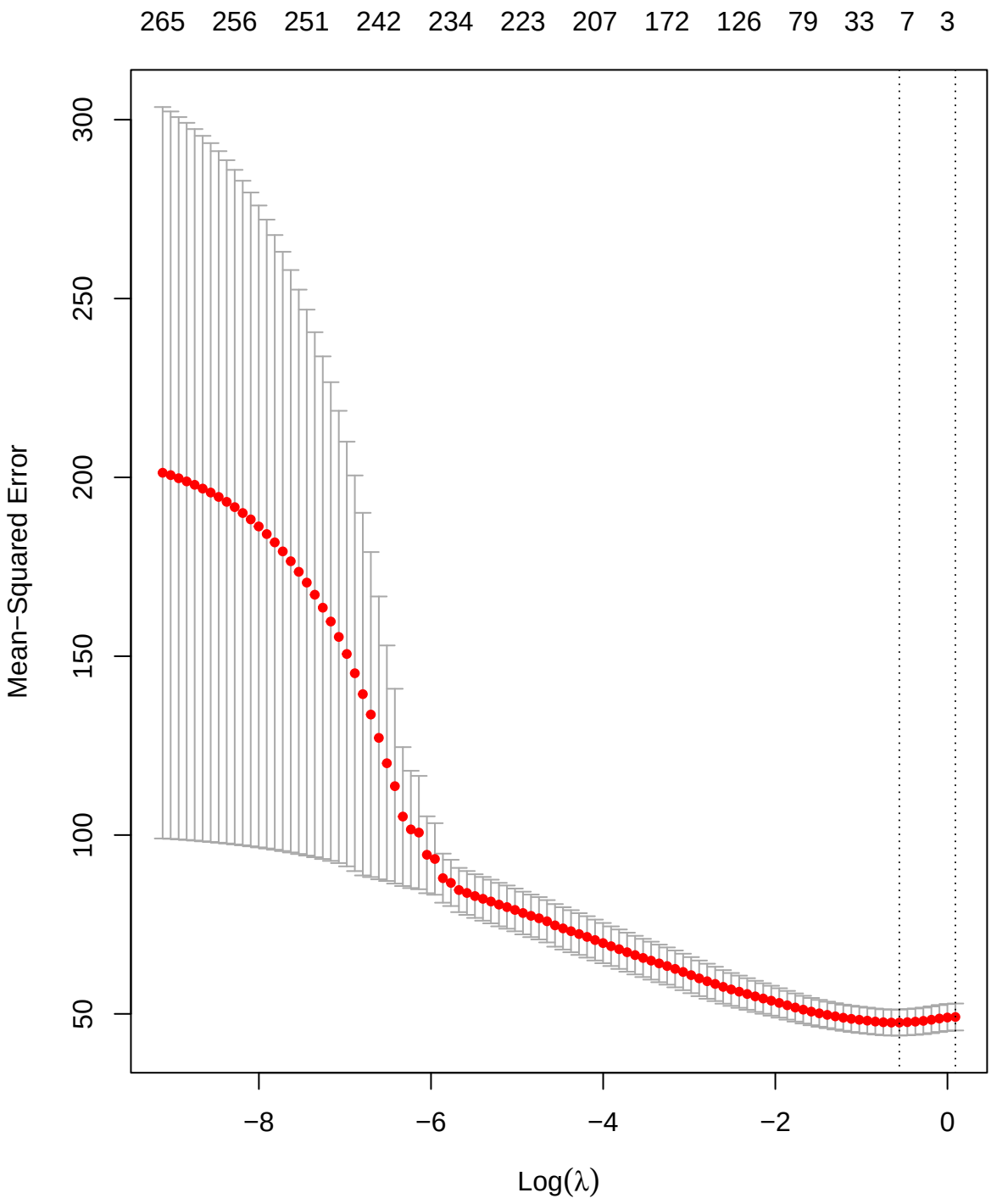


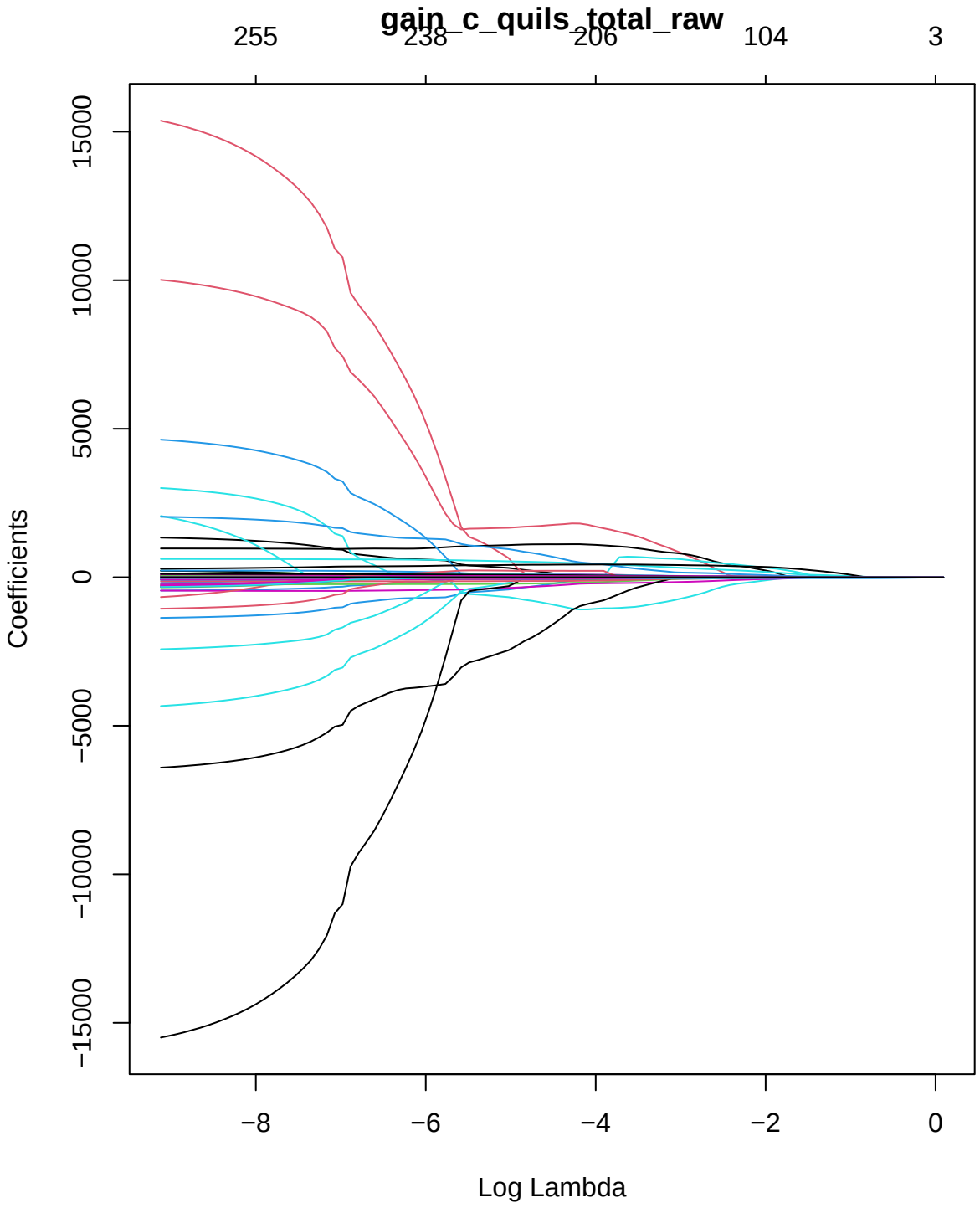

```
## [1] "gain_c_pbsa_total"
## [1] 882
## [1] 882 282
```


268 263 260 257 248 231 220 192 158 102 46 7
Mean−Squared Error
16
18
20
22
24
26
−10
−8
−6
−4
−2
Log(λ)

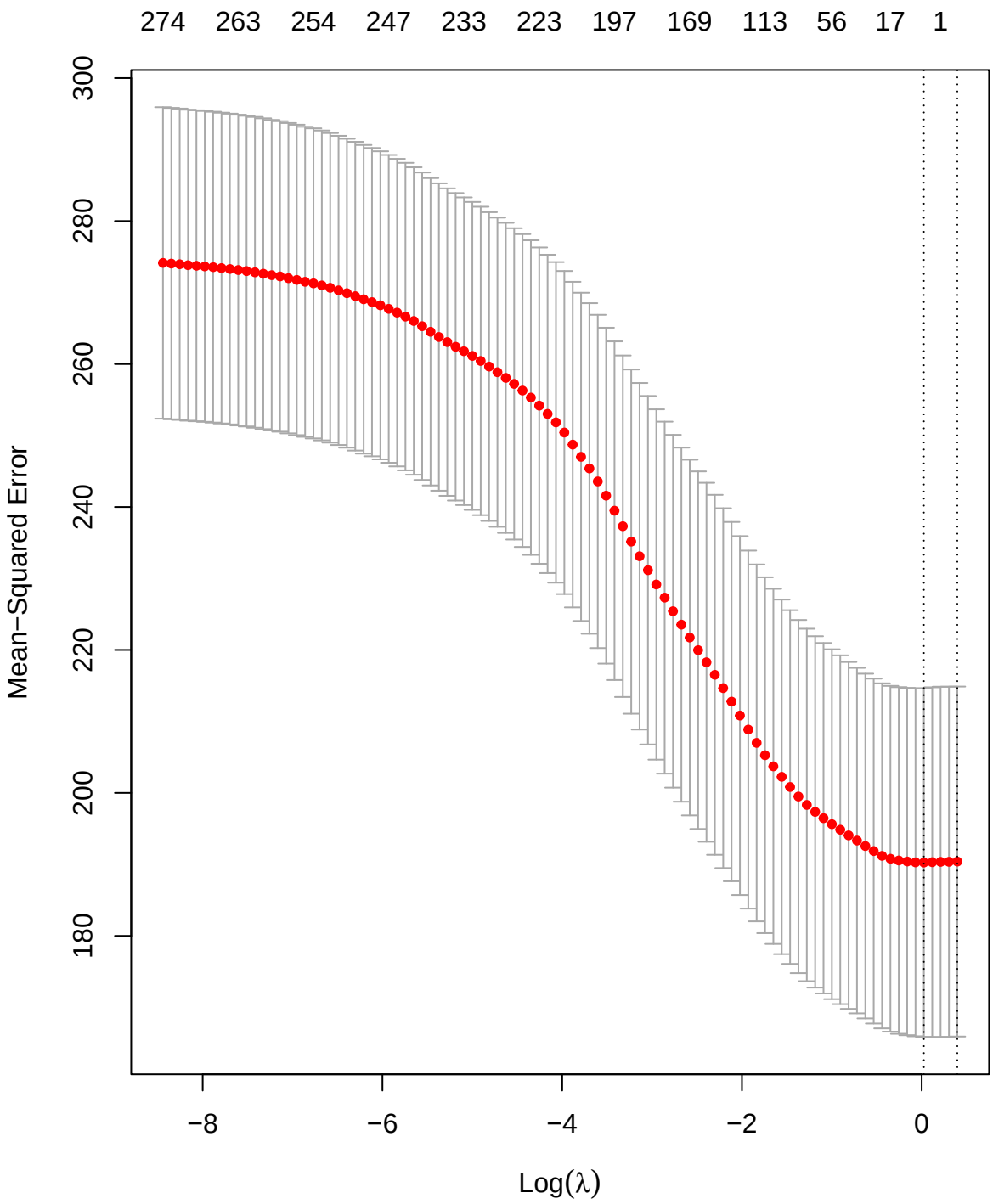


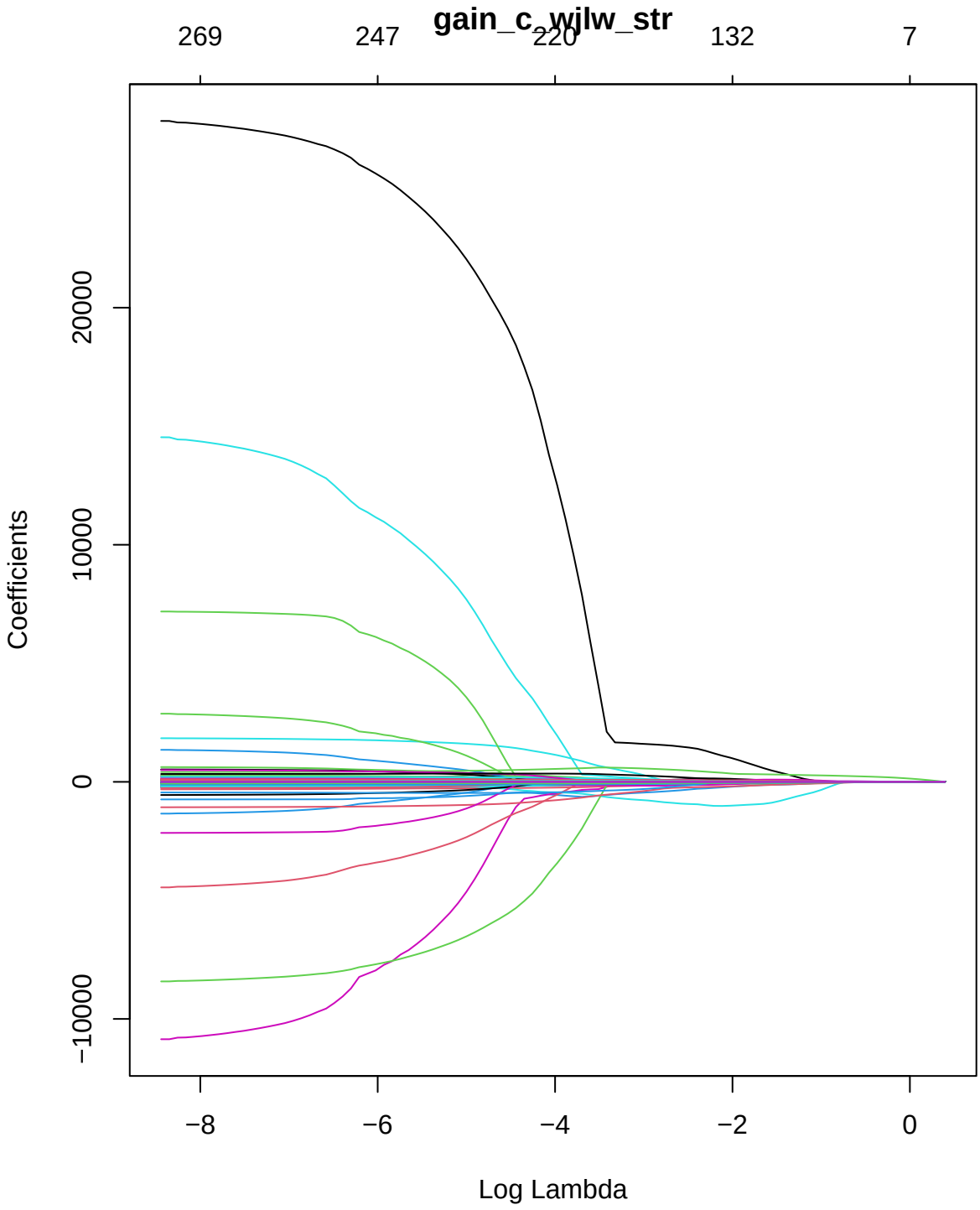

```
## [1] "gain_c_quils_total_raw"
## [1] 598
## [1] 598 282
```


265 256 251 242 234 223 207 172 126 79 33 7 3
Mean−Squared Error
300
250
200
150
100
50
−8
−6
−4
−2
0
Log(λ)

