MMAD Analysis

James Zhao

May 25, 2019

First, we obtain the MMAD even-level dataset and load it as follows.

```
setwd("~/GitHub/MMSS_311_2/Data Sets")
mmad <- read.csv("events.csv")
mmad <- mmad[,-(1:6)]
mmad <- na.omit(mmad)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyve
rse 1.2.1 --
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_co
nflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)
library(glmnet)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

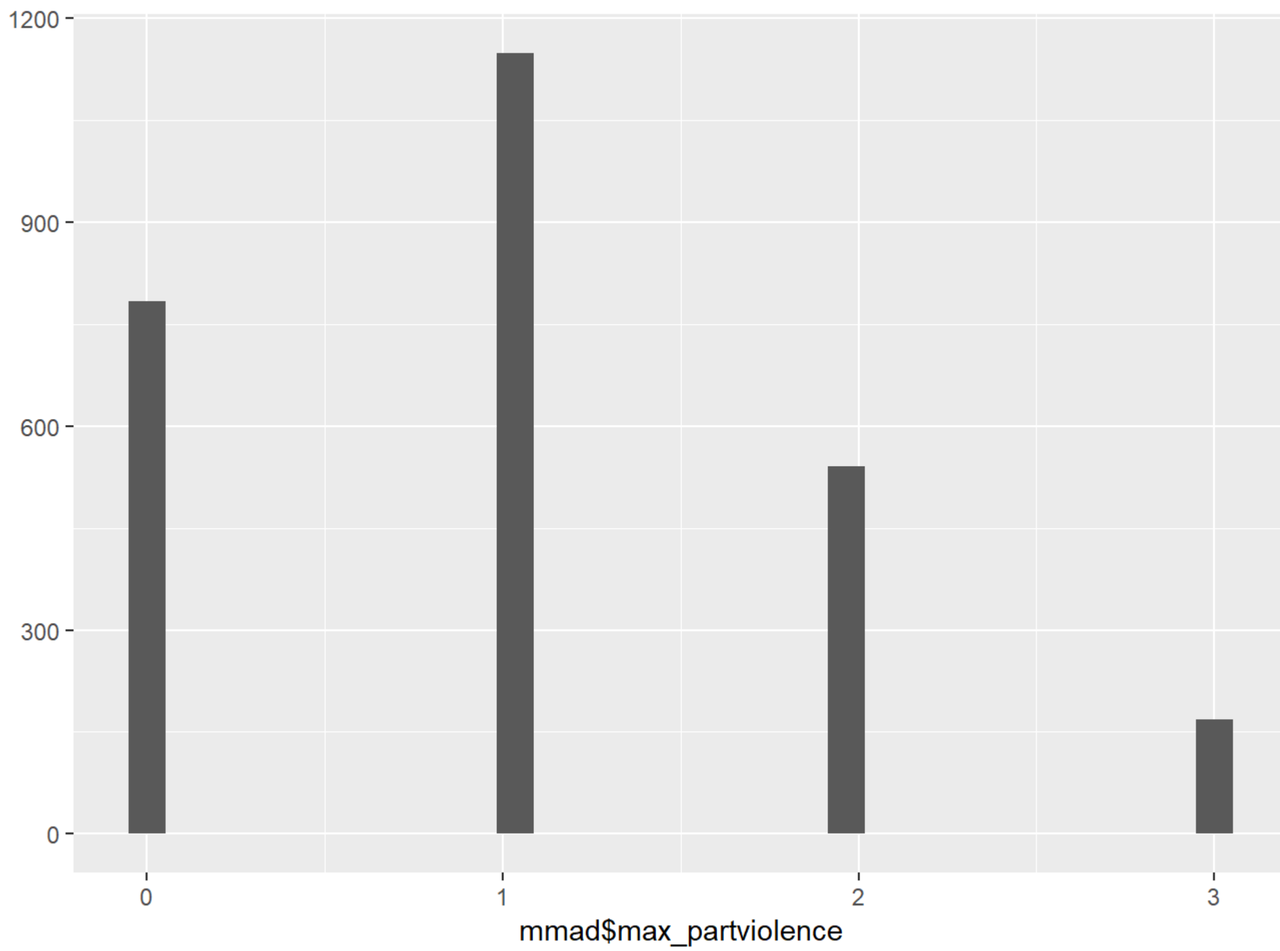
```
## Loaded glmnet 2.0-16
```

We plot the levels of max_partviolence in the dataset.

Then, we run a LASSO regression.

