JunjieZeng_STATS485_Unit3_Paper_V2

Junjie Zeng

2025-04-28

Contents

1 Overview **1**

2 Part 0: Data Loading **2**

3 Part 1: Data preprocessing **3**

4 Part 2: Identify Potential important variable **4**

5 Part 3: First Stage Best Subset Selection **14**

6 Part 4: Second Stage Best Subset Selection **19**

7 Part 5: Normalization **21**

8 Part 6: Splines **22**

9 Part 7: Best Model for Grade 5 Reading **26**

10 Part 8: Extremely Low Math Scores **28**

1 Overview

This file produces outcomes in paper Prognostic Modeling with Texas Education Data. This paper is based on the URPS project in Winter 2025 semester supervised by Prof. Ben Hansen and PhD student Julian Bernado. We used TEA_2019.csv data to model Texas students' math and reading test scores in grade 3-8 in 2019. Caroline Moy collaborated with me on this project. She was in charge of the modeling for grade 6-8 reading scores and grade 3-5 math scores. Since our paper depends on long-running computations, we will show the modeling process only for grade 5 reading scores as the representatives using 30 percent schools in this file. Other models can be built using basically the same way. This file is divided into several parts:

1. Data preprocessing
2. Identify Potential important variable
3. First Stage Best Subset Selection
4. Second Stage Best Subset Selection

5. Normalization
6. Splines
7. Best Model for Grade 5 Reading
8. Extremely Low Math Scores

2 Part 0: Data Loading

Reproduciability

```
set.seed(489)
```

We load the data and necessary packages.

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: ’dplyr’

## The following objects are masked from ’package:stats’:
##
##     filter, lag

## The following objects are masked from ’package:base’:
##
##     intersect, setdiff, setequal, union
```

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
library(purrr)
library(ggplot2)
library(stringr)
library(splines)
data <- read_csv("/home/rstudio/TEA_2019.csv")
```

```
## Rows: 2506956 Columns: 91

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): replacement_id
## dbl (64): acadyear, districtid_nces_enroll_m1, districtid_nces_enroll_p0, sc...
## lgl (26): frl_3yr, frl_5yr, frl_high, lep_3yr, lep_5yr, lep_high, rfep_now, ...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

3 Part 1: Data preprocessing

In this part, we use two functions `create_dataset` and `clean_dataset` to help us create datasets for each grade and subject. There will be 12 different datasets named as df_grade_subject, e.g. df_5_r is the dataset we are using in this file. We clean our data follow the following steps: 1. We randomly sample 30 percent of schools to run faster. 2. Remove all variables with only NA values. Those variables contains no information. 3. We also take out alternative test scores, they are not being modeled for now. If we are modeling primary school students, we also take out all variables ends with `_midd`. `replacement_id` and `acadyear` are also irrelevant. 4. If we are modeling reading scores, we take out all math related variables, vice versa. 5. We add `age_int` variable, which is the rounded version of `age`. We will treat age as categorical variable rather than continuous variable. Similarly for `attend_p0` and `attend_m1`, percentage of attendance present year/last year.

```r
#' Function to sample and clean our data
#'
#' @param data The TEA data
#' @param sample Proportion of schools to sample, default 0.3
#' @return A cleaned data with school sampled
create_dataset <- function(data, sample = 0.3){
  set.seed(489)
  unique_schools <- unique(data$schoolid_nces_enroll_p0)
  schools_sampled <- sample(unique_schools,
                            size = round(sample * length(unique_schools)))

  df <- data %>%
    filter(schoolid_nces_enroll_p0 %in% schools_sampled) %>%
    # Remove variables with only NA's
    select(where(~ !all(is.na(.)))) %>%
    mutate(age_int = round(age),
           attend_p0_d1 = round(attend_p0, 1),
           attend_m1_d1 = round(attend_m1, 1))

  return(df)
}

#' Function to create datasets for each grade and subject
#'
#' @param data The cleaned data
#' @return A list of datasets for each grade and subject of the form df_grade_subject, e.g. df_5_r
clean_dataset <- function(data){
  dfs <- list()
  for(grade in 3:8){
    for(subject in c("m", "r")){
      cols_to_drop_all <- c('glmath_alt_scr_m1', 'glmath_alt_scr_p0',
                            'readng_alt_scr_m1', 'readng_alt_scr_p0',
                            'replacement_id', 'acadyear')
      cols_to_drop_subject <- if (subject == "r") {
        c('glmath_ver_p0', 'glmath_lan_p0', 'glmath_scr_p0')
      } else {
        c('readng_ver_p0', 'readng_lan_p0', 'readng_scr_p0')
      }
      dfs[[paste("df", grade, subject, sep="_")]] <- data %>%
        select(where(~ !all(is.na(.)))) %>%
        select(-any_of(cols_to_drop_all)) %>%
        select(-any_of(cols_to_drop_subject)) %>%
        select(
          if (grade < 6) {
            any_of(names(data)[!stringr::str_ends(names(data), "_midd")])
          } else {
            everything()
          }
        )%>%
        filter(gradelevel == grade)
    }
  }
  return(dfs)
}
```

Now we use the functions to create dataframes.

```
df <- create_dataset(data)
dfs <- clean_dataset(df)
```

4 Part 2: Identify Potential important variable

We select categorical variables with number of categories less than 15 and create summary values for them to see whether there's a relatively big difference between categories. Here we only show the list for grade 5 reading.

```
#' Function to get numerical summary of one categorical variable from data
#'
#' @param var Categorical variable
#' @param data The TEA data
#' @return A dataframe containing numerical summary of one categorical variable from data
get_var_summary <- function(var, data1){
  data1 %>%
    group_by(!!sym(var)) %>%
    summarize(mean(readng_scr_p0, na.rm = T),
              median(readng_scr_p0, na.rm = T),
              count = n(),
              proportion = n()/nrow(data1))
}

#' Function to get numerical summary of categorical variables from data
#'
#' @param df data frame
#' @return numerical summaries of categorical variables less than 15 categories from data
get_var_summary_list <- function(df){
  vars <- names(df)[sapply(df, function(x) length(unique(x)) < 15)]
  summary_list <- map(vars, ~get_var_summary(var = .x, data1 = df))
  return(summary_list)
}
```

```
get_var_summary_list(dfs[["df_5_r"]])
```

```
## [[1]]
## # A tibble: 6 x 5
##   enrfay_school mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1         0.167                   1595.                   1599   1320     0.0108
## 2         0.333                   1575.                   1582   1473     0.0121
## 3         0.5                     1594.                   1582   2018     0.0166
## 4         0.667                   1575.                   1582   1416     0.0116
## 5         0.833                   1563.                   1564   1244     0.0102
## 6         1                       1635.                   1619 114457     0.939
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[2]]
## # A tibble: 6 x 5
##   enrfay_district ‘mean(readng_scr_p0, na.rm = T)‘ median(readng_scr_p0~1  count
##             <dbl>                            <dbl>                  <dbl>  <int>
## 1           0.167                            1598.                   1599   1217
## 2           0.333                            1582.                   1582   1141
## 3           0.5                              1598.                   1599   1634
## 4           0.667                            1585.                   1582   1110
## 5           0.833                            1551.                   1564   1092
## 6           1                                1635.                   1619 115734
## # i abbreviated name: 1: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[3]]
## # A tibble: 6 x 5
##   enrfay_state mean(readng_scr_p0, na~1 median(readng_scr_p0~2  count proportion
##          <dbl>                    <dbl>                  <dbl>  <int>      <dbl>
## 1        0.167                    1736                    1736    275    0.00226
## 2        0.333                    1521.                   1582    295    0.00242
## 3        0.5                      1583.                   1582    507    0.00416
## 4        0.667                    1609.                   1582    362    0.00297
## 5        0.833                    1485.                   1561    412    0.00338
## 6        1                        1633.                   1619 120077    0.985
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[4]]
## # A tibble: 1 x 5
##   gradelevel mean(readng_scr_p0, na.r~1 median(readng_scr_p0~2  count proportion
##        <dbl>                      <dbl>                  <dbl>  <int>      <dbl>
## 1          5                      1633.                   1619 121928          1
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[5]]
## # A tibble: 2 x 5
##   gender mean(readng_scr_p0, na.rm = T~1 median(readng_scr_p0~2 count proportion
##    <dbl>                           <dbl>                  <dbl> <int>      <dbl>
## 1      1                           1640.                   1619 59529      0.488
## 2      2                           1626.                   1619 62399      0.512
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[6]]
## # A tibble: 7 x 5
##   raceth mean(readng_scr_p0, na.rm = T~1 median(readng_scr_p0~2 count proportion
##    <dbl>                           <dbl>                  <dbl> <int>      <dbl>
## 1      1                           1632.                   1619   436    0.00358
## 2      2                           1711.                   1698  5731    0.0470
## 3      3                           1608.                   1599 15928    0.131
## 4      4                           1614.                   1599 63308    0.519
## 5      5                           1637.                   1619   162    0.00133
## 6      6                           1657.                   1642 33296    0.273
## 7     NA                           1655.                   1642  3067    0.0252
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[7]]
## # A tibble: 2 x 5
##   frl_now mean(readng_scr_p0, na.rm = ~1 median(readng_scr_p0~2 count proportion
##     <dbl>                          <dbl>                  <dbl> <int>      <dbl>
## 1       0                          1658.                   1642 59305      0.486
## 2       1                          1603.                   1599 62623      0.514
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[8]]
## # A tibble: 3 x 5
##   frl_2yr mean(readng_scr_p0, na.rm = ~1 median(readng_scr_p0~2 count proportion
##     <dbl>                          <dbl>                  <dbl> <int>      <dbl>
## 1       0                          1665.                   1642 48247     0.396
## 2       1                          1606.                   1599 71925     0.590
## 3      NA                          1646.                   1642  1756     0.0144
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[9]]
## # A tibble: 2 x 5
##   frl_ever mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2 count proportion
##      <dbl>                         <dbl>                  <dbl> <int>      <dbl>
## 1        0                         1664.                   1642 50003      0.410
## 2        1                         1606.                   1599 71925      0.590
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[10]]
## # A tibble: 2 x 5
##   frl_elem mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2 count proportion
##      <dbl>                         <dbl>                  <dbl> <int>      <dbl>
## 1        0                         1664.                   1642 50003      0.410
## 2        1                         1606.                   1599 71925      0.590
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[11]]
## # A tibble: 2 x 5
##   lep_now mean(readng_scr_p0, na.rm = ~1 median(readng_scr_p0~2 count proportion
##     <dbl>                          <dbl>                  <dbl> <int>      <dbl>
## 1       0                          1643.                   1642 96325      0.790
## 2       1                          1587.                   1582 25603      0.210
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[12]]
## # A tibble: 3 x 5
##   lep_2yr mean(readng_scr_p0, na.rm = ~1 median(readng_scr_p0~2 count proportion
##     <dbl>                          <dbl>                  <dbl> <int>      <dbl>
## 1       0                          1643.                   1642 91752     0.753
## 2       1                          1593.                   1582 27849     0.228
## 3      NA                          1645.                   1642  2327     0.0191
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[13]]
## # A tibble: 2 x 5
##   lep_ever mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2 count proportion
##      <dbl>                         <dbl>                  <dbl> <int>      <dbl>
## 1        0                         1643.                   1642 94079      0.772
## 2        1                         1593.                   1582 27849      0.228
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[14]]
## # A tibble: 2 x 5
##   lep_elem mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2 count proportion
##      <dbl>                         <dbl>                  <dbl> <int>      <dbl>
## 1        0                         1643.                   1642 94079      0.772
## 2        1                         1593.                   1582 27849      0.228
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[15]]
## # A tibble: 2 x 5
##   migrant_now mean(readng_scr_p0, na.~1 median(readng_scr_p0~2  count proportion
##         <dbl>                     <dbl>                  <dbl>  <int>      <dbl>
## 1           0                     1633.                   1619 121509    0.997
## 2           1                     1589.                   1582    419    0.00344
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[16]]
## # A tibble: 3 x 5
##   migrant_2yr mean(readng_scr_p0, na.~1 median(readng_scr_p0~2  count proportion
##         <dbl>                     <dbl>                  <dbl>  <int>      <dbl>
## 1           0                     1633.                   1619 118383    0.971
## 2           1                     1589.                   1582    514    0.00422
## 3          NA                     1631.                   1619   3031    0.0249
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
```