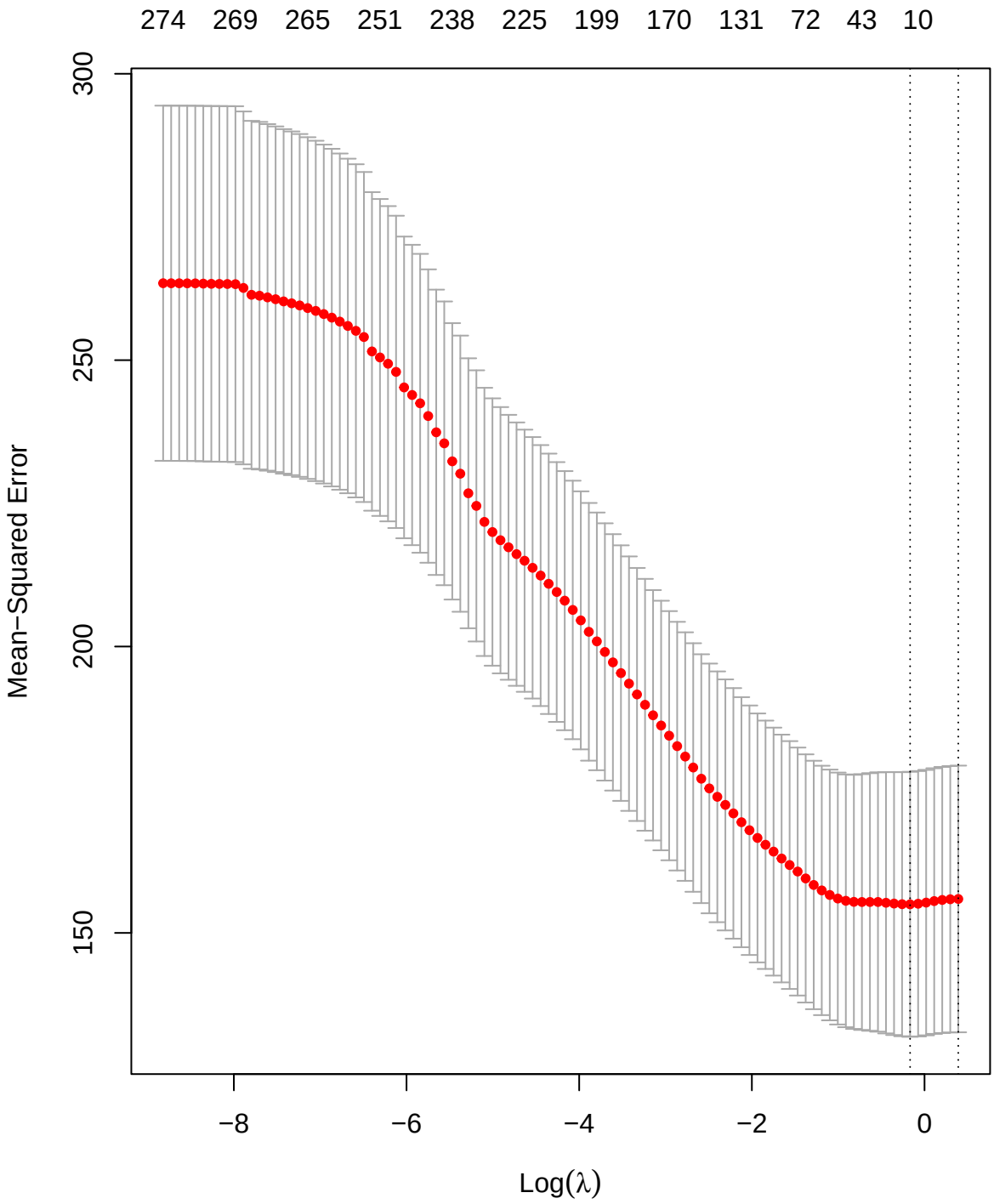


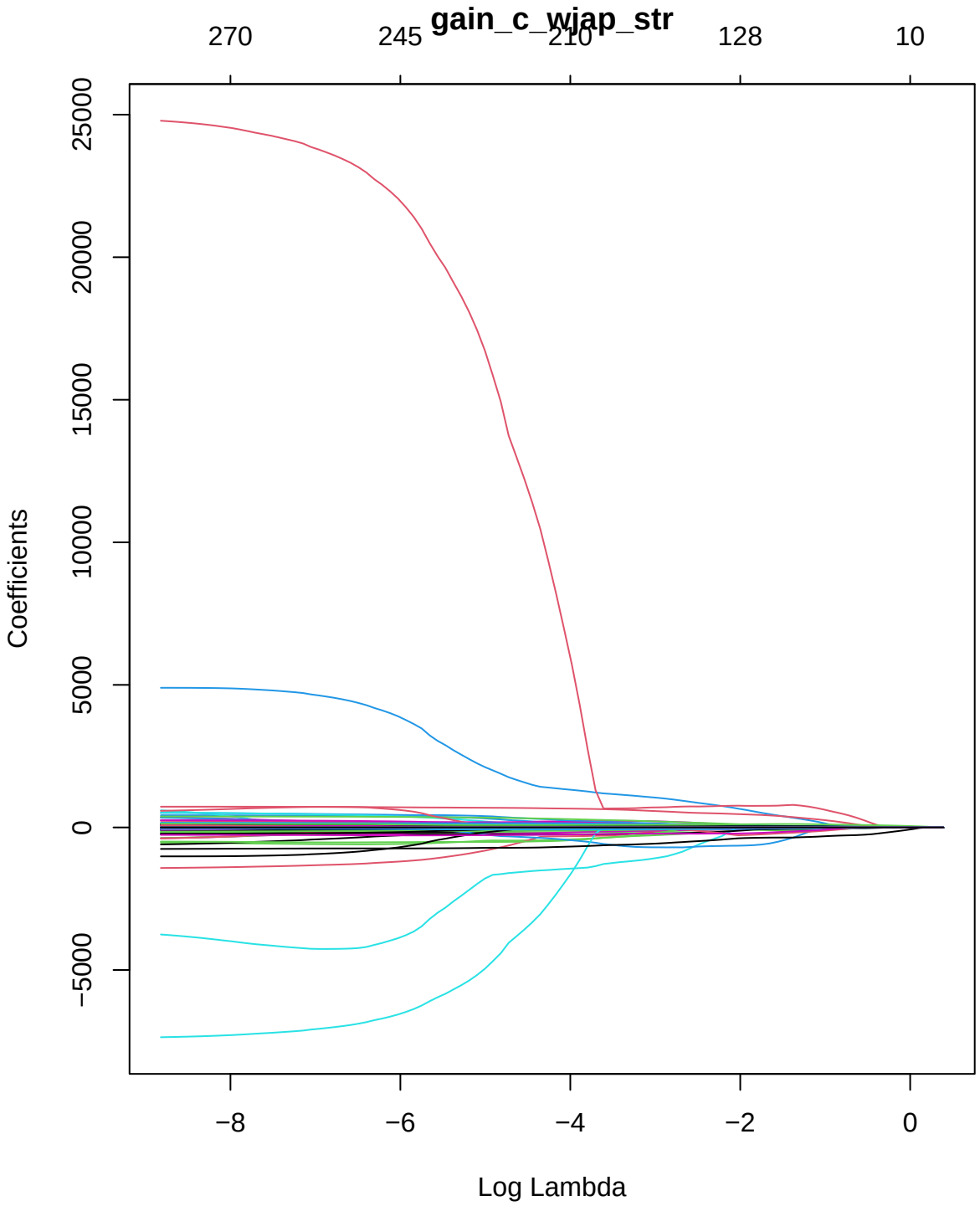

```
## [1] "gain_c_wjlw_str"
## [1] 963
## [1] 963 282
```


274 263 254 247 233 223 197 169 113 56 17 1
Mean−Squared Error
300
280
260
240
220
200
180
−8
−6
−4
−2
0
Log(λ)




```
## [1] "gain_c_wjap_str"
## [1] 930
## [1] 930 282
```


274 269 265 251 238 225 199 170 131 72 43 10
300
250
200
150
Mean−Squared Error
−8
−6
−4
−2
0
Log(λ)




```
for (i in gain_ind) {
  name
  hist(outcomes_and_obs_full_y1[, i])
}
```

```
# loop through the names associated with each outcome -- add 1 to corresponding entry
count_coefs_y1 <- list()
for (outcome in coefs_y1) {
  for (name in names(outcome)) {
```

```
    count_coefs_y1[[name]] <- ifelse(is.null(count_coefs_y1[[name]]), 1,
                                     count_coefs_y1[[name]] + 1)
  }
}

#Convert output to a df for plotting and such
count_coefs_y1 <- count_coefs_y1 %>%
  unlist() %>%
  as.data.frame(row.names=TRUE)
```

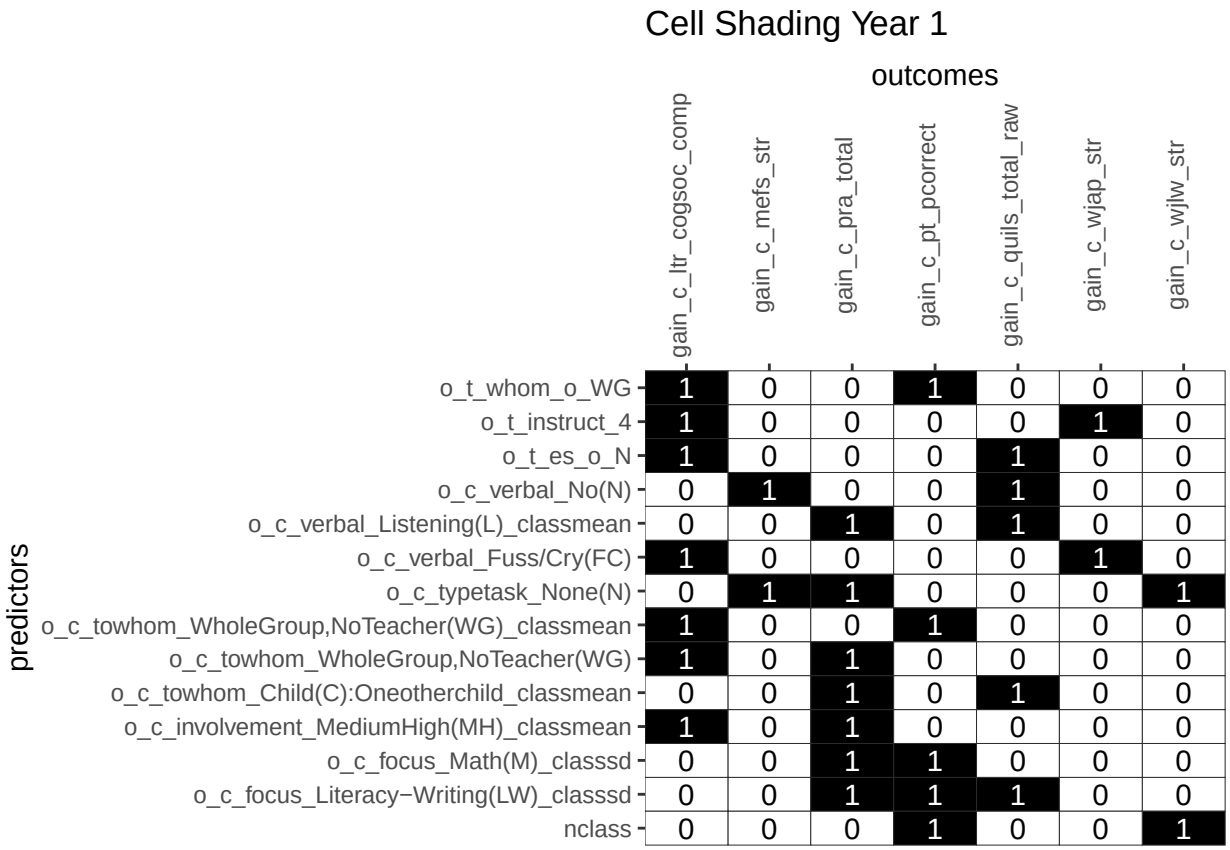
```
## Warning in as.data.frame.numeric(., row.names = TRUE): 'row.names' is not a
## character vector of length 66 -- omitting it. Will be an error!
```

```
count_coefs_y1$predictor <- row.names(count_coefs_y1)

names(count_coefs_y1) <- c("frequency", "predictor")
count_coefs_y1 <- count_coefs_y1[order(-count_coefs_y1$frequency), ]

# pull predictors that have more than 1 appearance

count_coefs_y1_top <- count_coefs_y1[count_coefs_y1$frequency > 1, ]
count_coefs_y1_top$outcomes <- rep(NA, nrow(count_coefs_y1_top))

list_store <- list()

for (top_predictor in count_coefs_y1_top$predictor) {
  for (i in 1:length(coefs_y1)) {
    outcome_name <- names(coefs_y1)[i]
    if (top_predictor %in% names(coefs_y1[[i]])) {
      if (is.null(list_store[[top_predictor]])) {
        list_store[[top_predictor]] <- list(outcome_name)
      }
      else {
        list_store[[top_predictor]] <- c(list_store[[top_predictor]], outcome_name)
      }
    }
  }
}
```

```
# all the unique outcomes that appear, so that they can be the columns in the
# binary table
unique_outcomes <- unique(unlist(unique(sapply(list_store, unlist))))
unique_predictors <- names(list_store)

# create data frame to translate into binary table
expand.grid(
  unique_predictors,
  unique_outcomes,
  stringsAsFactors = FALSE
) %>%
  set_names(c("predictors", "outcomes")) %>%
  mutate(value = rep(0, n())) -> binary_df

for (predictor in names(list_store)) {
  for (outcomes in list_store[[predictor]]) {
```

```
    outcomes_unlisted <- unlist(outcomes)
    for (outcome in outcomes_unlisted) {
        binary_df[binary_df$predictors == predictor &
                    binary_df$outcomes == outcomes_unlisted, ]$value <- 1
    }
  }
}
```

```
mutate(
  binary_df,
  fill = ifelse(value == 1, "black", "white"),
  color = ifelse(value == 1, "white", "black"),
  address = factor(predictors, levels = sort(unique(predictors), decreasing = TRUE))
) -> cell_shading_df

ggplot(cell_shading_df, aes(x = outcomes, y = predictors)) +
  geom_tile(
    aes(fill = I(fill)),
    color = "#2b2b2b", size=0.125,
  ) +
  geom_text(
    aes(label = value, color = I(color))
  ) +
  scale_x_discrete(expand=c(0,0), position = "top") +
  scale_y_discrete(expand=c(0,0)) +
  labs(title = "Cell Shading Year 1", x = "outcomes", y = "predictors") +
  # hrbrthemes::theme_ipsum_rc(grid="XY") +
  theme(axis.text.x = element_text(angle=90, vjust=0.5, hjust=0.5))
```




```
varimp_y1 <- list()
rf_models <- list()
rf_plots <- list()
test_data <- outcomes_and_obs_full_y1

# make sure this works with small subset of data
for (i in gain_ind) {
  name <- names(outcomes_and_obs_full_y1)[i]
  df_analysis <- test_data %>%
    filter(!is.na(test_data[[name]])) %>%
    dplyr::select(c(name, "o_c_verbal_Fuss/Cry(FC)":actualtype))
    # ask about verbal_fuss vs o_c_verbal_Talk(T) (original)
    # mutate_at(vars("o_c_verbal_Fuss/Cry (FC)":actualtype), replace.na)
  print(name)
  # options(na.action="na.pass")
  x = model.matrix(as.formula(paste(name, "~ .")), data = df_analysis)

  # Fit the random forest model
  rf_fit <- train(as.formula(paste(name, "~ .")), #Use all variables in the prediction
                  data = df_analysis, #Use the training data
                  method = "ranger",
                  importance = "permutation",
                  # ntree = 500,
                  na.action=na.pass)

  rf_plots[[name]] <- varImp(rf_fit) %>%
    pluck(1) %>%
```

```
    rownames_to_column("var") %>%
    ggplot(aes(x = reorder(var, Overall), y = Overall)) +
    geom_col(fill = "grey75") +
    coord_flip() +
    theme_minimal()

  # Store the full RF model
  rf_models[[name]] <- rf_fit

  # store variable importances as a data frame
  df <- as.data.frame(varImp(rf_fit)$importance)
  # arrange in descending order (most to least important), and put into list
  varimp_y1[[name]] <- df %>% arrange(desc(Overall))
}

rfoutput <- list(varimp_y1, rf_models, rf_plots)

saveRDS(rfoutput, file = "rfoutput.RDS")
```

```
#NOTES FROM MEETINGI 8/8
#1. Try Y2 observation data instead of Y1, and compare results
#2. Try a model with a bunch of predictors from Y1 Y2 -- looking at more demographic variables
# wait for Jonathan to see which demographic variables are important
#3. Use the particular caretype from the  setting that is being observed in. (prov_type)
#Collapse CC-Community Based & CC- License Exempt --
#4. Include leave-out standard deviation for each predictor as well
#5. Incorporte child-level covariates -- wait for this one as well
#6. Age and elapsed time for assessment
# look for date in the cleaning process -- flag it and let Jonathan take closer look

## 4 is a priority, plus other things we discussed (on google doc)

# comparing random forest and lasso results -- scatter the absolute value of coefficients (double check
# should hopefully see some
```

```
# As far as I can tell, variable importance is measuring either: a) the percentage that the prediction

# I'm not sure if I will try to "unstandardize" the random forest results. I will use the magnitude of

# todo:
# 1. extract coefficient values for each selected variable from lasso - maybe store in data frame?
# 2. find a way to extract numeric value of variable importance - also store in data frame
# 3. merge data frames -- thinking one data frame per variable? so list of data frames
# 4. scatter values against each other
```

```
#Merge in the outcomes data
#  Try Y2 observation data instead of y1, and compare results

# add this line in so that the merge is correctly executed

y2_obs <- y2_obs %>%
  filter(!is.na(cid)) %>%
  mutate(cid = as.numeric(cid))
```

```
outcomes_and_obs_y2 <- left_join(child_outcomes, y2_obs, by = "cid") %>%
  mutate(cid = as.character(cid))

#Add in the care type -- confirm that it is supposed to be year-specific
caretype <- read.dta13("y2caretype.dta", nonint.factors = TRUE) %>%
  mutate(cid = as.character(cid))

#Remove observations that have no care type or no classroom observation
outcomes_and_obs_full_y2 <- left_join(outcomes_and_obs_y2, caretype, by = "cid") %>%
  mutate(hasobservation = is.na(classid)) %>%
  filter(!is.na(caretype)) %>%
  filter(!is.na(classid))

# outcomes_and_obs_full_y1 <- outcomes_and_obs %>%
#   filter(!is.na(classid))

#Remove Y1 and Y2 data for cleanliness
outcomes_and_obs_full_y2 <- outcomes_and_obs_full_y2 %>%
  dplyr::select(-starts_with(c("y1", "y2")))

#Remove some irrelevant variables and remove illegal spaces
outcomes_and_obs_full_y2 <- outcomes_and_obs_full_y2 %>%
  # seems like provid is the same thing as famid in this case?
  dplyr::select(-c(provid, dob, actualtype_fcc:hasobservation)) %>%
  mutate(caretype = as.factor(caretype),
         actualtype = as.factor(actualtype)) %>%
  rename_at(vars(everything()), ~str_replace_all(., "\\s+", ""))
```

```
outcomes_and_obs_full_y2 <- outcomes_and_obs_full_y2 %>%
  mutate_at(vars("o_c_verbal_Fuss/Cry(FC)":actualtype), replace.na)

dim(outcomes_and_obs_full_y2)
```

```
## [1] 765 312
```

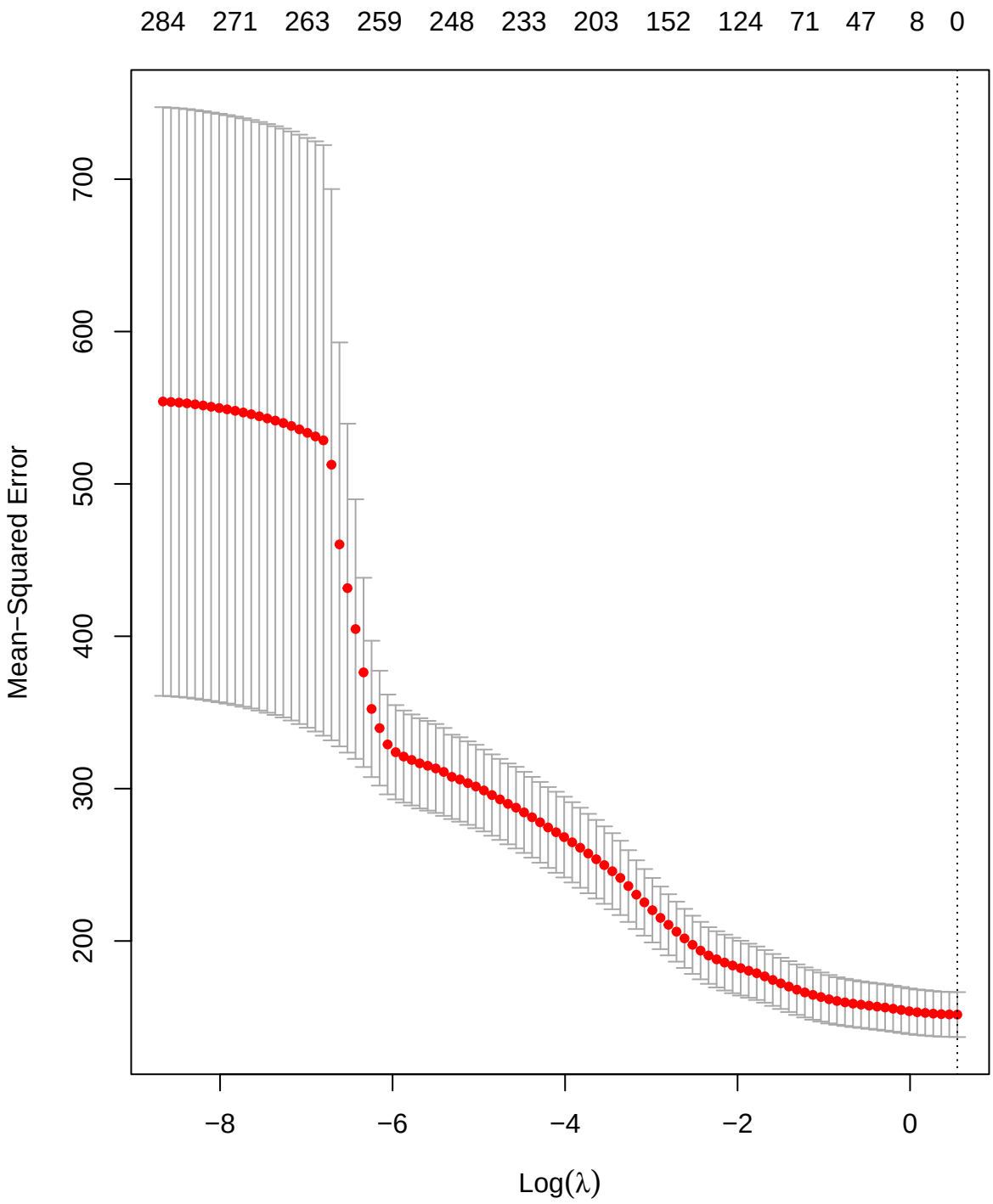
Analysis

We will first try a cross-validated LASSO, which will aggressively remove variables that do little to improve the predictive accuracy of the model.

