MSD 2019 Final Project

A replication of Greed and Grievance in Civil War by Paul Collier and Anke Hoeffler, 2000

Kiran Ramesh (kr2789), Sai Srujan Chinta (sc4401), Bhavya Shahi (bs3118)

2019-05-13 15:41:56

Contents

1. Introduction

Most major conflicts today aren't international conflicts but are civil wars. The authors cite the Stockholm International Peace Research Institute which claims that all of the 15 major armed conflicts in 2002 (the year before the paper was published) were civil wars. In a previous work by the same authors (published in 1998), the authors proposed that the post-conflict gains by the rebels would be the primary motivation for civil war. The paper focused on testing the implication that if the post-conflict pay-offs were high, there would be justification for prolonging the civil war. The authors revisit this assumption in this paper, as they recognize that the previous assumption was untenable: The rebel groups usually cover their costs during the conflict. Thus, they propose a more general theory in this paper which compares the opportunities for rebellion versus the constraints to rebellion.

Political science states that most rebellions occur when grievances are sufficiently dire that people feel the need to rebel. However economic theory states that rebellion is an industry which generates profits for the rebels (through looting and extortion), and thus rebels are motivated by greed. Thus, rebellions arises when there are atypical opportunities for profit available, and not due to motivation.

The paper compares the two theories- opportunity (greed) versus motivation (grievance).

2. Replicating the Original Study

2.1 Data description

The authors define civil war as "an internal conflict with at least 1000 combat-related deaths per year. In addition, to distinguish a war from a massacre, at least 5% of fatalities must be suffered by both the government and a rebel group.

The sample that is used in this paper covers around 750 five-year episodes (79 civil wars) spanning across 161 countries over the period 1960-1999 (ref 1). Data should be scaled to reflect the population of the country where appropriate.

To quantify opportunity, the authors consider several proxies including:

1) Extortion of natural resources: The proxy used is the ratio between primary commodity exports to the GDP of the country (i.e the dependence of the country on its primary export). If a country is rich in natural resources (such as Saudi Arabia), the government may be too well financed making rebellion untenable.

2) Financing from diasporas: The proxy is the proportion of a country's population living in the United States. This is also separated into the two- the people who emigrated due to the conflict, and those that emigrated for other reasons.

3) Finance from rebel governments : The proxy is the Cold War during which each great country financed rebellions in countries which were allied to the opposing country.

4) Forgone income: The income that the rebel loses by joining the rebellion. If this is low, the rebel has more of a reason to rebel. This is proxied using three variables - mean income per capita, male secondary schooling, and growth rate of the country.

5) Cost of rebellion: When the cost of weapons, skills etc. is low, the opportunity for rebellion may be higher. This is proxied by the time since the previous conflict.

6) Government military ability: If the terrain of the country is favorable to the rebels, the government has a weak military, or the country's population is dispersed, opportunity for conflict may be high.

7) Ethnic and religious diversity: The more diverse a society is, the recruitment pool for rebellion is reduced. The proxy used here is the index for ethno-linguistic fractionalization.

To quantify grievance, the authors consider the following proxies:

1) Ethnic and religious hatreds: Co-existence of multiple ethnicities and multiple religions in society can lead to conflicts. More specifically, inter-group tensions can be chalked up to polarization (more so than diversity).

2) Political Repression: Unless the degree of political repression is extreme, repression increases the likelihood of conflict. The authors also find a clear difference in the extents of political rights between conflict and peace episodes.

3) Political Exclusion: Counter-intuitively, the smaller the majority, the more exclusion the minority experiences. The plausible explanation for this is that the larger the minority, the greater the incentive for the majority to exploit the disparity in the population.

4) Economic Inequality: The poor will resort to conflict in order to reset the wealth distribution and the rich engage in inciting conflicts to pre-empt the aforementioned conflicts by the poor.

2.2 Regression Analysis

Models are developed which predict the risk of civil war in a five-year period using logistic regression. Two models are developed - one using proxies for opportunity, and one using proxies for grievances. The two are

compared and an integrated model is arrived at.

```
setwd(".")
options(scipen = 100, digits = 4)

data <- read.dta("data/G&G.dta")
```

```
summarize_into_table <- function(summary_obj) {
    # takes summary object as input, and returns a DF
    # res.  to print res, use print(summ(obj), quote =
    # FALSE) is you don't want quotes

    options(digits = 4)
    res <- t(round(summary_obj$coefficients, digits = 4))
    z.values <- res[4, ]
    Signif <- symnum(z.values, corr = FALSE, na = FALSE,
        cutpoints = c(0, 0.01, 0.05, 0.1, 1), symbols = c("***",
            "**", "*", ""))
    res <- rbind(res, Signif)
    res <- t(res)
    res <- res[-1, ]
    res
}

comma_sep = function(x) {
    x = strsplit(x, "")
}

convert2dArrayToDf = function(all_tests) {
    model_names <- all_tests[, 1]

    invisible(apply(all_tests, 2, as.numeric))
    invisible(sapply(all_tests, as.numeric))
    class(all_tests) <- "numeric"
    storage.mode(all_tests) <- "numeric"

    all_tests <- as.data.frame(all_tests)

    all_tests[, 1] <- model_names

    return(all_tests)
}
```

2.2.1 Opportunity Model

We first explore the intuition behind the choice of features for each regression model:

The first regression model excludes the "per capita" feature because this feature is highly correlated with the "enrollment in secondary schooling" feature. The diaspora measures are only available for 29 war episodes and hence are not explored for the first three regression models. (688 episodes from 123 countries)

Due to certain technicalities which are explained below, a dummy feature called "Previous War" was added in the first regression model. This feature is removed in the second regression model and all the other features remain the same. (688 episodes from 123 countries)

The third regression model replaces the "secondary schooling" feature with the "per capita income" feature (reverse of column 2). (750 episodes from 125 countries)

The fourth regression model introduces the "Diaspora/peace" feature and removes the "Post-coldwar", "Male secondary schooling","Previous War","Mountainous terrain", "Geographic dispersion" and "Social fractionalization" features. These features were removed because it was found that they did not add significant explanatory power and it was imperative to preserve the sample size. (29 episodes)

The "Diaspora/peace" feature is decomposed into two features which were computed as the difference between the actual diaspora and the estimated population of the diaspora had three been no conflict. The rest of the features remain the same as the fourth model.

```
# Opportunity Models

filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,prevwara,mount,geogia,frac,lnpop"
    "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,frac,lnpop",
    "warsa,sxp,sxp2,coldwar,lngdp_,gy1,peace,mount,geogia,frac,lnpop",
    "warsa,sxp,sxp2,lngdp_,peace,lnpop,diaspeaa", "warsa,sxp,sxp2,lngdp_,peace,lnpop,difdpeaa,diahpeaa")

regression_formula_list <- list("warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + prevwara + mount +
    "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + frac + lnpop",
    "warsa ~  sxp + sxp2 + coldwar + lngdp_ + gy1 + peace + mount + geogia + frac + lnpop",
    "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + diaspeaa",
    "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + difdpeaa + diahpeaa")


for (i in c(1:5)) {
    print(paste0("Opportunity Model ", i))

    filtering_columns <- strsplit(filtering_columns_list[[i]],
        ",")[[1]]
    print(filtering_columns)
    opportunity.data <- data[, filtering_columns]

    opportunity.data <- na.omit(opportunity.data)
    print(paste0("N : ", nrow(opportunity.data)))
    print(paste0("No of wars : ", nrow(opportunity.data[opportunity.data$warsa ==
        1, ])))

    opportunity_fit <- glm(as.formula(regression_formula_list[[i]]),
        family = binomial(link = "logit"), data = opportunity.data)

    print(paste0("Pseudo R2 : ", round(PseudoR2(opportunity_fit),
        digits = 2)))
    print(paste0("Log likelihood : ", round(logLik(opportunity_fit),
        digits = 2)))

    print(summarize_into_table(summary(opportunity_fit)),
        quote = FALSE)
}
```

```
## [1] "Opportunity Model 1"
##  [1] "warsa"    "sxp"      "sxp2"     "coldwar"  "secm"     "gy1"
##  [7] "peace"    "prevwara" "mount"    "geogia"   "frac"     "lnpop"
## [1] "N : 688"
## [1] "No of wars : 46"
```

```
## [1] "Pseudo R2 : 0.24"
## [1] "Log likelihood : -128.49"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      18.1486  6.0065     3.0215  0.0025   ***
## sxp2     -27.4453 11.9963    -2.2878 0.0221   **
## coldwar  -0.3257  0.4695     -0.6937 0.4879
## secm     -0.0248  0.0103     -2.4028 0.0163   **
## gy1      -0.117   0.0437     -2.6782 0.0074   ***
## peace    -0.0025  0.0017     -1.5234 0.1277
## prevwara 0.4639   0.5467     0.8487  0.3961
## mount    0.0129   0.0093     1.3929  0.1636
## geogia   -2.2115  1.0377     -2.1311 0.0331   **
## frac     -0.0002  0.0001     -1.6019 0.1092
## lnpop    0.6688   0.1631     4.1016  0        ***
## [1] "Opportunity Model 2"
##  [1] "warsa"   "sxp"     "sxp2"    "coldwar" "secm"    "gy1"     "peace"
##  [8] "mount"   "geogia"  "frac"    "lnpop"
## [1] "N : 688"
## [1] "No of wars : 46"
## [1] "Pseudo R2 : 0.24"
## [1] "Log likelihood : -128.85"
##         Estimate Std. Error z value Pr(>|z|) Signif
## sxp     18.8998  5.9478     3.1776  0.0015   ***
## sxp2    -29.1226 11.9047    -2.4463 0.0144   **
## coldwar -0.207   0.4496     -0.4605 0.6451
## secm    -0.0239  0.0101     -2.3565 0.0184   **
## gy1     -0.1182  0.0438     -2.6967 0.007    ***
## peace   -0.0036  0.0011     -3.1021 0.0019   ***
## mount   0.0137   0.0091     1.5097  0.1311
## geogia  -2.1291  1.0324     -2.0622 0.0392   **
## frac    -0.0002  0.0001     -1.5449 0.1224
## lnpop   0.6855   0.1617     4.2396  0        ***
## [1] "Opportunity Model 3"
##  [1] "warsa"   "sxp"     "sxp2"    "coldwar" "lngdp_"  "gy1"     "peace"
##  [8] "mount"   "geogia"  "frac"    "lnpop"
## [1] "N : 750"
## [1] "No of wars : 52"
## [1] "Pseudo R2 : 0.22"
## [1] "Log likelihood : -146.86"
##         Estimate Std. Error z value Pr(>|z|) Signif
## sxp     16.4757  5.2071     3.1641  0.0016   ***
## sxp2    -23.0168 9.9722     -2.3081 0.021    **
## coldwar -0.4543  0.4162     -1.0916 0.275
## lngdp_  -0.837   0.2532     -3.3055 0.0009   ***
## gy1     -0.1051  0.0421     -2.4933 0.0127   **
## peace   -0.0035  0.0011     -3.3355 0.0009   ***
## mount   0.0084   0.0085     0.9966  0.3189
## geogia  -0.8655  0.9482     -0.9127 0.3614
## frac    -0.0002  0.0001     -1.9716 0.0487   **
## lnpop   0.4927   0.1286     3.8308  0.0001   ***
## [1] "Opportunity Model 4"
## [1] "warsa"    "sxp"      "sxp2"     "lngdp_"   "peace"    "lnpop"
## [7] "diaspeaa"
## [1] "N : 595"
```

```
## [1] "No of wars : 32"
## [1] "Pseudo R2 : 0.25"
## [1] "Log likelihood : -93.27"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      17.5671  6.7436     2.605   0.0092   ***
## sxp2     -28.8151 15.35      -1.8772 0.0605   *
## lngdp_   -1.2366  0.2826     -4.3756 0        ***
## peace    -0.002   0.0014     -1.4716 0.1411
## lnpop    0.2949   0.1414     2.0859  0.037    **
## diaspeaa 700.9343 363.2903   1.9294  0.0537   *
## [1] "Opportunity Model 5"
## [1] "warsa"    "sxp"      "sxp2"     "lngdp_"   "peace"    "lnpop"
## [7] "difdpeaa" "diahpeaa"
## [1] "N : 595"
## [1] "No of wars : 32"
## [1] "Pseudo R2 : 0.25"
## [1] "Log likelihood : -93.23"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      17.4034  6.7493     2.5785  0.0099   ***
## sxp2     -28.4562 15.3642    -1.8521 0.064    *
## lngdp_   -1.2426  0.2837     -4.3794 0        ***
## peace    -0.002   0.0014     -1.4816 0.1385
## lnpop    0.2962   0.1413     2.0959  0.0361   **
## difdpeaa 823.9412 556.0224   1.4818  0.1384
## diahpeaa 741.1547 387.6344   1.912   0.0559   *
```