```
##
## [[17]]
## # A tibble: 2 x 5
##   migrant_ever mean(readng_scr_p0, na~1 median(readng_scr_p0~2  count proportion
##          <dbl>                    <dbl>                  <dbl>  <int>      <dbl>
## 1            0                    1633.                   1619 121414    0.996
## 2            1                    1589.                   1582    514    0.00422
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[18]]
## # A tibble: 2 x 5
##   migrant_elem mean(readng_scr_p0, na~1 median(readng_scr_p0~2  count proportion
##          <dbl>                    <dbl>                  <dbl>  <int>      <dbl>
## 1            0                    1633.                   1619 121414    0.996
## 2            1                    1589.                   1582    514    0.00422
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[19]]
## # A tibble: 2 x 5
##   homeless_now mean(readng_scr_p0, na~1 median(readng_scr_p0~2  count proportion
##          <dbl>                    <dbl>                  <dbl>  <int>      <dbl>
## 1            0                    1634.                   1619 120039     0.985
## 2            1                    1580.                   1582   1889     0.0155
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[20]]
## # A tibble: 3 x 5
##   homeless_2yr mean(readng_scr_p0, na~1 median(readng_scr_p0~2  count proportion
##          <dbl>                    <dbl>                  <dbl>  <int>      <dbl>
## 1            0                    1635.                   1619 113043     0.927
## 2            1                    1602.                   1599   5977     0.0490
## 3           NA                    1633.                   1619   2908     0.0239
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[21]]
## # A tibble: 2 x 5
##   homeless_ever mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             0                   1635.                   1619 115951     0.951
## 2             1                   1602.                   1599   5977     0.0490
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[22]]
## # A tibble: 2 x 5
##   homeless_elem mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             0                   1635.                   1619 115951     0.951
## 2             1                   1602.                   1599   5977     0.0490
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[23]]
## # A tibble: 2 x 5
##   specialed_now mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             0                   1640.                   1619 106919      0.877
## 2             1                   1530.                   1532  15009      0.123
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[24]]
## # A tibble: 3 x 5
##   specialed_2yr mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             0                   1641.                   1619 103425     0.848
## 2             1                   1539.                   1532  15733     0.129
## 3            NA                   1637.                   1619   2770     0.0227
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[25]]
## # A tibble: 2 x 5
##   specialed_ever mean(readng_scr_p0, ~1 median(readng_scr_p0~2  count proportion
##            <dbl>                  <dbl>                  <dbl>  <int>      <dbl>
## 1              0                  1640.                   1619 106195      0.871
## 2              1                  1539.                   1532  15733      0.129
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[26]]
## # A tibble: 2 x 5
##   specialed_elem mean(readng_scr_p0, ~1 median(readng_scr_p0~2  count proportion
##            <dbl>                  <dbl>                  <dbl>  <int>      <dbl>
## 1              0                  1640.                   1619 106195      0.871
## 2              1                  1539.                   1532  15733      0.129
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[27]]
## # A tibble: 2 x 5
##   withdrawal_date_p0 mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2  count
##                <dbl>                         <dbl>                  <dbl>  <int>
## 1           20181026                          NaN                      NA      1
## 2                 NA                         1633.                   1619 121927
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[28]]
## # A tibble: 2 x 5
##   dropout_inferred_m1 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   0                        1633.                   1619 118887
## 2                  NA                        1631.                   1619   3041
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[29]]
## # A tibble: 2 x 5
##   dropout_inferred_p0 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   0                        1633.                   1619 119407
## 2                   1                        1647.                   1642   2521
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[30]]
## # A tibble: 2 x 5
##   persist_inferred_m1 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   1                        1633.                   1619 118887
## 2                  NA                        1631.                   1619   3041
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[31]]
## # A tibble: 2 x 5
##   persist_inferred_p0 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   0                        1647.                   1642   2521
## 2                   1                        1633.                   1619 119407
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[32]]
## # A tibble: 4 x 5
##   transferred_out_m1 mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2  count
##                <dbl>                         <dbl>                  <dbl>  <int>
## 1                  0                         1636.                   1619 105342
## 2                  1                         1611.                   1599   6149
## 3                  2                         1616.                   1599   7396
## 4                 NA                         1631.                   1619   3041
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[33]]
## # A tibble: 3 x 5
##   transferred_out_p0 mean(readng_scr_p0, na.rm =~1 median(readng_scr_p0~2  count
##                <dbl>                         <dbl>                  <dbl>  <int>
## 1                  0                         1635.                   1619 108916
## 2                  1                         1623.                   1619   2610
## 3                  2                         1614.                   1599  10402
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[34]]
## # A tibble: 3 x 5
##   chronic_absentee_m1 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   0                        1635.                   1619 112582
## 2                   1                        1592.                   1582   6305
## 3                  NA                        1631.                   1619   3041
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[35]]
## # A tibble: 2 x 5
##   chronic_absentee_p0 mean(readng_scr_p0, na.rm ~1 median(readng_scr_p0~2  count
##                 <dbl>                        <dbl>                  <dbl>  <int>
## 1                   0                        1635.                   1619 114955
## 2                   1                        1585.                   1582   6973
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
## # i 1 more variable: proportion <dbl>
##
## [[36]]
## # A tibble: 3 x 5
##   glmath_ver_m1 mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             1                   1634.                   1619 114176     0.936 
## 2             2                   1319.                   1311   1743     0.0143
## 3            NA                   1615.                   1619   6009     0.0493
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[37]]
## # A tibble: 4 x 5
##   glmath_lan_m1 mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             5                   1635.                   1619 111901     0.918 
## 2             6                   1594.                   1597   2275     0.0187
## 3             9                   1319.                   1311   1743     0.0143
## 4            NA                   1615.                   1619   6009     0.0493
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[38]]
## # A tibble: 3 x 5
##   readng_ver_m1 mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             1                   1634.                   1619 114188     0.937 
## 2             2                   1319.                   1311   1747     0.0143
## 3            NA                   1615.                   1619   5993     0.0492
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[39]]
## # A tibble: 3 x 5
##   readng_ver_p0 mean(readng_scr_p0, na~1 median(readng_scr_p0~2 count proportion
##           <dbl>                    <dbl>                  <dbl> <int>      <dbl>
## 1             1                    1633.                   1619 87018     0.714
## 2             2                     NaN                      NA  1736     0.0142
## 3            NA                     NaN                      NA 33174     0.272
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[40]]
## # A tibble: 4 x 5
##   readng_lan_m1 mean(readng_scr_p0, n~1 median(readng_scr_p0~2  count proportion
##           <dbl>                   <dbl>                  <dbl>  <int>      <dbl>
## 1             5                   1636.                   1619 108287     0.888
## 2             6                   1603.                   1597   5901     0.0484
## 3             9                   1319.                   1311   1747     0.0143
## 4            NA                   1615.                   1619   5993     0.0492
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[41]]
## # A tibble: 4 x 5
##   readng_lan_p0 mean(readng_scr_p0, na~1 median(readng_scr_p0~2 count proportion
##           <dbl>                    <dbl>                  <dbl> <int>      <dbl>
## 1             5                    1633.                   1619 83932     0.688
## 2             6                    1628.                   1616  3086     0.0253
## 3             9                     NaN                      NA  1736     0.0142
## 4            NA                     NaN                      NA 33174     0.272
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[42]]
## # A tibble: 7 x 5
##   age_int mean(readng_scr_p0, na.rm = ~1 median(readng_scr_p0~2 count proportion
##     <dbl>                          <dbl>                  <dbl> <int>      <dbl>
## 1       2                          1547                    1547     1 0.00000820
## 2      10                          1719.                   1698    61 0.000500
## 3      11                          1632.                   1619 49900 0.409
## 4      12                          1638.                   1619 65979 0.541
## 5      13                          1559.                   1547  5819 0.0477
## 6      14                          1547.                   1564   165 0.00135
## 7      15                          1776                    1776     3 0.0000246
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[43]]
## # A tibble: 11 x 5
##    attend_p0_d1 mean(readng_scr_p0, na~1 median(readng_scr_p0~2 count proportion
##           <dbl>                    <dbl>                  <dbl> <int>      <dbl>
##  1          0                       NaN                     NA      1 0.00000820
##  2          0.1                     NaN                     NA      1 0.00000820
##  3          0.2                     NaN                     NA      5 0.0000410
##  4          0.3                    1438.                  1438.    12 0.0000984
##  5          0.4                    1350.                  1444     21 0.000172
##  6          0.5                    1463.                  1582     44 0.000361
##  7          0.6                    1434.                  1526.    89 0.000730
##  8          0.7                    1507.                  1556.   327 0.00268
##  9          0.8                    1564.                  1582   1835 0.0150
## 10          0.9                    1617.                  1599  27834 0.228
## 11          1                      1638.                  1619  91759 0.753
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
##
## [[44]]
## # A tibble: 12 x 5
##    attend_m1_d1 mean(readng_scr_p0, na~1 median(readng_scr_p0~2 count proportion
##           <dbl>                    <dbl>                  <dbl> <int>      <dbl>
##  1          0.1                    1517                   1517      2  0.0000164
##  2          0.2                    1532                   1532      6  0.0000492
##  3          0.3                    1564.                  1564.     3  0.0000246
##  4          0.4                    1321                   1356     14  0.000115
##  5          0.5                    1578                   1582.    16  0.000131
##  6          0.6                    1505.                  1564     68  0.000558
##  7          0.7                    1552.                  1564    251  0.00206
##  8          0.8                    1573.                  1582   1630  0.0134
##  9          0.9                    1619.                  1599  26911  0.221
## 10          1                      1638.                  1619  89973  0.738
## 11          9                      1556.                  1547     13  0.000107
## 12         NA                      1631.                  1619   3041  0.0249
## # i abbreviated names: 1: ‘mean(readng_scr_p0, na.rm = T)‘,
## #   2: ‘median(readng_scr_p0, na.rm = T)‘
```

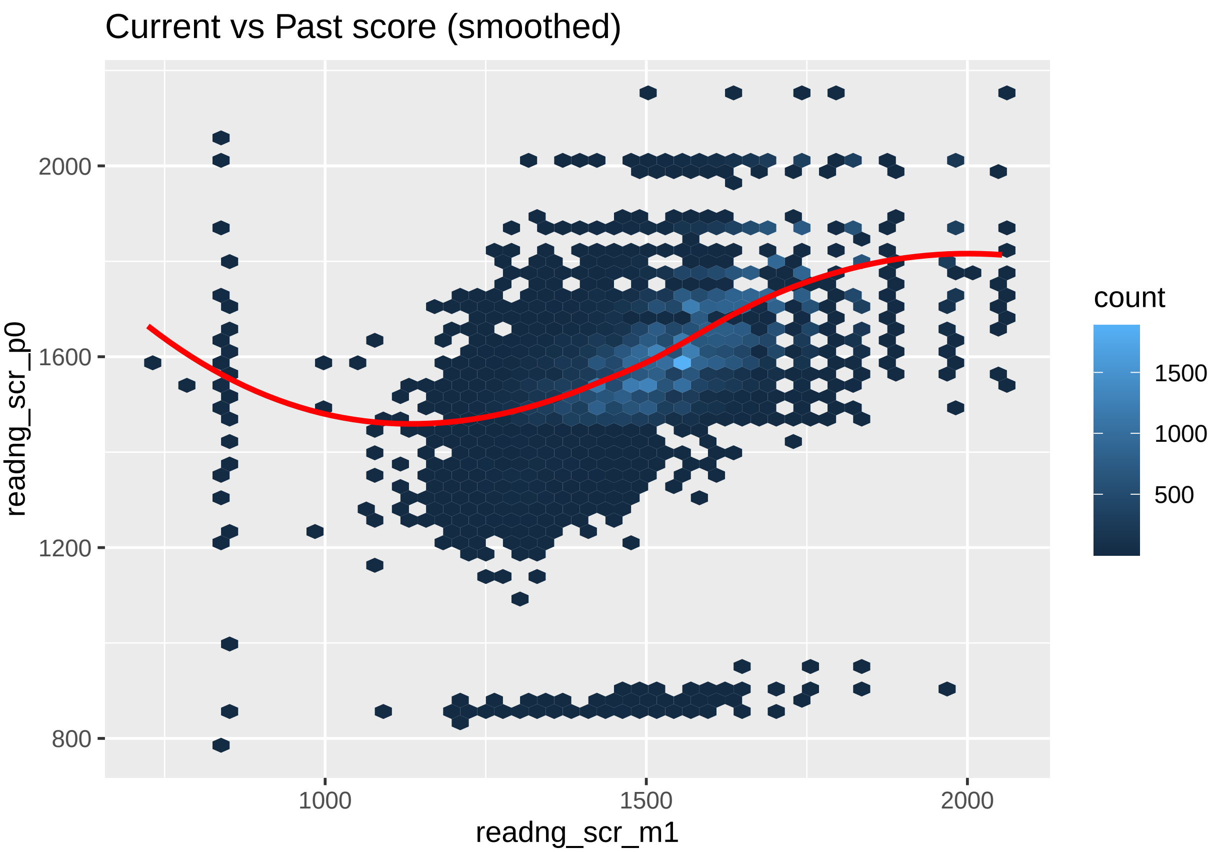
From the summaries, it seem every category variable in the list has some impact on reading scores. What about continuous variable reading score from last year?