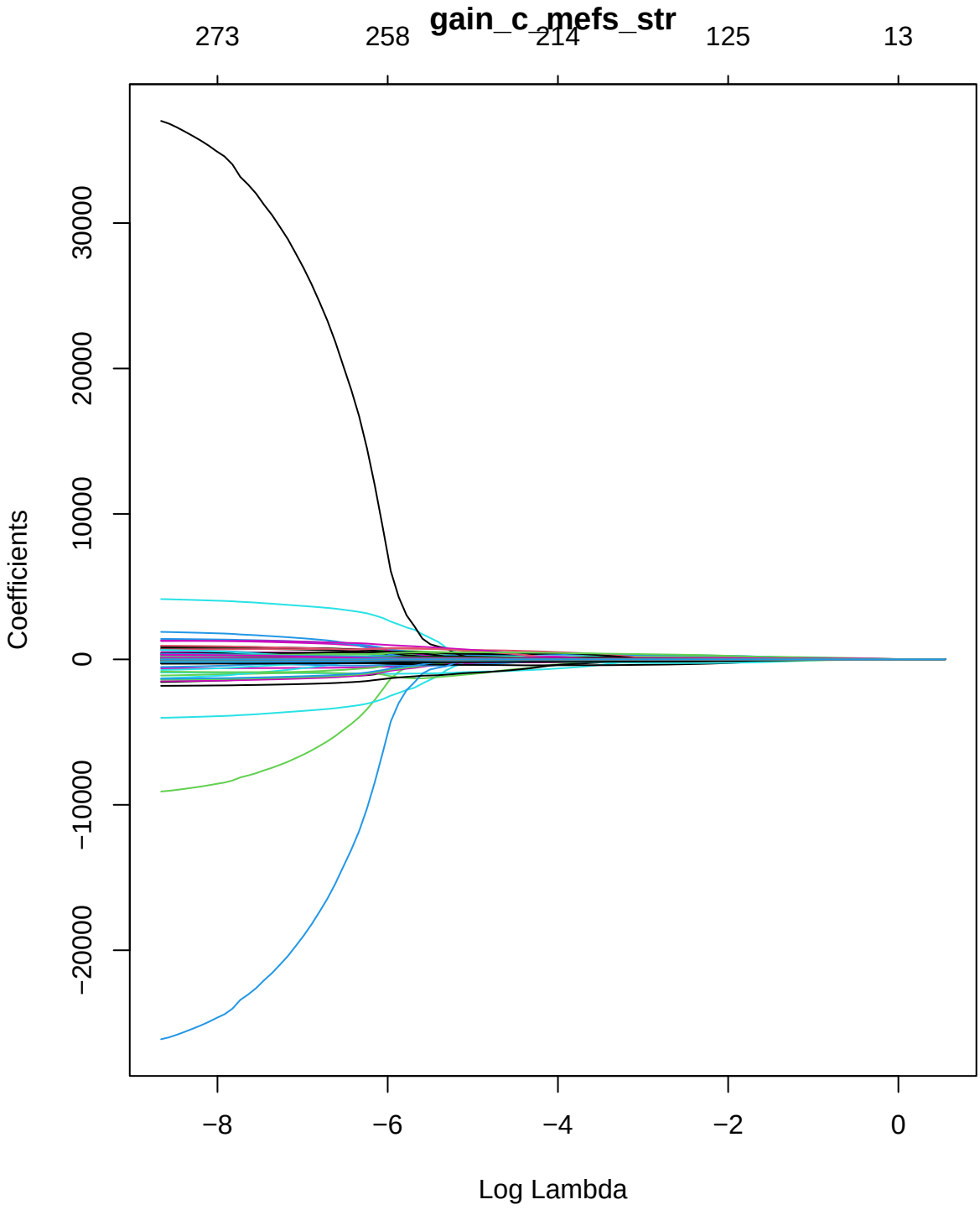
```
set.seed(4224)

gain_ind <- which(startsWith(colnames(outcomes_and_obs_full_y2), "gain"))

models_y2 <- list()
coefs_y2 <- list()

for (i in gain_ind) {
  name <- names(outcomes_and_obs_full_y2)[i]
  df_analysis <- outcomes_and_obs_full_y2 %>%
    filter(!is.na(outcomes_and_obs_full_y2[[name]])) %>%
    dplyr::select(c(name, "o_c_verbal_Fuss/Cry(FC)":actualtype))
  allSd <- apply(df_analysis[, -i], 2, sd)
  print(name)
```

```
  options(na.action="na.pass")
  x = model.matrix(as.formula(paste(name, "~ .")), data = df_analysis)

  y = df_analysis[[name]]
  print(dim(x))
  x = x[, -1]


  # call cv.glmnet()
  model_lasso <- cv.glmnet(x = x, y = y, alpha = 1)
  plot(model_lasso)
  plot(model_lasso$glmnet.fit, "lambda", main=name)

  models_y2[[name]] <- model_lasso

  cc = coef(model_lasso, s = model_lasso$lambda.min)

  # print out the model coefficients and store in a list.
  cc = cc[cc[,1]!=0,1][-1]
  # remove backticks for ease of standardizing
  names(cc)<- gsub("`","", names(cc))
  coefs_y2[[name]] <- cc * allSd[names(cc)]
}
```

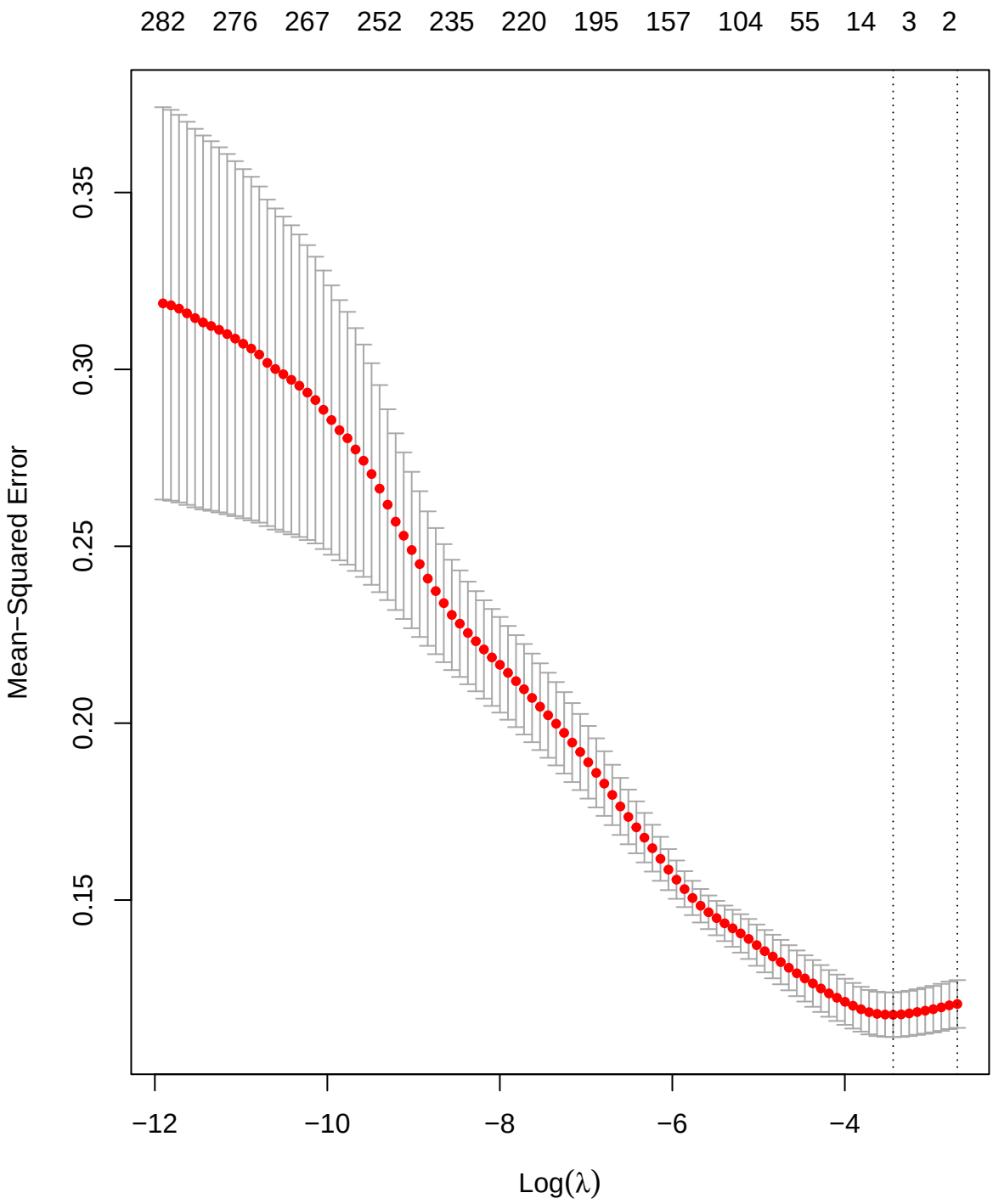
```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## [1] "gain_c_mefs_str"
## [1] 522 290
```



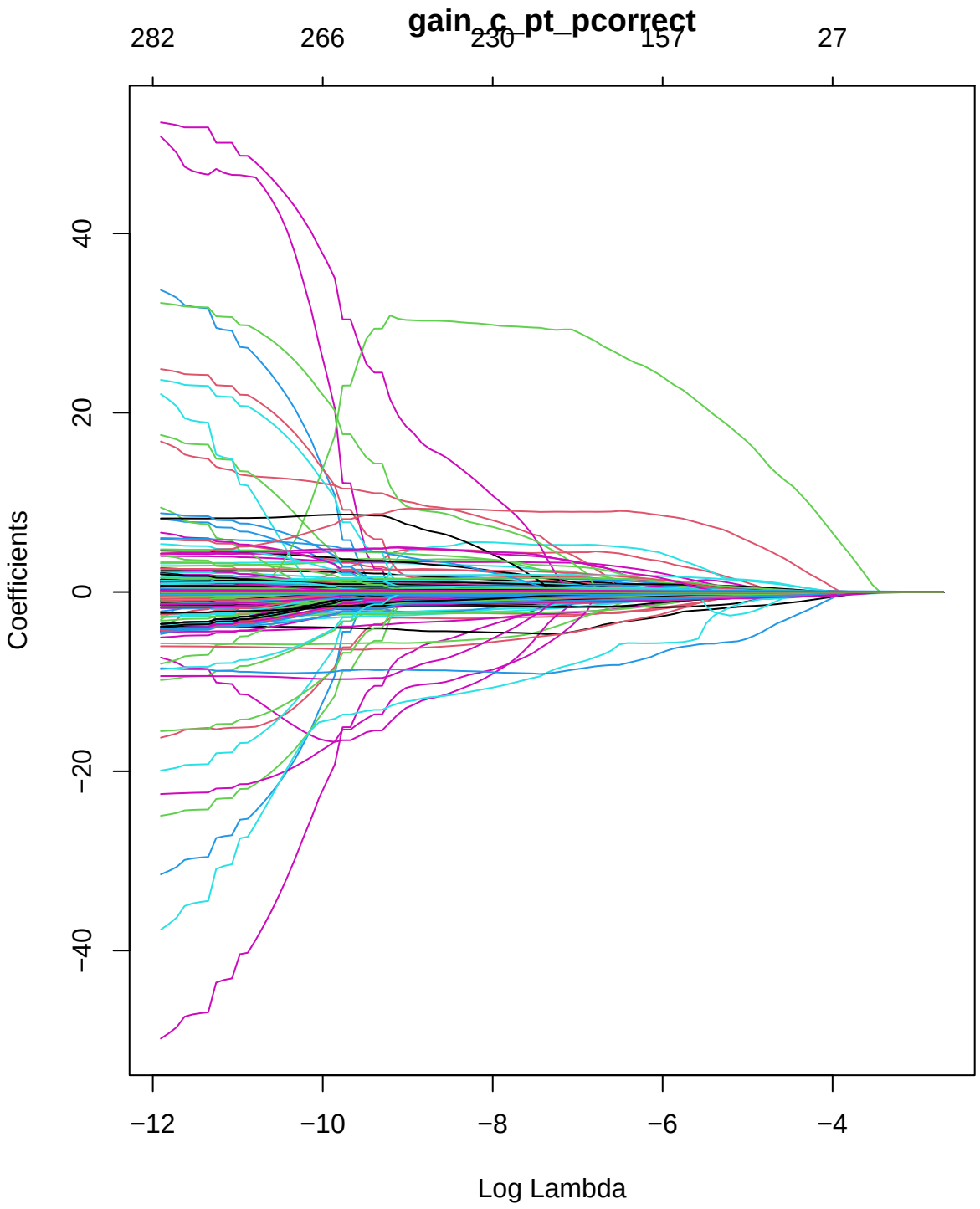

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



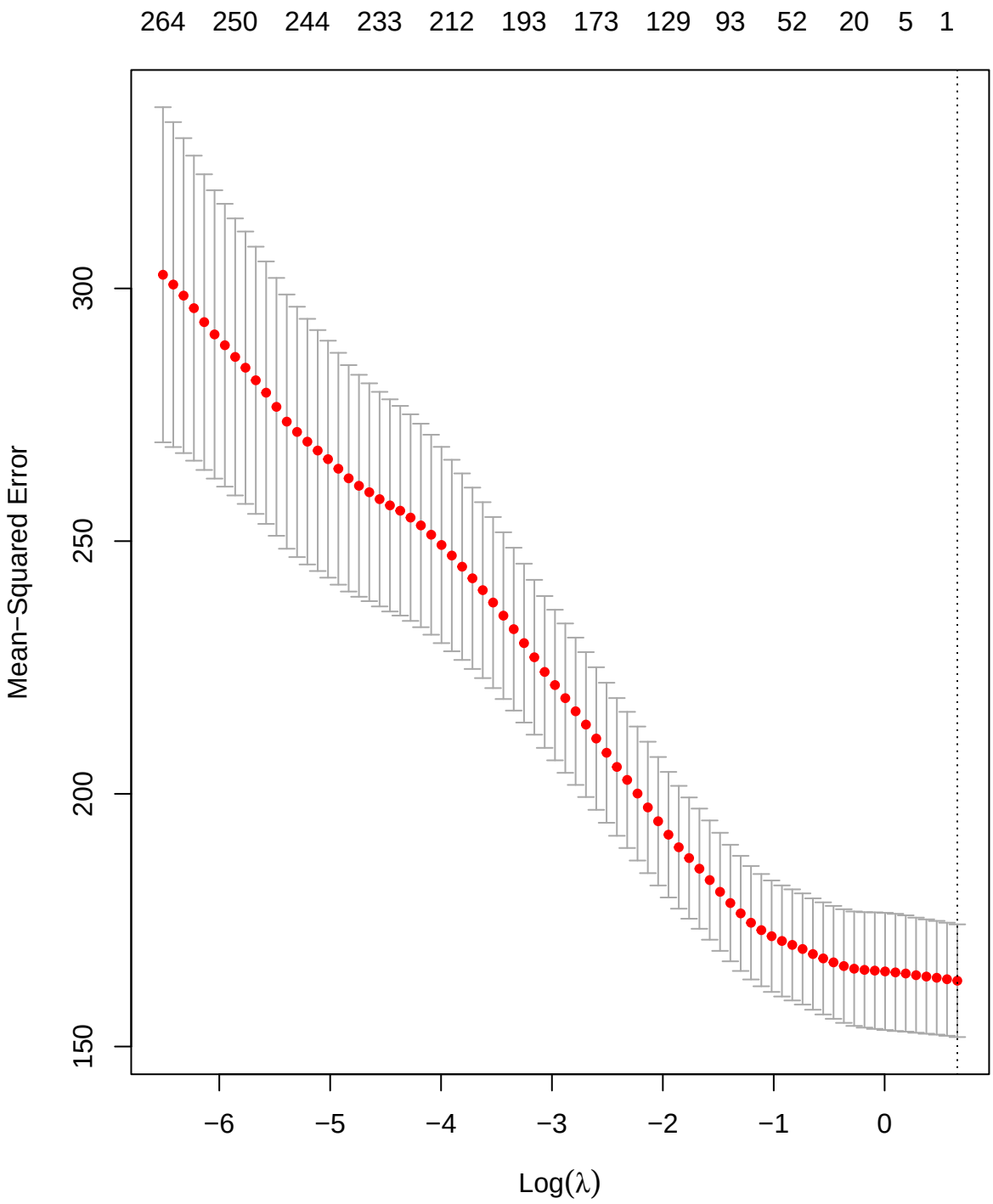

```
## [1] "gain_c_pt_pcorrect"
## [1] 609 290
```