Now, we explore the explanatory strength of each feature as per the result of the regression analysis of these models:

1) Primary commodity exports: highly significant.
2) End of the Cold War: expected sign but insignificant.
3) Secondary Schooling: significant with expected sign.
4) GDP Growth: significant with expected sign.
5) Number of months since previous conflict: borderline significant with dummy variable, highly significant after dummy variable is dropped.
6) Mountainous Terrain: Marginally significant.
7) Population dispersion: Marginally significant.
8) Social fractionalization: Marginally significant.
9) Per Capita: Highly significant with negative sign.
10) Diaspora/Peace: Significant and positive sign.
11) Corrected Diaspora features: Significant with positive sign.

The only comparison across regression models is between the second and third regression models: Model 2 gives a better fit whereas Model 3 permits a slightly larger sample.

2.2.2 Grievance Model

Three models are learnt for grievance. In the first model, we exclude income inequality and land inequality due to sample size. We introduce them in the second and third model respectively. We see that ethnic fractionalization is significant, as is democracy (the more repressed the population is, the higher the chance of rebellion). The time since the previous conflict is also highly significant, while income inequality and land inequality is insignificant.

```
# Grievance Models
```

```r
filtering_columns_list <- list("warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop",
    "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,ygini",
    "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,lgini")

regression_formula_list <- list("warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia +
    "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + ygini",
    "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + lgini")


for (i in c(1:3)) {
    print(paste0("Grievance Model ", i))

    filtering_columns <- strsplit(filtering_columns_list[[i]],
        ",")[[1]]
    print(filtering_columns)
    grievance.data <- data[, filtering_columns]

    grievance.data <- na.omit(grievance.data)
    print(paste0("N : ", nrow(grievance.data)))
    print(paste0("No of wars : ", nrow(grievance.data[grievance.data$warsa ==
        1, ])))

    grievance_fit <- glm(as.formula(regression_formula_list[[i]]),
        family = binomial(link = "logit"), data = grievance.data)

    print(paste0("Pseudo R2 : ", round(PseudoR2(grievance_fit),
        digits = 2)))
    print(paste0("Log likelihood : ", round(logLik(grievance_fit),
        digits = 2)))

    print(summarize_into_table(summary(grievance_fit)),
        quote = FALSE)
}
```

```
## [1] "Grievance Model 1"
##  [1] "warsa"    "elfo"     "rf"       "pol16"    "etdo4590" "dem"
##  [7] "peace"    "mount"    "geogia"   "lnpop"
## [1] "N : 850"
## [1] "No of wars : 59"
## [1] "Pseudo R2 : 0.13"
## [1] "Log likelihood : -185.57"
##          Estimate Std. Error z value Pr(>|z|) Signif
## elfo     0.0104   0.0057     1.8068  0.0708   *
## rf       -0.0032  0.0067     -0.4723 0.6367
## pol16    -3.0675  7.0207     -0.4369 0.6622
## etdo4590 0.4136   0.4958     0.8342  0.4042
## dem      -0.1091  0.0445     -2.4552 0.0141   **
## peace    -0.0037  0.001      -3.78   0.0002   ***
## mount    0.0109   0.0068     1.6033  0.1089
## geogia   -0.5092  0.8564     -0.5946 0.5521
## lnpop    0.2215   0.0959     2.3102  0.0209   **
## [1] "Grievance Model 2"
##  [1] "warsa"    "elfo"     "rf"       "pol16"    "etdo4590" "dem"
##  [7] "peace"    "mount"    "geogia"   "lnpop"    "ygini"
```

```
## [1] "N : 604"
## [1] "No of wars : 41"
## [1] "Pseudo R2 : 0.11"
## [1] "Log likelihood : -133.46"
##          Estimate Std. Error z value Pr(>|z|) Signif
## elfo     0.0108   0.0067     1.6199  0.1053
## rf       -0.0064  0.0082     -0.7851 0.4324
## pol16    -4.6818  8.2671     -0.5663 0.5712
## etdo4590 0.5747   0.5863     0.9802  0.327
## dem      -0.0834  0.0508     -1.6398 0.1011
## peace    -0.0031  0.0012     -2.6649 0.0077   ***
## mount    0.007    0.0089     0.7911  0.4289
## geogia   -0.7632  1.0531     -0.7247 0.4686
## lnpop    0.2461   0.1188     2.071   0.0384   **
## ygini    0.0153   0.0179     0.8543  0.393
## [1] "Grievance Model 3"
##  [1] "warsa"    "elfo"     "rf"       "pol16"    "etdo4590" "dem"
##  [7] "peace"    "mount"    "geogia"   "lnpop"    "lgini"
## [1] "N : 603"
## [1] "No of wars : 38"
## [1] "Pseudo R2 : 0.17"
## [1] "Log likelihood : -117.12"
##          Estimate Std. Error z value Pr(>|z|) Signif
## elfo     0.0117   0.0084     1.3867  0.1655
## rf       -0.0037  0.0094     -0.3992 0.6897
## pol16    -6.536   8.5782     -0.7619 0.4461
## etdo4590 1.0841   0.6285     1.7249  0.0846   *
## dem      -0.1211  0.0533     -2.2728 0.023    **
## peace    -0.0044  0.0013     -3.4254 0.0006   ***
## mount    -0.0001  0.0093     -0.0083 0.9934
## geogia   -1.2926  1.1016     -1.1734 0.2406
## lnpop    0.2991   0.1331     2.2465  0.0247   **
## lgini    0.4607   1.3052     0.353   0.7241
```

2.2.3 Combined Model

The authors use 7 regression models. Let us first explore the parameters chosen in each model and the reason for doing so:

