data modeling

Dean Gladish

February 18, 2020

```r
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 3.5.3
```

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

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

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## 
## 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
```

```r
library(DataCombine)
```

```
## Warning: package 'DataCombine' was built under R version 3.5.3
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.0.1     v stringr 1.3.1
## v tidyr   0.8.2     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(fGarch)
```

```
## Warning: package 'fGarch' was built under R version 3.5.3
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Warning: package 'timeSeries' was built under R version 3.5.3
## Loading required package: fBasics
```

```
## Warning: package 'fBasics' was built under R version 3.5.3
```

```
library(sn)
```

```
## Warning: package 'sn' was built under R version 3.5.3
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'sn'
```

```
## The following object is masked from 'package:fBasics':
##
##     vech
```

```
## The following object is masked from 'package:stats':
##
##     sd
```

```
library(arm)
```

```
## Warning: package 'arm' was built under R version 3.5.3
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.5.3
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
##
## arm (Version 1.10-1, built: 2018-4-12)
```

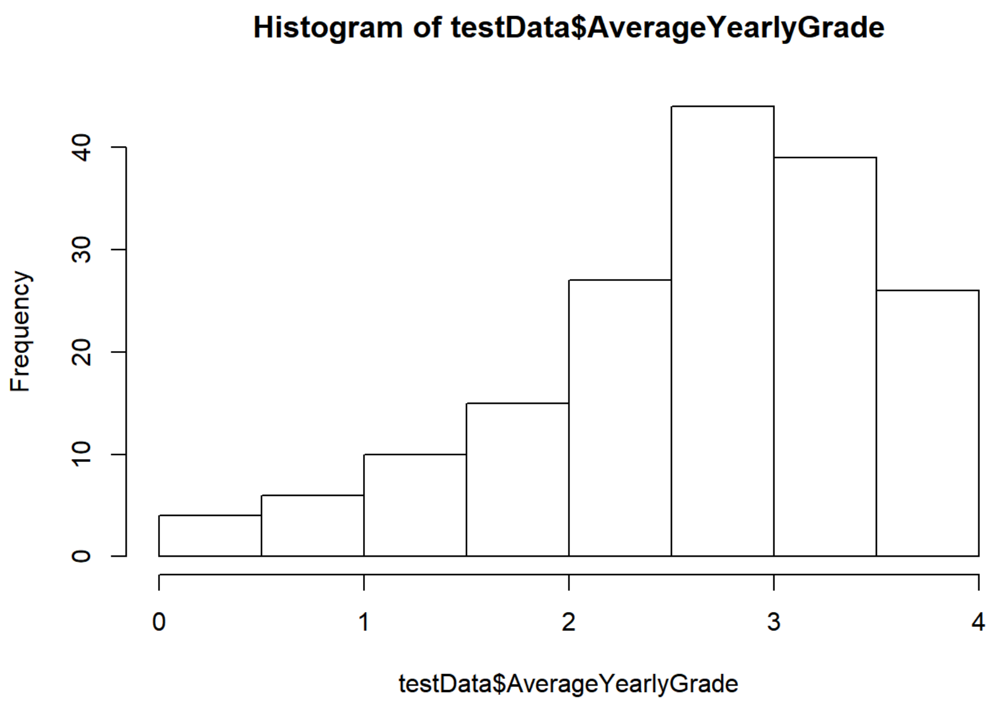
```
## Working directory is K:/math280-00-w20/Common/spps
```

```
modeling_data <- read_csv("derived_data/modeling_data.csv")

testData <- modeling_data %>%
  group_by(ID, YearInProgram) %>%
  summarise(AverageYearlyGrade = mean(EnglishGrade))
```

```
# EDA.  It looks normal but also left skewed.

hist(testData$AverageYearlyGrade)
```



```
testDatanoNA <- DropNA(testData)
```

```
## No Var specified. Dropping all NAs from the data frame.
```

```
## 1283 rows dropped from the data frame because of missing values.
```

```
mean(testDatanoNA$AverageYearlyGrade)
```

```
## [1] 2.679386
```

```
# This function generates data solely for the purpose of evaluating our model's power.
# max.years describes the maximum number of years in the program.
# n.students describes the maximum student ID value to be generated.