```r
ggplot(dfs[["df_5_r"]], aes(x = readng_scr_m1, y = readng_scr_p0)) +
  geom_hex(bins = 50) +
  geom_smooth(method = "loess", se = FALSE, color = "red") +
  labs(title = "Current vs Past score (smoothed)")
```

```
## Warning: Removed 37714 rows containing non-finite values (‘stat_binhex()‘).
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

```
## Warning: Removed 37714 rows containing non-finite values (‘stat_smooth()‘).
```



We see past year reading score and present year reading score are nonlinearly related. This also indicates the usage of polynomial terms/spines.

5 Part 3: First Stage Best Subset Selection

Last time we used pseudo forward selection. This time we apply a different methods. 1. We first decide some fixed variables that we want to include in every model: readng_scr_m1/glmath_scr_m1, race, gender. Preliminary study has shown previous year test score is a strong predictor of present year score. And race and gender are two natural variables people concern.

2. Now we categories the variables into different groups. Variables in a same group measure the same thing, but may be vary in space or time. For example, (enrfay_school, enrfay_state, enrfay_district) is one group, (frl_now, frl_2yrs, frl_ever,. . .) is another group. We select one representative from each group because they overlap greatly and thus we believe the existence of strong multicollinearity within each group. But we do keep the lagged variables. For example we keep both “persist_inferred_m1” and “persist_inferred_p0”. The selection of representatives can be investigated further in future, but for now, here’s the list of representatives:

“enrfay_school” “frl_ever” “lep_ever” “migrant_ever” “homeless_ever” “specialed_ever” “persist_inferred_m1”
“persist_inferred_p0”
“transferred_out_m1”
“transferred_out_p0”
“chronic_absentee_m1”
“chronic_absentee_p0” “readng_lan_m1”

“attend_p0_d1”
“attend_m1_d1”

3. We again divided representatives into two groups: group 1 and group 2. We will build model for all combination of variables in group 1. Group 2 contains all the lagged variables.

Group 1: “enrfay_school” “frl_ever” “lep_ever” “migrant_ever” “homeless_ever” “specialed_ever” “persist_inferred_p0”
“transferred_out_p0”
“chronic_absentee_p0” “attend_p0_d1”

Group 2: “persist_inferred_m1” “transferred_out_m1”
“chronic_absentee_m1” “readng_lan_m1” “attend_m1_d1”

4. We build simple models for all combination of variables in group 1. Here we only show models for grade 5 reading. A general formula would be: reading_score_past_year ~ fixed variables + a combination of group 1 variables + school random effect

```r
#' Function to create a list to keep lists of models
#'
#' @param
#' @return a list of lists named `c_mod_grade_subject` for future use
create_candidate_mod_list <- function(){
  c_mod_list <- list()
  for(grade in 3:8){
    for(subject in c("m", "r")){
      c_mod_list[[paste("c_mod", grade, subject, sep="_")]] <- list()
    }
  }
  return(c_mod_list)
}

#' Function to create formulas for modeling
#'
#' @param subject `r` for reading, `m` for math
#' @param predictors predictors to be written in formula
#' @return a formula to be passed in `lmer` function
create_formula <- function(subject, predictors){
  if(!(subject %in% c("m", "r"))){
    stop("Subject should be m or r")
  }
  if(subject == "m"){
    response <- "glmath_scr_p0"
  } else {
    response <- "readng_scr_p0"
  }
  fixed_part <- paste(predictors, collapse = " + ")
  formula <- as.formula(paste0(response, " ~ ", fixed_part, " + (1 | schoolid_nces_enroll_p0)"))
  return(formula)
}

#' Function to generate candidate models for a given subject and grade
#'
#' @param candidate_vars candidate variables whose combination will be modeled
#' @param fixed_vars fixed variables that will be included in every model
#' @param subject `r` for reading, `m` for math
#' @param grade an integer from 3-8
#' @param list1 a list to save models, default `c_mod_list`
#' @return candidate models passed into the `c_mod_grade_subject` list in the `c_mod_list`
create_candidate_mod <- function(candidate_vars, fixed_vars, subject, grade, list1) {
  all_combinations <- map(0:length(candidate_vars), ~ combn(candidate_vars, m = .x, simplify = FALSE))

  formulas <- map(all_combinations, ~ create_formula(subject = subject, predictors = c(fixed_vars, .x)))

  list1[[paste("c_mod", grade, subject, sep="_")]] <-
    map(formulas, ~ lmer(.x, data = dfs[[paste("df", grade, subject, sep = "_")]], REML = FALSE))

  return(list1)
}
```

```r
candidate_vars_1 <- c("frl_ever",
                      "lep_ever",
                      "attend_p0_d1",
                      "specialed_ever",
                      "transferred_out_p0",
                      "enrfay_school",
                      "chronic_absentee_p0",
                      "homeless_ever",
                      "persist_inferred_p0",
                      "migrant_ever")

fixed_vars_1 <- c("readng_scr_m1",
                  "gender",
                  "raceth")
c_mod_list <- create_candidate_mod_list()
c_mod_list <- create_candidate_mod(candidate_vars_1, fixed_vars_1, "r", 5,c_mod_list)
```