1. The first model is essentially the opportunity model with "Male secondary schooling" and no "GDP per capita" (since both cannot co-exist due to their high correlation).
2. The second model is the grievance model with all the corresponding features.
3. The third model is the combined model which contains all the features pertaining to both opportunity and grievance.
4. We remove the feature "inequality" in the fourth model because it is consistently insignificant.
5. The following insignificant features are removed in the fifth model: "post Cold War", "religious fractionalization", "democracy", "polarization", "ethnic fractionalization" and "mountainous terrain".
6. The sixth model is the same as the fifth model but with "per capita" instead of "secondary enrollment".
7. The seventh model explores various types of primary commodities and chooses features which conform well with this segregation.

```
# Combined Models

filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,grievxb",
```

```
    "warsa,peace,mount,geogia,lnpop,elfo,rf,pol16,etdo4590,dem,greedxb",
    "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem,ygini",
    "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem",
    "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590",
    "warsa,sxp,sxp2,lngdp_,gy1,peace,geogia,lnpop,frac,etdo4590",
    "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590,oilsxp,oilsxp2")

regression_formula_list <- list("warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + l
    "warsa ~  peace + mount + geogia + lnpop + elfo + rf + pol16 + etdo4590 + dem + greedxb",
    "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf + p
    "warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf + pol
    "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590",
    "warsa ~  sxp + sxp2 + lngdp_ + gy1 + peace + geogia + lnpop + frac + etdo4590",
    "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590 +  oilsxp + oilsxp2")


for (i in c(1:7)) {
    print(paste0("Combined Model ", i))

    filtering_columns <- strsplit(filtering_columns_list[[i]],
        ",")[[1]]
    print(filtering_columns)
    combined.data <- data[, filtering_columns]

    combined.data <- na.omit(combined.data)
    print(paste0("N : ", nrow(combined.data)))
    print(paste0("No of wars : ", nrow(combined.data[combined.data$warsa ==
        1, ])))

    combined_fit <- glm(as.formula(regression_formula_list[[i]]),
        family = binomial(link = "logit"), data = combined.data)

    print(paste0("Pseudo R2 : ", round(PseudoR2(combined_fit),
        digits = 2)))
    print(paste0("Log likelihood : ", round(logLik(combined_fit),
        digits = 2)))

    print(summarize_into_table(summary(combined_fit)),
        quote = FALSE)
}
```

```
## [1] "Combined Model 1"
##  [1] "warsa"   "sxp"     "sxp2"    "coldwar" "secm"    "gy1"     "peace"
##  [8] "mount"   "geogia"  "lnpop"   "frac"    "grievxb"
## [1] "N : 665"
## [1] "No of wars : 46"
## [1] "Pseudo R2 : 0.24"
## [1] "Log likelihood : -126.7"
##         Estimate Std. Error z value Pr(>|z|) Signif
## sxp     19.1073  5.9961     3.1866  0.0014   ***
## sxp2    -30.2619 12.0145    -2.5188 0.0118   **
## coldwar -0.2084  0.4572     -0.4559 0.6484
## secm    -0.0212  0.0106     -1.9984 0.0457   **
## gy1     -0.1084  0.0437     -2.4794 0.0132   **
```

```
## peace   -0.0003  0.0021     -0.1381 0.8902
## mount   0.0052   0.01       0.5198  0.6032
## geogia  -1.9761  1.0495     -1.883  0.0597   *
## lnpop   0.4886   0.1929     2.5329  0.0113   **
## frac    -0.0002  0.0001     -2.0499 0.0404   **
## grievxb 0.7648   0.4129     1.8524  0.064    *
## [1] "Combined Model 2"
##  [1] "warsa"    "peace"    "mount"    "geogia"   "lnpop"    "elfo"
##  [7] "rf"       "pol16"    "etdo4590" "dem"      "greedxb"
## [1] "N : 665"
## [1] "No of wars : 46"
## [1] "Pseudo R2 : 0.25"
## [1] "Log likelihood : -125.29"
##          Estimate Std. Error z value Pr(>|z|) Signif
## peace    0.0005   0.0014     0.3607  0.7183
## mount    0.0009   0.008      0.1123  0.9106
## geogia   0.0533   1.1006     0.0484  0.9614
## lnpop    -0.022   0.1364     -0.1614 0.8718
## elfo     0.0083   0.0071     1.1706  0.2418
## rf       -0.0048  0.0081     -0.6005 0.5482
## pol16    -9.3378  8.7336     -1.0692 0.285
## etdo4590 1.2103   0.6484     1.8666  0.062    *
## dem      -0.0365  0.0537     -0.6791 0.4971
## greedxb  1.0435   0.2106     4.9546  0        ***
## [1] "Combined Model 3"
##  [1] "warsa"    "sxp"      "sxp2"     "coldwar"  "secm"     "gy1"
##  [7] "peace"    "mount"    "geogia"   "lnpop"    "frac"     "elfo"
## [13] "rf"       "pol16"    "etdo4590" "dem"      "ygini"
## [1] "N : 479"
## [1] "No of wars : 32"
## [1] "Pseudo R2 : 0.24"
## [1] "Log likelihood : -89.55"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      37.0716  10.2919    3.602   0.0003   ***
## sxp2     -69.2696 21.6957    -3.1928 0.0014   ***
## coldwar  -0.873   0.6438     -1.356  0.1751
## secm     -0.0288  0.0133     -2.1632 0.0305   **
## gy1      -0.0455  0.0619     -0.7352 0.4622
## peace    -0.0003  0.0015     -0.2283 0.8194
## mount    0.0054   0.0116     0.4673  0.6403
## geogia   -4.0317  1.4898     -2.7061 0.0068   ***
## lnpop    0.9272   0.2501     3.707   0.0002   ***
## frac     -0.0008  0.0004     -2.2865 0.0222   **
## elfo     0.0412   0.0188     2.1902  0.0285   **
## rf       0.0148   0.0202     0.7312  0.4647
## pol16    -25.2763 13.3891    -1.8878 0.059    *
## etdo4590 2.0202   0.915      2.2079  0.0273   **
## dem      -0.0177  0.0618     -0.2861 0.7748
## ygini    0.0252   0.024      1.0507  0.2934
## [1] "Combined Model 4"
##  [1] "warsa"    "sxp"      "sxp2"     "coldwar"  "secm"     "gy1"
##  [7] "peace"    "mount"    "geogia"   "lnpop"    "frac"     "elfo"
## [13] "rf"       "pol16"    "etdo4590" "dem"
## [1] "N : 665"
```

```
## [1] "No of wars : 46"
## [1] "Pseudo R2 : 0.26"
## [1] "Log likelihood : -124.6"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      23.3851  6.6915     3.4947  0.0005   ***
## sxp2     -36.3352 12.9976    -2.7955 0.0052   ***
## coldwar  -0.2811  0.459      -0.6124 0.5403
## secm     -0.022   0.0108     -2.0372 0.0416   **
## gy1      -0.1078  0.0446     -2.4169 0.0157   **
## peace    -0.0032  0.0012     -2.7143 0.0066   ***
## mount    0.015    0.0093     1.6068  0.1081
## geogia   -1.9622  1.1491     -1.7076 0.0877   *
## lnpop    0.6974   0.1807     3.8604  0.0001   ***
## frac     -0.0005  0.0003     -1.5846 0.113
## elfo     0.0228   0.0149     1.5362  0.1245
## rf       0.0136   0.0185     0.7344  0.4627
## pol16    -15.9917 10.5178    -1.5204 0.1284
## etdo4590 1.5918   0.746      2.1338  0.0329   **
## dem      -0.0418  0.0542     -0.7706 0.441
## [1] "Combined Model 5"
##  [1] "warsa"    "sxp"      "sxp2"     "secm"     "gy1"      "peace"
##  [7] "geogia"   "lnpop"    "frac"     "etdo4590"
## [1] "N : 688"
## [1] "No of wars : 46"
## [1] "Pseudo R2 : 0.24"
## [1] "Log likelihood : -128.21"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      18.937   5.8651     3.2287  0.0012   ***
## sxp2     -29.4432 11.7813    -2.4992 0.0124   **
## secm     -0.0316  0.0098     -3.2252 0.0013   ***
## gy1      -0.1152  0.0431     -2.6753 0.0075   ***
## peace    -0.0037  0.0011     -3.397  0.0007   ***
## geogia   -2.487   1.0052     -2.4741 0.0134   **
## lnpop    0.7677   0.1658     4.6318  0        ***
## frac     -0.0002  0.0001     -2.3451 0.019    **
## etdo4590 0.6704   0.3535     1.8963  0.0579   *
## [1] "Combined Model 6"
##  [1] "warsa"    "sxp"      "sxp2"     "lngdp_"   "gy1"      "peace"
##  [7] "geogia"   "lnpop"    "frac"     "etdo4590"
## [1] "N : 750"
## [1] "No of wars : 52"
## [1] "Pseudo R2 : 0.22"
## [1] "Log likelihood : -146.84"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      16.7734  5.2064     3.2217  0.0013   ***
## sxp2     -23.8005 10.0396    -2.3707 0.0178   **
## lngdp_   -0.9504  0.2454     -3.8723 0.0001   ***
## gy1      -0.098   0.0415     -2.3625 0.0182   **
## peace    -0.0038  0.001      -3.8085 0.0001   ***
## geogia   -0.9919  0.9093     -1.0909 0.2753
## lnpop    0.5105   0.1284     3.9751  0.0001   ***
## frac     -0.0002  0.0001     -2.6953 0.007    ***
## etdo4590 0.4801   0.3283     1.4625  0.1436
## [1] "Combined Model 7"
```

```
##  [1] "warsa"    "sxp"      "sxp2"     "secm"     "gy1"      "peace"
##  [7] "geogia"   "lnpop"    "frac"     "etdo4590" "oilsxp"   "oilsxp2"
## [1] "N : 654"
## [1] "No of wars : 45"
## [1] "Pseudo R2 : 0.3"
## [1] "Log likelihood : -114.2"
##          Estimate  Std. Error z value Pr(>|z|) Signif
## sxp      50.6076   13.0928    3.8653  0.0001   ***
## sxp2     -130.9982 42.9311    -3.0514 0.0023   ***
## secm     -0.0343   0.0105     -3.2697 0.0011   ***
## gy1      -0.1335   0.0464     -2.8762 0.004    ***
## peace    -0.0032   0.0012     -2.8139 0.0049   ***
## geogia   -2.8712   1.1298     -2.5415 0.011    **
## lnpop    1.1235    0.2258     4.9752  0        ***
## frac     -0.0003   0.0001     -2.8706 0.0041   ***
## etdo4590 0.7688    0.3681     2.0886  0.0367   **
## oilsxp   -28.2748  9.3506     -3.0239 0.0025   ***
## oilsxp2  106.4589  38.7041    2.7506  0.0059   ***
```

After analyzing the effect of each feature and its explanatory power, the authors conclude that only one proxy for grievance, namely "ethnic dominance" is worth including in the combined model. Within the proxies for rebel military advantage (which is in-turn a proxy for the opportunity model), the only features which survive are "population dispersion" and "social fractionalization". The proxies for foregone earnings are deemed significant and are retained. All features related to diasporas and the Cold War are excluded.

2.3 Robustness Checks

We replicate the robustness checks with respect to any outliers in the data, and the definitions of certain dependent and independent variables.

1. We check the characteristics of the sample in which conflict occurs. We find two cases that are atypical: The events in Iran and Romania both have high secondary schooling rates as compared to other events. We remove these events from the data, and the results of the regression improves.
2. There are four events where there was highly negative economic growth before the conflict. We remove these events as well, and find that the growth rate of a country is still significant. Thus, we conclude, that conflict does not arise solely due to a sudden collapse in the economy but also through a sustained period of slow growth.
3. We also remove certain significant events as defined by Pregibon (1981), and find that the fit of the model improves and the coefficients remain significant.
4. We also need to determine whether a country has several short wars with periods of peace in between or one long war. We reclassify wars as continuous which have shorter than one month of peace and then those with peace shorter than 1 year. We do not find any significant changes to the model except that the growth rate becomes marginally insignificant.

```
# Robustness Check 1
robustness.data <- data %>% select(warsa, country,
    year, sxp, sxp2, secm, gy1, peace, geogia, frac,
    etdo4590, lnpop)

for (i in c(1:6)) {
    print(paste0("Robustness Check ", i))

    if (i == 1) {
        robustness.subdata <- robustness.data %>% filter(country !=
            "Iran") %>% filter(country != "Romania")
```

```
}

if (i == 2) {
    robustness.subdata <- robustness.data %>% filter(country !=
        "Iran") %>% filter(country != "Romania") %>%
        filter(!(country == "Angola" & year ==
            "1975")) %>% filter(!(country == "Iraq" &
        year == "1985")) %>% filter(!(country ==
        "Zaire" & year == "1995"))
}

if (i == 3) {
    robustness.subdata <- robustness.data %>% filter(!(country ==
        "Iran" & year == "1970")) %>% filter(!(country ==
        "Romania" & year == "1985")) %>% filter(!(country ==
        "Congo" & year == "1995"))
}

if (i == 4) {
    robustness.subdata <- robustness.data %>% filter(country !=
        "Saudi Arabia") %>% filter(country != "Guyana") %>%
        filter(country != "Oman") %>% filter(country !=
        "Trinidad and Tobago")
}

if (i == 5) {
    robustness.subdata <- robustness.data %>% filter(!(country ==
        "Angola" & year == "1975")) %>% filter(!(country ==
        "Somalia" & year == "1985"))
}

if (i == 6) {
    robustness.subdata <- robustness.data %>% filter(!(country ==
        "Angola" & year == "1975")) %>% filter(!(country ==
        "Somalia" & year == "1985")) %>% filter(!(country ==
        "Mozambique" & year == "1975")) %>% filter(!(country ==
        "Sierra Leone" & year == "1995")) %>% filter(!(country ==
        "Zaire" & year == "1995"))
}

robustness.1.data <- na.omit(robustness.subdata)
print(paste0("N : ", nrow(robustness.subdata)))
print(paste0("No of wars : ", nrow(robustness.subdata[robustness.subdata$warsa ==
    1, ])))

robustness.subdata <- glm(warsa ~ sxp + sxp2 +
    secm + gy1 + peace + geogia + frac + etdo4590 +
    lnpop, family = binomial(link = "logit"), data = robustness.1.data)

print(paste0("Pseudo R2 : ", round(PseudoR2(robustness.subdata),
    digits = 2)))
print(paste0("Log likelihood : ", round(logLik(robustness.subdata),
    digits = 2)))
```

```
    print(summarize_into_table(summary(robustness.subdata)),
        quote = FALSE)
}
```

```
## [1] "Robustness Check 1"
## [1] "N : 1272"
## [1] "No of wars : 195"
## [1] "Pseudo R2 : 0.25"
## [1] "Log likelihood : -118.4"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      19.6965  6.6069     2.9812  0.0029   ***
## sxp2     -34.0902 14.3523    -2.3752 0.0175   **
## secm     -0.0345  0.0109     -3.1699 0.0015   ***
## gy1      -0.1398  0.0468     -2.9878 0.0028   ***
## peace    -0.0037  0.0012     -3.1601 0.0016   ***
## geogia   -2.1135  1.0796     -1.9577 0.0503   *
## frac     -0.0002  0.0001     -2.0407 0.0413   **
## etdo4590 0.727    0.3678     1.9766  0.0481   **
## lnpop    0.7472   0.1739     4.2968  0        ***
## [1] "Robustness Check 2"
## [1] "N : 1269"
## [1] "No of wars : 192"
## [1] "Pseudo R2 : 0.22"
## [1] "Log likelihood : -116.17"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      19.0288  6.6699     2.8529  0.0043   ***
## sxp2     -33.2498 14.604     -2.2768 0.0228   **
## secm     -0.0372  0.0112     -3.3188 0.0009   ***
## gy1      -0.0996  0.0518     -1.9247 0.0543   *
## peace    -0.0032  0.0012     -2.6524 0.008    ***
## geogia   -2.2718  1.09       -2.0842 0.0371   **
## frac     -0.0002  0.0001     -2.0445 0.0409   **
## etdo4590 0.7315   0.3695     1.9797  0.0477   **
## lnpop    0.7429   0.175      4.2456  0        ***
## [1] "Robustness Check 3"
## [1] "N : 1285"
## [1] "No of wars : 196"
## [1] "Pseudo R2 : 0.29"
## [1] "Log likelihood : -114.04"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      28.745   7.8622     3.6561  0.0003   ***
## sxp2     -54.818  17.7813    -3.0829 0.002    ***
## secm     -0.0415  0.0114     -3.626  0.0003   ***
## gy1      -0.1375  0.0459     -2.9965 0.0027   ***
## peace    -0.0041  0.0012     -3.5129 0.0004   ***
## geogia   -2.8902  1.1361     -2.5439 0.011    **
## frac     -0.0003  0.0001     -2.7478 0.006    ***
## etdo4590 0.6554   0.3717     1.7633  0.0778   *
## lnpop    0.8986   0.1955     4.5969  0        ***
## [1] "Robustness Check 4"
## [1] "N : 1256"
## [1] "No of wars : 199"
## [1] "Pseudo R2 : 0.24"
```

```
## [1] "Log likelihood : -127.55"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      18.7714  6.0632     3.0959  0.002    ***
## sxp2     -28.4663 12.2989    -2.3145 0.0206   **
## secm     -0.0312  0.0098     -3.1933 0.0014   ***
## gy1      -0.1215  0.044      -2.7622 0.0057   ***
## peace    -0.0036  0.0011     -3.2808 0.001    ***
## geogia   -2.4491  1.0079     -2.4299 0.0151   **
## frac     -0.0002  0.0001     -2.3871 0.017    **
## etdo4590 0.6468   0.3536     1.8293  0.0674   *
## lnpop    0.7722   0.1682     4.5904  0        ***
## [1] "Robustness Check 5"
## [1] "N : 1286"
## [1] "No of wars : 197"
## [1] "Pseudo R2 : 0.23"
## [1] "Log likelihood : -126.33"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      19.1466  5.9393     3.2237  0.0013   ***
## sxp2     -30.1497 12.0316    -2.5059 0.0122   **
## secm     -0.0306  0.0097     -3.1538 0.0016   ***
## gy1      -0.1024  0.0441     -2.3198 0.0204   **
## peace    -0.0034  0.0011     -3.0647 0.0022   ***
## geogia   -2.5411  1.0115     -2.5121 0.012    **
## frac     -0.0002  0.0001     -2.2107 0.0271   **
## etdo4590 0.7318   0.3573     2.0484  0.0405   **
## lnpop    0.7822   0.167      4.6841  0        ***
## [1] "Robustness Check 6"
## [1] "N : 1283"
## [1] "No of wars : 194"
## [1] "Pseudo R2 : 0.21"
## [1] "Log likelihood : -124.2"
##          Estimate Std. Error z value Pr(>|z|) Signif
## sxp      19.7632  5.9958     3.2962  0.001    ***
## sxp2     -31.0063 12.0769    -2.5674 0.0102   **
## secm     -0.0308  0.0097     -3.1734 0.0015   ***
## gy1      -0.0795  0.0458     -1.7345 0.0828   *
## peace    -0.0031  0.0011     -2.7503 0.006    ***
## geogia   -2.7197  1.0203     -2.6656 0.0077   ***
## frac     -0.0002  0.0001     -2.4131 0.0158   **
## etdo4590 0.6984   0.3602     1.9389  0.0525   *
## lnpop    0.7949   0.1689     4.7069  0        ***
```