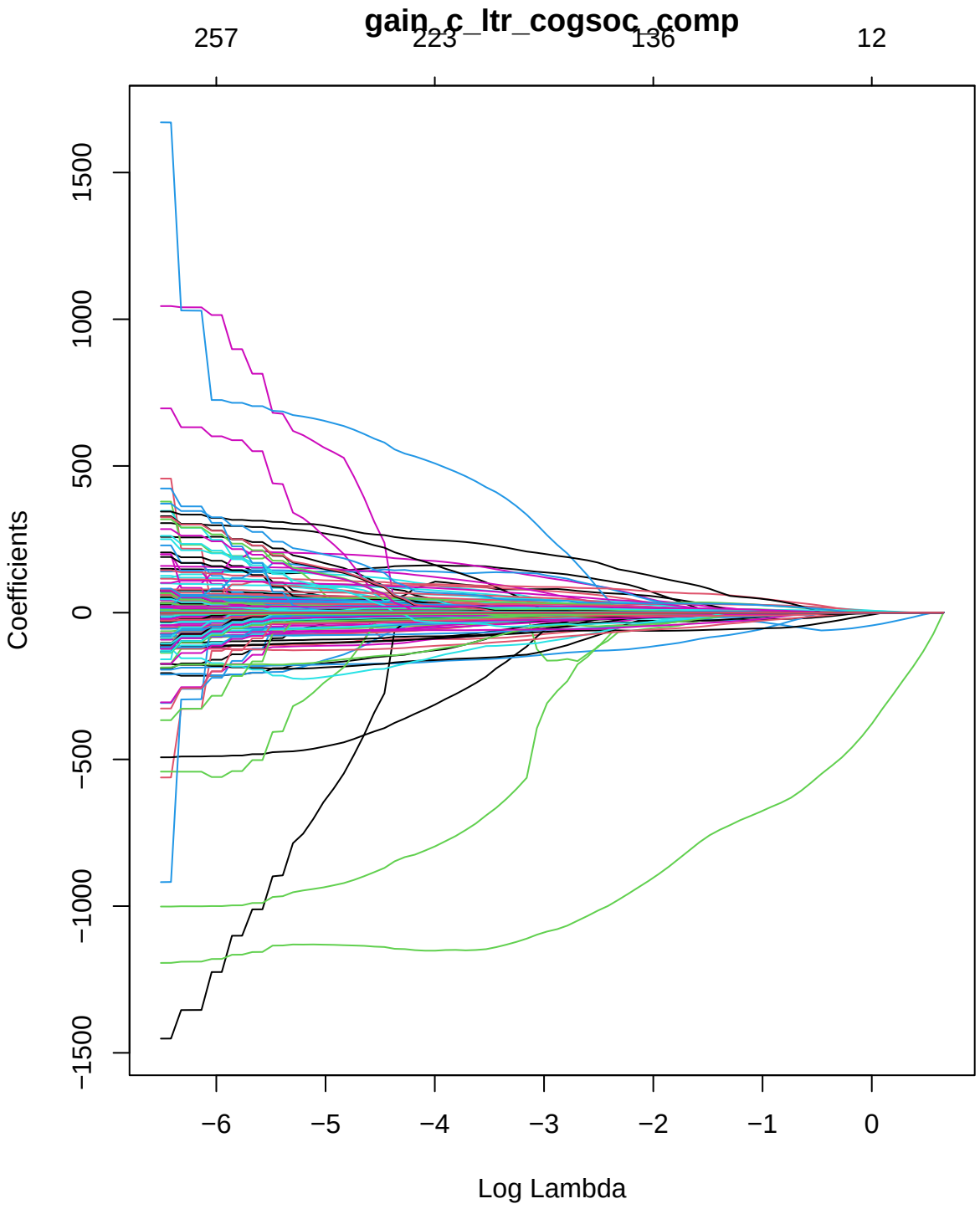

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



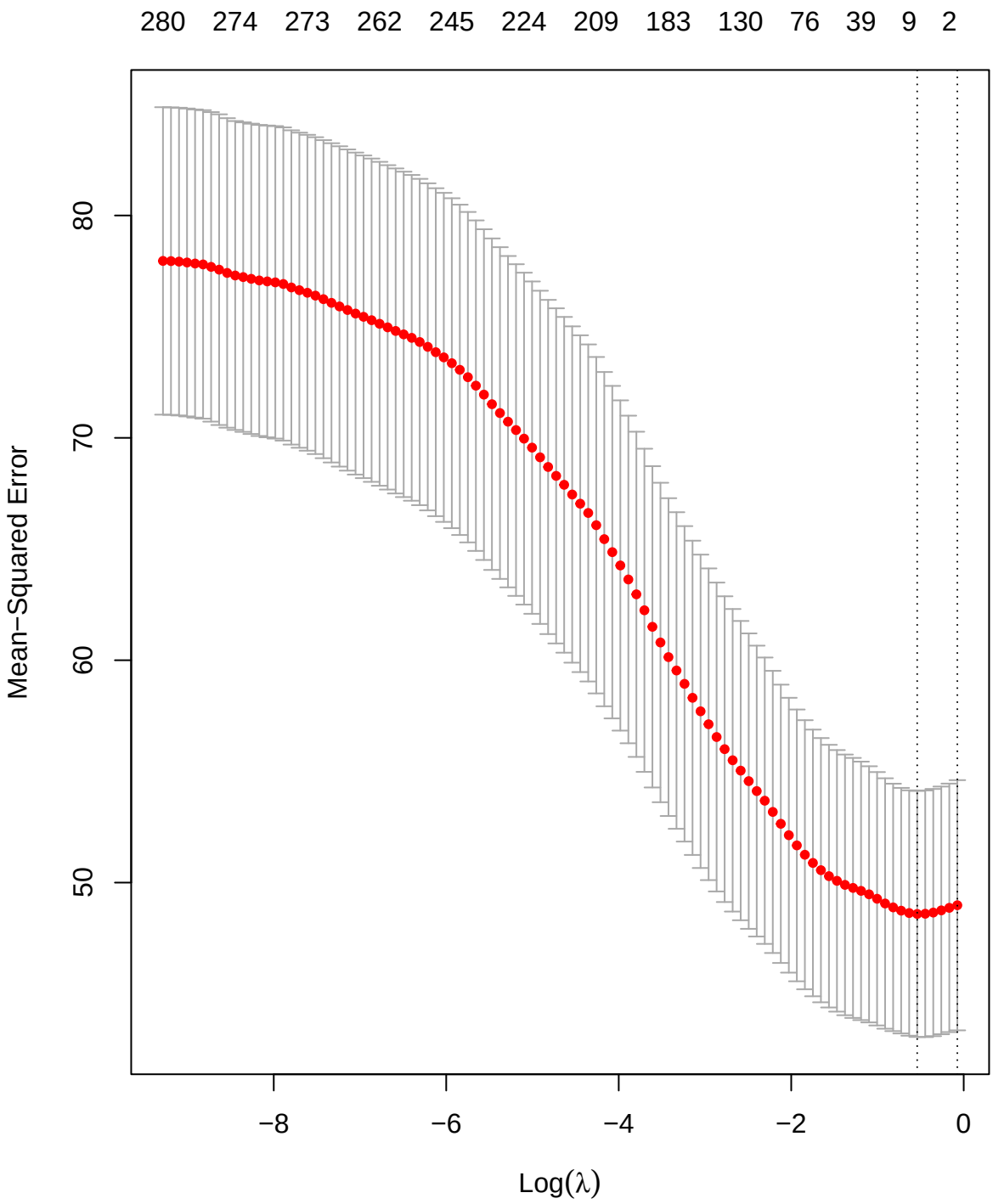

```
## [1] "gain_c_ltr_cogsoc_comp"
## [1] 687 290
```



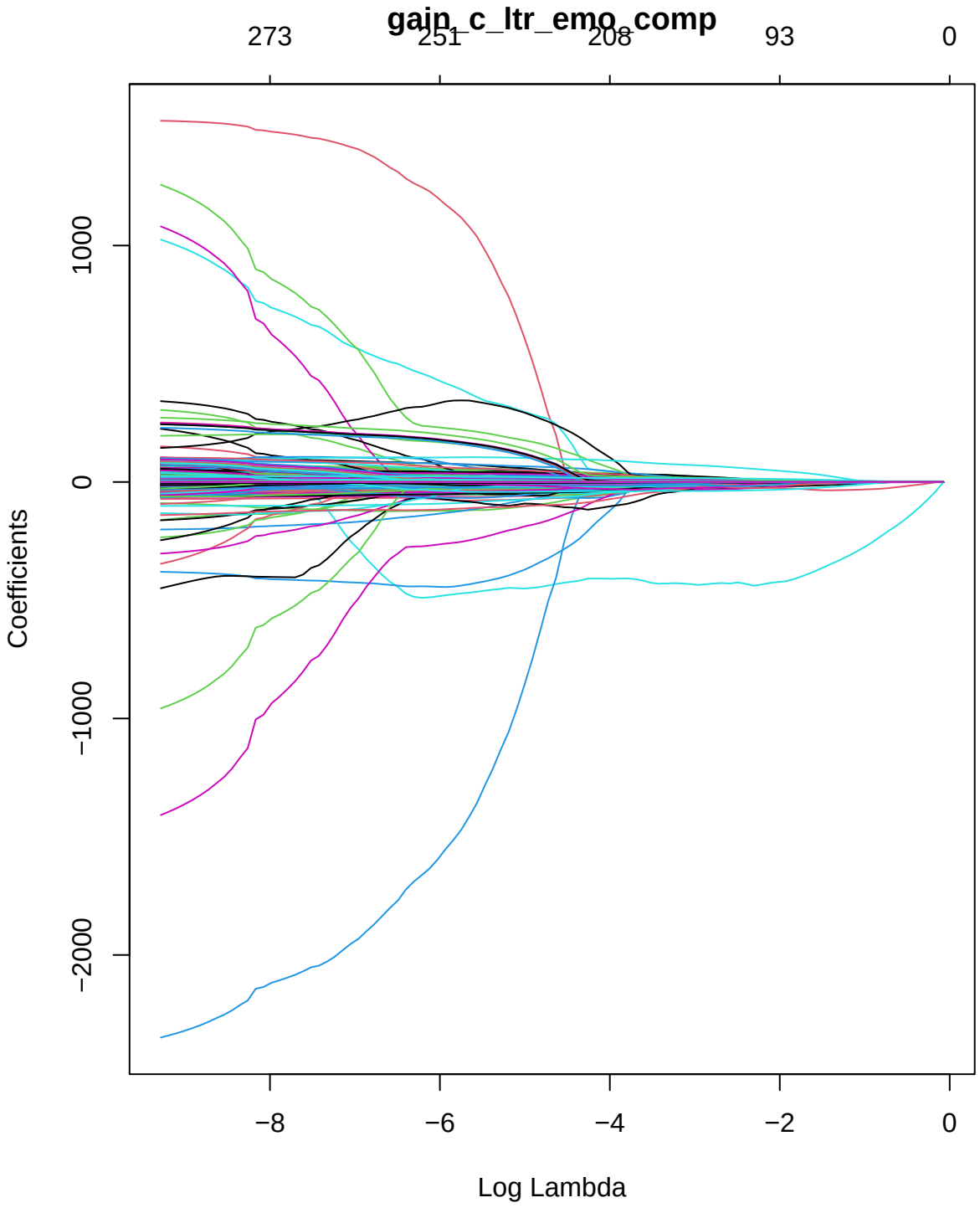

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



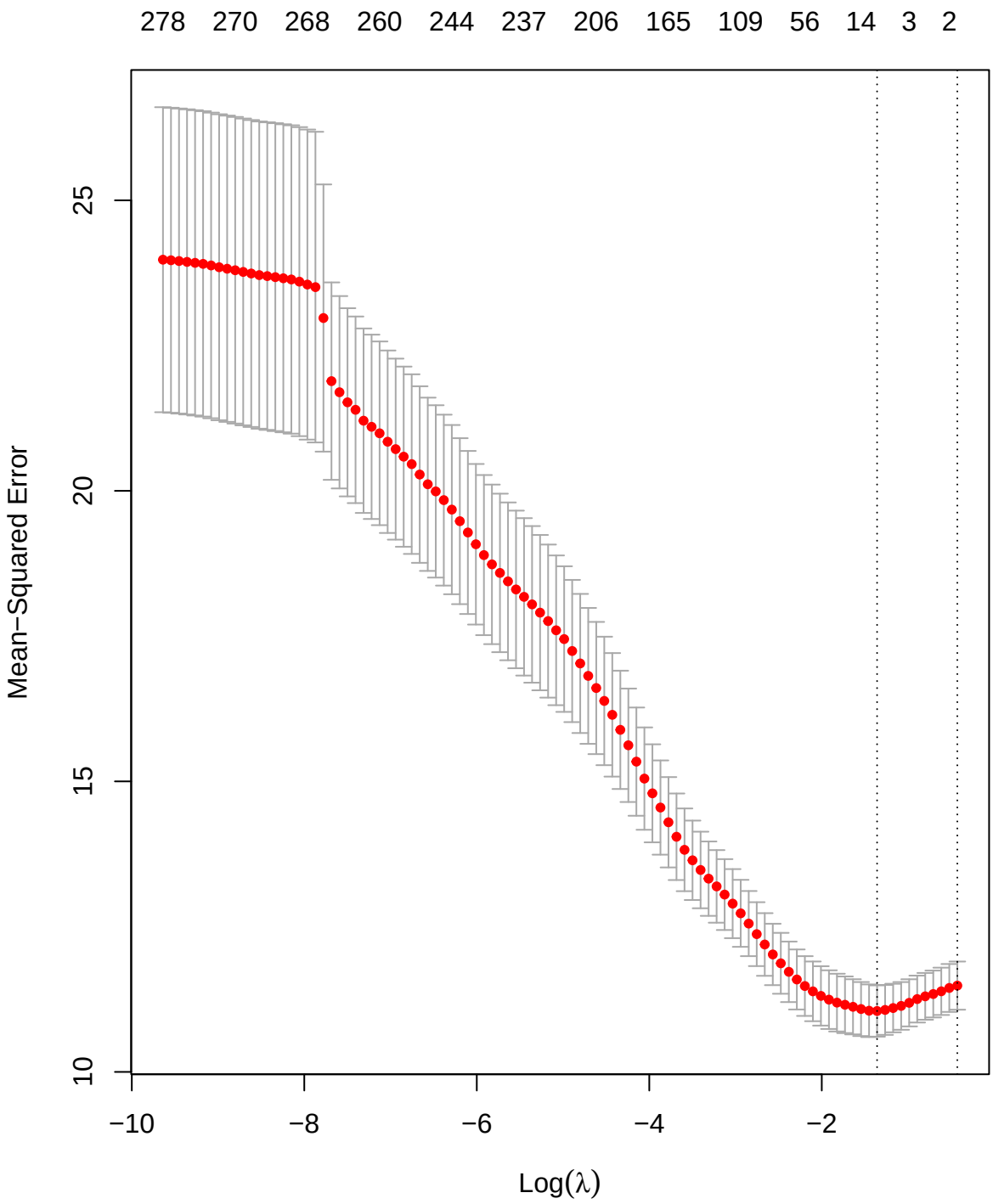

```
## [1] "gain_c_ltr_emo_comp"
## [1] 687 290
```



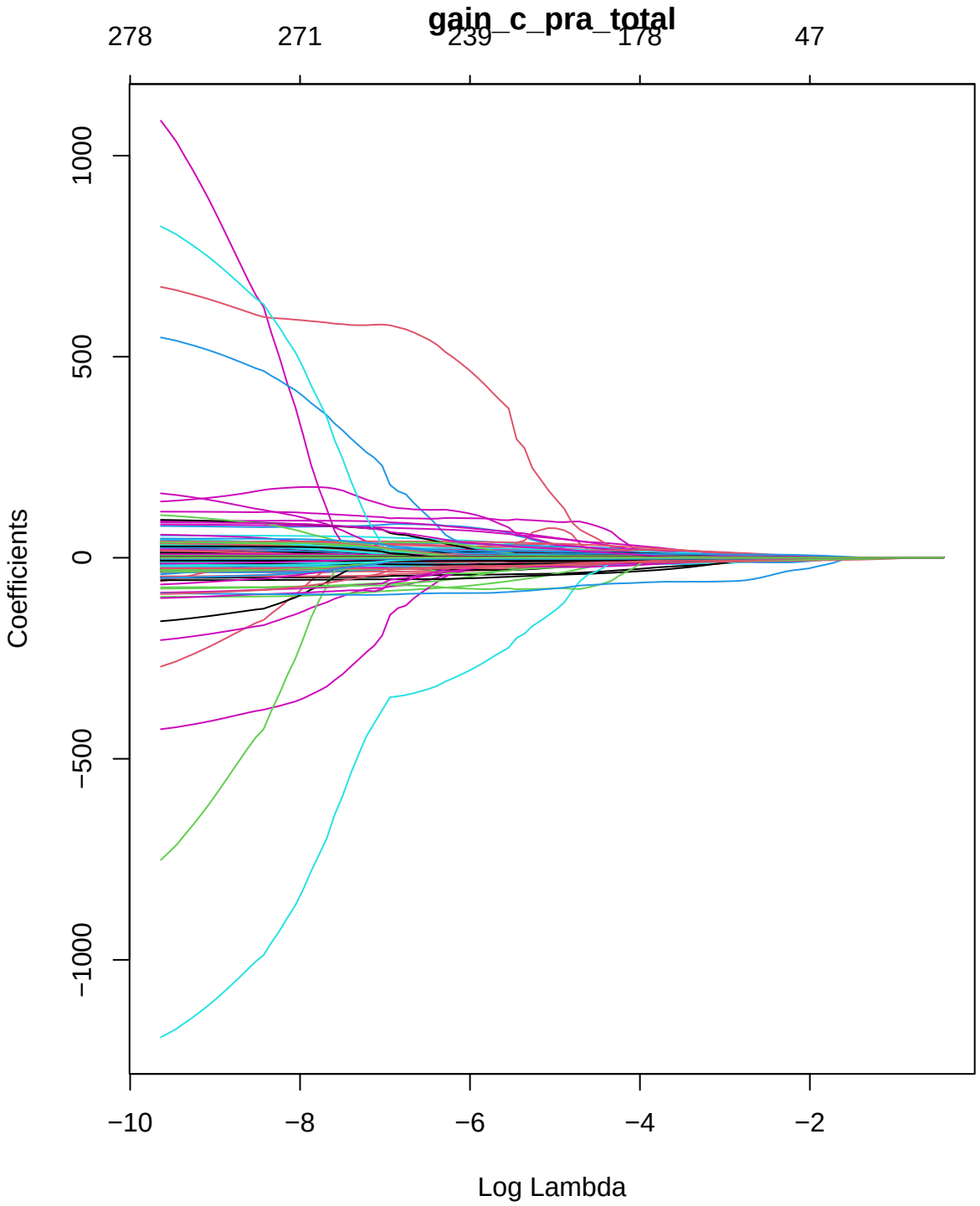

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



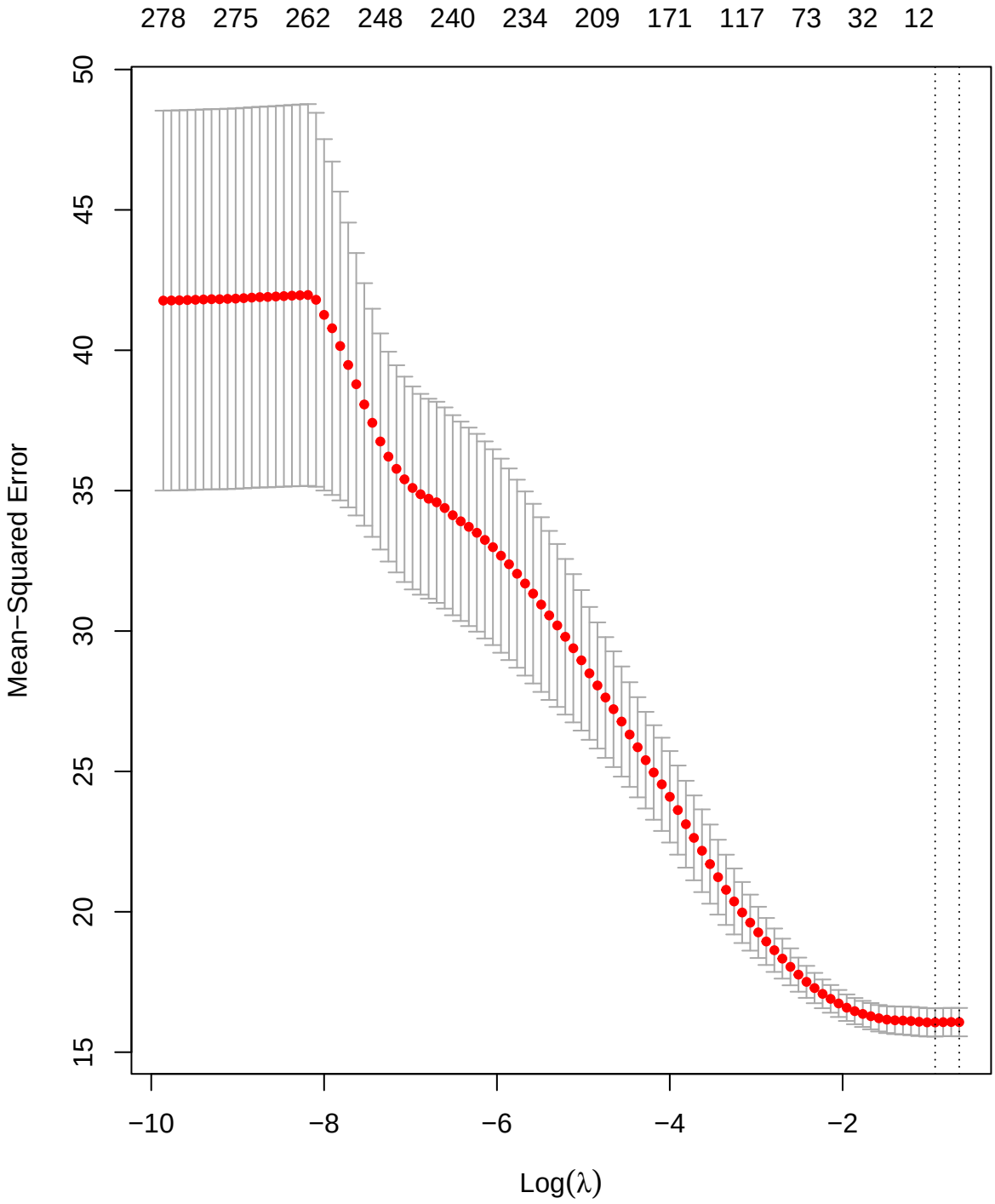

```
## [1] "gain_c_pra_total"
## [1] 612 290
```