```
qplot (mmad$max_partviolence)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
x <- as.matrix(mmad[,-4])
y <- (mmad$max_partviolence)

grid =  10 ^ seq(-2, 2, length = 100)
lasso_mod = glmnet(x, y, alpha = 1, lambda = grid)
lasso_mod
```

```
##
## Call:  glmnet(x = x, y = y, alpha = 1, lambda = grid)
##
##        Df    %Dev    Lambda
##   [1,]  0 0.00000 100.00000
##   [2,]  0 0.00000  91.12000
##   [3,]  0 0.00000  83.02000
##   [4,]  0 0.00000  75.65000
##   [5,]  0 0.00000  68.93000
##   [6,]  0 0.00000  62.80000
##   [7,]  0 0.00000  57.22000
##   [8,]  0 0.00000  52.14000
##   [9,]  0 0.00000  47.51000
##  [10,]  0 0.00000  43.29000
##  [11,]  0 0.00000  39.44000
##  [12,]  0 0.00000  35.94000
##  [13,]  0 0.00000  32.75000
##  [14,]  0 0.00000  29.84000
##  [15,]  0 0.00000  27.19000
##  [16,]  0 0.00000  24.77000
##  [17,]  0 0.00000  22.57000
##  [18,]  0 0.00000  20.57000
##  [19,]  0 0.00000  18.74000
##  [20,]  0 0.00000  17.07000
##  [21,]  0 0.00000  15.56000
##  [22,]  0 0.00000  14.17000
##  [23,]  0 0.00000  12.92000
##  [24,]  0 0.00000  11.77000
##  [25,]  0 0.00000  10.72000
##  [26,]  0 0.00000   9.77000
##  [27,]  0 0.00000   8.90200
##  [28,]  0 0.00000   8.11100
##  [29,]  0 0.00000   7.39100
##  [30,]  0 0.00000   6.73400
##  [31,]  0 0.00000   6.13600
##  [32,]  0 0.00000   5.59100
##  [33,]  0 0.00000   5.09400
##  [34,]  0 0.00000   4.64200
##  [35,]  0 0.00000   4.22900
##  [36,]  0 0.00000   3.85400
##  [37,]  0 0.00000   3.51100
##  [38,]  0 0.00000   3.19900
##  [39,]  0 0.00000   2.91500
##  [40,]  0 0.00000   2.65600
##  [41,]  0 0.00000   2.42000
##  [42,]  0 0.00000   2.20500
##  [43,]  0 0.00000   2.00900
##  [44,]  0 0.00000   1.83100
##  [45,]  0 0.00000   1.66800
##  [46,]  0 0.00000   1.52000
##  [47,]  0 0.00000   1.38500
##  [48,]  0 0.00000   1.26200
##  [49,]  0 0.00000   1.15000
```