Now we have 2^10, which is 1024 models. We want to choose the best model. The best model is chosen by AIC/BIC/MSE values. However, those three criteria do not always agree on the best model. As a compromise, we take the top 50 models based on MSE first, then take the mean of AIC and BIC rank to select the best model. The motivation is that MSE prefers complexity, we don't want our model gets too complexed. Choosing top 50 based on MSE first to make sure the MSEs are not too further away.

```r
#' Function to generate candidate models for a given subject and grade
#'
#' @param candidate_vars candidate variables whose combination will be modeled
#' @param fixed_vars fixed variables that will be included in every model
#' @param subject `r` for reading, `m` for math
#' @param grade an integer from 3-8
#' @return candidate models passed into the `c_mod_grade_subject` list in the `c_mod_list`
model_selection <- function(grade, subject, list1){
  list_name <- paste("c_mod", grade, subject, sep="_")
  aic_values <- map_dbl(list1[[list_name]], AIC)
  bic_values <- map_dbl(list1[[list_name]], BIC)
  mse_values <- map_dbl(list1[[list_name]], function(model) {
    preds <- predict(model)
    truth <- model@frame[[1]]
    mean((preds - truth)^2)
  })

# Combine everything nicely into a tibble
  comparison_table <- tibble(
    model_id = seq_along(list1[[list_name]]),
    AIC = aic_values,
    BIC = bic_values,
    MSE = mse_values
  )

  print(comparison_table)

  comparison_table <- comparison_table %>%
    arrange(MSE) %>%
    mutate(mse_rank = row_number()) %>%
    slice(1:50) %>%
    mutate(
      rank_aic = rank(AIC, ties.method = "first"),
      rank_bic = rank(BIC, ties.method = "first"),
      mean_rank = (rank_aic + rank_bic) / 2
    ) %>%
    arrange(mean_rank)

  return(comparison_table)
}
```

```r
model_selection(5, "r",c_mod_list)
```

```
## # A tibble: 1,024 x 4
##    model_id     AIC     BIC   MSE
##       <int>   <dbl>   <dbl> <dbl>
##  1        1 977682. 977738. 8576.
##  2        2 977104. 977169. 8533.
##  3        3 977571. 977636. 8566.
##  4        4 977473. 977538. 8557.
##  5        5 976715. 976780. 8470.
##  6        6 977668. 977733. 8576.
##  7        7 977605. 977670. 8569.
##  8        8 977583. 977648. 8567.
##  9        9 977659. 977725. 8574.
## 10       10 977682. 977747. 8576.
## # i 1,014 more rows
```

```
## # A tibble: 50 x 8
##    model_id     AIC     BIC   MSE mse_rank rank_aic rank_bic mean_rank
##       <int>   <dbl>   <dbl> <dbl>    <int>    <int>    <int>     <dbl>
##  1      849 975858. 975979. 8401.       11        6        5       5.5
##  2      969 975856. 975987. 8401.        4        2       14       8
##  3      639 975862. 975974. 8402.       27       15        2       8.5
##  4      850 975860. 975981. 8402.       20       10        8       9
##  5      971 975857. 975988. 8401.        7        5       16      10.5
##  6      852 975861. 975982. 8402.       23       13        9      11
##  7      644 975864. 975976. 8402.       19       22        3      12.5
##  8      970 975858. 975989. 8401.        9        8       18      13
##  9      859 975862. 975983. 8401.       12       16       10      13
## 10     1015 975856. 975996. 8401.        2        1       26      13.5
## # i 40 more rows
```

```
summary(c_mod_list[["c_mod_5_r"]][[849]])
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: readng_scr_p0 ~ readng_scr_m1 + gender + raceth + frl_ever +
##     lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 +
##     enrfay_school + chronic_absentee_p0 + (1 | schoolid_nces_enroll_p0)
##    Data: dfs[[paste("df", grade, subject, sep = "_")]]
##
##       AIC       BIC    logLik  deviance  df.resid
##  975857.6  975978.7 -487915.8  975831.6     81983
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -10.6688  -0.6081  -0.0905   0.5051   8.6820
##
## Random effects:
##  Groups                  Name        Variance Std.Dev.
##  schoolid_nces_enroll_p0 (Intercept)  314     17.72
##  Residual                            8484     92.11
## Number of obs: 81996, groups:  schoolid_nces_enroll_p0, 1248
##
## Fixed effects:
##                      Estimate Std. Error t value
## (Intercept)         725.44378    9.68838  74.878
## readng_scr_m1         0.53498    0.00266 201.149
## gender               -8.67440    0.64941 -13.357
## raceth                0.67202    0.30128   2.231
## frl_ever            -16.41300    0.76721 -21.393
## lep_ever             -9.40571    0.91392 -10.292
## attend_p0_d1         75.92552    8.35815   9.084
## specialed_ever      -40.41309    1.30753 -30.908
## transferred_out_p0   -1.74071    0.60984  -2.854
## enrfay_school        21.38837    3.18632   6.713
## chronic_absentee_p0  -4.94192    1.97710  -2.500
##
## Correlation of Fixed Effects:
##             (Intr) rdn__1 gender raceth frl_vr lep_vr at_0_1 spcld_ trn__0
## rdng_scr_m1 -0.399
## gender      -0.110  0.013
## raceth      -0.178 -0.013 -0.006
## frl_ever    -0.184  0.136  0.015  0.156
## lep_ever    -0.035  0.151 -0.013  0.108 -0.127
## attnd_p0_d1 -0.823 -0.033  0.006  0.053  0.060 -0.055
## speciald_vr -0.109  0.207 -0.081 -0.031 -0.008  0.033  0.025
## trnsfrrd__0 -0.066  0.015  0.005  0.019 -0.014  0.008  0.032  0.006
## enrfay_schl -0.256 -0.034  0.002 -0.019  0.035 -0.037 -0.061  0.005  0.057
## chrnc_bsn_0 -0.416  0.018 -0.008 -0.001 -0.028  0.018  0.461 -0.013 -0.013
##             enrfy_
## rdng_scr_m1
## gender
## raceth
## frl_ever
## lep_ever
## attnd_p0_d1
## speciald_vr
## trnsfrrd__0
## enrfay_schl
## chrnc_bsn_0  0.044
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

The comparison table tells us that model 849 is the best model. The formula for it is readng_scr_p0 ~ readng_scr_m1 + gender + raceth + frl_ever + lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 + enrfay_school + chronic_absentee_p0 + (1 | schoolid_nces_enroll_p0)

6 Part 4: Second Stage Best Subset Selection

Now we have the best simple model from fixed variables and group 1. We now treat the variables in the best simple model as the fixed variables, and so a best subset selection over group 2. Note this time we only have 32 models, that means we are only looking at AIC/BIC this time. Fixed variables: readng_scr_m1 + gender + raceth + frl_ever + lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 + enrfay_school + chronic_absentee_p0 Group 2: "persist_inferred_m1" "transferred_out_m1" "chronic_absentee_m1" "readng_lan_m1" "attend_m1_d1"

```
candidate_vars_2 <- c("persist_inferred_m1",
                      "transferred_out_m1",
                      "chronic_absentee_m1",
                      "readng_lan_m1",
                      "attend_m1_d1")

fixed_vars_2 <- c("readng_scr_m1",
                  "gender",
                  "raceth",
                  "frl_ever",
                  "lep_ever",
                  "attend_p0_d1",
                  "specialed_ever",
                  "transferred_out_p0",
                  "enrfay_school",
                  "chronic_absentee_p0")
c_mod_list_2 <- create_candidate_mod_list()
c_mod_list_2 <- create_candidate_mod(candidate_vars_2, fixed_vars_2, "r", 5, c_mod_list_2)
```

```
model_selection(5, "r", c_mod_list_2)
```

```
## # A tibble: 32 x 4
##    model_id     AIC     BIC   MSE
##       <int>   <dbl>   <dbl> <dbl>
##  1        1 975858. 975979. 8401.
##  2        2 975858. 975979. 8401.
##  3        3 975858. 975989. 8401.
##  4        4 975847. 975977. 8400.
##  5        5 975790. 975920. 8396.
##  6        6 975860. 975990. 8401.
##  7        7 975858. 975989. 8401.
##  8        8 975847. 975977. 8400.
##  9        9 975790. 975920. 8396.
## 10       10 975860. 975990. 8401.
## # i 22 more rows
```

```
## # A tibble: 32 x 8
##    model_id     AIC     BIC   MSE mse_rank rank_aic rank_bic mean_rank
##       <int>   <dbl>   <dbl> <dbl>    <int>    <int>    <int>     <dbl>
##  1       14 975778. 975918. 8395.        7        1        1         1
##  2       20 975778. 975918. 8395.        8        2        2         2
##  3       23 975779. 975928. 8394.        3        3        5         4
##  4       27 975779. 975928. 8394.        4        4        6         5
##  5       26 975780. 975929. 8395.        5        5        7         6
##  6        5 975790. 975920. 8396.       15        9        3         6
##  7       30 975780. 975929. 8395.        6        6        8         7
##  8        9 975790. 975920. 8396.       16       10        4         7
##  9       31 975781. 975939. 8394.        1        7       13        10
## 10       12 975791. 975930. 8396.       11       11        9        10
## # i 22 more rows
```

```r
summary(c_mod_list_2[["c_mod_5_r"]][[14]])
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: readng_scr_p0 ~ readng_scr_m1 + gender + raceth + frl_ever +
##     lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 +
##     enrfay_school + chronic_absentee_p0 + chronic_absentee_m1 +
##     readng_lan_m1 + (1 | schoolid_nces_enroll_p0)
##    Data: dfs[[paste("df", grade, subject, sep = "_")]]
##
##       AIC       BIC    logLik  deviance  df.resid
##  975778.5  975918.2 -487874.2  975748.5     81981
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -10.6789  -0.6071  -0.0895   0.5056   8.6885
##
## Random effects:
##  Groups                  Name        Variance Std.Dev.
##  schoolid_nces_enroll_p0 (Intercept)  308.3   17.56
##  Residual                            8476.9   92.07
## Number of obs: 81996, groups:  schoolid_nces_enroll_p0, 1248
##
## Fixed effects:
##                      Estimate Std. Error t value
## (Intercept)         642.60878   13.39682  47.967
## readng_scr_m1         0.53559    0.00266 201.374
## gender               -8.57914    0.64921 -13.215
## raceth                0.61290    0.30108   2.036
## frl_ever            -16.60232    0.76697 -21.647
## lep_ever            -12.52842    0.99025 -12.652
## attend_p0_d1         82.06025    8.52664   9.624
## specialed_ever      -40.26887    1.30726 -30.804
## transferred_out_p0   -1.73782    0.60947  -2.851
## enrfay_school        21.55596    3.18533   6.767
## chronic_absentee_p0  -7.08294    2.06470  -3.430
## chronic_absentee_m1   7.26206    1.98310   3.662
## readng_lan_m1        15.11322    1.80335   8.381
```

```
##
## Correlation matrix not shown by default, as p = 13 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