generatedData <- function (n.students, max.years){
    YearInProgram <- 5*(rep(seq(0,1,length = max.years), n.students) + 0.2)         # 4 values for each stu
dent (this is arbitrary)
    ID <- rep(1:n.students, each = max.years)                             # Student IDs


    g.0.true <- -.55 # g.0 and g.1 are used to construct the mean for the b.true sampling distribution
    g.1.true <- .5

    sigma.averageYearlyGrade.true <- .7
    sigma.a.true <- 1.3
    sigma.b.true <- .7

    mu.a.true <- 2.667
    a.true <- rnorm (n.students, mu.a.true, sigma.a.true)
    b.true <- rnorm (n.students, g.0.true + g.1.true, sigma.b.true)

    df <- modeling_data[1:n.students,]


    df$EnglishGrade <- sample(modeling_data$EnglishGrade[!is.na(modeling_data$EnglishGrade)], n.students, re
place = TRUE)
    df$AverageYearlyGrade <- df$EnglishGrade
    df$ReadingGrade <- sample(modeling_data$ReadingGrade[!is.na(modeling_data$ReadingGrade)], n.students, re
place = TRUE)

    df$`ACT-English` <- sample(modeling_data$`ACT-English`[!is.na(modeling_data$`ACT-English`)], n.students,
replace = TRUE)
replace = TRUE)

    df$RestrictedLanguage <- sample(modeling_data$RestrictedLanguage[!is.na(modeling_data$RestrictedLanguage
)], n.students, replace = TRUE)
    df$Receiving_Services <- sample(modeling_data$`Receiving Services`[!is.na(modeling_data$`Receiving Servi
ces`)], n.students, replace = TRUE)
    df$SchoolLevel <- sample(modeling_data$SchoolLevel, n.students, replace = TRUE)
    df$In_Program <- sample(modeling_data$In_Program, n.students, replace = TRUE)


    df$RestrictedLanguage <- as.character(df$RestrictedLanguage)
    df$Receiving_Services <- as.character(df$Receiving_Services)
    df$Receiving_Services <- as.character(df$Receiving_Services)
    df$SchoolLevel <- as.character(df$SchoolLevel)
    df$In_Program <- as.character(df$In_Program)

    df <- modeling_data[sample(1:nrow(modeling_data), n.students),]
    return (df)
}

model.power <- function(n.students, max.years, specificModel, multiple, numberSims = 10) { # Includes Number
of Simulations
  signif <- rep(NA, numberSims)
  for(s in 1:numberSims) {
    generated_data <- generatedData(n.students, max.years) # Call the other function
    lme.power <- eval(parse(text = specificModel)) # Model

    fixedEffects <- fixef(lme.power)["YearInProgram"] # Store the fixed/random effects
    fixedEffectsSD <- se.fixef(lme.power)["YearInProgram"]
    names(fixedEffects) <- c() # and remove column names
    names(fixedEffectsSD) <- c()

    if (specificModel %in% c(English.lmer, Reading.lmer, ACTEnglish.lmer)) {



      fixedEffectsTemp <- fixef(lme.power)["In_ProgramStill in Program"]
      fixedEffectsSDTemp <- se.fixef(lme.power)["In_ProgramStill in Program"]
      names(fixedEffectsTemp) <- c()
      names(fixedEffectsSDTemp) <- c()

      fixedEffects <- fixedEffects + fixedEffectsTemp
      fixedEffectsSD <- fixedEffectsSD + fixedEffectsSDTemp

      if (specificModel %in% c(English.lmer, Reading.lmer)) {
        fixedEffectsTemp <- fixef(lme.power)["SchoolLevelJH"]
        fixedEffectsSDTemp <- se.fixef(lme.power)["SchoolLevelJH"]
        names(fixedEffectsTemp) <- c()
        names(fixedEffectsSDTemp) <- c()

        fixedEffects <- fixedEffects + fixedEffectsTemp
        fixedEffectsSD <- fixedEffectsSD + fixedEffectsSDTemp
      }