3. Challenges

The challenges we faced during replication are as follows:

1) The paper explores Estimation issues during the robustness checks which include random effects, fixed effects, time effects, and rare events logits. The paper doesn't mention which variables are used to induce this randomness, and thus could not be replicated.
2) Measurements are taken at each 5 year period, however during robustness checks some conflicts fall in the middle of a period, and thus we need to decide which five year period to take into account. We do this through trial and error to arrive at the same results as the paper.

4. Extending the original study

4.1 In sample testing

In sample testing is done for all the models (opportunity, grievance and combined). The Specificity, Sensitivity, AUC, and Accuracy are recorded at thresholds 0.1 and 0.5

```
all_tests <- matrix(data = 0, nrow = 15, ncol = 5)
colnames(all_tests) <- c("model", "sens", "spec", "auc",
    "accuracy")

thresholding_flag = TRUE
sens_index <- 2
spec_index <- 3
auc_index <- 4
accuracy_index <- 5
all_tests_index <- 1
```

4.1.1 Opportunity Model testing

```
# Opportunity Models
opportunity_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,prevwara,mount,geogia,frac,lnp
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,frac,lnpop",
        "warsa,sxp,sxp2,coldwar,lngdp_,gy1,peace,mount,geogia,frac,lnpop",
        "warsa,sxp,sxp2,lngdp_,peace,lnpop,diaspeaa",
        "warsa,sxp,sxp2,lngdp_,peace,lnpop,difdpeaa,diahpeaa")

    regression_formula_list <- list("warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + prevwara + mou
        "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + frac + lnpop",
        "warsa ~  sxp + sxp2 + coldwar + lngdp_ + gy1 + peace + mount + geogia + frac + lnpop",
        "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + diaspeaa",
        "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + difdpeaa + diahpeaa")


    for (i in c(1:5)) {

        filtering_columns <- strsplit(filtering_columns_list[[i]],
            ",")[[1]]

        opportunity.data <- data[, filtering_columns]

        testData <- opportunity.data
        trainData <- opportunity.data

        # trainData <- na.omit(trainData) testData <-
        # na.omit(testData)

        opportunity_fit <- glm(as.formula(regression_formula_list[[i]]),
            family = binomial(link = "logit"), data = trainData)

        opportunity_predict <- predict(opportunity_fit,
```

```
        newdata = testData, type = "response")

    opportunity_y.hat <- as.matrix(opportunity_predict)

    y <- as.matrix(testData$warsa)

    all_tests[all_tests_index, 1] <- paste(c("opportunity",
        i), collapse = ".")
    if (thresholding_flag == TRUE) {

        opportunity_y.hat_normalized <- opportunity_y.hat

        opportunity_y.hat_normalized[opportunity_y.hat_normalized >=
            threshold_value] <- 1
        opportunity_y.hat_normalized[opportunity_y.hat_normalized <
            threshold_value] <- 0

        opp_predict_normalized <- prediction(opportunity_y.hat_normalized,
            y)

        len <- length(opp_predict_normalized@fp[[1]])
        fp <- as.numeric(opp_predict_normalized@fp[[1]][[len -
            1]])
        tp <- as.numeric(opp_predict_normalized@tp[[1]][[len -
            1]])
        fn <- as.numeric(opp_predict_normalized@fn[[1]][[len -
            1]])
        tn <- as.numeric(opp_predict_normalized@tn[[1]][[len -
            1]])

        all_tests[all_tests_index, sens_index] <- tp/(tp +
            fn)
        all_tests[all_tests_index, spec_index] <- tn/(tn +
            fp)

        all_tests[all_tests_index, accuracy_index] <- (tp +
            tn)/(tp + tn + fp + fn)

        opp_predict <- prediction(opportunity_y.hat,
            y)
        opp_auc <- performance(opp_predict, measure = "auc")
        all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(opp_auc,
            "y.values")))
        all_tests_index <- all_tests_index + 1
    } else {

        opp_predict <- prediction(opportunity_y.hat,
            y)

        opp_f <- performance(opp_predict, measure = "f")
        opp_where.F <- which.max(as.numeric(unlist(slot(opp_f,
            "y.values"))))
        opp_what.F <- performance(opp_predict,
```

```
                measure = "sens", x.measure = "spec")

            all_tests[all_tests_index, sens_index] <- as.numeric(unlist(slot(opp_what.F,
                "y.values")))[opp_where.F]
            all_tests[all_tests_index, spec_index] <- as.numeric(unlist(slot(opp_what.F,
                "x.values")))[opp_where.F]

            opp_auc <- performance(opp_predict, measure = "auc")
            all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(opp_auc,
                "y.values")))
            all_tests_index <- all_tests_index + 1
        }
    }
    return(all_tests)
}
temp1_1 <- opportunity_model(0.1)
temp1_2 <- opportunity_model(0.5)
print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp1_1[1:5, ])
```

```
##      model           sens                spec
## [1,] "opportunity.1" "0.717391304347826" "0.828660436137072"
## [2,] "opportunity.2" "0.717391304347826" "0.827102803738318"
## [3,] "opportunity.3" "0.692307692307692" "0.805157593123209"
## [4,] "opportunity.4" "0.5625"            "0.873889875666075"
## [5,] "opportunity.5" "0.5625"            "0.875666074600355"
##      auc                 accuracy
## [1,] "0.853988893403763" "0.821220930232558"
## [2,] "0.855817418393604" "0.819767441860465"
## [3,] "0.836014987877453" "0.797333333333333"
## [4,] "0.86101243339254"  "0.857142857142857"
## [5,] "0.860956927175844" "0.858823529411765"
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp1_2[1:5, ])
```

```
##      model           sens                 spec
## [1,] "opportunity.1" "0.0652173913043478" "0.995327102803738"
## [2,] "opportunity.2" "0.0652173913043478" "0.995327102803738"
## [3,] "opportunity.3" "0.0769230769230769" "0.995702005730659"
## [4,] "opportunity.4" "0.0625"             "0.998223801065719"
## [5,] "opportunity.5" "0.09375"            "0.998223801065719"
##      auc                 accuracy
## [1,] "0.853988893403763" "0.933139534883721"
## [2,] "0.855817418393604" "0.933139534883721"
## [3,] "0.836014987877453" "0.932"
## [4,] "0.86101243339254"  "0.947899159663866"
## [5,] "0.860956927175844" "0.949579831932773"
```