We see the best model is model 14. And its formula is readng_scr_m1 + gender + raceth + frl_ever + lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 + enrfay_school + chronic_absentee_p0 + chronic_absentee_m1 + readng_lan_m1

7 Part 5: Normalization

In this part, we investigate whether normalization will help with our model. We will use the winner model as the control.

```r
unscaled <- lmer(readng_scr_p0 ~ readng_scr_m1 + gender + raceth + frl_ever + lep_ever
                 + attend_p0_d1 + specialed_ever + transferred_out_p0
                 +  enrfay_school + chronic_absentee_p0
                 + chronic_absentee_m1 +  readng_lan_m1
                 + (1 | schoolid_nces_enroll_p0),
                 data = dfs[["df_5_r"]], REML = FALSE)

scaled <- lmer(scale(readng_scr_p0) ~ scale(readng_scr_m1) + gender + raceth + frl_ever + lep_ever
                 + attend_p0_d1 + specialed_ever + transferred_out_p0
                 +  enrfay_school + chronic_absentee_p0
                 + chronic_absentee_m1 +  readng_lan_m1
                 + (1 | schoolid_nces_enroll_p0),
                 data = dfs[["df_5_r"]], REML = FALSE)

cat("\nThe AIC for unnormalized model is", AIC(unscaled))
```

```
##
## The AIC for unnormalized model is 975778.5
```

```r
cat("\nThe AIC for normalized model is", AIC(scaled))
```

```
##
## The AIC for normalized model is 185181.5
```

```
cat("\nThe BIC for unnormalized model is", BIC(unscaled))
```

```
##
## The BIC for unnormalized model is 975918.2
```

```
cat("\nThe BIC for normalized model is", BIC(scaled))
```

```
##
## The BIC for normalized model is 185321.3
```

We see normalization results in a better model. So we update our best model to the normalized one.

8 Part 6: Splines

Now we explore the use of spline for our model. `mod0` is without splines, `modi` is has spline of degree i. We create 11 models, first one without any splines, the rest natural splines 1-10.

```
splines <- list()
splines[['mod0']] <- lmer(scale(readng_scr_p0) ~ scale(readng_scr_m1)
                          + gender + raceth + frl_ever + lep_ever
                          + attend_p0_d1 + specialed_ever + transferred_out_p0
                          +  enrfay_school + chronic_absentee_p0
                          + chronic_absentee_m1 +  readng_lan_m1
                          + (1 | schoolid_nces_enroll_p0),
                          data = dfs[["df_5_r"]], REML = FALSE)
for(i in 1:10){
  splines[[paste('mod', i)]] <- lmer(scale(readng_scr_p0) ~ ns(scale(readng_scr_m1),i)
                                      + gender + raceth + frl_ever + lep_ever
                                      + attend_p0_d1 + specialed_ever + transferred_out_p0
                                      +  enrfay_school + chronic_absentee_p0
                                      + chronic_absentee_m1 +  readng_lan_m1
                                      + (1 | schoolid_nces_enroll_p0),
                                      data = dfs[["df_5_r"]], REML = FALSE)
}
```

Now like we did before, we get AIC/BIC/MSEs for each mdoel.

```
splines_Sum <- list()

for(i in seq_along(splines)) {
  splines_Sum[[i]] <- list(
    model = splines[[i]],
    AIC = AIC(splines[[i]]),
    BIC = BIC(splines[[i]]),
    MSE = mean(residuals(splines[[i]])^2)
  )
}
names(splines_Sum) <- names(splines)

aic_values <- sapply(splines_Sum, function(x) x$AIC)
aic_values
```

```
##     mod0    mod 1    mod 2    mod 3    mod 4    mod 5    mod 6    mod 7
## 185181.5 185181.5 184833.5 182413.1 182334.9 182190.2 182096.2 182018.7
##    mod 8    mod 9   mod 10
## 181966.2 181954.1 181933.9
```

```
bic_values <- sapply(splines_Sum, function(x) x$BIC)
bic_values
```

```
##     mod0    mod 1    mod 2    mod 3    mod 4    mod 5    mod 6    mod 7
## 185321.3 185321.3 184982.6 182571.4 182502.5 182367.2 182282.5 182214.3
##    mod 8    mod 9   mod 10
## 182171.1 182168.3 182157.4
```

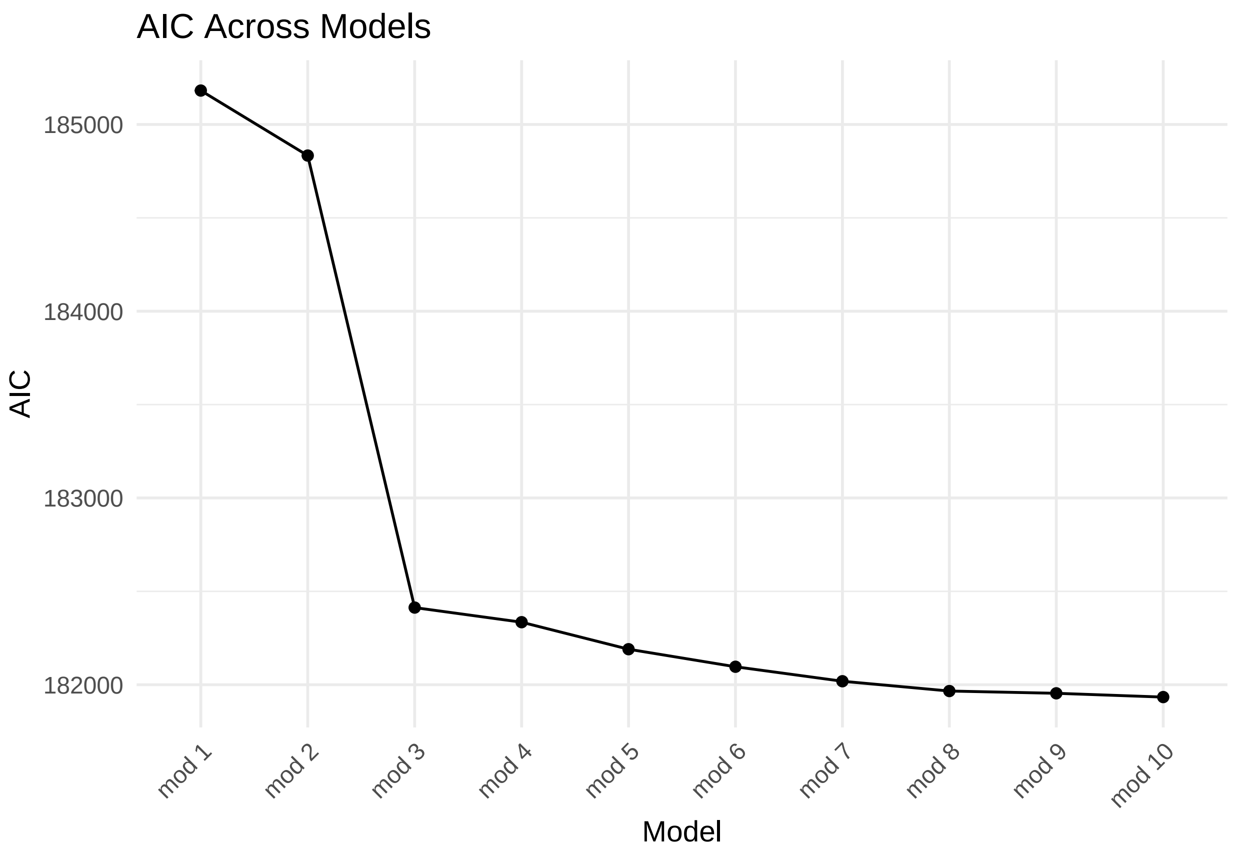
```
mse_values <- sapply(splines_Sum, function(x) x$MSE)
mse_values
```

```
##      mod0     mod 1     mod 2     mod 3     mod 4     mod 5     mod 6     mod 7
## 0.5452290 0.5452290 0.5430690 0.5269380 0.5264600 0.5255274 0.5249242 0.5244335
##     mod 8     mod 9    mod 10
## 0.5240913 0.5240015 0.5238651
```