    }



    if (specificModel == specificmodel1) {
      theta.hat <- fixedEffects + ranef(lme.power)$ID[,1]  # Add Random effects
      theta.se <- fixedEffectsSD + se.ranef(lme.power)$ID[,"(Intercept)"]
      names(theta.se) <- c()
      names(theta.hat) <- c()
    }
    if (specificModel %in% c(English.lmer, Reading.lmer, ACTEnglish.lmer)) {
      theta.hat <- fixedEffects + sum(ranef(lme.power)$RestrictedLanguage[,1])
      theta.se <- fixedEffectsSD + sum(se.ranef(lme.power)$RestrictedLanguage[,"(Intercept)"])
      names(theta.se) <- c()
      names(theta.hat) <- c()
      theta.hat <- theta.hat + sum(ranef(lme.power)$`Receiving Services`[,1])
      theta.se <- theta.se + sum(se.ranef(lme.power)$`Receiving Services`[,"(Intercept)"])
      names(theta.se) <- c()
      names(theta.hat) <- c()
    }

    signif[s] <- (theta.hat - multiple*theta.se) > 0 # Vector of true/false values
  }
  power <- mean(signif)
  return(power)
}
```

```
specificmodel1 <- "lmer(EnglishGrade ~ YearInProgram + (1 | ID), data = generated_data)"
English.lmer <- "lmer(EnglishGrade ~ YearInProgram + (1 | RestrictedLanguage) + (1 | `Receiving Services`) +
SchoolLevel + In_Program, data=generated_data)"
Reading.lmer <- "lmer(ReadingGrade ~ YearInProgram + (1 | RestrictedLanguage) + (1 | `Receiving Services`) +
SchoolLevel + In_Program, data=generated_data)"
ACTEnglish.lmer <- "lmer(`ACT-English` ~ YearInProgram + (1 | RestrictedLanguage) + (1 | `Receiving Services
`) + In_Program, data = generated_data)"

powervalues1 <- rep(NA, 5)
powervaluesEnglish <- rep(NA, 5)
powervaluesReading <- rep(NA, 5)
powervaluesACTEnglish <- rep(NA, 5)
for (i in (1:5)*1000) {
  powervalues1[i/1000] <- model.power(i, 6, specificmodel1, 2)
  powervaluesEnglish[i/1000] <- model.power(i, 6, English.lmer, .2)
  powervaluesReading[i/1000] <- model.power(i, 6, Reading.lmer, -.2)
  powervaluesACTEnglish[i/1000] <- model.power(i, 6, ACTEnglish.lmer, .05)

}
```

```
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00307611
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## boundary (singular) fit: see ?isSingular
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.094005
## (tol = 0.002, component 1)
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00318176
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00622385
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.031704
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00654102
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.0048175
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.0145743
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00213647
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00711824
## (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00459449
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length