4.1.2 Grievance Model testing

```
# Grievance Models
grievance_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop",
        "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,ygini",
        "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,lgini")

    regression_formula_list <- list("warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogi
        "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + ygini",
        "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + lgini")


    for (i in c(1:3)) {

        filtering_columns <- strsplit(filtering_columns_list[[i]],
            ",")[[1]]

        grievance.data <- data[, filtering_columns]

        testData <- grievance.data
        trainData <- grievance.data

        # trainData <- na.omit(trainData) testData <-
        # na.omit(testData)

        grievance_fit <- glm(as.formula(regression_formula_list[[i]]),
            family = binomial(link = "logit"), data = trainData)

        grievance_predict <- predict(grievance_fit,
            newdata = testData, type = "response")

        grievance_y.hat <- as.matrix(grievance_predict)

        y <- as.matrix(testData$warsa)

        all_tests[all_tests_index, 1] <- paste(c("grievance",
            i), collapse = ".")

        if (thresholding_flag == TRUE) {
            grievance_y.hat_normalized <- grievance_y.hat

            grievance_y.hat_normalized[grievance_y.hat_normalized >=
                threshold_value] <- 1
            grievance_y.hat_normalized[grievance_y.hat_normalized <
                threshold_value] <- 0

            griev_predict_normalized <- prediction(grievance_y.hat_normalized,
                y)

            len <- length(griev_predict_normalized@fp[[1]])
            fp <- as.numeric(griev_predict_normalized@fp[[1]][[len -
                1]])
```

```
            tp <- as.numeric(griev_predict_normalized@tp[[1]][[len -
                1]])
            fn <- as.numeric(griev_predict_normalized@fn[[1]][[len -
                1]])
            tn <- as.numeric(griev_predict_normalized@tn[[1]][[len -
                1]])

            all_tests[all_tests_index, sens_index] <- tp/(tp +
                fn)
            all_tests[all_tests_index, spec_index] <- tn/(tn +
                fp)

            all_tests[all_tests_index, accuracy_index] <- (tp +
                tn)/(tp + tn + fp + fn)

            griev_predict <- prediction(grievance_y.hat,
                y)
            griev_auc <- performance(griev_predict,
                measure = "auc")
            all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(griev_auc,
                "y.values")))
            all_tests_index <- all_tests_index + 1
        } else {

            griev_predict <- prediction(grievance_y.hat,
                y)

            griev_f <- performance(griev_predict, measure = "f")
            griev_where.F <- which.max(as.numeric(unlist(slot(griev_f,
                "y.values"))))
            griev_what.F <- performance(griev_predict,
                measure = "sens", x.measure = "spec")

            all_tests[all_tests_index, sens_index] <- as.numeric(unlist(slot(griev_what.F,
                "y.values")))[griev_where.F]
            all_tests[all_tests_index, spec_index] <- as.numeric(unlist(slot(griev_what.F,
                "x.values")))[griev_where.F]

            griev_auc <- performance(griev_predict,
                measure = "auc")
            all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(griev_auc,
                "y.values")))
            all_tests_index <- all_tests_index + 1
        }

    }
    return(all_tests)
}
temp2_1 <- grievance_model(0.1)
temp2_2 <- grievance_model(0.5)

print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp2_1[1:3, ])
```

```
##      model         sens                spec
## [1,] "grievance.1" "0.644067796610169" "0.806573957016435"
## [2,] "grievance.2" "0.560975609756098" "0.793960923623446"
## [3,] "grievance.3" "0.684210526315789" "0.847787610619469"
##      auc                 accuracy
## [1,] "0.777411129443529" "0.795294117647059"
## [2,] "0.759260061517131" "0.778145695364238"
## [3,] "0.806474149976711" "0.837479270315091"
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp2_2[1:3, ])
```

```
##      model         sens spec auc                 accuracy
## [1,] "grievance.1" "0"  "1"  "0.777411129443529" "0.930588235294118"
## [2,] "grievance.2" "0"  "1"  "0.759260061517131" "0.932119205298013"
## [3,] "grievance.3" "0"  "1"  "0.806474149976711" "0.93698175787728"
```

4.1.3 Combined Model testing

Generating the combined opportunity and grievance models

```
# Combined Models
combined_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,grievi
        "warsa,peace,mount,geogia,lnpop,elfo,rf,pol16,etdo4590,dem,greedxb",
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem,ygini
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem",
        "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590",
        "warsa,sxp,sxp2,lngdp_,gy1,peace,geogia,lnpop,frac,etdo4590",
        "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590,oilsxp,oilsxp2")

    regression_formula_list <- list("warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia
        "warsa ~  peace + mount + geogia + lnpop + elfo + rf + pol16 + etdo4590 + dem + greedxb",
        "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf
        "warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf
        "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590",
        "warsa ~  sxp + sxp2 + lngdp_ + gy1 + peace + geogia + lnpop + frac + etdo4590",
        "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590 +  oilsxp + oilsxp


    for (i in c(1:7)) {

        filtering_columns <- strsplit(filtering_columns_list[[i]],
            ",")[[1]]

        combined.data <- data[, filtering_columns]

        testData <- combined.data
        trainData <- combined.data
```

```
        # trainData <- na.omit(trainData) testData <-
        # na.omit(testData)

        combined_fit <- glm(as.formula(regression_formula_list[[i]]),
            family = binomial(link = "logit"), data = trainData)

        combined_predict <- predict(combined_fit, newdata = testData,
            type = "response")

        combined_y.hat <- as.matrix(combined_predict)

        y <- as.matrix(testData$warsa)

        all_tests[all_tests_index, 1] <- paste(c("combined",
            i), collapse = ".")

        if (thresholding_flag == TRUE) {
            combined_y.hat_normalized <- combined_y.hat

            combined_y.hat_normalized[combined_y.hat_normalized >=
                threshold_value] <- 1
            combined_y.hat_normalized[combined_y.hat_normalized <
                threshold_value] <- 0

            comb_predict_normalized <- prediction(combined_y.hat_normalized,
                y)

            len <- length(comb_predict_normalized@fp[[1]])
            fp <- as.numeric(comb_predict_normalized@fp[[1]][[len -
                1]])
            tp <- as.numeric(comb_predict_normalized@tp[[1]][[len -
                1]])
            fn <- as.numeric(comb_predict_normalized@fn[[1]][[len -
                1]])
            tn <- as.numeric(comb_predict_normalized@tn[[1]][[len -
                1]])

            all_tests[all_tests_index, sens_index] <- tp/(tp +
                fn)
            all_tests[all_tests_index, spec_index] <- tn/(tn +
                fp)

            all_tests[all_tests_index, accuracy_index] <- (tp +
                tn)/(tp + tn + fp + fn)

            comb_predict <- prediction(combined_y.hat,
                y)
            comb_auc <- performance(comb_predict, measure = "auc")
            all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(comb_auc,
                "y.values")))
            all_tests_index <- all_tests_index + 1
        } else {
```

```
            comb_predict <- prediction(combined_y.hat,
                y)

            comb_f <- performance(comb_predict, measure = "f")
            comb_where.F <- which.max(as.numeric(unlist(slot(comb_f,
                "y.values"))))
            comb_what.F <- performance(comb_predict,
                measure = "sens", x.measure = "spec")

            all_tests[all_tests_index, sens_index] <- as.numeric(unlist(slot(comb_what.F,
                "y.values")))[comb_where.F]
            all_tests[all_tests_index, spec_index] <- as.numeric(unlist(slot(comb_what.F,
                "x.values")))[comb_where.F]

            comb_auc <- performance(comb_predict, measure = "auc")
            all_tests[all_tests_index, auc_index] <- as.numeric(unlist(slot(comb_auc,
                "y.values")))
            all_tests_index <- all_tests_index + 1
        }
    }
    return(all_tests)
}
temp3_1 <- combined_model(0.1)
temp3_2 <- combined_model(0.5)

print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp3_1[1:7, ])
```

```
##      model        sens                spec
## [1,] "combined.1" "0.760869565217391" "0.810985460420032"
## [2,] "combined.2" "0.717391304347826" "0.804523424878837"
## [3,] "combined.3" "0.71875"           "0.803131991051454"
## [4,] "combined.4" "0.760869565217391" "0.807754442649435"
## [5,] "combined.5" "0.739130434782609" "0.828660436137072"
## [6,] "combined.6" "0.673076923076923" "0.820916905444126"
## [7,] "combined.7" "0.822222222222222" "0.824302134646962"
##      auc                 accuracy
## [1,] "0.85727330195968"  "0.807518796992481"
## [2,] "0.860188241904898" "0.798496240601504"
## [3,] "0.849832214765098" "0.797494780793319"
## [4,] "0.861417433448061" "0.804511278195489"
## [5,] "0.859779222538261" "0.822674418604651"
## [6,] "0.835794577914921" "0.810666666666667"
## [7,] "0.894252873563215" "0.824159021406728"
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp3_2[1:7, ])
```

```
##      model        sens                spec
## [1,] "combined.1" "0.0869565217391304" "0.995153473344103"
```

```
## [2,] "combined.2" "0.0869565217391304" "0.993537964458804"
## [3,] "combined.3" "0.03125"            "1"
## [4,] "combined.4" "0.0869565217391304" "0.991922455573506"
## [5,] "combined.5" "0.0652173913043478" "0.992211838006231"
## [6,] "combined.6" "0.0769230769230769" "0.995702005730659"
## [7,] "combined.7" "0.0888888888888889" "0.988505747126437"
##      auc                 accuracy
## [1,] "0.85727330195968"  "0.932330827067669"
## [2,] "0.860188241904898" "0.930827067669173"
## [3,] "0.849832214765098" "0.935281837160752"
## [4,] "0.861417433448061" "0.929323308270677"
## [5,] "0.859779222538261" "0.930232558139535"
## [6,] "0.835794577914921" "0.932"
## [7,] "0.894252873563215" "0.926605504587156"
```

```
# print('Results for threshold = 0.1')

# print('Opportunity Models') print(temp1_1)
# print('Grievance Models') print(temp2_1)
# print('Combined Models') print(temp3_1)
# print('Results for threshold = 0.5')
# print('Opportunity Models') print(temp1_2)
# print('Grievance Models') print(temp2_2)
# print('Combined Models') print(temp3_2)
```

4.2 Out-sample testing

Out sample testing is done for all the models (opportunity, grievance and combined). The Specificity, Sensitivity, AUC, and Accuracy are recorded at thresholds 0.1 and 0.5. 5-fold crossvalidation is used.

```
k <- 5

all_tests <- matrix(data = 0, nrow = 15 * k, ncol = 6)
colnames(all_tests) <- c("model", "test_index", "sens",
    "spec", "auc", "accuracy")

set.seed(42)
shuffled_data <- data[sample(nrow(data)), ]
folds <- cut(seq(1, nrow(shuffled_data)), breaks = k,
    labels = FALSE)

thresholding_flag = TRUE
# threshold_value = as.numeric(params$threshold)

sens_index <- 3
spec_index <- 4
auc_index <- 5
accuracy_index <- 6
```