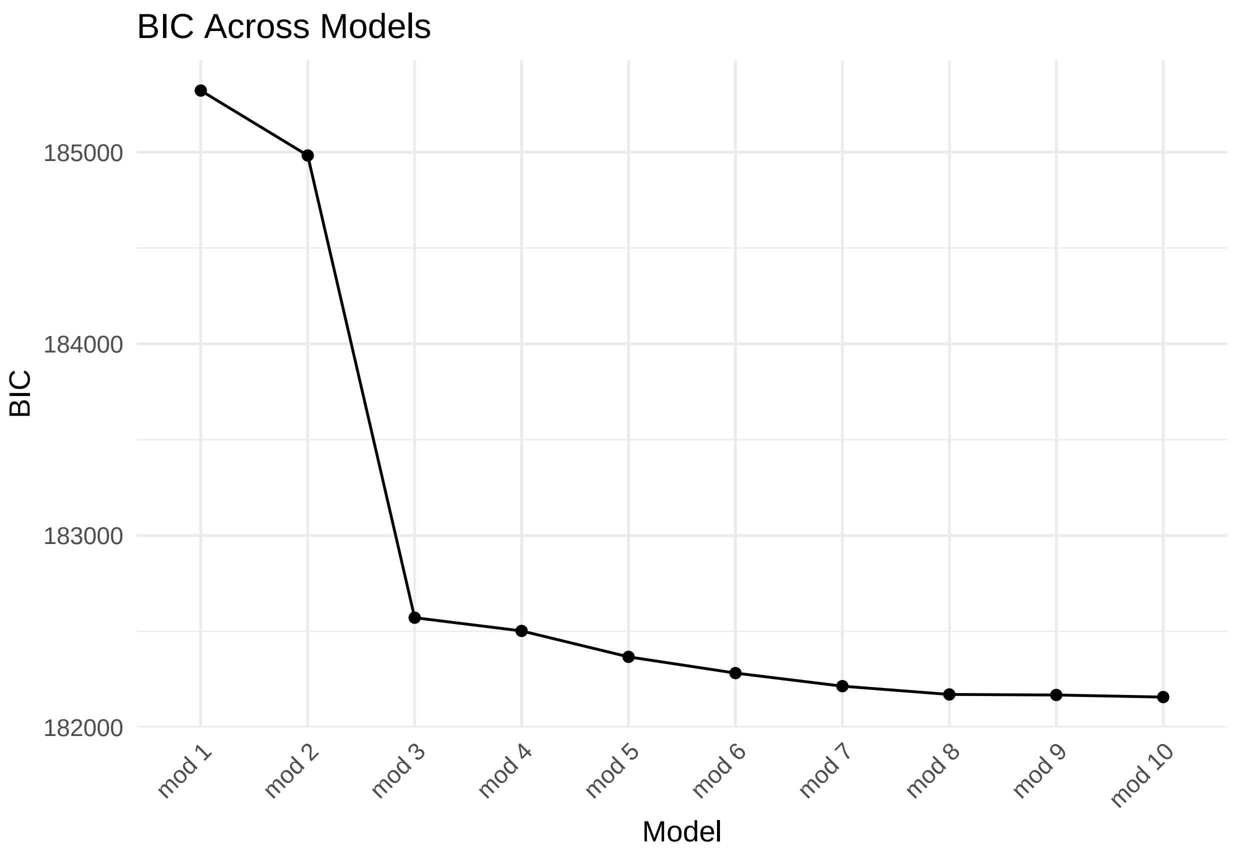
We create line plots to visualize how AIC/BIC/MSE change with the increase of spline degrees.

```
aic_df <- data.frame(
  Model = names(aic_values),
  AIC = aic_values
)
aic_df <- subset(aic_df, Model != "mod0")
aic_df$Order <- as.numeric(str_extract(aic_df$Model, "\\d+"))
aic_df <- aic_df[order(aic_df$Order), ]
aic_df$Model <- factor(aic_df$Model, levels = aic_df$Model)

ggplot(aic_df, aes(x = Model, y = AIC, group = 1)) +
  geom_line() +
  geom_point() +
  labs(title = "AIC Across Models", x = "Model", y = "AIC") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

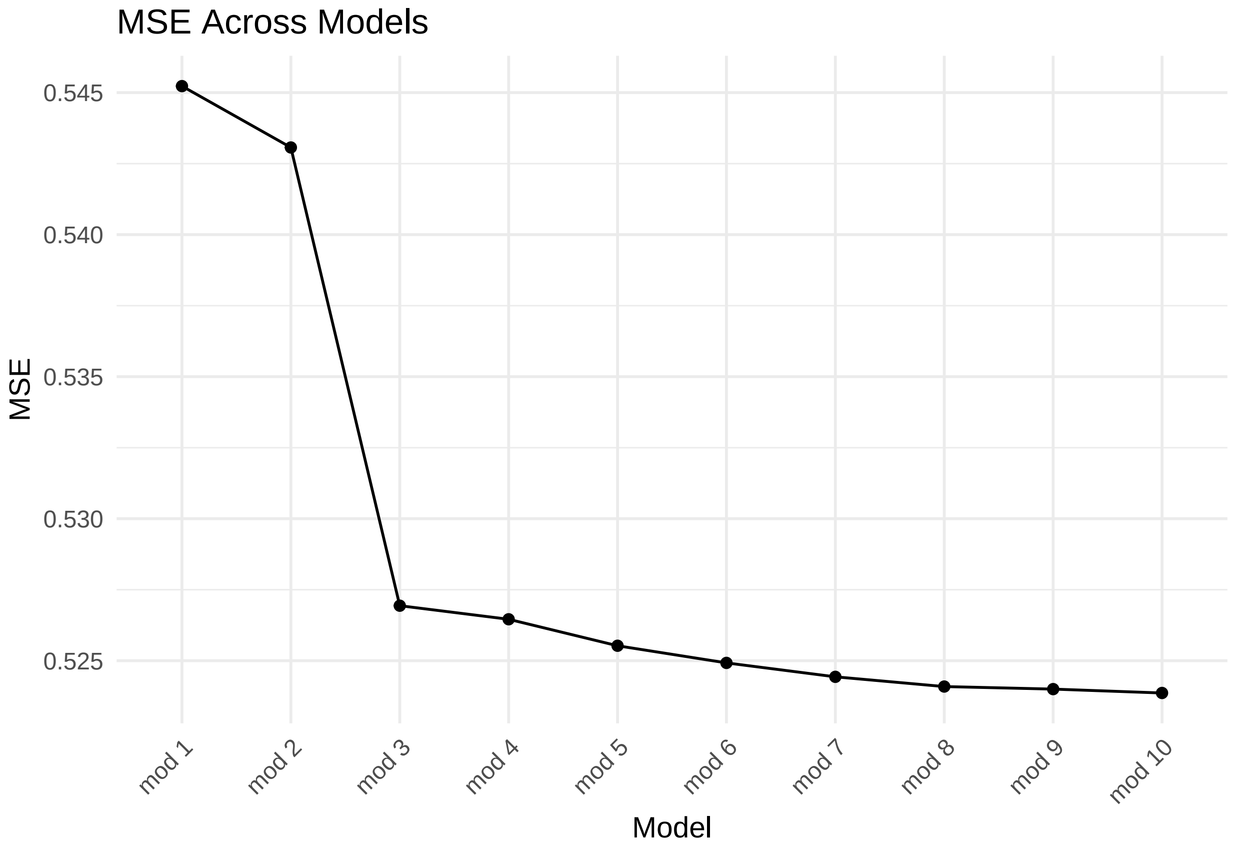


```r
bic_df <- data.frame(
  Model = names(bic_values),
  BIC = bic_values
)
bic_df <- subset(bic_df, Model != "mod0")
bic_df$Order <- as.numeric(str_extract(bic_df$Model, "\\d+"))
bic_df <- bic_df[order(bic_df$Order), ]
bic_df$Model <- factor(bic_df$Model, levels = bic_df$Model)
ggplot(bic_df, aes(x = Model, y = BIC, group = 1)) +
  geom_line() +
  geom_point() +
  labs(title = "BIC Across Models", x = "Model", y = "BIC") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
mse_df <- data.frame(
  Model = names(mse_values),
  MSE = mse_values
)
mse_df <- subset(mse_df, Model != "mod0")
mse_df$Order <- as.numeric(str_extract(mse_df$Model, "\\d+"))
mse_df <- mse_df[order(mse_df$Order), ]
mse_df$Model <- factor(mse_df$Model, levels = mse_df$Model)

ggplot(mse_df, aes(x = Model, y = MSE, group = 1)) +
  geom_line() +
  geom_point() +
  labs(title = "MSE Across Models", x = "Model", y = "MSE") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



9 Part 7: Best Model for Grade 5 Reading

So based on our analysis, the best model we have for grade 5 reading is scale(readng_scr_p0) ~ ns(scale(readng_scr_m1),3) + gender + raceth + frl_ever + lep_ever + attend_p0_d1 + specialed_ever + transferred_out_p0 + enrfay_school + chronic_absentee_p0 + chronic_absentee_m1 + readng_lan_m1 + (1 | schoolid_nces_enroll_p0)

```r
best_model_5_r <- lmer(scale(readng_scr_p0) ~ ns(scale(readng_scr_m1),3)
                                      + gender + raceth + frl_ever + lep_ever
                                      + attend_p0_d1 + specialed_ever + transferred_out_p0
                                      +  enrfay_school + chronic_absentee_p0
                                      + chronic_absentee_m1 +  readng_lan_m1
                                      + (1 | schoolid_nces_enroll_p0),
                                      data = dfs[["df_5_r"]], REML = FALSE)
summary(best_model_5_r)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: scale(readng_scr_p0) ~ ns(scale(readng_scr_m1), 3) + gender +
##     raceth + frl_ever + lep_ever + attend_p0_d1 + specialed_ever +
##     transferred_out_p0 + enrfay_school + chronic_absentee_p0 +
##     chronic_absentee_m1 + readng_lan_m1 + (1 | schoolid_nces_enroll_p0)
##    Data: dfs[["df_5_r"]]
##
##      AIC      BIC   logLik deviance df.resid
## 182413.1 182571.4 -91189.5 182379.1    81979
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -10.4397  -0.6005  -0.0734   0.5140   6.1401
##
## Random effects:
##  Groups                  Name        Variance Std.Dev.
##  schoolid_nces_enroll_p0 (Intercept) 0.01983  0.1408
##  Residual                            0.53214  0.7295
## Number of obs: 81996, groups:  schoolid_nces_enroll_p0, 1248
##
## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)                  -1.593689   0.123237 -12.932
## ns(scale(readng_scr_m1), 3)1  1.879024   0.038963  48.226
## ns(scale(readng_scr_m1), 3)2  0.286015   0.138498   2.065
## ns(scale(readng_scr_m1), 3)3  2.801586   0.040593  69.016
## gender                       -0.068875   0.005144 -13.390
## raceth                        0.002882   0.002388   1.207
## frl_ever                     -0.117291   0.006090 -19.261
## lep_ever                     -0.079599   0.007863 -10.123
## attend_p0_d1                  0.672555   0.067564   9.954
## specialed_ever               -0.316033   0.010455 -30.227
## transferred_out_p0           -0.012686   0.004831  -2.626
## enrfay_school                 0.177180   0.025243   7.019
## chronic_absentee_p0          -0.050766   0.016360  -3.103
## chronic_absentee_m1           0.022531   0.015737   1.432
## readng_lan_m1                 0.116227   0.014320   8.117
```

```
##
## Correlation matrix not shown by default, as p = 15 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```r
cat("\nThe AIC of the best model is ", AIC(best_model_5_r))
```

```
##
## The AIC of the best model is  182413.1
```

```r
cat("\nThe BIC of the best model is ", BIC(best_model_5_r))
```

```
##
## The BIC of the best model is  182571.4
```

```r
cat("\nThe MSE of the best model is ", mean(residuals(best_model_5_r)^2))
```

```
##
## The MSE of the best model is  0.526938
```