## Warning in signif[s] <- (theta.hat - multiple * theta.se) > 0: number of
## items to replace is not a multiple of replacement length
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.0304526
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifi
able: large eigenvalue ratio
##  - Rescale variables?
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues
```

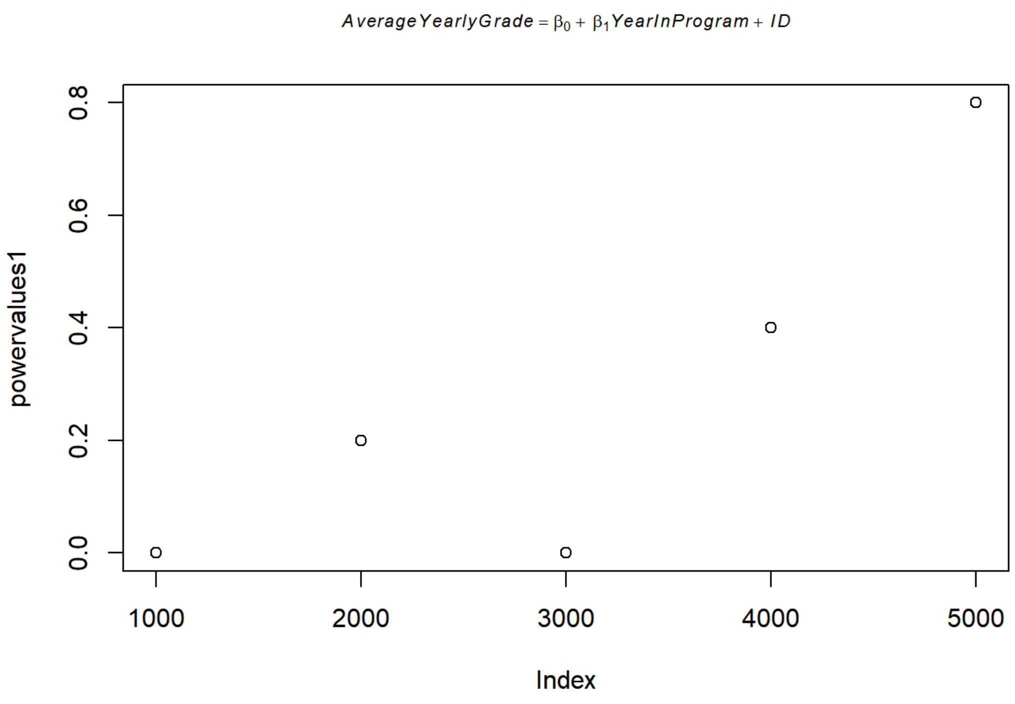
```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

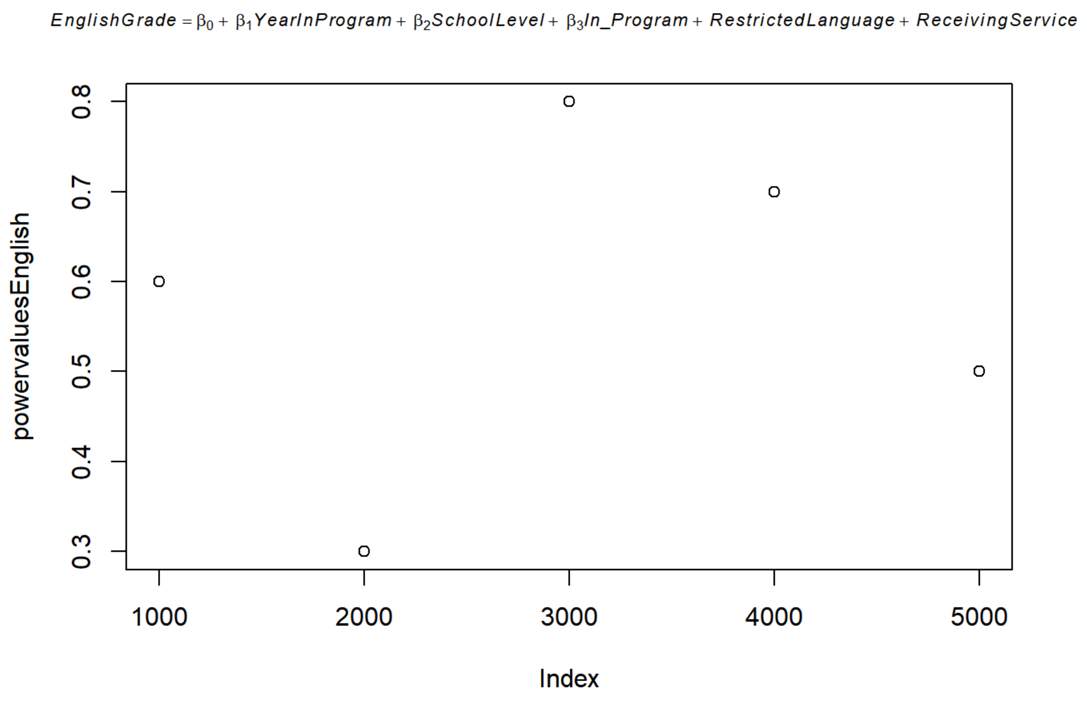
```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Hessian is numerically singular: parameters are not
## uniquely determined
```