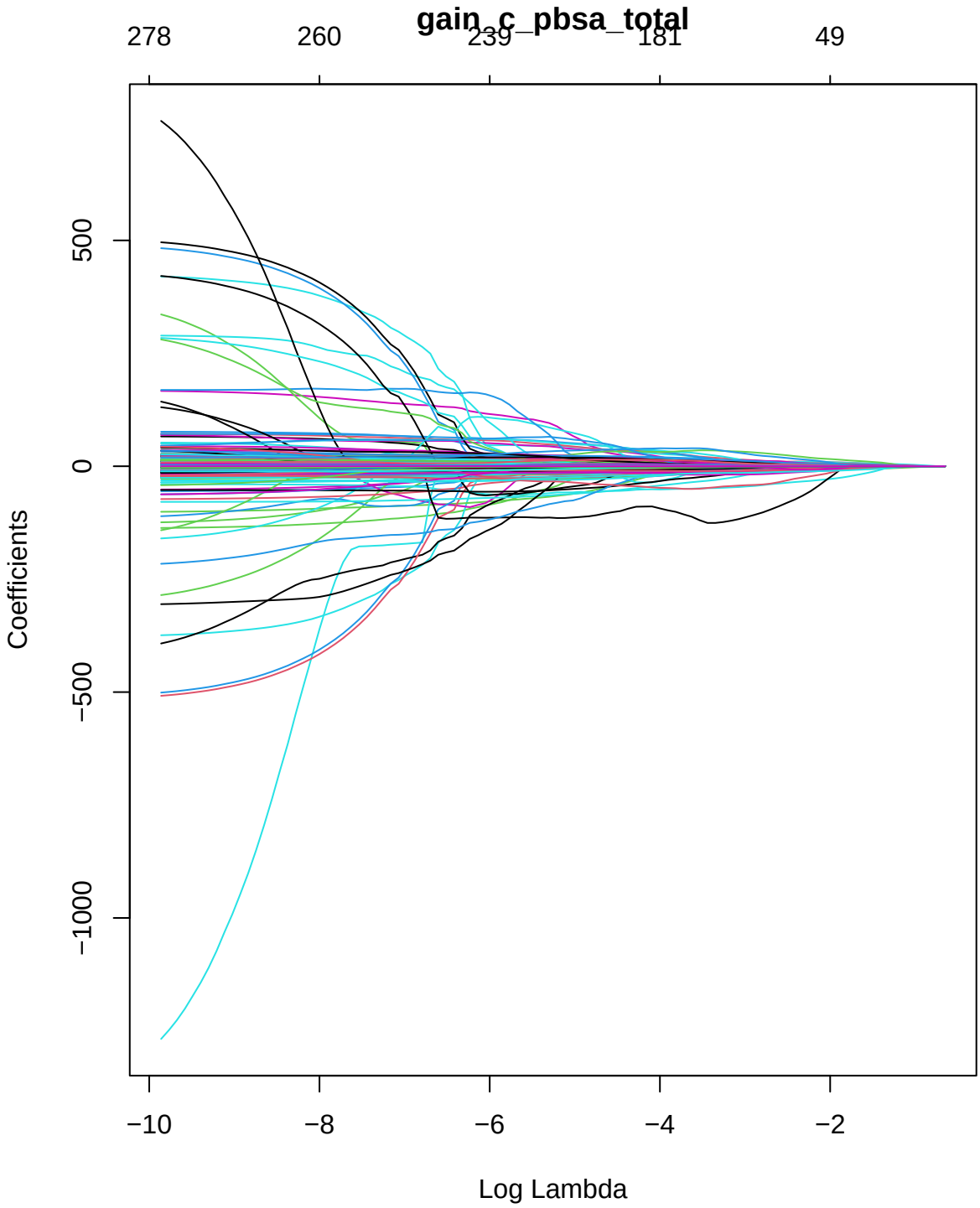

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



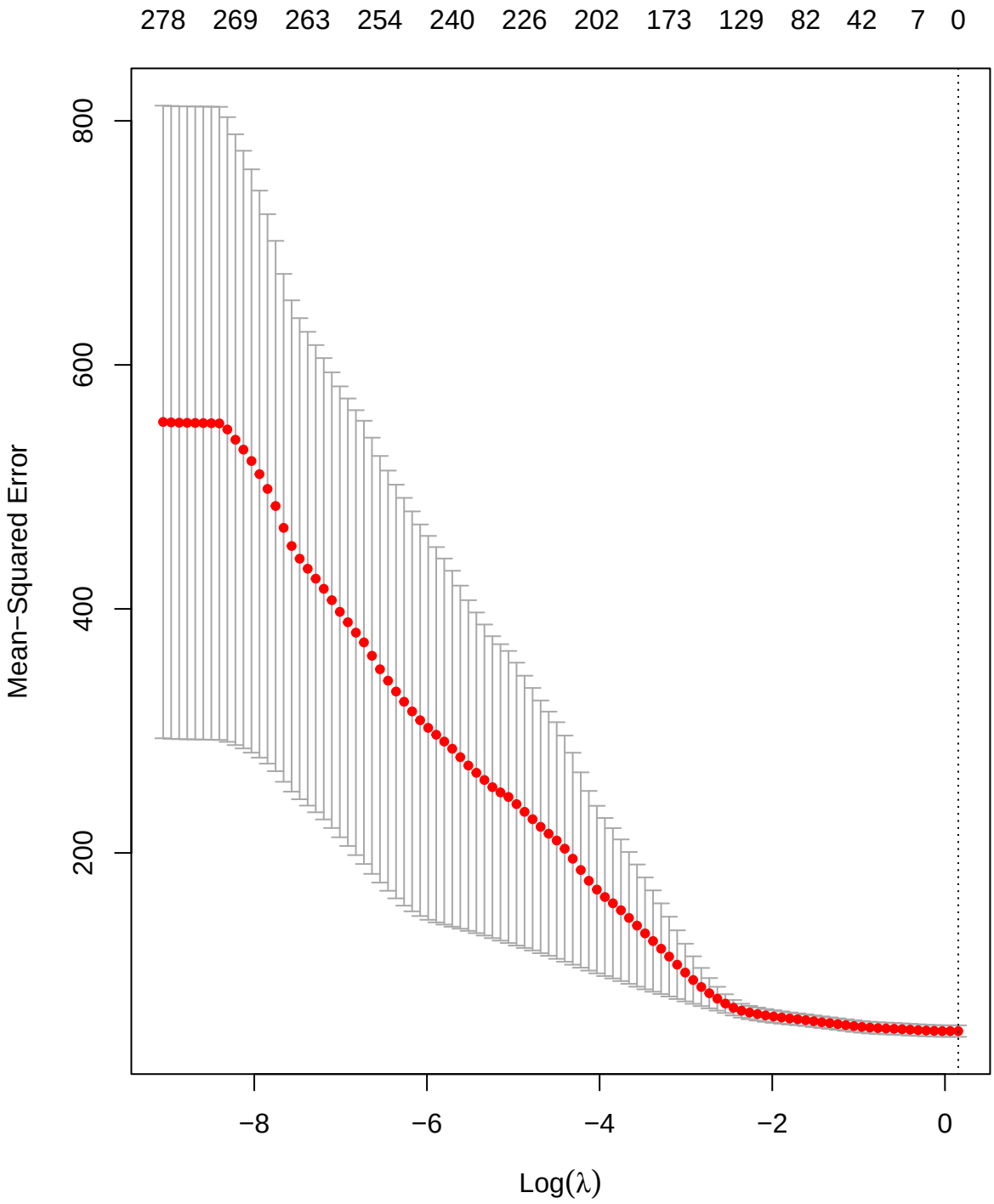

```
## [1] "gain_c_pbsa_total"
## [1] 622 290
```



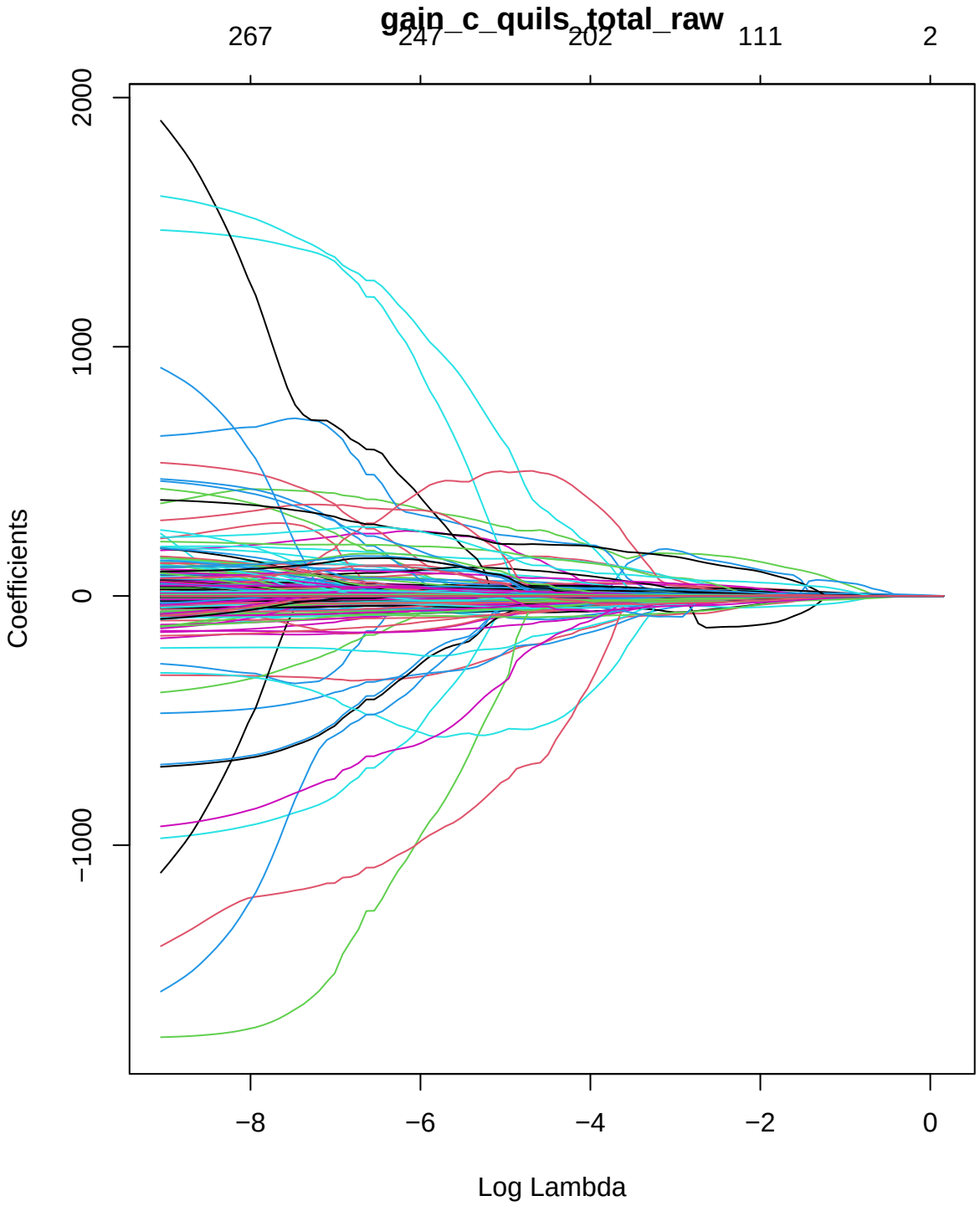

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



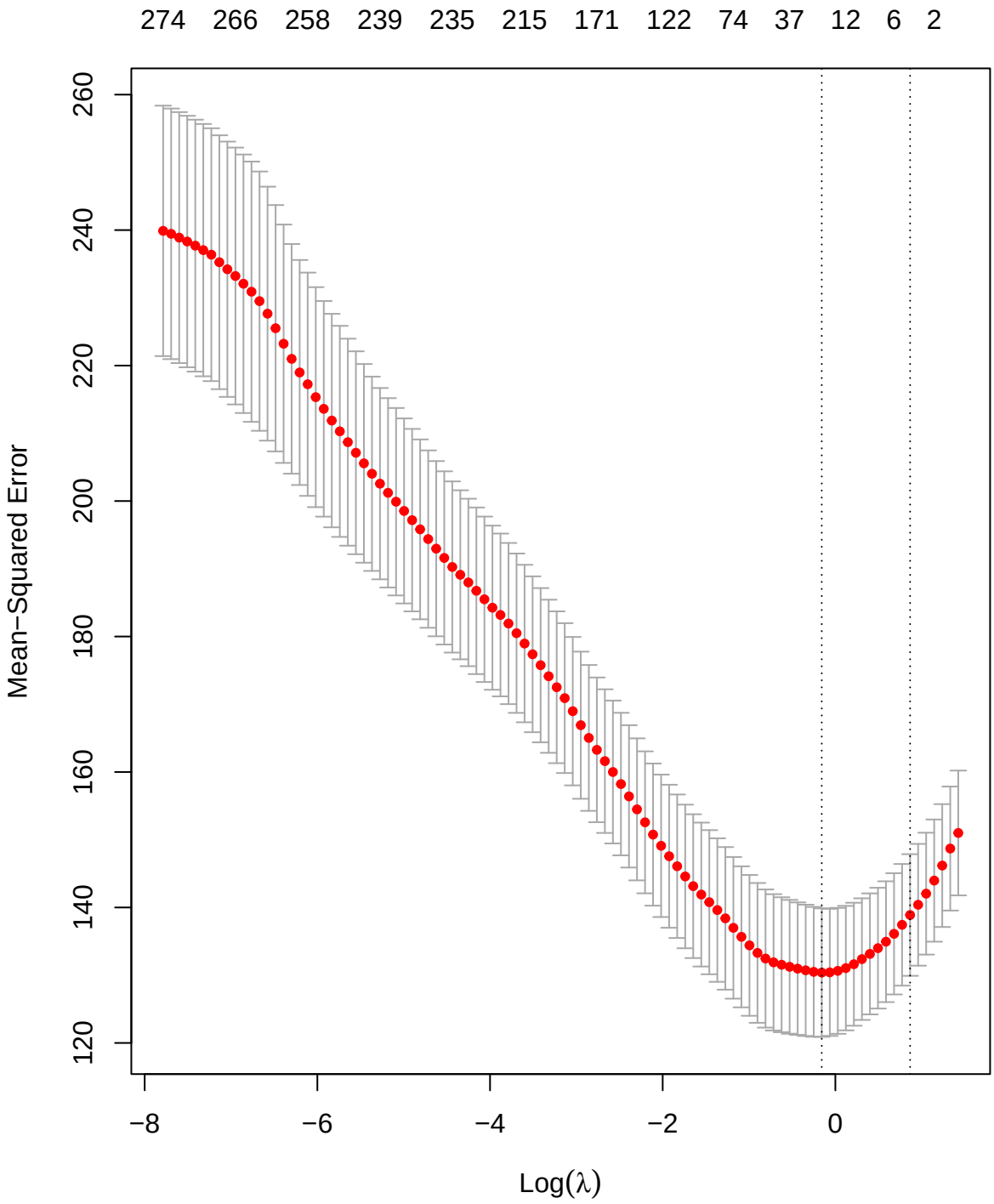

```
## [1] "gain_c_quils_total_raw"
## [1] 434 290
```