4.2.1 Opportunity Model testing

```r
# Opportunity Models
opportunity_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,prevwara,mount,geogia,frac,lnp
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,frac,lnpop",
        "warsa,sxp,sxp2,coldwar,lngdp_,gy1,peace,mount,geogia,frac,lnpop",
        "warsa,sxp,sxp2,lngdp_,peace,lnpop,diaspeaa",
        "warsa,sxp,sxp2,lngdp_,peace,lnpop,difdpeaa,diahpeaa")

    regression_formula_list <- list("warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + prevwara + mou
        "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + frac + lnpop",
        "warsa ~  sxp + sxp2 + coldwar + lngdp_ + gy1 + peace + mount + geogia + frac + lnpop",
        "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + diaspeaa",
        "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + difdpeaa + diahpeaa")


    for (i in c(1:5)) {

        for (testIndex in 1:k) {

            filtering_columns <- strsplit(filtering_columns_list[[i]],
                ",")[[1]]

            opportunity.data <- shuffled_data[, filtering_columns]

            # Segement your data by fold using the which()
            # function
            testIndexes <- which(folds == testIndex,
                arr.ind = TRUE)
            testData <- opportunity.data[testIndexes,
                ]
            trainData <- opportunity.data[-testIndexes,
                ]

            # trainData <- na.omit(trainData) testData <-
            # na.omit(testData)

            opportunity_fit <- glm(as.formula(regression_formula_list[[i]]),
                family = binomial(link = "logit"),
                data = trainData)

            opportunity_predict <- predict(opportunity_fit,
                newdata = testData, type = "response")

            opportunity_y.hat <- as.matrix(opportunity_predict)

            y <- as.matrix(testData$warsa)

            all_tests[(i - 1) * k + testIndex, 1] <- paste(c("opportunity",
                i), collapse = ".")
            all_tests[(i - 1) * k + testIndex, 2] <- as.numeric(testIndex)

            if (thresholding_flag == TRUE) {
```

```
        opportunity_y.hat_normalized <- opportunity_y.hat

        opportunity_y.hat_normalized[opportunity_y.hat_normalized >=
          threshold_value] <- 1
        opportunity_y.hat_normalized[opportunity_y.hat_normalized <
          threshold_value] <- 0

        opp_predict_normalized <- prediction(opportunity_y.hat_normalized,
          y)

        len <- length(opp_predict_normalized@fp[[1]])
        fp <- as.numeric(opp_predict_normalized@fp[[1]][[len -
          1]])
        tp <- as.numeric(opp_predict_normalized@tp[[1]][[len -
          1]])
        fn <- as.numeric(opp_predict_normalized@fn[[1]][[len -
          1]])
        tn <- as.numeric(opp_predict_normalized@tn[[1]][[len -
          1]])

        all_tests[(i - 1) * k + testIndex,
          sens_index] <- tp/(tp + fn)
        all_tests[(i - 1) * k + testIndex,
          spec_index] <- tn/(tn + fp)

        all_tests[(i - 1) * k + testIndex,
          accuracy_index] <- (tp + tn)/(tp +
          tn + fp + fn)

        opp_predict <- prediction(opportunity_y.hat,
          y)
        opp_auc <- performance(opp_predict,
          measure = "auc")
        all_tests[(i - 1) * k + testIndex,
          auc_index] <- as.numeric(unlist(slot(opp_auc,
          "y.values")))

      } else {

        opp_predict <- prediction(opportunity_y.hat,
          y)

        opp_f <- performance(opp_predict, measure = "f")
        opp_where.F <- which.max(as.numeric(unlist(slot(opp_f,
          "y.values"))))
        opp_what.F <- performance(opp_predict,
          measure = "sens", x.measure = "spec")

        all_tests[(i - 1) * k + testIndex,
          sens_index] <- as.numeric(unlist(slot(opp_what.F,
          "y.values")))[opp_where.F]
        all_tests[(i - 1) * k + testIndex,
          spec_index] <- as.numeric(unlist(slot(opp_what.F,
```

```
                  "x.values")))[opp_where.F]

                opp_auc <- performance(opp_predict,
                  measure = "auc")
                all_tests[(i - 1) * k + testIndex,
                  auc_index] <- as.numeric(unlist(slot(opp_auc,
                  "y.values")))

            }

        }

    }

    lower_lim = 1
    upper_lim = 5 * k
    return(convert2dArrayToDf(all_tests[lower_lim:upper_lim,
        1:6]))
}
temp1_1 <- opportunity_model(0.1)
temp1_2 <- opportunity_model(0.5)

print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp1_1)
```

```
##            model test_index   sens   spec    auc accuracy
## 1  opportunity.1          1 0.5000 0.8583 0.7291   0.8321
## 2  opportunity.1          2 0.5000 0.8760 0.9109   0.8647
## 3  opportunity.1          3 0.8182 0.8370 0.8343   0.8356
## 4  opportunity.1          4 0.7000 0.7712 0.8390   0.7656
## 5  opportunity.1          5 0.4545 0.7895 0.7949   0.7639
## 6  opportunity.2          1 0.5000 0.8504 0.7394   0.8248
## 7  opportunity.2          2 0.5000 0.8760 0.9089   0.8647
## 8  opportunity.2          3 0.8182 0.8370 0.8350   0.8356
## 9  opportunity.2          4 0.7000 0.7712 0.8347   0.7656
## 10 opportunity.2          5 0.6364 0.7820 0.8373   0.7708
## 11 opportunity.3          1 0.5000 0.8357 0.7196   0.8092
## 12 opportunity.3          2 1.0000 0.8686 0.9387   0.8732
## 13 opportunity.3          3 0.8182 0.8041 0.8243   0.8050
## 14 opportunity.3          4 0.7273 0.7099 0.7793   0.7113
## 15 opportunity.3          5 0.6154 0.7746 0.8386   0.7613
## 16 opportunity.4          1 0.4286 0.9126 0.8322   0.8818
## 17 opportunity.4          2 0.5000 0.8852 0.7664   0.8730
## 18 opportunity.4          3 0.8000 0.8376 0.8855   0.8361
## 19 opportunity.4          4 0.3750 0.8800 0.6650   0.8426
## 20 opportunity.4          5 0.5000 0.8347 0.8533   0.8140
## 21 opportunity.5          1 0.2857 0.9223 0.8044   0.8818
## 22 opportunity.5          2 0.5000 0.8852 0.7664   0.8730
## 23 opportunity.5          3 0.8000 0.8376 0.8855   0.8361
## 24 opportunity.5          4 0.3750 0.8700 0.6325   0.8333
## 25 opportunity.5          5 0.5000 0.8347 0.8543   0.8140
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp1_2)
```

```
##            model test_index    sens   spec    auc accuracy
## 1  opportunity.1          1 0.00000 0.9843 0.7291   0.9124
## 2  opportunity.1          2 0.00000 1.0000 0.9109   0.9699
## 3  opportunity.1          3 0.09091 0.9852 0.8343   0.9178
## 4  opportunity.1          4 0.10000 0.9915 0.8390   0.9219
## 5  opportunity.1          5 0.09091 0.9925 0.7949   0.9236
## 6  opportunity.2          1 0.00000 0.9843 0.7394   0.9124
## 7  opportunity.2          2 0.00000 1.0000 0.9089   0.9699
## 8  opportunity.2          3 0.09091 0.9852 0.8350   0.9178
## 9  opportunity.2          4 0.10000 0.9915 0.8347   0.9219
## 10 opportunity.2          5 0.09091 0.9850 0.8373   0.9167
## 11 opportunity.3          1 0.16667 0.9857 0.7196   0.9211
## 12 opportunity.3          2 0.20000 1.0000 0.9387   0.9718
## 13 opportunity.3          3 0.09091 0.9932 0.8243   0.9308
## 14 opportunity.3          4 0.09091 0.9924 0.7793   0.9225
## 15 opportunity.3          5 0.00000 0.9930 0.8386   0.9097
## 16 opportunity.4          1 0.14286 0.9903 0.8322   0.9364
## 17 opportunity.4          2 0.25000 1.0000 0.7664   0.9762
## 18 opportunity.4          3 0.00000 0.9829 0.8855   0.9426
## 19 opportunity.4          4 0.00000 1.0000 0.6650   0.9259
## 20 opportunity.4          5 0.00000 1.0000 0.8533   0.9380
## 21 opportunity.5          1 0.14286 1.0000 0.8044   0.9455
## 22 opportunity.5          2 0.25000 1.0000 0.7664   0.9762
## 23 opportunity.5          3 0.00000 0.9829 0.8855   0.9426
## 24 opportunity.5          4 0.00000 1.0000 0.6325   0.9259
## 25 opportunity.5          5 0.00000 1.0000 0.8543   0.9380
```

4.2.2 Grievance Model testing

```
# Grievance Models Grievance Models
grievance_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop",
        "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,ygini",
        "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,lgini")

    regression_formula_list <- list("warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geog
        "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + ygini",
        "warsa ~  elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + lgini")


    for (i in c(1:3)) {

        for (testIndex in 1:k) {

            filtering_columns <- strsplit(filtering_columns_list[[i]],
                ",")[[1]]
```

```
        grievance.data <- shuffled_data[, filtering_columns]

        # Segement your data by fold using the which()
        # function
        testIndexes <- which(folds == testIndex,
            arr.ind = TRUE)
        testData <- grievance.data[testIndexes,
            ]
        trainData <- grievance.data[-testIndexes,
            ]

        # trainData <- na.omit(trainData) testData <-
        # na.omit(testData)

        grievance_fit <- glm(as.formula(regression_formula_list[[i]]),
            family = binomial(link = "logit"),
            data = trainData)

        grievance_predict <- predict(grievance_fit,
            newdata = testData, type = "response")

        grievance_y.hat <- as.matrix(grievance_predict)

        y <- as.matrix(testData$warsa)

        all_tests[5 * k + (i - 1) * k + testIndex,
            1] <- paste(c("grievance", i), collapse = ".")
        all_tests[5 * k + (i - 1) * k + testIndex,
            2] <- as.numeric(testIndex)

        if (thresholding_flag == TRUE) {
            grievance_y.hat_normalized <- grievance_y.hat

            grievance_y.hat_normalized[grievance_y.hat_normalized >=
              threshold_value] <- 1
            grievance_y.hat_normalized[grievance_y.hat_normalized <
              threshold_value] <- 0

            griev_predict_normalized <- prediction(grievance_y.hat_normalized,
              y)

            len <- length(griev_predict_normalized@fp[[1]])
            fp <- as.numeric(griev_predict_normalized@fp[[1]][[len -
              1]])
            tp <- as.numeric(griev_predict_normalized@tp[[1]][[len -
              1]])
            fn <- as.numeric(griev_predict_normalized@fn[[1]][[len -
              1]])
            tn <- as.numeric(griev_predict_normalized@tn[[1]][[len -
              1]])

            all_tests[5 * k + (i - 1) * k + testIndex,
              sens_index] <- tp/(tp + fn)
```

```
                all_tests[5 * k + (i - 1) * k + testIndex,
                  spec_index] <- tn/(tn + fp)

                all_tests[5 * k + (i - 1) * k + testIndex,
                  accuracy_index] <- (tp + tn)/(tp +
                  tn + fp + fn)

                griev_predict <- prediction(grievance_y.hat,
                  y)
                griev_auc <- performance(griev_predict,
                  measure = "auc")
                all_tests[5 * k + (i - 1) * k + testIndex,
                  auc_index] <- as.numeric(unlist(slot(griev_auc,
                  "y.values")))

            } else {

                griev_predict <- prediction(grievance_y.hat,
                  y)

                griev_f <- performance(griev_predict,
                  measure = "f")
                griev_where.F <- which.max(as.numeric(unlist(slot(griev_f,
                  "y.values"))))
                griev_what.F <- performance(griev_predict,
                  measure = "sens", x.measure = "spec")

                all_tests[5 * k + (i - 1) * k + testIndex,
                  sens_index] <- as.numeric(unlist(slot(griev_what.F,
                  "y.values")))[griev_where.F]
                all_tests[5 * k + (i - 1) * k + testIndex,
                  spec_index] <- as.numeric(unlist(slot(griev_what.F,
                  "x.values")))[griev_where.F]

                griev_auc <- performance(griev_predict,
                  measure = "auc")
                all_tests[5 * k + (i - 1) * k + testIndex,
                  auc_index] <- as.numeric(unlist(slot(griev_auc,
                  "y.values")))

            }

        }

    }

    lower_lim = 5 * k + 1
    upper_lim = 5 * k + 3 * k
    return(convert2dArrayToDf(all_tests[lower_lim:upper_lim,
        1:6]))
}
temp2_1 <- grievance_model(0.1)
temp2_2 <- grievance_model(0.5)
```

```
print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp2_1)
```

```
##          model test_index   sens   spec    auc accuracy
## 1  grievance.1          1 0.3077 0.8408 0.6316   0.8000
## 2  grievance.1          2 0.8000 0.8129 0.8877   0.8125
## 3  grievance.1          3 0.7500 0.7758 0.7758   0.7740
## 4  grievance.1          4 0.6429 0.7792 0.7389   0.7679
## 5  grievance.1          5 0.7333 0.8250 0.7617   0.8171
## 6  grievance.2          1 0.2500 0.8333 0.5868   0.7931
## 7  grievance.2          2 0.2500 0.7778 0.7986   0.7589
## 8  grievance.2          3 0.5714 0.7586 0.6638   0.7480
## 9  grievance.2          4 0.4615 0.7826 0.7090   0.7500
## 10 grievance.2          5 0.4444 0.7931 0.6868   0.7680
## 11 grievance.3          1 0.2500 0.9083 0.6778   0.8632
## 12 grievance.3          2 0.0000 0.8468 0.8018   0.8319
## 13 grievance.3          3 0.6667 0.7647 0.8193   0.7600
## 14 grievance.3          4 0.6364 0.8624 0.7665   0.8417
## 15 grievance.3          5 0.6364 0.8462 0.7855   0.8281
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp2_2)
```

```
##          model test_index sens   spec    auc accuracy
## 1  grievance.1          1    0 1.0000 0.6316   0.9235
## 2  grievance.1          2    0 1.0000 0.8877   0.9688
## 3  grievance.1          3    0 1.0000 0.7758   0.9322
## 4  grievance.1          4    0 1.0000 0.7389   0.9167
## 5  grievance.1          5    0 1.0000 0.7617   0.9143
## 6  grievance.2          1    0 1.0000 0.5868   0.9310
## 7  grievance.2          2    0 1.0000 0.7986   0.9643
## 8  grievance.2          3    0 1.0000 0.6638   0.9431
## 9  grievance.2          4    0 1.0000 0.7090   0.8984
## 10 grievance.2          5    0 1.0000 0.6868   0.9280
## 11 grievance.3          1    0 1.0000 0.6778   0.9316
## 12 grievance.3          2    0 1.0000 0.8018   0.9823
## 13 grievance.3          3    0 0.9916 0.8193   0.9440
## 14 grievance.3          4    0 1.0000 0.7665   0.9083
## 15 grievance.3          5    0 0.9829 0.7855   0.8984
```