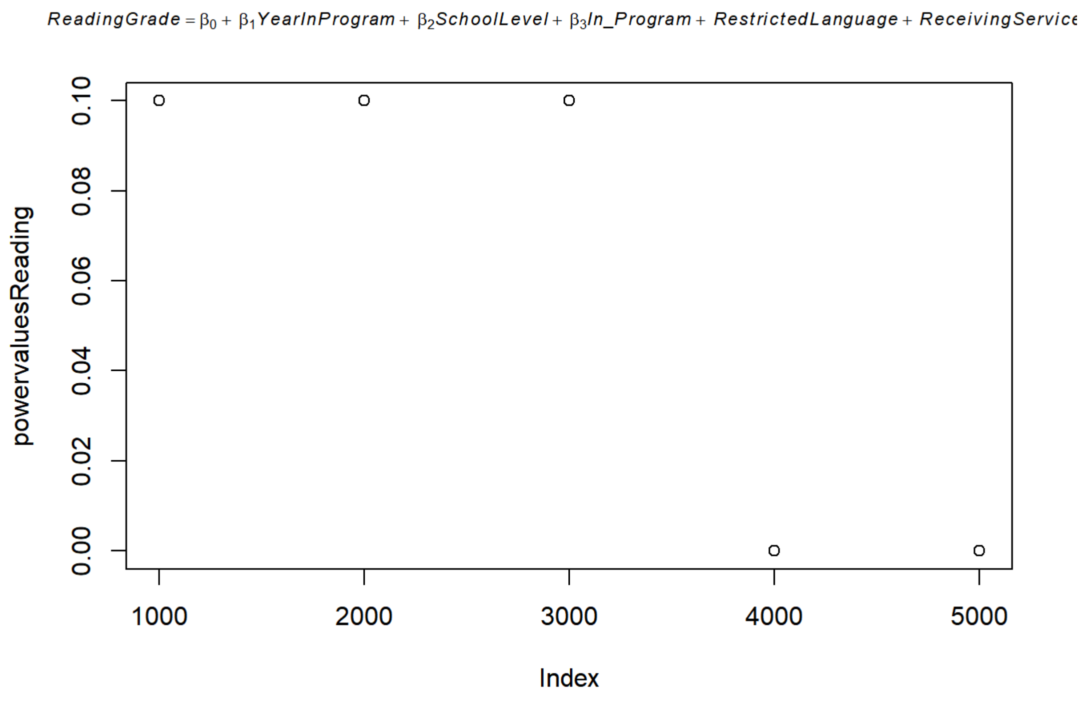
```
plot(powervalues1, xaxt = "n", main = expression(italic(AverageYearlyGrade) == beta[0] ~+~ beta[1]*italic(Ye
arInProgram) ~+~ italic(ID)), cex.main = 0.7)
axis(1, at=1:5, labels=(1:5)*1000)
```