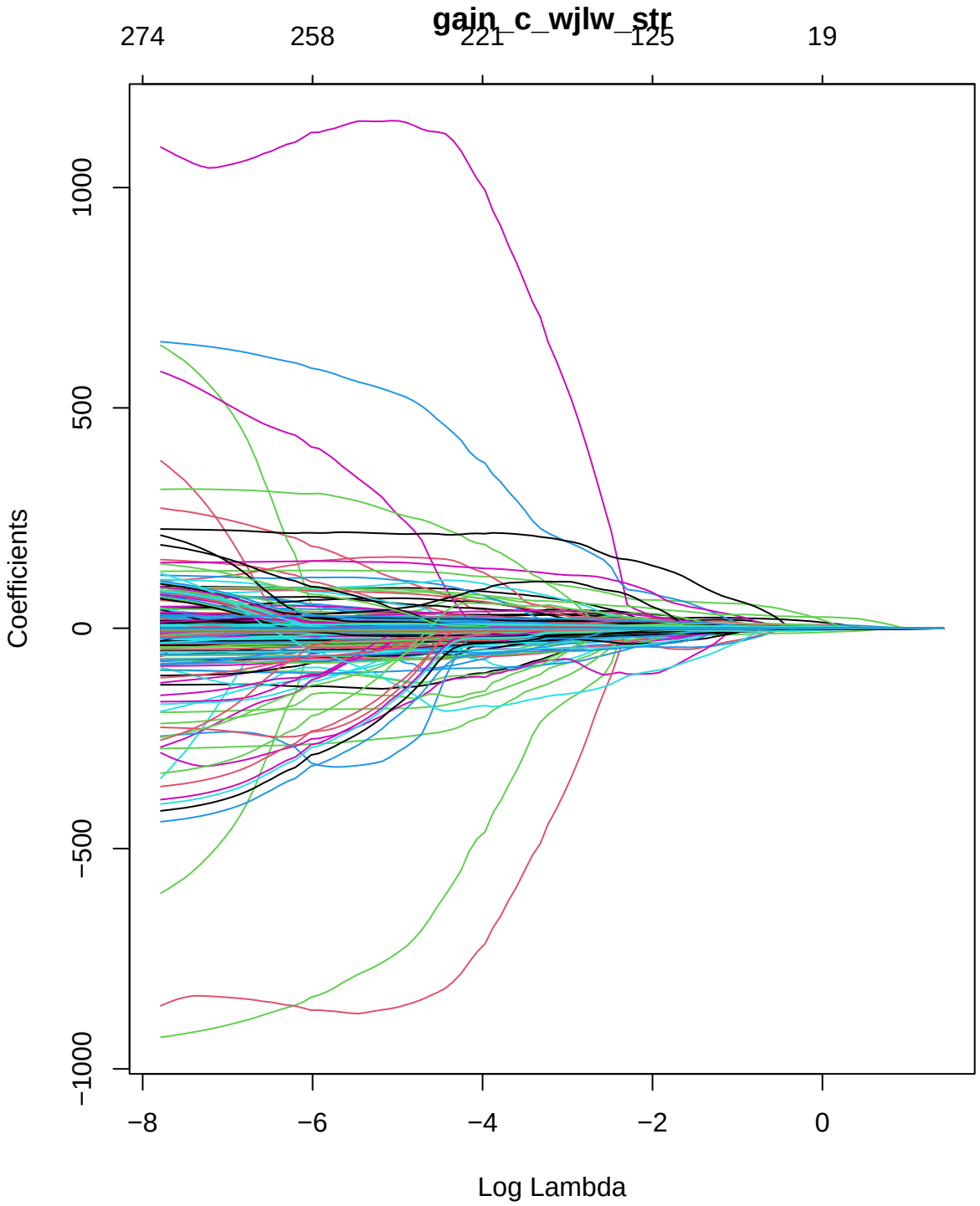

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```



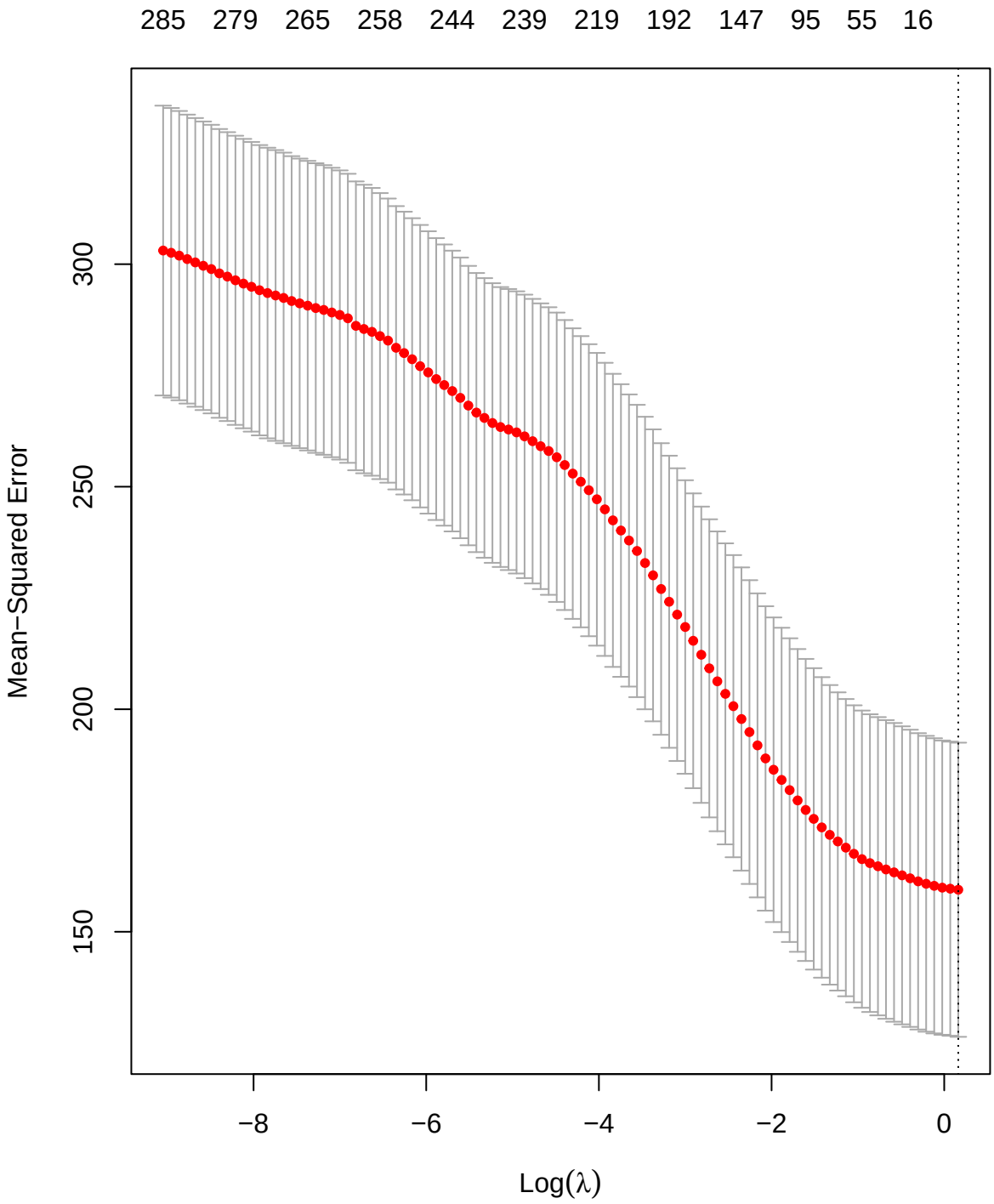

```
## [1] "gain_c_wjlw_str"
## [1] 683 290
```

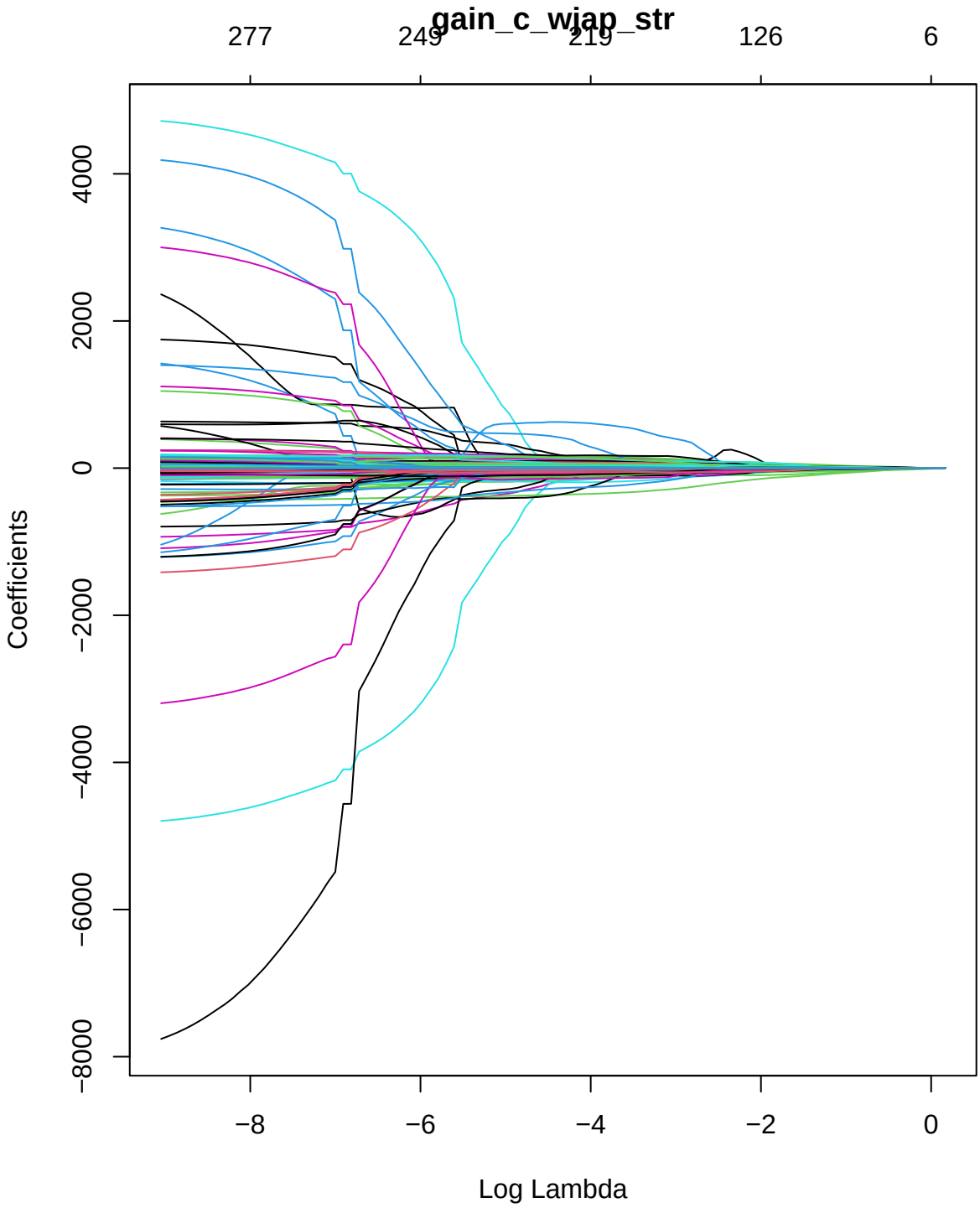



```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```




```
## [1] "gain_c_wjap_str"
## [1] 661 290
```


285 279 265 258 244 239 219 192 147 95 55 16
Mean−Squared Error
300
250
200
150
−8
−6
−4
−2
0
Log(λ)




```
# loop through the names associated with each outcome -- add 1 to corresponding entry
count_coefs_y2 <- list()
for (outcome in coefs_y2) {
  for (name in names(outcome)) {
    count_coefs_y2[[name]] <- ifelse(is.null(count_coefs_y2[[name]]), 1,
                                     count_coefs_y2[[name]] + 1)
  }
}
```

```r
#Convert output to a df for plotting and such
count_coefs_y2 <- count_coefs_y2 %>%
  unlist() %>%
  as.data.frame(row.names=TRUE)
```

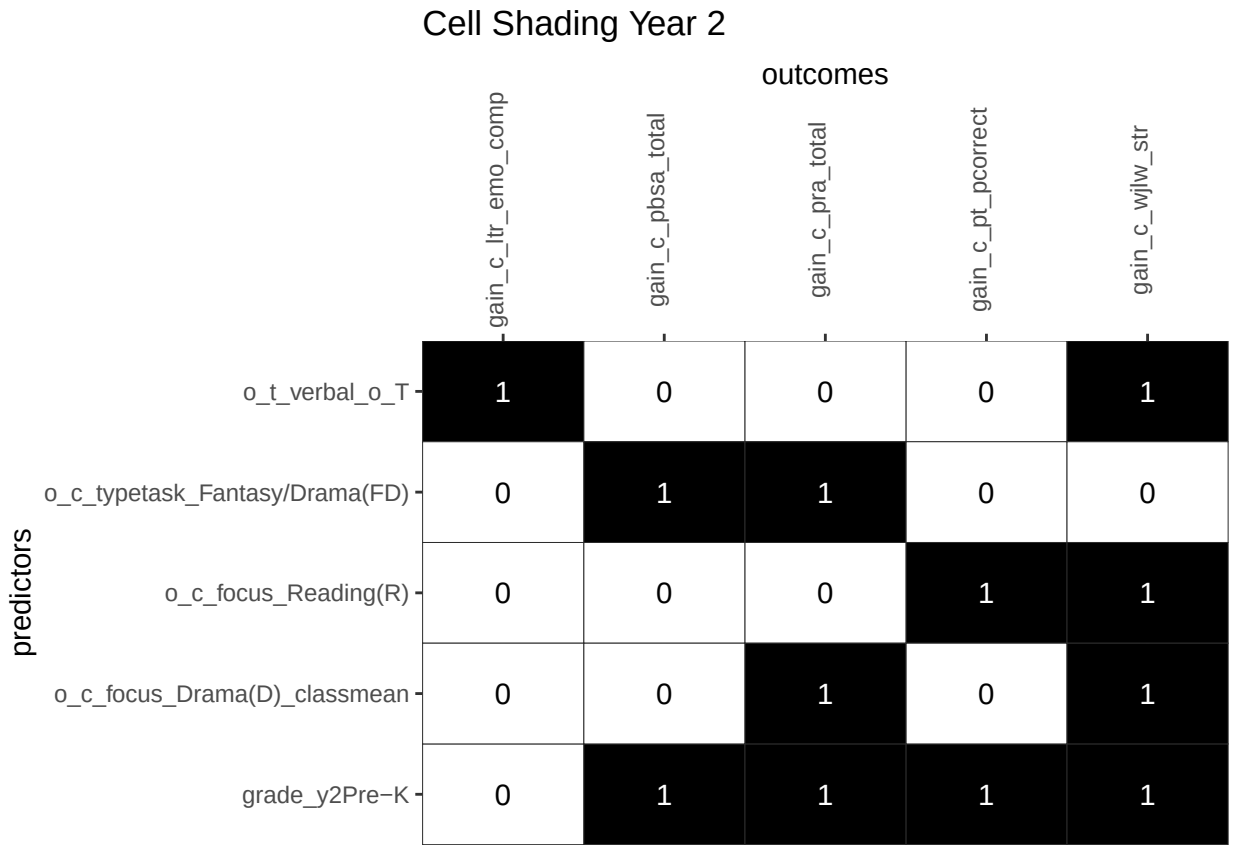
```
## Warning in as.data.frame.numeric(., row.names = TRUE): 'row.names' is not a
## character vector of length 36 -- omitting it. Will be an error!
```

```r
count_coefs_y2$predictor <- row.names(count_coefs_y2)

names(count_coefs_y2) <- c("frequency", "predictor")
count_coefs_y2 <- count_coefs_y2[order(-count_coefs_y2$frequency), ]

# pull predictors that have more than 1 appearance

count_coefs_y2_top <- count_coefs_y2[count_coefs_y2$frequency > 1, ]
count_coefs_y2_top$outcomes <- rep(NA, nrow(count_coefs_y2_top))

list_store <- list()

for (top_predictor in count_coefs_y2_top$predictor) {
  for (i in 1:length(coefs_y2)) {
    outcome_name <- names(coefs_y2)[i]
    if (top_predictor %in% names(coefs_y2[[i]])) {
      if (is.null(list_store[[top_predictor]])) {
        list_store[[top_predictor]] <- list(outcome_name)
      }
      else {
        list_store[[top_predictor]] <- c(list_store[[top_predictor]], outcome_name)
      }
    }
  }
}
```

```r
# all the unique outcomes that appear, so that they can be the columns in the
# binary table
unique_outcomes <- unique(unlist(unique(sapply(list_store, unlist))))
unique_predictors <- names(list_store)

# create data frame to translate into binary table
expand.grid(
  unique_predictors,
  unique_outcomes,
  stringsAsFactors = FALSE
) %>%
  set_names(c("predictors", "outcomes")) %>%
  mutate(value = rep(0, n())) -> binary_df

for (predictor in names(list_store)) {
  for (outcomes in list_store[[predictor]]) {
    outcomes_unlisted <- unlist(outcomes)
    for (outcome in outcomes_unlisted) {
        binary_df[binary_df$predictors == predictor &
                    binary_df$outcomes == outcomes_unlisted, ]$value <- 1
```

```
    }
  }
}
```

```
mutate(
  binary_df,
  fill = ifelse(value == 1, "black", "white"),
  color = ifelse(value == 1, "white", "black"),
  address = factor(predictors, levels = sort(unique(predictors), decreasing = TRUE))
) -> cell_shading_df

ggplot(cell_shading_df, aes(x = outcomes, y = predictors)) +
  geom_tile(
    aes(fill = I(fill)),
    color = "#2b2b2b", size=0.125,
  ) +
  geom_text(
    aes(label = value, color = I(color))
  ) +
  scale_x_discrete(expand=c(0,0), position = "top") +
  scale_y_discrete(expand=c(0,0)) +
  labs(title = "Cell Shading Year 2", x = "outcomes", y = "predictors") +
  # hrbrthemes::theme_ipsum_rc(grid="XY") +
  theme(axis.text.x = element_text(angle=90, vjust=0.5, hjust=0.5))
```



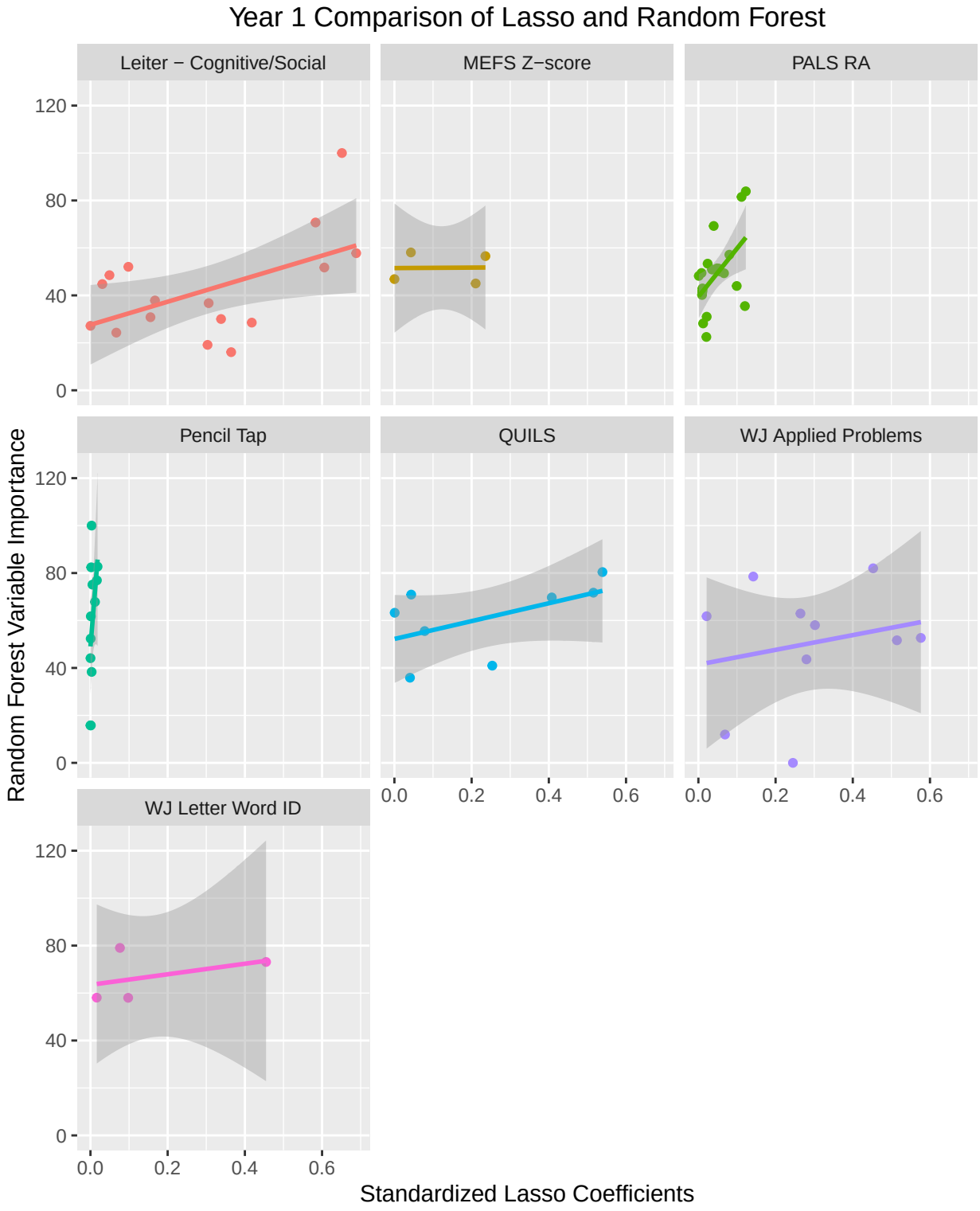