```
##  [50,]  0 0.00000   1.04800
##  [51,]  0 0.00000   0.95450
##  [52,]  0 0.00000   0.86970
##  [53,]  0 0.00000   0.79250
##  [54,]  0 0.00000   0.72210
##  [55,]  0 0.00000   0.65790
##  [56,]  0 0.00000   0.59950
##  [57,]  0 0.00000   0.54620
##  [58,]  1 0.03996   0.49770
##  [59,]  1 0.09559   0.45350
##  [60,]  1 0.14180   0.41320
##  [61,]  1 0.18010   0.37650
##  [62,]  1 0.21190   0.34300
##  [63,]  1 0.23840   0.31260
##  [64,]  1 0.26030   0.28480
##  [65,]  1 0.27850   0.25950
##  [66,]  1 0.29360   0.23640
##  [67,]  1 0.30620   0.21540
##  [68,]  1 0.31660   0.19630
##  [69,]  1 0.32530   0.17890
##  [70,]  1 0.33250   0.16300
##  [71,]  1 0.33840   0.14850
##  [72,]  1 0.34340   0.13530
##  [73,]  1 0.34750   0.12330
##  [74,]  1 0.35090   0.11230
##  [75,]  1 0.35370   0.10240
##  [76,]  1 0.35610   0.09326
##  [77,]  1 0.35800   0.08498
##  [78,]  1 0.35970   0.07743
##  [79,]  1 0.36100   