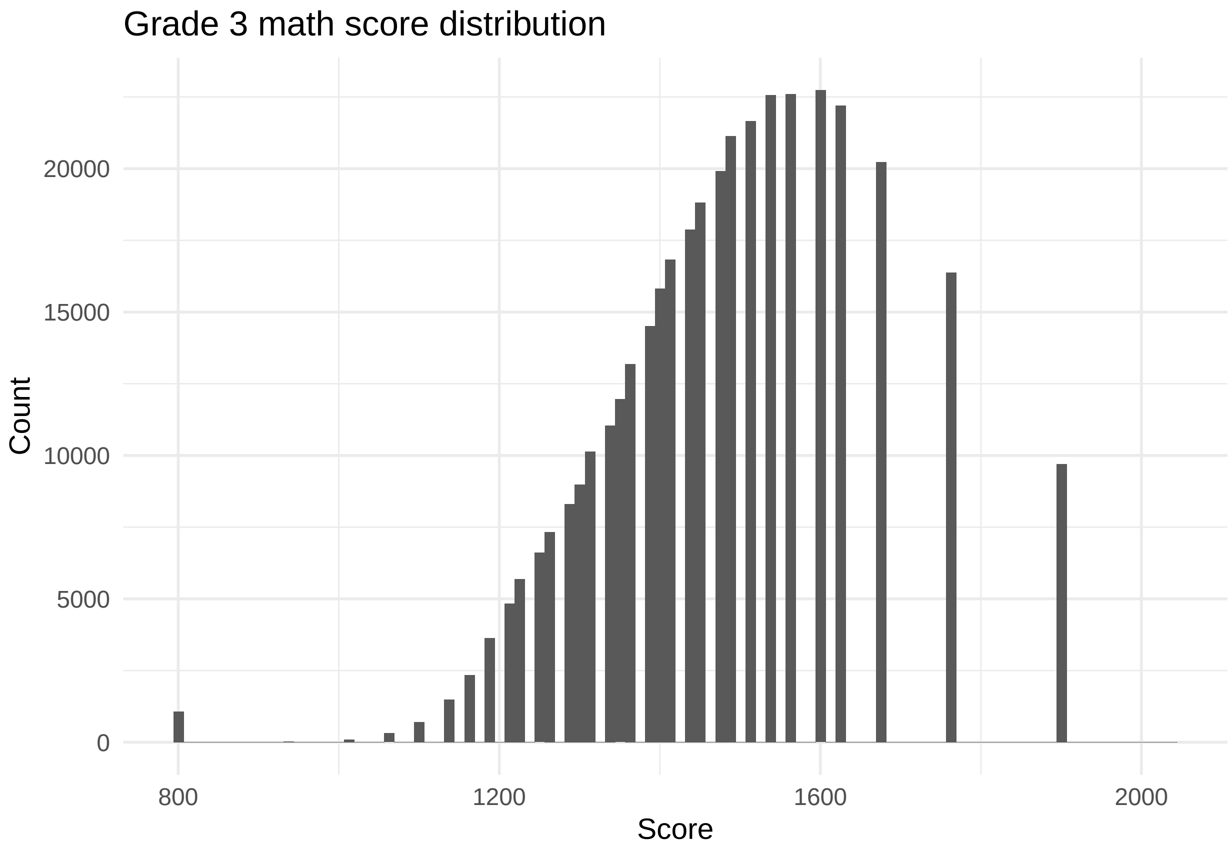
10 Part 8: Extremely Low Math Scores

One worth-noting pattern in our data is extremely low math scores for students in grade 6-8. We have lots of students scored 1043, which we can reasonably guess corresponding to raw score 0 in STAAR math test based on past grading schemes (Grading scheme for 2019 was not found online). Here's a histogram show the pattern. Grade 6 math score distribution:

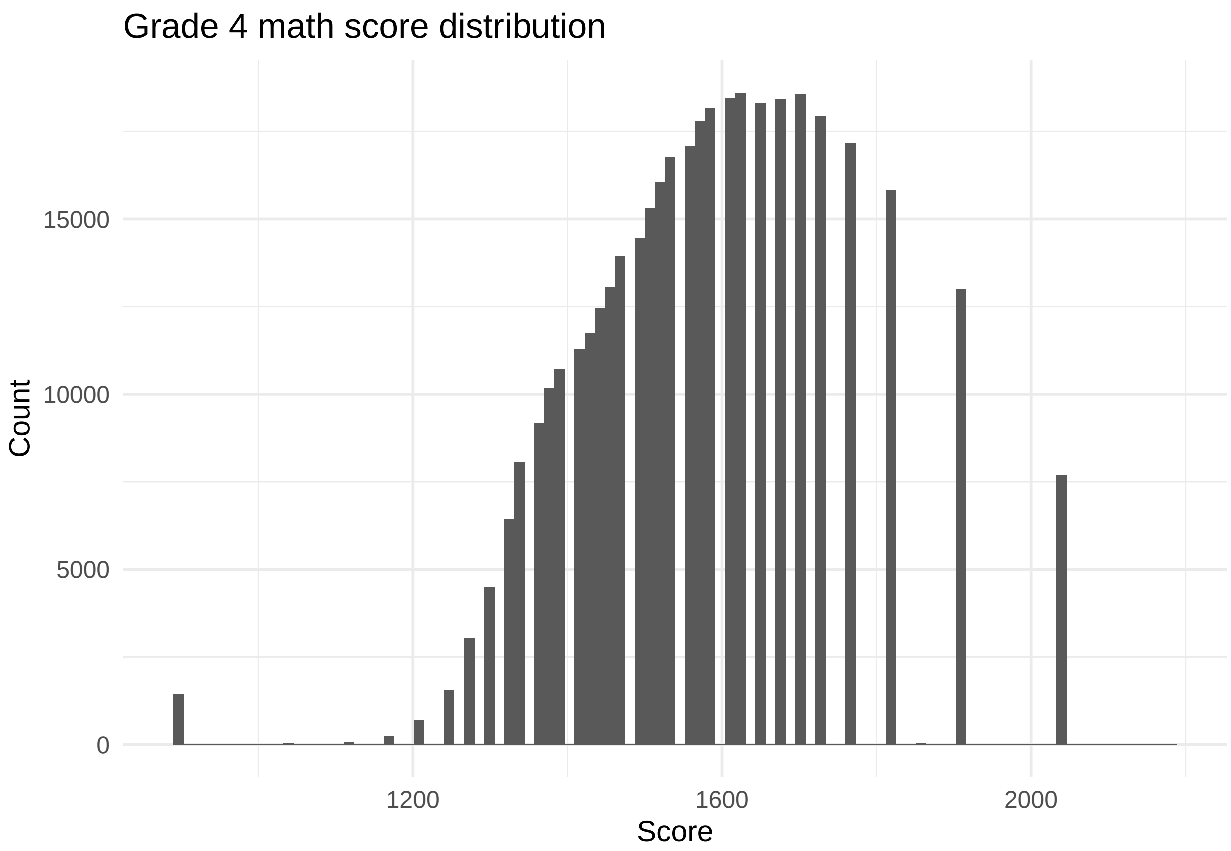
```r
for(i in 3:8){
  p <- data %>%
    filter(gradelevel == i) %>%
    ggplot(aes(x = glmath_scr_p0)) +
    geom_histogram(bins = 100) +
    labs(
      title = paste("Grade", i, "math score distribution"),
      x = "Score",
      y = "Count"
    ) +
    theme_minimal()

  print(p)
}
```

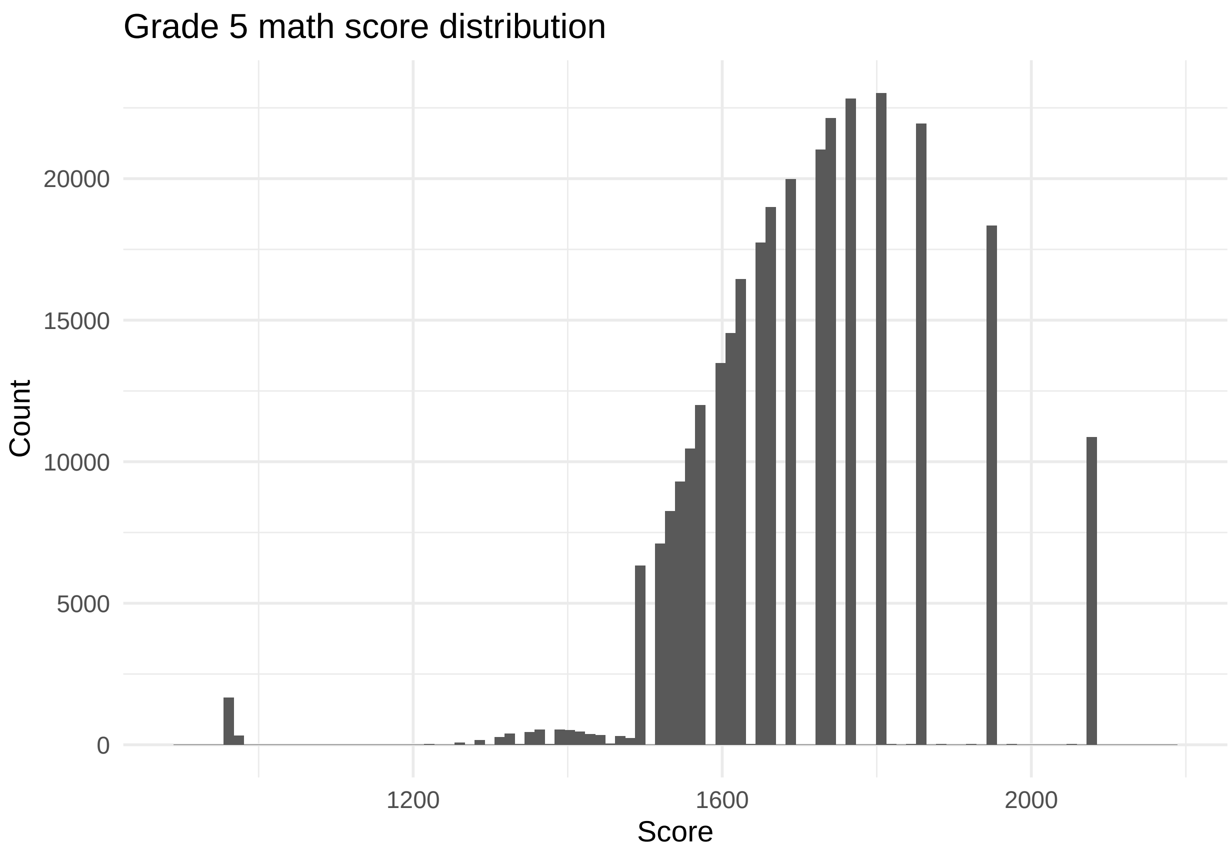
```
## Warning: Removed 24244 rows containing non-finite values (‘stat_bin()‘).
```