```
varimp_y2 <- list()
# NOTE: change to full data set here
```

```r
test_data <- head(outcomes_and_obs_full_y2, 100)
# make sure this works with small subset of data
for (i in gain_ind) {
  name <- names(outcomes_and_obs_full_y2)[i]
  df_analysis <- test_data %>%
    filter(!is.na(test_data[[name]])) %>%
    dplyr::select(c(name, "o_c_verbal_Fuss/Cry(FC)":actualtype))
    # ask about verbal_fuss vs o_c_verbal_Talk(T) (original)
    # mutate_at(vars("o_c_verbal_Fuss/Cry (FC)":actualtype), replace.na)
  print(name)
  # options(na.action="na.pass")
  x = model.matrix(as.formula(paste(name, "~ .")), data = df_analysis)

  # Fit the random forest model
  rf_fit <- train(as.formula(paste(name, "~ .")), #Use all variables in the prediction
                data = df_analysis, #Use the training data
                method = "ranger",
                importance = "permutation",
                # ntree = 500,
                na.action=na.pass)

  varImp(rf_fit) %>%
    pluck(1) %>%
    rownames_to_column("var") %>%
    ggplot(aes(x = reorder(var, Overall), y = Overall)) +
    geom_col(fill = "grey75") +
    coord_flip() +
    theme_minimal()

  # store variable importances as a data frame
  df <- as.data.frame(varImp(rf_fit)$importance)
  # arrange in descending order (most to least important), and put into list
  varimp_y2[[name]] <- df %>% arrange(desc(Overall))
}
```

```r
# comparing lasso and random forest results, for year 1
outcomes <- names(outcomes_and_obs_full_y1)[gain_ind]
varimp_y1 <- rf_y1_obj[[1]]
comparison_df <- data.frame(matrix(ncol = 4, nrow = 0))

for (outcome in outcomes) {
  # print(outcome)
  lasso_coefs <- coefs_y1[[outcome]]
  # get rid of intercept
  if (length(lasso_coefs) == 0) {
    next
  }
  lasso_coefs <- lasso_coefs[2:length(lasso_coefs)]
  rf_coefs <- varimp_y1[[outcome]]
  # removing illegal characters for consistency
  row.names(rf_coefs) <- gsub(" ", "", row.names(rf_coefs))
  row.names(rf_coefs) <- gsub("`", "", row.names(rf_coefs))
  # this will be in the order of the coefficients in the lasso model
  overlap <- rf_coefs[names(lasso_coefs),]
```

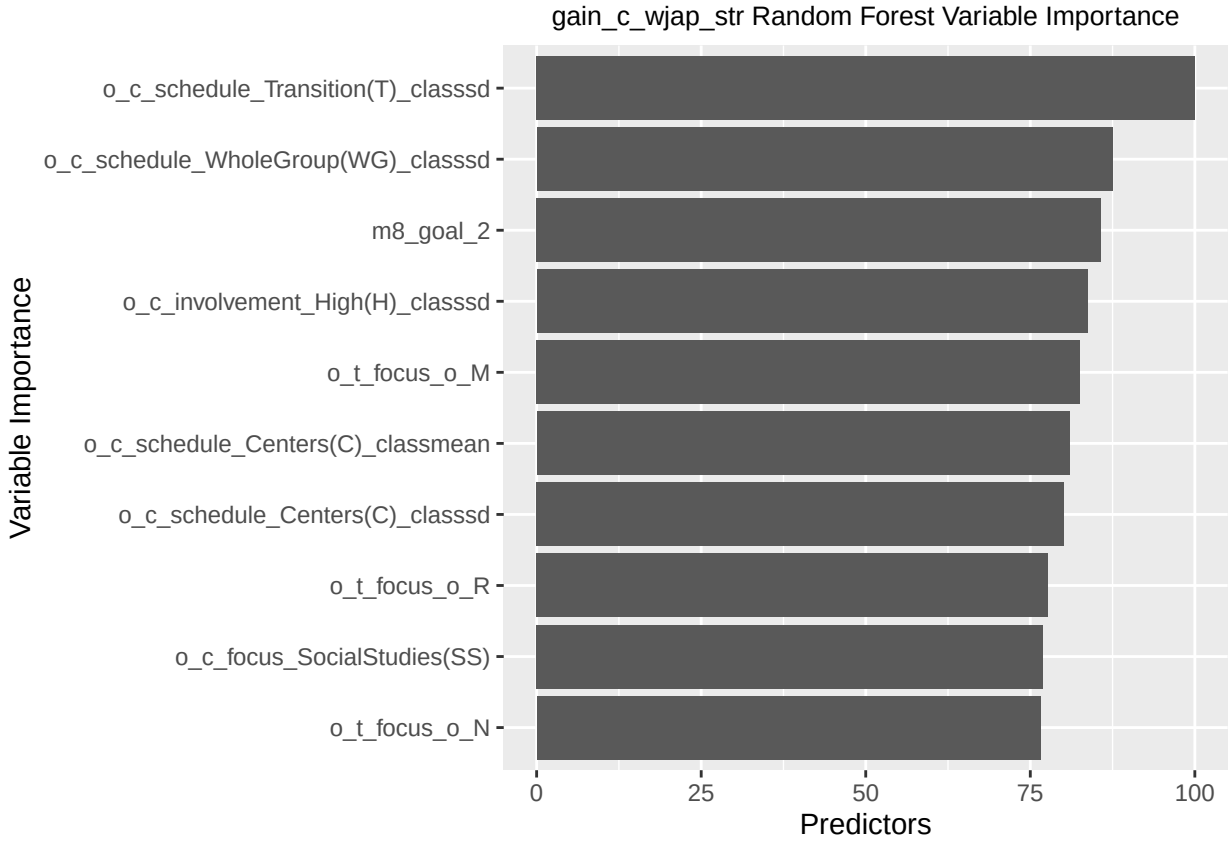
```
  x <- rep(NA, length(lasso_coefs))
  for (i in 1:length(lasso_coefs)) {
    x[i] <- abs(lasso_coefs[[i]])
  }
  comparison_df <- rbind(comparison_df, cbind(x, overlap,
                                              rep(outcome, length(x)),
                                              rep(proper_names[[outcome]],
                                                  length(x))))
}

comparison_df <- comparison_df %>%
  rename("lasso" = "x", "rf" = "overlap", "gains" = "V3", "outcome"=V4) %>%
  mutate_at(vars(lasso, rf), as.numeric)

## TO DO: need to pull the actual names of the outcomes for better plot titles
ggplot(comparison_df, aes(lasso, rf, color = outcome)) +
  geom_point() +
  facet_wrap(~outcome) +
  geom_smooth(method='lm') +
  theme(legend.position="none") +
  labs(title="Year 1 Comparison of Lasso and Random Forest",
       x ="Standardized Lasso Coefficients",
       y = "Random Forest Variable Importance") +
  theme(plot.title = element_text(hjust = 0.5))
```

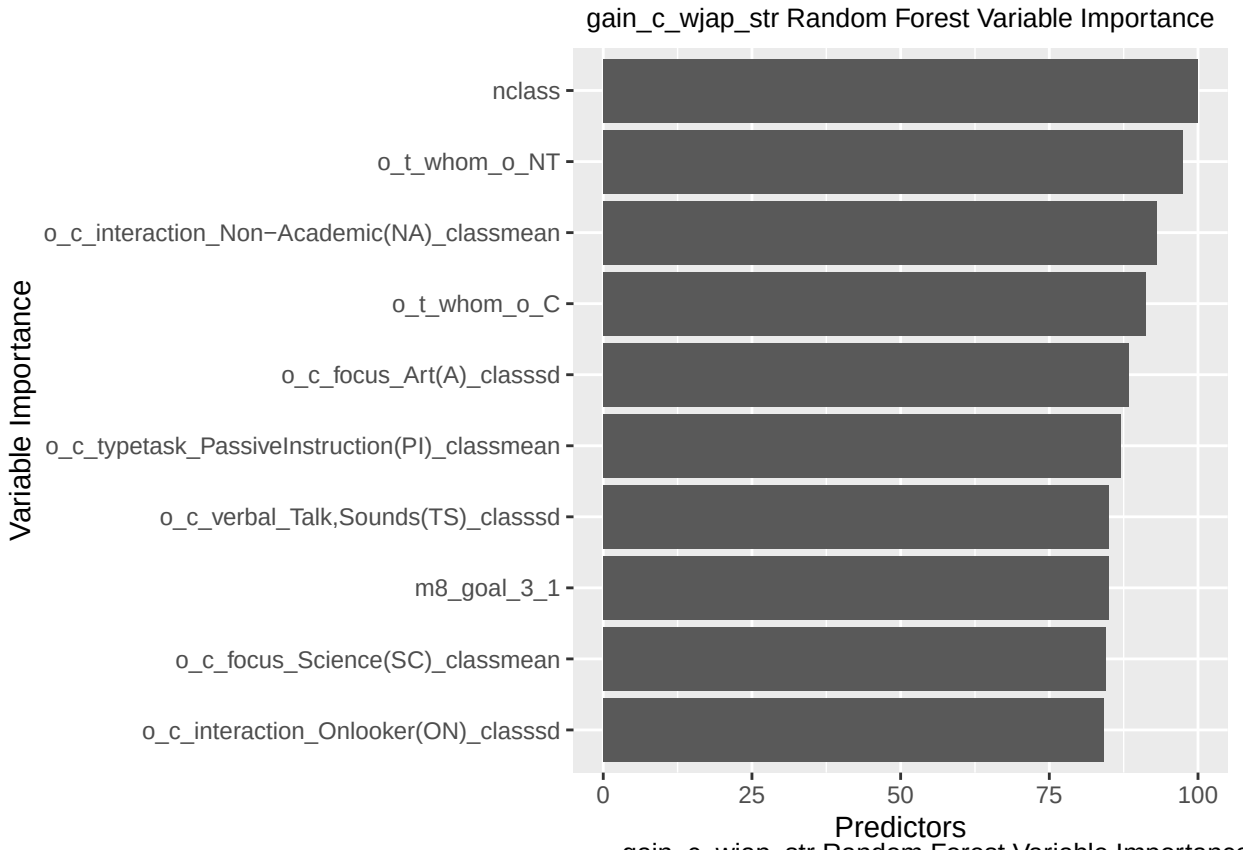
```
## `geom_smooth()` using formula 'y ~ x'
```

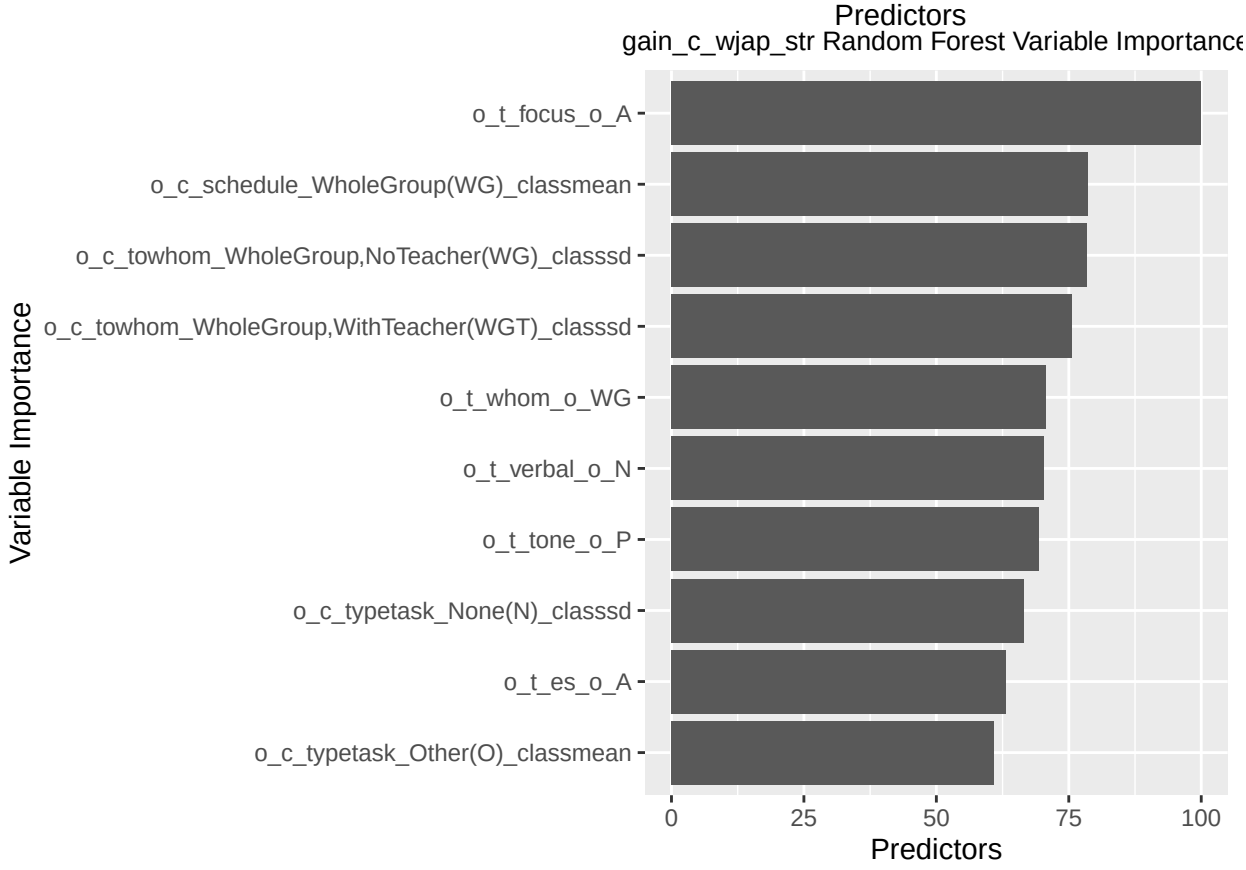


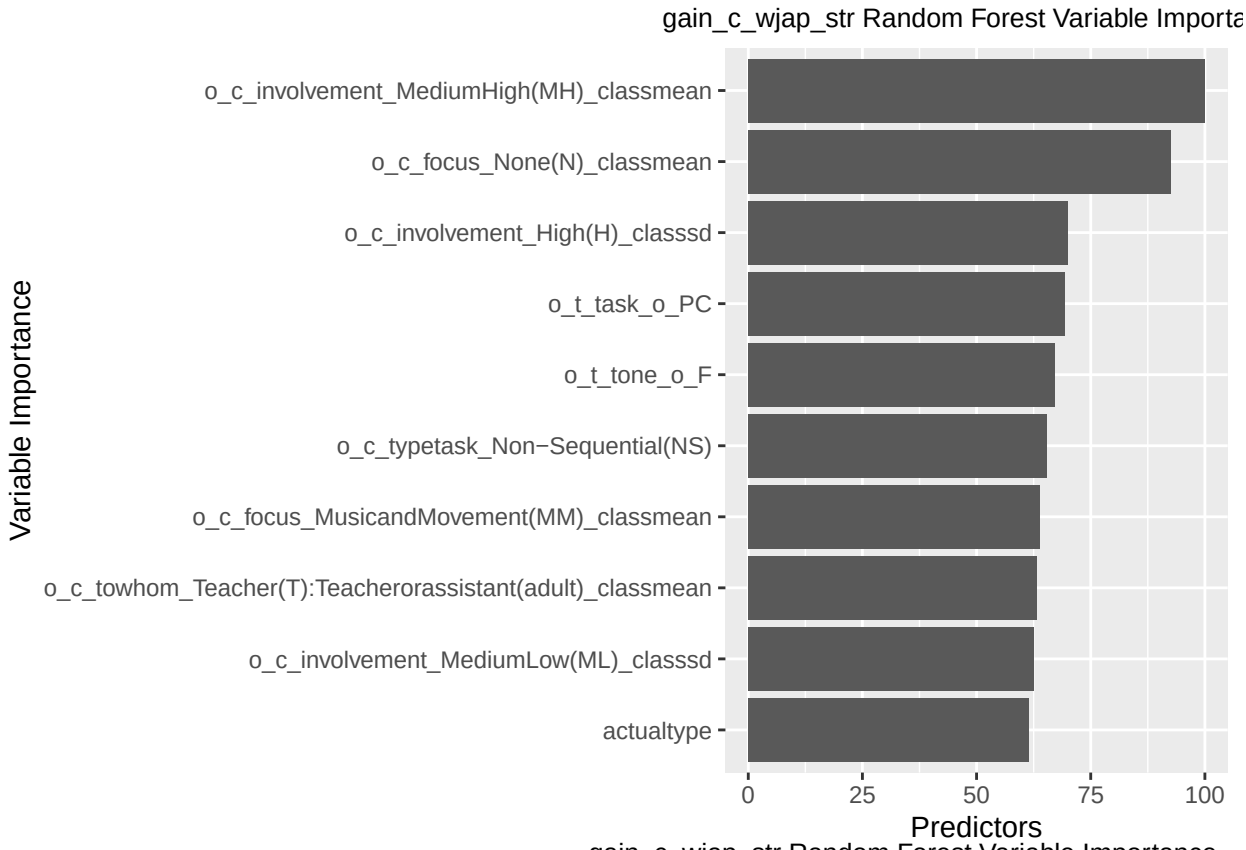