0.07055
##  [80,]  2 0.36220   0.06428
##  [81,]  2 0.36390   0.05857
##  [82,]  3 0.36640   0.05337
##  [83,]  3 0.36850   0.04863
##  [84,]  3 0.37020   0.04431
##  [85,]  3 0.37160   0.04037
##  [86,]  3 0.37270   0.03678
##  [87,]  3 0.37370   0.03352
##  [88,]  4 0.37470   0.03054
##  [89,]  4 0.37550   0.02783
##  [90,]  4 0.37620   0.02535
##  [91,]  4 0.37680   0.02310
##  [92,]  4 0.37730   0.02105
##  [93,]  4 0.37770   0.01918
##  [94,]  4 0.37800   0.01748
##  [95,]  4 0.37830   0.01592
##  [96,]  5 0.37860   0.01451
##  [97,]  5 0.37880   0.01322
##  [98,]  5 0.37900   0.01205
##  [99,]  5 0.37920   0.01097
## [100,]  5 0.37940   0.01000
```

Then, we use tidy from the broom package to extract the data from the regression into a useable format and use ggplot2 to plot the coefficient estimates as lambda changes.

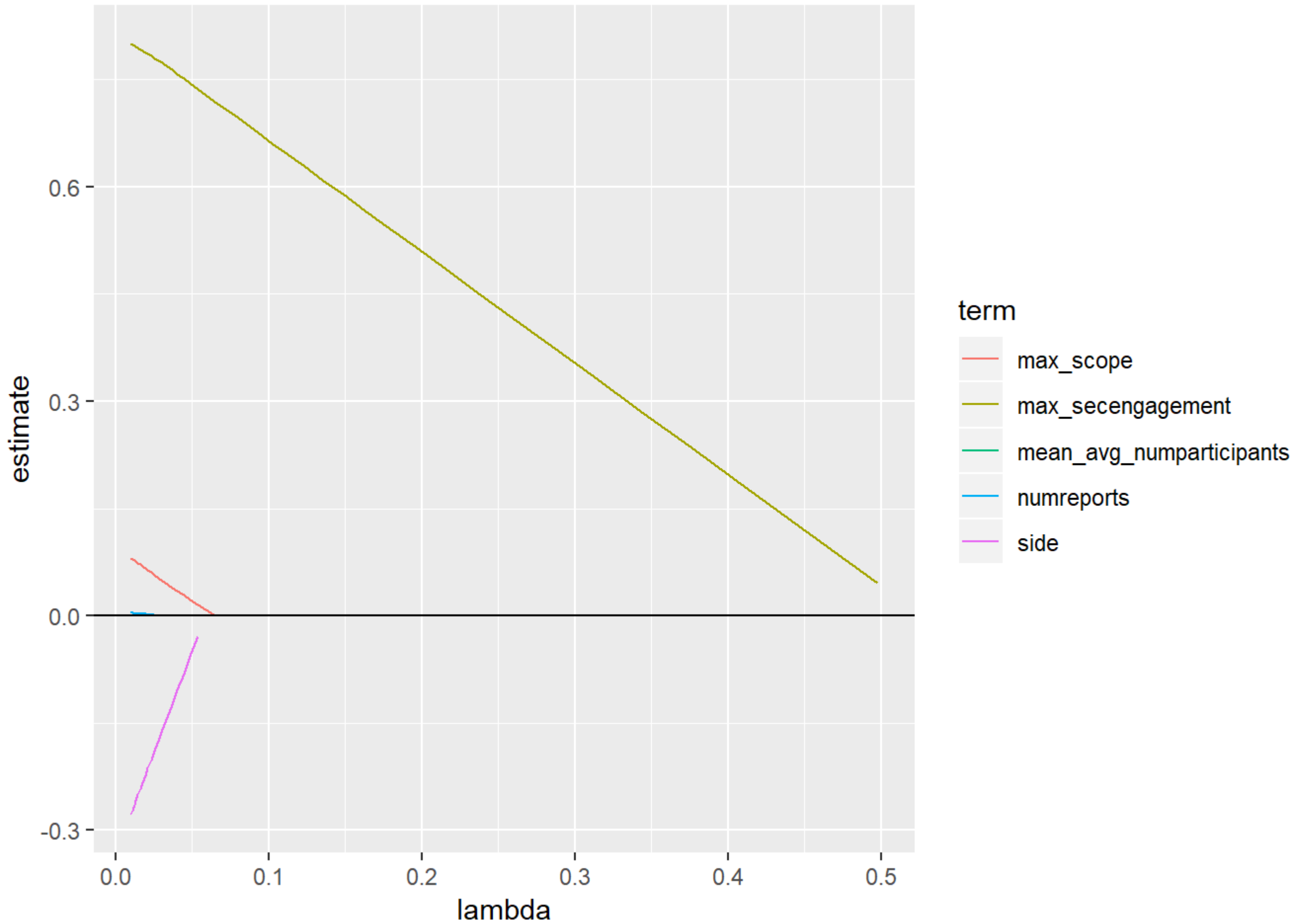
```
lasso_output <- broom::tidy(lasso_mod)