4.2.3 Combined Model testing

```
# Combined Models
combined_model <- function(threshold_value) {
    filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,grievi
        "warsa,peace,mount,geogia,lnpop,elfo,rf,pol16,etdo4590,dem,greedxb",
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem,ygini
        "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem",
```

```
    "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590",
    "warsa,sxp,sxp2,lngdp_,gy1,peace,geogia,lnpop,frac,etdo4590",
    "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590,oilsxp,oilsxp2")

regression_formula_list <- list("warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia
    "warsa ~  peace + mount + geogia + lnpop + elfo + rf + pol16 + etdo4590 + dem + greedxb",
    "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf
    "warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf +
    "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590",
    "warsa ~  sxp + sxp2 + lngdp_ + gy1 + peace + geogia + lnpop + frac + etdo4590",
    "warsa ~  sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590 +  oilsxp + oilsxp2


for (i in c(1:7)) {

    for (testIndex in 1:k) {

        filtering_columns <- strsplit(filtering_columns_list[[i]],
            ",")[[1]]

        combined.data <- shuffled_data[, filtering_columns]

        # Segement your data by fold using the which()
        # function
        testIndexes <- which(folds == testIndex,
            arr.ind = TRUE)
        testData <- combined.data[testIndexes,
            ]
        trainData <- combined.data[-testIndexes,
            ]

        # trainData <- na.omit(trainData) testData <-
        # na.omit(testData)

        combined_fit <- glm(as.formula(regression_formula_list[[i]]),
            family = binomial(link = "logit"),
            data = trainData)

        combined_predict <- predict(combined_fit,
            newdata = testData, type = "response")

        combined_y.hat <- as.matrix(combined_predict)

        y <- as.matrix(testData$warsa)

        all_tests[(5 + 3) * k + (i - 1) * k + testIndex,
            1] <- paste(c("combined", i), collapse = ".")
        all_tests[(5 + 3) * k + (i - 1) * k + testIndex,
            2] <- as.numeric(testIndex)

        if (thresholding_flag == TRUE) {
            combined_y.hat_normalized <- combined_y.hat
```

```
            combined_y.hat_normalized[combined_y.hat_normalized >=
              threshold_value] <- 1
            combined_y.hat_normalized[combined_y.hat_normalized <
              threshold_value] <- 0

            comb_predict_normalized <- prediction(combined_y.hat_normalized,
              y)

            len <- length(comb_predict_normalized@fp[[1]])
            fp <- as.numeric(comb_predict_normalized@fp[[1]][[len -
              1]])
            tp <- as.numeric(comb_predict_normalized@tp[[1]][[len -
              1]])
            fn <- as.numeric(comb_predict_normalized@fn[[1]][[len -
              1]])
            tn <- as.numeric(comb_predict_normalized@tn[[1]][[len -
              1]])

            all_tests[(5 + 3) * k + (i - 1) * k +
              testIndex, sens_index] <- tp/(tp +
              fn)
            all_tests[(5 + 3) * k + (i - 1) * k +
              testIndex, spec_index] <- tn/(tn +
              fp)

            all_tests[(5 + 3) * k + (i - 1) * k +
              testIndex, accuracy_index] <- (tp +
              tn)/(tp + tn + fp + fn)

            comb_predict <- prediction(combined_y.hat,
              y)
            comb_auc <- performance(comb_predict,
              measure = "auc")
            all_tests[(5 + 3) * k + (i - 1) * k +
              testIndex, auc_index] <- as.numeric(unlist(slot(comb_auc,
              "y.values")))

        } else {

            comb_predict <- prediction(combined_y.hat,
              y)

            comb_f <- performance(comb_predict,
              measure = "f")
            comb_where.F <- which.max(as.numeric(unlist(slot(comb_f,
              "y.values"))))
            comb_what.F <- performance(comb_predict,
              measure = "sens", x.measure = "spec")

            all_tests[(5 + 3) * k + (i - 1) * k +
              testIndex, sens_index] <- as.numeric(unlist(slot(comb_what.F,
              "y.values")))[comb_where.F]
            all_tests[(5 + 3) * k + (i - 1) * k +
```

```
                  testIndex, spec_index] <- as.numeric(unlist(slot(comb_what.F,
                  "x.values")))[comb_where.F]

                comb_auc <- performance(comb_predict,
                  measure = "auc")
                all_tests[(5 + 3) * k + (i - 1) * k +
                  testIndex, auc_index] <- as.numeric(unlist(slot(comb_auc,
                  "y.values")))

            }

        }
    }

    lower_lim = 5 * k + 3 * k + 1
    upper_lim = 5 * k + 3 * k + 7 * k
    return(convert2dArrayToDf(all_tests[lower_lim:upper_lim,
        1:6]))
}
temp3_1 <- combined_model(0.1)
temp3_2 <- combined_model(0.5)

print("For threshold = 0.1 :")
```

```
## [1] "For threshold = 0.1 :"
```

```
print(temp3_1)
```

```
##         model test_index   sens   spec    auc accuracy
## 1  combined.1          1 0.5000 0.8710 0.7492   0.8433
## 2  combined.1          2 0.7500 0.8468 0.9194   0.8438
## 3  combined.1          3 0.7273 0.7812 0.8246   0.7770
## 4  combined.1          4 0.8000 0.7368 0.8386   0.7419
## 5  combined.1          5 0.6364 0.7984 0.8259   0.7857
## 6  combined.2          1 0.5000 0.8548 0.7685   0.8284
## 7  combined.2          2 1.0000 0.8468 0.9415   0.8516
## 8  combined.2          3 0.6364 0.8125 0.7940   0.7986
## 9  combined.2          4 0.8000 0.7368 0.8289   0.7419
## 10 combined.2          5 0.7273 0.7907 0.8598   0.7857
## 11 combined.3          1 0.3333 0.8571 0.7546   0.8247
## 12 combined.3          2 0.3333 0.8391 0.8084   0.8222
## 13 combined.3          3 0.5000 0.8617 0.6348   0.8400
## 14 combined.3          4 0.6000 0.8000 0.7765   0.7789
## 15 combined.3          5 0.5714 0.7667 0.7968   0.7526
## 16 combined.4          1 0.4000 0.8548 0.7419   0.8209
## 17 combined.4          2 1.0000 0.8548 0.9355   0.8594
## 18 combined.4          3 0.7273 0.7969 0.7798   0.7914
## 19 combined.4          4 0.8000 0.7456 0.8298   0.7500
## 20 combined.4          5 0.6364 0.7907 0.8161   0.7786
## 21 combined.5          1 0.5000 0.8661 0.7402   0.8394
## 22 combined.5          2 0.5000 0.8450 0.9070   0.8346
## 23 combined.5          3 0.8182 0.8296 0.8155   0.8288
## 24 combined.5          4 0.8000 0.8051 0.8449   0.8047
## 25 combined.5          5 0.5455 0.8120 0.8640   0.7917
```

```
## 26 combined.6          1 0.4167 0.8571 0.7185   0.8224
## 27 combined.6          2 0.8000 0.8759 0.9401   0.8732
## 28 combined.6          3 0.7273 0.8108 0.8237   0.8050
## 29 combined.6          4 0.7273 0.7557 0.7911   0.7535
## 30 combined.6          5 0.6923 0.7958 0.8505   0.7871
## 31 combined.7          1 0.6667 0.8417 0.8120   0.8295
## 32 combined.7          2 0.7500 0.8468 0.8770   0.8438
## 33 combined.7          3 0.7273 0.8080 0.8335   0.8015
## 34 combined.7          4 0.8000 0.7876 0.8903   0.7886
## 35 combined.7          5 0.6364 0.8189 0.8690   0.8043
```

```
print("For threshold = 0.5 :")
```

```
## [1] "For threshold = 0.5 :"
```

```
print(temp3_2)
```

```
##         model test_index    sens   spec    auc accuracy
## 1  combined.1          1 0.00000 0.9839 0.7492   0.9104
## 2  combined.1          2 0.00000 1.0000 0.9194   0.9688
## 3  combined.1          3 0.09091 0.9688 0.8246   0.8993
## 4  combined.1          4 0.20000 0.9912 0.8386   0.9274
## 5  combined.1          5 0.09091 0.9845 0.8259   0.9143
## 6  combined.2          1 0.10000 0.9919 0.7685   0.9254
## 7  combined.2          2 0.00000 1.0000 0.9415   0.9688
## 8  combined.2          3 0.09091 0.9688 0.7940   0.8993
## 9  combined.2          4 0.10000 0.9912 0.8289   0.9194
## 10 combined.2          5 0.00000 0.9922 0.8598   0.9143
## 11 combined.3          1 0.00000 1.0000 0.7546   0.9381
## 12 combined.3          2 0.00000 1.0000 0.8084   0.9667
## 13 combined.3          3 0.00000 0.9787 0.6348   0.9200
## 14 combined.3          4 0.00000 0.9882 0.7765   0.8842
## 15 combined.3          5 0.00000 0.9778 0.7968   0.9072
## 16 combined.4          1 0.00000 0.9839 0.7419   0.9104
## 17 combined.4          2 0.00000 1.0000 0.9355   0.9688
## 18 combined.4          3 0.09091 0.9531 0.7798   0.8849
## 19 combined.4          4 0.10000 0.9912 0.8298   0.9194
## 20 combined.4          5 0.09091 0.9845 0.8161   0.9143
## 21 combined.5          1 0.00000 0.9921 0.7402   0.9197
## 22 combined.5          2 0.00000 1.0000 0.9070   0.9699
## 23 combined.5          3 0.18182 0.9704 0.8155   0.9110
## 24 combined.5          4 0.10000 1.0000 0.8449   0.9297
## 25 combined.5          5 0.09091 0.9925 0.8640   0.9236
## 26 combined.6          1 0.08333 0.9929 0.7185   0.9211
## 27 combined.6          2 0.00000 1.0000 0.9401   0.9648
## 28 combined.6          3 0.09091 0.9865 0.8237   0.9245
## 29 combined.6          4 0.00000 0.9924 0.7911   0.9155
## 30 combined.6          5 0.07692 0.9930 0.8505   0.9161
## 31 combined.7          1 0.22222 1.0000 0.8120   0.9457
## 32 combined.7          2 0.00000 1.0000 0.8770   0.9688
## 33 combined.7          3 0.18182 0.9360 0.8335   0.8750
## 34 combined.7          4 0.00000 0.9912 0.8903   0.9106
## 35 combined.7          5 0.00000 1.0000 0.8690   0.9203
```