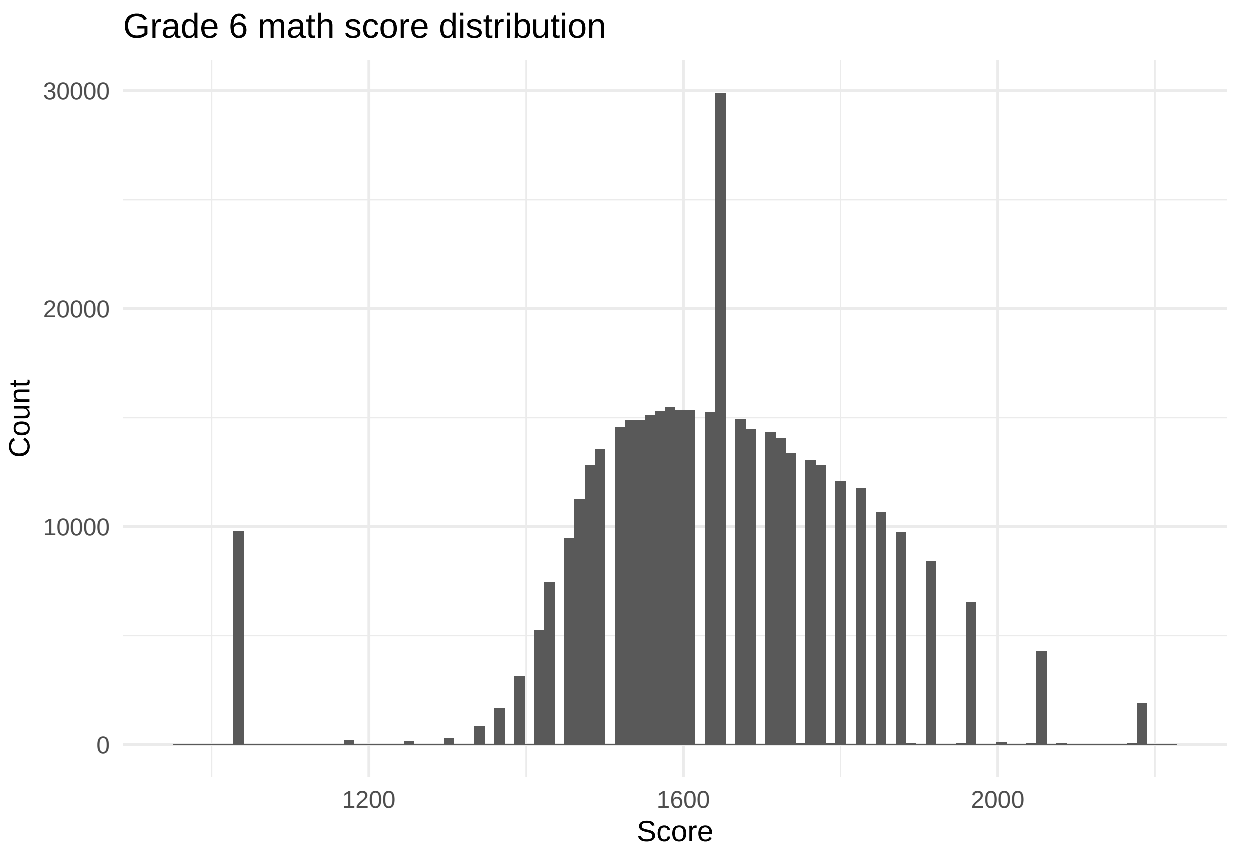
```
## Warning: Removed 22483 rows containing non-finite values (‘stat_bin()‘).
```



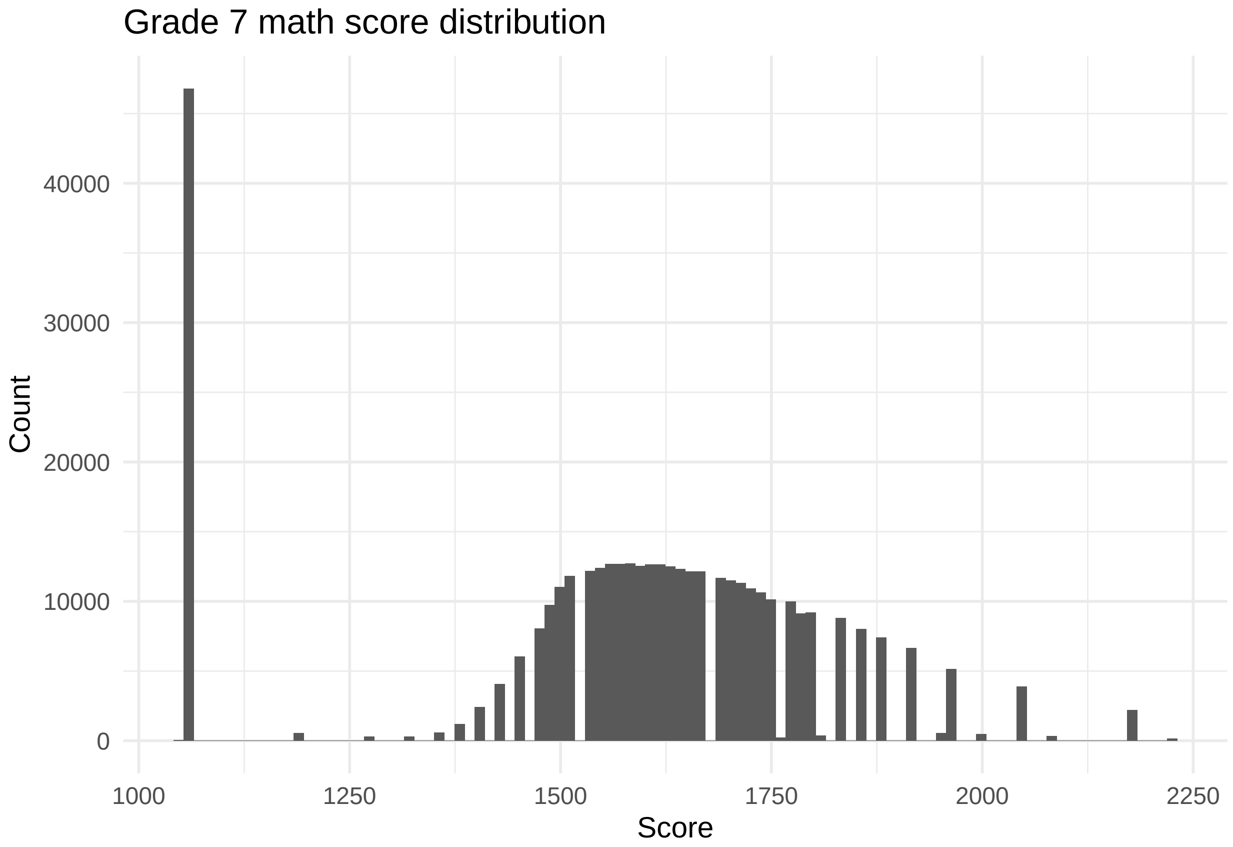
```
## Warning: Removed 123908 rows containing non-finite values (‘stat_bin()‘).
```



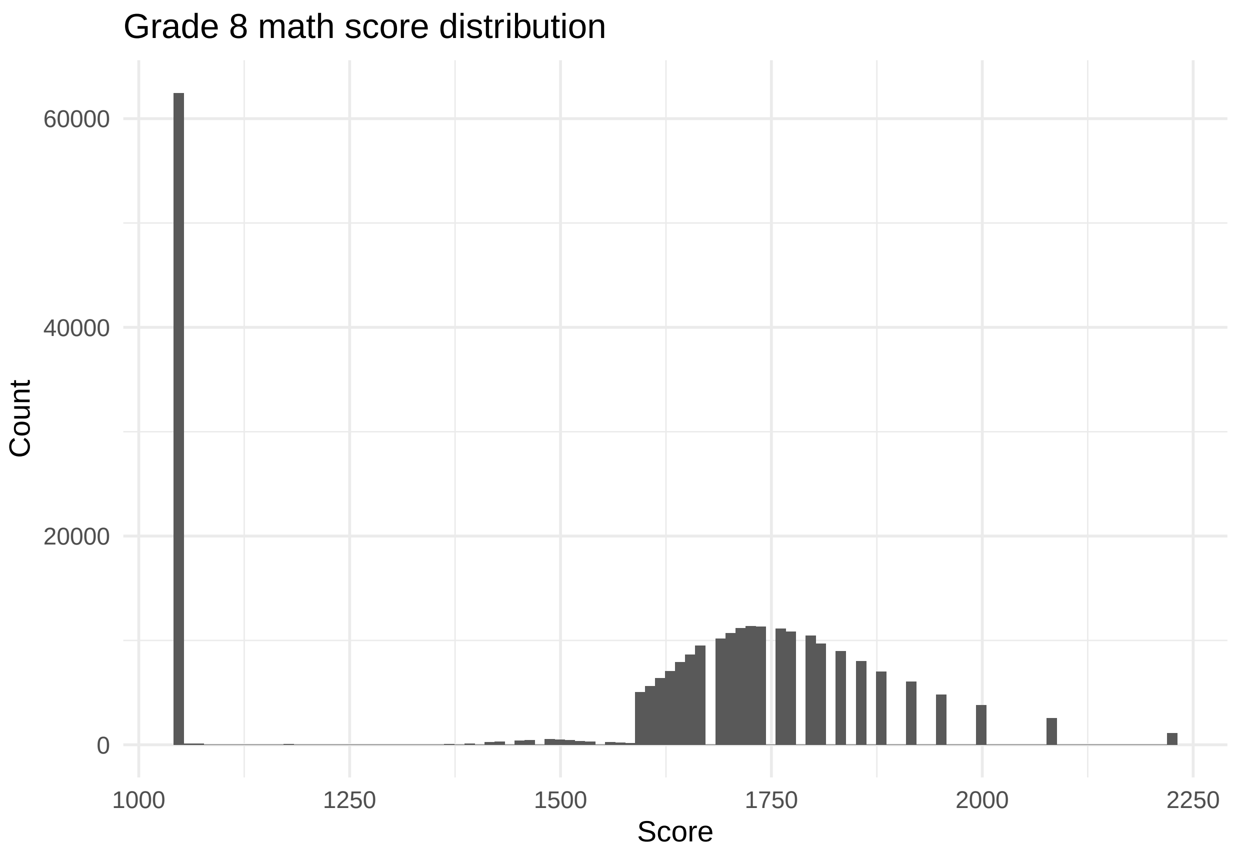
```
## Warning: Removed 20894 rows containing non-finite values (‘stat_bin()‘).
```



```
## Warning: Removed 23639 rows containing non-finite values (‘stat_bin()‘).
```



```
## Warning: Removed 156657 rows containing non-finite values (‘stat_bin()‘).
```


Grade 8 math score distribution
Count
60000
40000
20000
0
1000
1250
1500
1750
2000
2250
Score