```
var_imp_y1 <- rf_y1_obj[[1]]
for (name in names(var_imp_y1)) {
  varimp_df <- head(var_imp_y1[[name]], 10)

  row.names(varimp_df) <- gsub(" ", "", row.names(varimp_df))
  row.names(varimp_df) <- gsub("`", "", row.names(varimp_df))
```

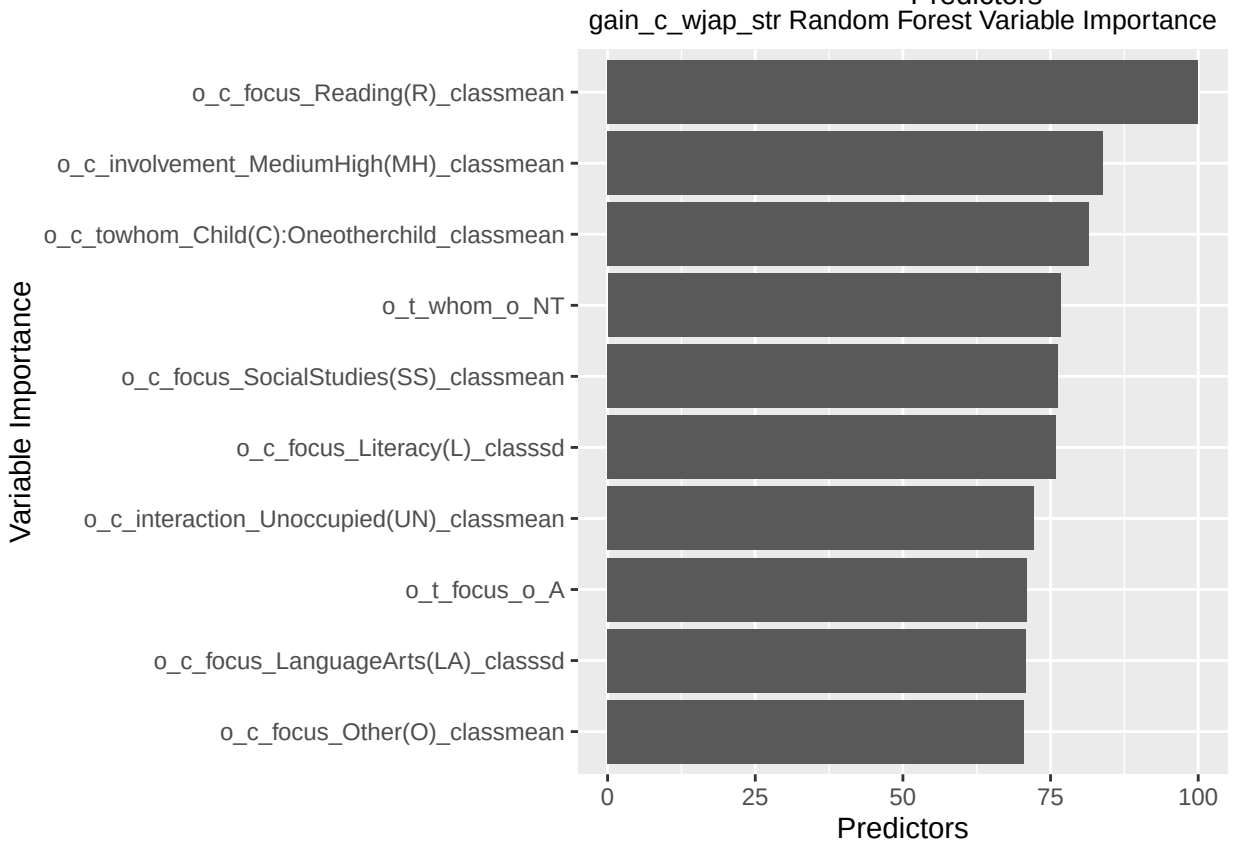
```
  varimp_df$predictors <- row.names(varimp_df)

  plot <- ggplot(varimp_df, aes(x=reorder(predictors, +Overall), y=Overall)) +
    geom_col() +
    coord_flip() +
    labs(title=paste(outcome, "Random Forest Variable Importance"),
       x ="Variable Importance",
       y = "Predictors") +
    theme(plot.title = element_text(size = 10, hjust = 0.5))
  print(plot)
}
```

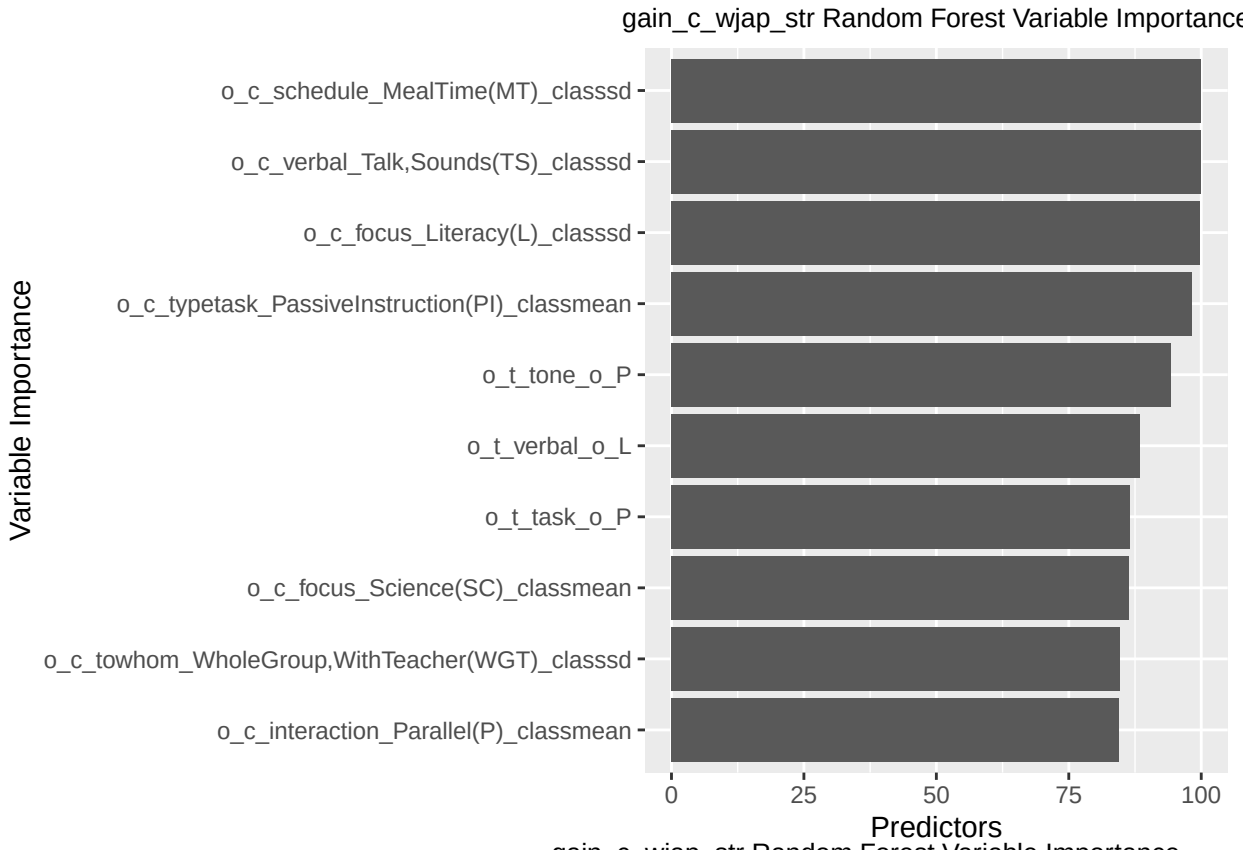


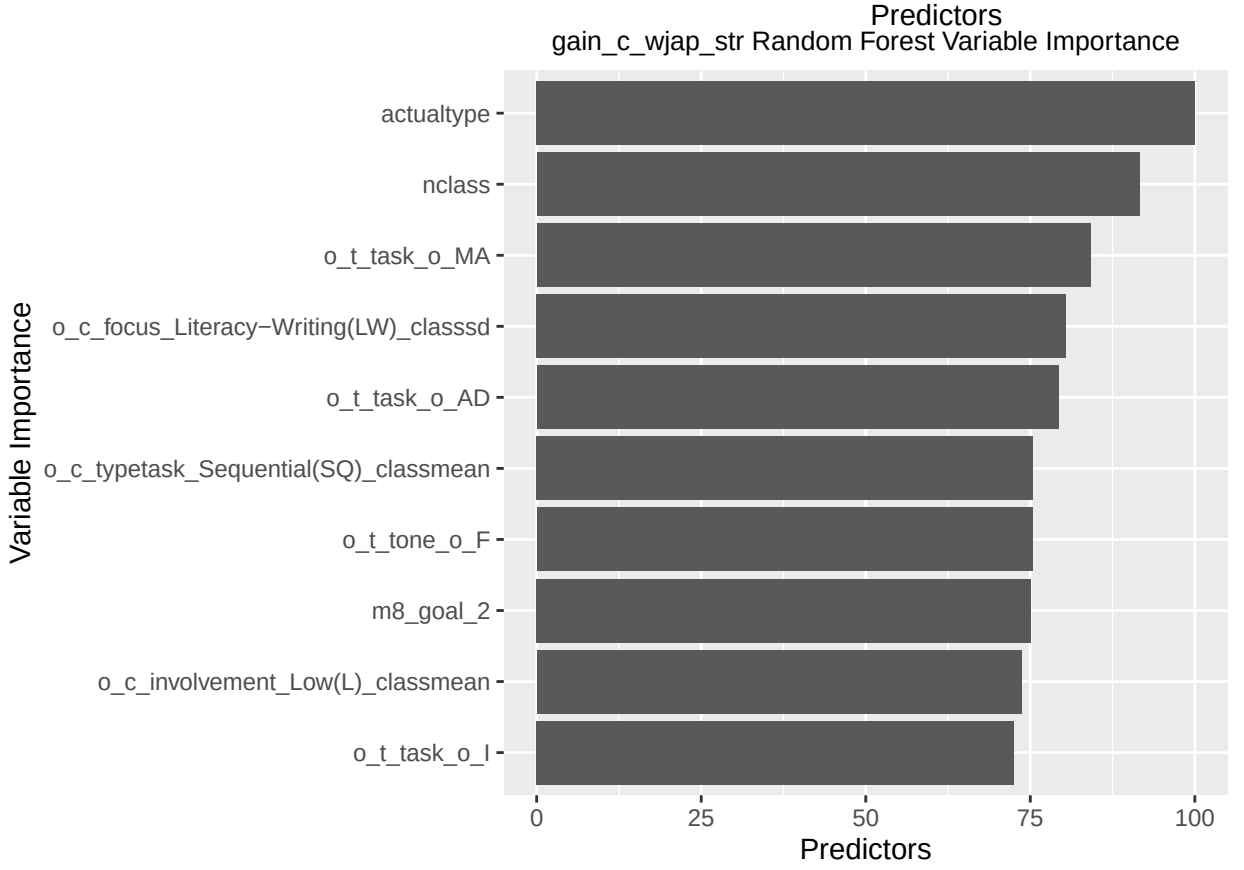

gain_c_wjap_str Random Forest Variable Importance
nclass
o_t_whom_o_NT
o_c_interaction_Non−Academic(NA)_classmean
o_t_whom_o_C
o_c_focus_Art(A)_classsd
o_c_typetask_PassiveInstruction(PI)_classmean
o_c_verbal_Talk,Sounds(TS)_classsd
m8_goal_3_1
o_c_focus_Science(SC)_classmean
o_c_interaction_Onlooker(ON)_classsd
Variable Importance
0
25
50
75
100
Predictors

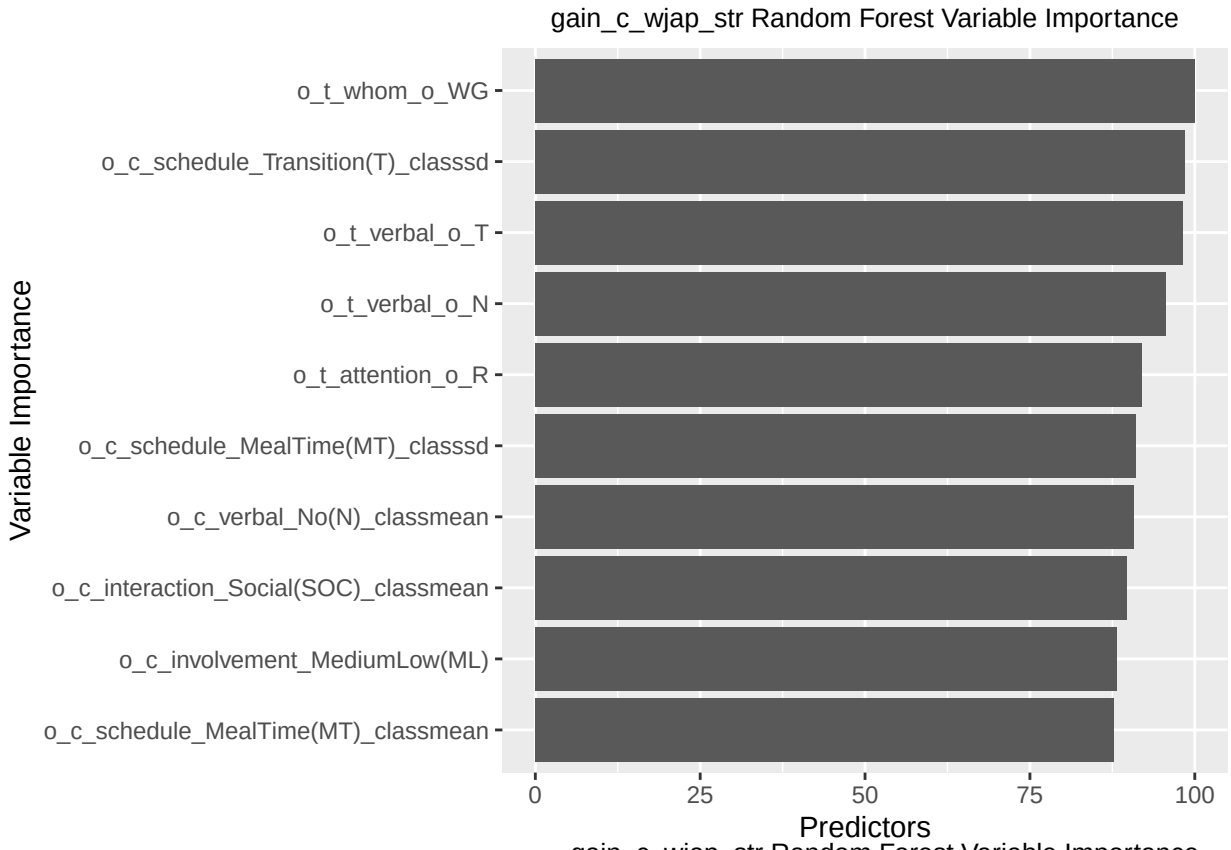

gain_c_wjap_str Random Forest Variable Importance
o_t_focus_o_A
o_c_schedule_WholeGroup(WG)_classmean
o_c_towhom_WholeGroup,NoTeacher(WG)_classsd
o_c_towhom_WholeGroup,WithTeacher(WGT)_classsd
o_t_whom_o_WG
o_t_verbal_o_N
o_t_tone_o_P
o_c_typetask_None(N)_classsd
o_t_es_o_A
o_c_typetask_Other(O)_classmean
Variable Importance
0
25
50
75
100
Predictors

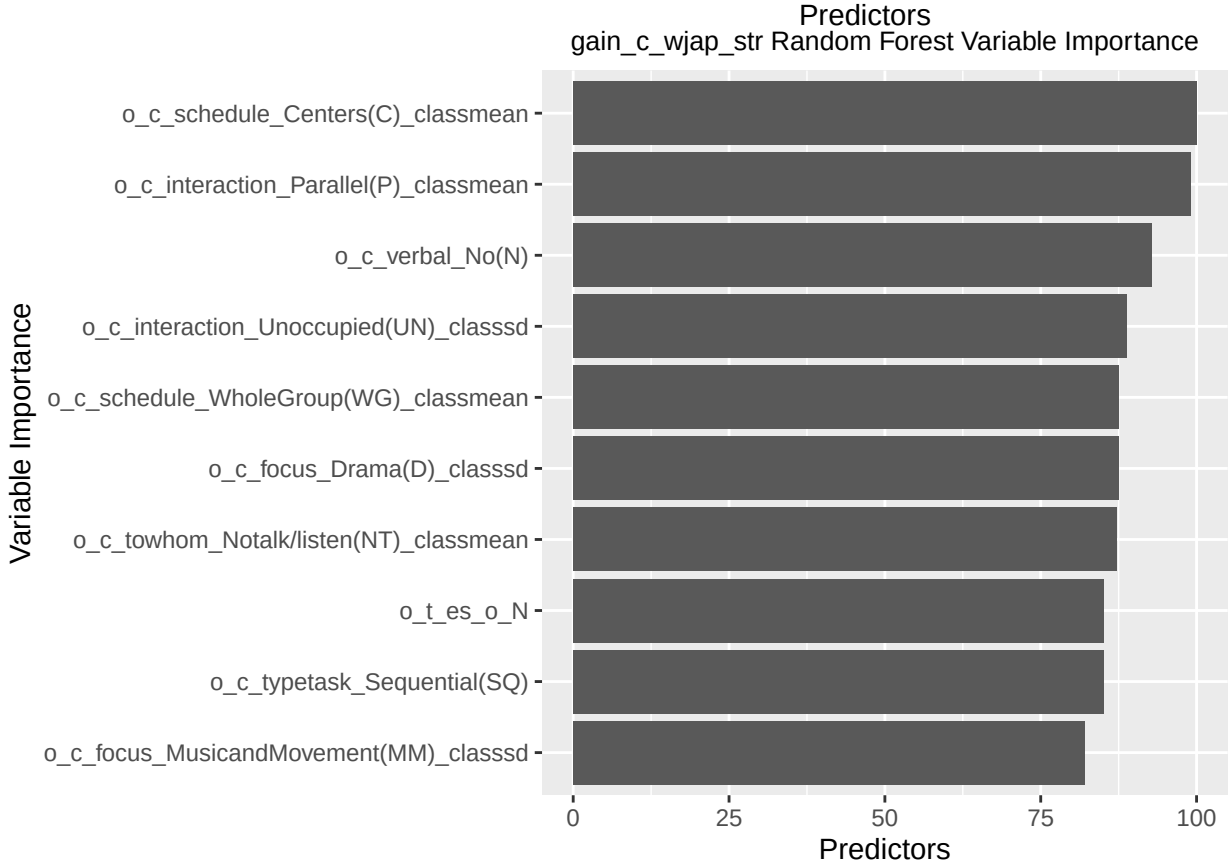

gain_c_wjap_str Random Forest Variable Importa
o_c_involvement_MediumHigh(MH)_classmean
o_c_focus_None(N)_classmean
o_c_involvement_High(H)_classsd
o_t_task_o_PC
o_t_tone_o_F
o_c_typetask_Non−Sequential(NS)
o_c_focus_MusicandMovement(MM)_classmean
o_c_towhom_Teacher(T):Teacherorassistant(adult)_classmean
o_c_involvement_MediumLow(ML)_classsd
actualtype
Variable Importance
0
25
50
75
100
Predictors


gain_c_wjap_str Random Forest Variable Importance
o_c_focus_Reading(R)_classmean
o_c_involvement_MediumHigh(MH)_classmean
o_c_towhom_Child(C):Oneotherchild_classmean
o_t_whom_o_NT
o_c_focus_SocialStudies(SS)_classmean
o_c_focus_Literacy(L)_classsd
o_c_interaction_Unoccupied(UN)_classmean
o_t_focus_o_A
o_c_focus_LanguageArts(LA)_classsd
o_c_focus_Other(O)_classmean
Variable Importance
0
25
50
75
100
Predictors


gain_c_wjap_str Random Forest Variable Importance
o_c_schedule_MealTime(MT)_classsd
o_c_verbal_Talk,Sounds(TS)_classsd
o_c_focus_Literacy(L)_classsd
o_c_typetask_PassiveInstruction(PI)_classmean
o_t_tone_o_P
o_t_verbal_o_L
o_t_task_o_P
o_c_focus_Science(SC)_classmean
o_c_towhom_WholeGroup,WithTeacher(WGT)_classsd
o_c_interaction_Parallel(P)_classmean
Variable Importance
0
25
50
75
100
Predictors


gain_c_wjap_str Random Forest Variable Importance
actualtype
nclass
o_t_task_o_MA
o_c_focus_Literacy−Writing(LW)_classsd
o_t_task_o_AD
o_c_typetask_Sequential(SQ)_classmean
o_t_tone_o_F
m8_goal_2
o_c_involvement_Low(L)_classmean
o_t_task_o_I
Variable Importance
0
25
50
75
100
Predictors


gain_c_wjap_str Random Forest Variable Importance
o_t_whom_o_WG
o_c_schedule_Transition(T)_classsd
o_t_verbal_o_T
o_t_verbal_o_N
o_t_attention_o_R
o_c_schedule_MealTime(MT)_classsd
o_c_verbal_No(N)_classmean
o_c_interaction_Social(SOC)_classmean
o_c_involvement_MediumLow(ML)
o_c_schedule_MealTime(MT)_classmean
Variable Importance
0
25
50
75
100
Predictors


gain_c_wjap_str Random Forest Variable Importance
o_c_schedule_Centers(C)_classmean
o_c_interaction_Parallel(P)_classmean
o_c_verbal_No(N)
o_c_interaction_Unoccupied(UN)_classsd
o_c_schedule_WholeGroup(WG)_classmean
o_c_focus_Drama(D)_classsd
o_c_towhom_Notalk/listen(NT)_classmean
o_t_es_o_N
o_c_typetask_Sequential(SQ)
o_c_focus_MusicandMovement(MM)_classsd
Variable Importance
0
25
50
75
100
Predictors