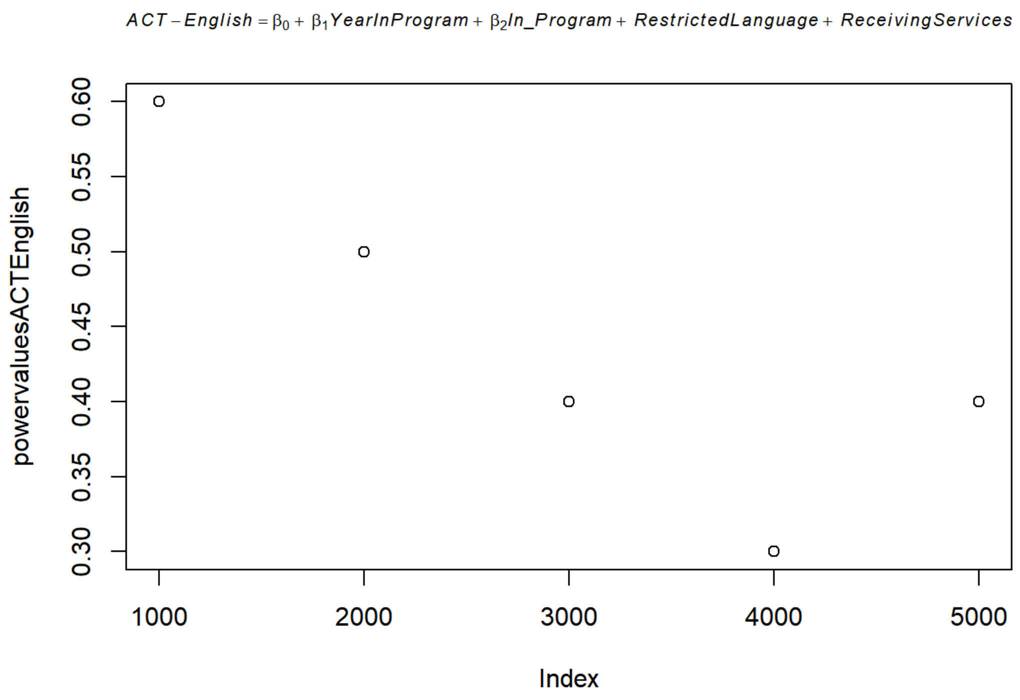
```
plot(powervaluesEnglish, xaxt = "n", main = expression(italic(EnglishGrade) == beta[0] ~+~ beta[1]*italic(Ye
arInProgram)~+~ beta[2]*italic(SchoolLevel)~+~ beta[3]*italic(In_Program)~+~ italic(RestrictedLanguage)~+~ i
talic(ReceivingServices)), cex.main = 0.7)
axis(1, at=1:5, labels=(1:5)*1000)
```



```
plot(powervaluesReading, xaxt = "n", main = expression(italic(ReadingGrade) == beta[0] ~+~ beta[1]*italic(Ye
arInProgram)~+~ beta[2]*italic(SchoolLevel)~+~ beta[3]*italic(In_Program)~+~ italic(RestrictedLanguage)~+~ i
talic(ReceivingServices)), cex.main = 0.7)
axis(1, at=1:5, labels=(1:5)*1000)
```



```
plot(powervaluesACTEnglish, xaxt = "n", main = expression(italic(ACT-English) == beta[0] ~+~ beta[1]*italic(
YearInProgram)~+~beta[2]*italic(In_Program)~+~ italic(RestrictedLanguage)~+~ italic(ReceivingServices)), cex
.main = 0.7)
axis(1, at=1:5, labels=(1:5)*1000)
```



```
#lme.power <- lmer(EnglishGrade ~ YearInProgram + (1 | RestrictedLanguage) + (1 | `Receiving Services`) + Sc
hoolLevel + In_Program, data=modeling_data[1:500,])
```

```
model <- lmer(AverageYearlyGrade ~ YearInProgram + (1 | ID), data = testData)
model
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: AverageYearlyGrade ~ YearInProgram + (1 | ID)
##    Data: testData
## REML criterion at convergence: 410.382
## Random effects:
##  Groups   Name        Std.Dev.
##  ID       (Intercept) 0.6577
##  Residual             0.5737
## Number of obs: 171, groups:  ID, 94
## Fixed Effects:
##   (Intercept)  YearInProgram
##       2.61347        0.01692
```

```
coef(model)$ID[1]
```

```
##        (Intercept)
## 165009    3.296461
## 170042    2.953377
## 170415    2.721767
## 170649    2.203276
## 170943    2.638802
## 180034    1.819951
## 180035    3.278803
## 180044    1.712843
## 180050    2.662174
## 180059    3.170696
## 180095    1.777359
## 180100    2.926212
## 180555    2.084098
## 180798    2.529812
## 180825    2.654562
## 190061    3.034192
## 190063    2.588821
## 190064    1.418891
## 190066    2.661941
## 190071    2.329575
## 190076    3.020897
## 190153    2.322730
## 190192    3.154489
## 190208    2.645507
## 190668    2.165593
## 190871    1.902998
## 191012    3.098257
## 200053    2.243160
## 200794    2.927835
## 200812    2.672672
## 200877    2.255477
## 200882    2.955728
## 200934    2.482632
## 210030    2.376106
## 210055    1.441214
## 210099    2.389401
## 210132    2.756967
## 210209    3.080723
## 210318    1.724668
## 210398    3.267099
## 210400    2.595468
## 210639    3.179751
## 210785    3.678983
## 210870    2.509682
## 210930    2.340660
## 220052    2.841419
## 220348    3.193728
## 220383    2.927835
## 220433    1.700586
## 220541    2.030445
## 220549    3.248989
## 220584    3.060781
## 220651    1.610036
## 220698    3.047487
## 220705    2.430431
## 220738    3.300085
## 220795    3.173786
## 220874    2.524590
## 220878    3.240259
## 220908    2.838865
## 220957    1.730180
## 234571    2.151075
## 