4.2.4 Computing Averages of the k-fold Validation

```
# print('Results for threshold = 0.1')
# print('Opportunity Models') print(temp1_1)
# print('Grievance Models') print(temp2_1)
# print('Combined Models') print(temp3_1)

all_tests[1:25, ] <- as.matrix(temp1_1)
all_tests[26:40, ] <- as.matrix(temp2_1)
all_tests[41:75, ] <- as.matrix(temp3_1)

all_tests <- convert2dArrayToDf(all_tests)

result <- aggregate(all_tests[, 3:6], list(all_tests$model),
    mean)

names(result)[1] <- "model"
print("Averaged K-Means results (for threshold=0.1)")
```

```
## [1] "Averaged K-Means results (for threshold=0.1)"
```

```
print(result)
```

```
##            model   sens   spec    auc accuracy
## 1     combined.1 0.6827 0.8068 0.8315   0.7983
## 2     combined.2 0.7327 0.8083 0.8385   0.8012
## 3     combined.3 0.4676 0.8249 0.7542   0.8037
## 4     combined.4 0.7127 0.8086 0.8206   0.8001
## 5     combined.5 0.6327 0.8316 0.8343   0.8198
## 6     combined.6 0.6727 0.8191 0.8248   0.8082
## 7     combined.7 0.7161 0.8206 0.8564   0.8135
## 8    grievance.1 0.6468 0.8067 0.7591   0.7943
## 9    grievance.2 0.3955 0.7891 0.6890   0.7636
## 10   grievance.3 0.4379 0.8457 0.7702   0.8250
## 11 opportunity.1 0.5945 0.8264 0.8216   0.8124
## 12 opportunity.2 0.6309 0.8233 0.8311   0.8123
## 13 opportunity.3 0.7322 0.7986 0.8201   0.7920
## 14 opportunity.4 0.5207 0.8700 0.8005   0.8495
## 15 opportunity.5 0.4921 0.8700 0.7886   0.8476
```

```
# print('For threshold value = 0.5')

all_tests <- matrix(data = 0, nrow = 15 * k, ncol = 6)
colnames(all_tests) <- c("model", "test_index", "sens",
    "spec", "auc", "accuracy")

all_tests[1:25, ] <- as.matrix(temp1_2)
all_tests[26:40, ] <- as.matrix(temp2_2)
all_tests[41:75, ] <- as.matrix(temp3_2)

all_tests <- convert2dArrayToDf(all_tests)

result <- aggregate(all_tests[, 3:6], list(all_tests$model),
    mean)
```

```r
names(result)[1] <- "model"
print("Averaged K-Means results (for threshold=0.5)")
```

```
## [1] "Averaged K-Means results (for threshold=0.5)"
```

```r
print(result)
```

```
##            model    sens   spec    auc accuracy
## 1     combined.1 0.07636 0.9857 0.8315   0.9240
## 2     combined.2 0.05818 0.9888 0.8385   0.9254
## 3     combined.3 0.00000 0.9889 0.7542   0.9232
## 4     combined.4 0.05636 0.9825 0.8206   0.9196
## 5     combined.5 0.07455 0.9910 0.8343   0.9308
## 6     combined.6 0.05023 0.9930 0.8248   0.9284
## 7     combined.7 0.08081 0.9854 0.8564   0.9241
## 8    grievance.1 0.00000 1.0000 0.7591   0.9311
## 9    grievance.2 0.00000 1.0000 0.6890   0.9330
## 10   grievance.3 0.00000 0.9949 0.7702   0.9329
## 11 opportunity.1 0.05636 0.9907 0.8216   0.9291
## 12 opportunity.2 0.05636 0.9892 0.8311   0.9277
## 13 opportunity.3 0.10970 0.9929 0.8201   0.9312
## 14 opportunity.4 0.07857 0.9946 0.8005   0.9438
## 15 opportunity.5 0.07857 0.9966 0.7886   0.9456
```

```r
# write.csv(result, file =
# paste0('Project_Extension_2_Threshold_',
# params$threshold, '.csv'))
```

4.3 Regularized Model

descriptiion lorum impsonvefonvefv

```r
k <- 5

all_tests <- matrix(data = 0, nrow = k, ncol = 6)
colnames(all_tests) <- c("model", "test_index", "sens",
    "spec", "auc", "accuracy")

set.seed(42)
shuffled_data <- data[sample(nrow(data)), ]
folds <- cut(seq(1, nrow(shuffled_data)), breaks = k,
    labels = FALSE)

thresholding_flag = TRUE

sens_index <- 3
spec_index <- 4
auc_index <- 5
accuracy_index <- 6
```

```r
reg_model <- function(threshold_value) {
    for (testIndex in 1:k) {

        regularized.data <- shuffled_data %>% select(warsa,
```

```
    coldwar, prevwara, peace, elfo, rf, frac,
    geogia, mount, lnpop, sxp, sxp2, dem, ygini,
    lgini, lngdp_, gy1, secm, diaspeaa, difdpeaa,
    diahpeaa, pol16, oilsxp, oilsxp2, etdo4590)

# Segement your data by fold using the which()
# function
testIndexes <- which(folds == testIndex, arr.ind = TRUE)
testData <- regularized.data[testIndexes, ]
trainData <- regularized.data[-testIndexes,
    ]

trainData <- na.omit(trainData)
testData <- na.omit(testData)

train_x <- as.matrix(trainData[, 2:ncol(trainData)])
train_y <- as.matrix(trainData$warsa)

test_x <- as.matrix(testData[, 2:ncol(testData)])
test_y <- as.matrix(testData$warsa)

regularized_fit <- cv.glmnet(x = train_x, y = train_y,
    family = "binomial", type.measure = "auc")
regularized_predict <- predict(regularized_fit,
    test_x, type = "response", s = "lambda.min")

regularized_y.hat <- as.matrix(regularized_predict)

all_tests[testIndex, 1] <- paste(c("regularized"),
    collapse = ".")
all_tests[testIndex, 2] <- as.numeric(testIndex)

regularized_y.hat_normalized <- regularized_y.hat

regularized_y.hat_normalized[regularized_y.hat_normalized >=
    threshold_value] <- 1
regularized_y.hat_normalized[regularized_y.hat_normalized <
    threshold_value] <- 0

regularized_predict_normalized <- prediction(regularized_y.hat_normalized,
    test_y)

len <- length(regularized_predict_normalized@fp[[1]])
fp <- as.numeric(regularized_predict_normalized@fp[[1]][[len -
    1]])
tp <- as.numeric(regularized_predict_normalized@tp[[1]][[len -
    1]])
fn <- as.numeric(regularized_predict_normalized@fn[[1]][[len -
    1]])
tn <- as.numeric(regularized_predict_normalized@tn[[1]][[len -
    1]])

all_tests[testIndex, sens_index] <- tp/(tp +
```

```
            fn)
        all_tests[testIndex, spec_index] <- tn/(tn +
            fp)

        all_tests[testIndex, accuracy_index] <- (tp +
            tn)/(tp + tn + fp + fn)

        regularized_predict <- prediction(regularized_y.hat,
            test_y)
        regular_auc <- performance(regularized_predict,
            measure = "auc")
        all_tests[testIndex, auc_index] <- as.numeric(unlist(slot(regular_auc,
            "y.values")))
    }

    lower_lim = 1
    upper_lim = k
    return(convert2dArrayToDf(all_tests[lower_lim:upper_lim,
        1:6]))
}
```

```
# For threshold = 0.1

temp1 <- reg_model(0.1)
all_tests[1:5, ] <- as.matrix(temp1)
all_tests <- convert2dArrayToDf(all_tests)
result <- aggregate(all_tests[, 3:6], list(all_tests$model),
    mean)
names(result)[1] <- "model"

print("For threshold = 0.1")
```

```
## [1] "For threshold = 0.1"
```

```
print(result)
```

```
##         model   sens   spec   auc accuracy
## 1 regularized 0.4583 0.8483 0.817    0.816
```

```
# For threshold = 0.5

all_tests <- matrix(data = 0, nrow = k, ncol = 6)
colnames(all_tests) <- c("model", "test_index", "sens",
    "spec", "auc", "accuracy")
temp1 <- reg_model(0.5)
all_tests[1:5, ] <- as.matrix(temp1)
all_tests <- convert2dArrayToDf(all_tests)
result <- aggregate(all_tests[, 3:6], list(all_tests$model),
    mean)
names(result)[1] <- "model"

print("For threshold = 0.5")
```

```
## [1] "For threshold = 0.5"
```

```
print(result)
```

```
##         model sens spec    auc accuracy
## 1 regularized 0.05    1 0.8108   0.9403
```

4.4 Observations

1) The accuracy and specificity is higher, while the sensitivity is very low for models at threshold 0.5 (because the model almost always predicts 0). The accuracy and specificity decrease but the sensitivity increase dramatically at threshold 0.1. So the number of civil wars we're actually predicting at this threshold is very low, and while the accuracy and specificity increase (because the dataset is skewed), this does not mean that this model performs better.

2) Broadly speaking, the performance of the first three opportunity models is comparable to the grievance models. However, the last two opportunity models tend to outperform even the combined models (in terms of accuracy).

3) The performance of the regularized GLM is not necessarily better than the proposed models. The sensitivity of the 7th combined model is significantly better than that of the regularized model while all the other parameters are at least comparable.

4) Our findings seem to conform to the authors' claim that "greed is a better indicator than grievance for predicting civil wars"

The following is a list of all packages used to generate these results. (Leave at very end of file.)

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2    glmnet_2.0-16     foreach_1.4.4
##  [4] ROCR_1.0-7        gplots_3.0.1.1    lme4_1.1-21
##  [7] Matrix_1.2-15     DescTools_0.99.28 foreign_0.8-71
## [10] forcats_0.3.0     stringr_1.3.1     dplyr_0.7.8
## [13] purrr_0.3.0       readr_1.3.1       tidyr_0.8.2
## [16] tibble_2.0.1      ggplot2_3.1.0     tidyverse_1.2.1
## [19] scales_1.0.0      here_0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0         lubridate_1.7.4    mvtnorm_1.0-10
##  [4] lattice_0.20-38    gtools_3.8.1       assertthat_0.2.0
##  [7] rprojroot_1.3-2    digest_0.6.18      R6_2.3.0
## [10] cellranger_1.1.0   plyr_1.8.4         backports_1.1.3
```

```
## [13] evaluate_0.12      httr_1.4.0         pillar_1.3.1
## [16] rlang_0.3.1        lazyeval_0.2.1     readxl_1.2.0
## [19] rstudioapi_0.9.0   minqa_1.2.4        gdata_2.18.0
## [22] nloptr_1.2.1       rmarkdown_1.11     splines_3.5.2
## [25] munsell_0.5.0      broom_0.5.1        compiler_3.5.2
## [28] modelr_0.1.2       xfun_0.4           pkgconfig_2.0.2
## [31] manipulate_1.0.1   htmltools_0.3.6    tidyselect_0.2.5
## [34] expm_0.999-4       codetools_0.2-15   crayon_1.3.4
## [37] withr_2.1.2        MASS_7.3-51.1      bitops_1.0-6
## [40] grid_3.5.2         nlme_3.1-137       jsonlite_1.6
## [43] gtable_0.2.0       formatR_1.6        magrittr_1.5
## [46] KernSmooth_2.23-15 cli_1.0.1          stringi_1.2.4
## [49] xml2_1.2.0         generics_0.0.2     boot_1.3-20
## [52] iterators_1.0.10   tools_3.5.2        glue_1.3.0
## [55] hms_0.4.2          yaml_2.2.0         colorspace_1.4-0
## [58] caTools_1.17.1.2   rvest_0.3.2        knitr_1.21
## [61] bindr_0.1.1        haven_2.0.0
```