head(lasso_output, 12)
```

```
## # A tibble: 12 x 5
##    term         step estimate lambda dev.ratio
##    <chr>       <dbl>    <dbl>  <dbl>     <dbl>
##  1 (Intercept)     1     1.04  100           0
##  2 (Intercept)     2     1.04   91.1         0
##  3 (Intercept)     3     1.04   83.0         0
##  4 (Intercept)     4     1.04   75.6         0
##  5 (Intercept)     5     1.04   68.9         0
##  6 (Intercept)     6     1.04   62.8         0
##  7 (Intercept)     7     1.04   57.2         0
##  8 (Intercept)     8     1.04   52.1         0
##  9 (Intercept)     9     1.04   47.5         0
## 10 (Intercept)    10     1.04   43.3         0
## 11 (Intercept)    11     1.04   39.4         0
## 12 (Intercept)    12     1.04   35.9         0
```

Next, we use cross validation with cv.glmnet to pick the value of lambda that will minimize mean squared error, and give the coefficients you get when using that lambda.

```
lasso_output %>%
filter(term != '(Intercept)') %>%
ggplot(aes(x = lambda, y = estimate, group = term, color = term)) +
geom_line() +
geom_hline(yintercept = 0)
```



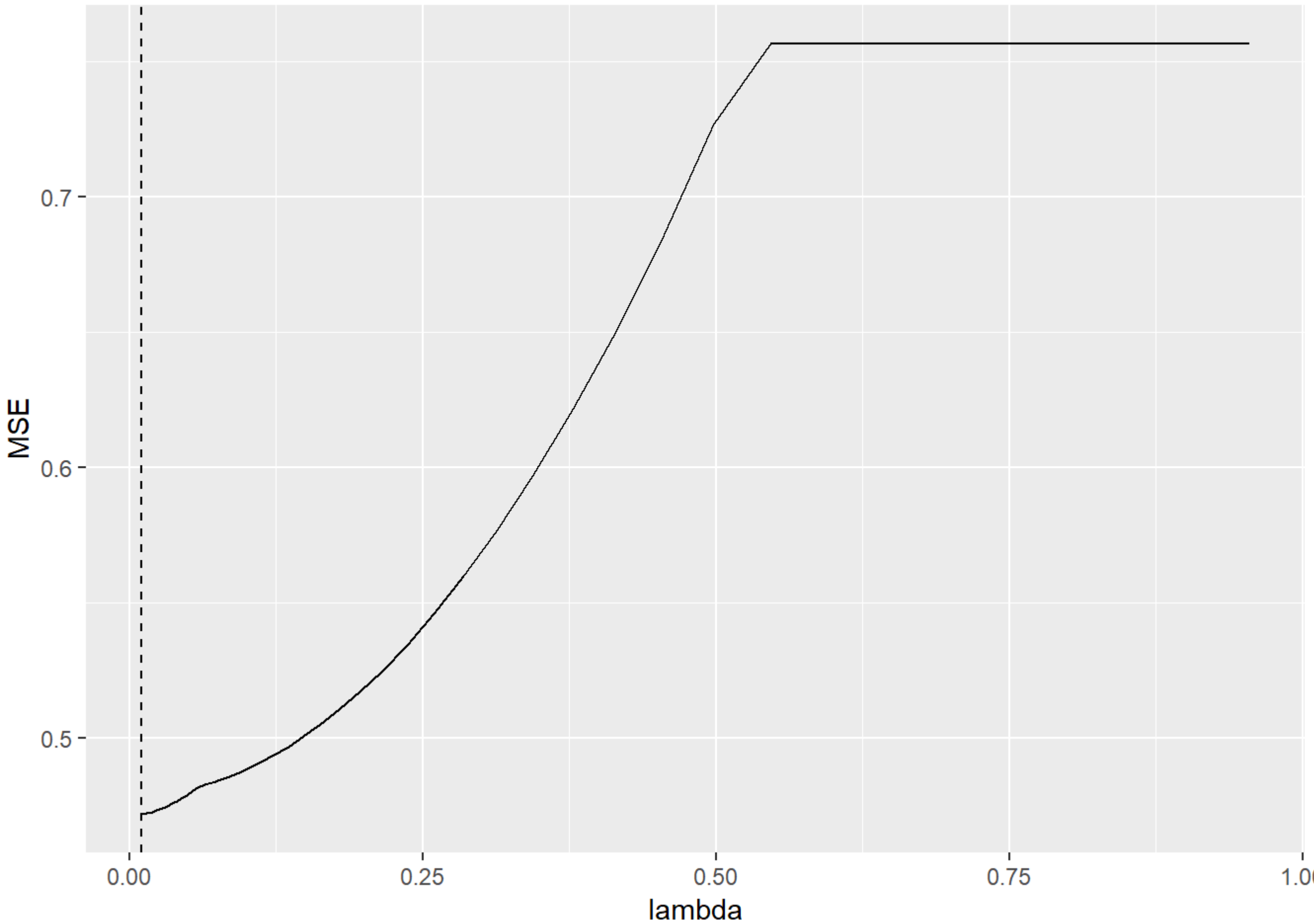

```
#cross validation
lasso_cv <- cv.glmnet(x = x, y = y, alpha = 1, nfolds = 15, lambda = grid)

#lambda with the best (lowest) MSE
lasso_cv$lambda.min
```

```
## [1] 0.01
```

We visualize our results.

```
broom::tidy(lasso_cv) %>%
filter(lambda <= 1) %>%
ggplot(aes(x = lambda, y = estimate)) +
geom_line() +
geom_vline(xintercept = lasso_cv$lambda.min,
linetype = 'dashed') +
labs(y = 'MSE')
```



Lastly, we refit our LASSO model with the best lambda value found and print the coeeficients for each of the variables.

```
lasso_final <- glmnet(x = x, y = y, alpha = 1, lambda = lasso_cv$lambda.min)
coef(lasso_final)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                                    s0
## (Intercept)             -2.280423e-01
## side                    -2.779140e-01
## numreports               4.016611e-03
## max_scope                8.087961e-02
## max_secengagement        7.996387e-01
## mean_avg_numparticipants -1.640865e-07
```