234606    2.723799
## 234614    3.031669
## 234923    2.378230
## 234975    2.497425
## 235004    2.280105
## 235042    2.995449
## 235056    1.130121
## 235090    2.463413
## 235266    1.854520
## 235303    1.800190
## 235384    1.573577
## 235430    2.586601
## 235434    2.146487
## 235451    2.430431
## 235466    2.596210
## 235469    2.922746
## 240015    3.319834
## 240049    3.362426
## 240092    2.326029
## 240406    2.652565
## 240413    3.135270
## 240414    2.595776
## 240430    3.263045
## 240432    3.177862
## 240436    2.609973
## 240443    3.277242
## 240451    3.319834
## 240453    3.035890
## 240461    2.879720
## 240542    3.021693
## 240580    2.510593
## 240663    3.064284
```

```
coef(model)$ID[2]
```

```
##        YearInProgram
## 165009    0.01692122
## 170042    0.01692122
## 170415    0.01692122
## 170649    0.01692122
## 170943    0.01692122
## 180034    0.01692122
## 180035    0.01692122
## 180044    0.01692122
## 180050    0.01692122
## 180059    0.01692122
## 180095    0.01692122
## 180100    0.01692122
## 180555    0.01692122
## 180798    0.01692122
## 180825    0.01692122
## 190061    0.01692122
## 190063    0.01692122
## 190064    0.01692122
## 190066    0.01692122
## 190071    0.01692122
## 190076    0.01692122
## 190153    0.01692122
## 190192    0.01692122
## 190208    0.01692122
## 190668    0.01692122
## 190871    0.01692122
## 191012    0.01692122
## 200053    0.01692122
## 200794    0.01692122
## 200812    0.01692122
## 200877    0.01692122
## 200882    0.01692122
## 200934    0.01692122
## 210030    0.01692122
## 210055    0.01692122
## 210099    0.01692122
## 210132    0.01692122
## 210209    0.01692122
## 210318    0.01692122
## 210398    0.01692122
## 210400    0.01692122
## 210639    0.01692122
## 210785    0.01692122
## 210870    0.01692122
## 210930    0.01692122
## 220052    0.01692122
## 220348    0.01692122
## 220383    0.01692122
## 220433    0.01692122
## 220541    0.01692122
## 220549    0.01692122
## 220584    0.01692122
## 220651    0.01692122
## 220698    0.01692122
## 220705    0.01692122
## 220738    0.01692122
## 220795    0.01692122
## 220874    0.01692122
## 220878    0.01692122
## 220908    0.01692122
## 220957    0.01692122
## 234571    0.01692122
## 234606    0.01692122
## 234614    0.01692122
## 234923    0.01692122
## 234975    0.01692122
## 235004    0.01692122
## 235042    0.01692122
## 235056    0.01692122
## 235090    0.01692122
## 235266    0.01692122
## 235303    0.01692122
## 235384    0.01692122
## 235430    0.01692122
## 235434    0.01692122
## 235451    0.01692122
## 235466    0.01692122
## 235469    0.01692122
## 240015    0.01692122
## 240049    0.01692122
## 240092    0.01692122
## 240406    0.01692122
## 240413    0.01692122
## 240414    0.01692122
## 240430    0.01692122
## 240432    0.01692122
## 240436    0.01692122
## 240443    0.01692122
## 240451    0.01692122
## 240453    0.01692122
## 240461    0.01692122
## 240542    0.01692122
## 240580    0.01692122
## 240663    0.01692122
```