# Ex4_Keel_Scruton

Note: all the first section here is a direct copy of ex3 work, some explanations have been left out as such.

## Load data

Load the following data: + applications from `app_data_sample.parquet` + edges from `edges_sample.csv`

```
data_path <- "/Users/keelscruton/Desktop/Org Network Analysis/672_project_data/"
applications <- read_parquet(paste0(data_path,"app_data_sample.parquet"))
edges <- read_csv(paste0(data_path,"edges_sample.csv"))
```

```
## Rows: 32906 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): application_number
## dbl  (2): ego_examiner_id, alter_examiner_id
## date (1): advice_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
applications
```

```
## # A tibble: 2,018,477 x 16
##    application_number filing_date examiner_name_last examiner_name_first
##    <chr>              <date>      <chr>              <chr>
##  1 08284457           2000-01-26  HOWARD             JACQUELINE
##  2 08413193           2000-10-11  YILDIRIM           BEKIR
##  3 08531853           2000-05-17  HAMILTON           CYNTHIA
##  4 08637752           2001-07-20  MOSHER             MARY
##  5 08682726           2000-04-10  BARR               MICHAEL
##  6 08687412           2000-04-28  GRAY               LINDA
##  7 08716371           2004-01-26  MCMILLIAN          KARA
##  8 08765941           2000-06-23  FORD               VANESSA
##  9 08776818           2000-02-04  STRZELECKA         TERESA
## 10 08809677           2002-02-20  KIM                SUN
## # ... with 2,018,467 more rows, and 12 more variables:
## #   examiner_name_middle <chr>, examiner_id <dbl>, examiner_art_unit <dbl>,
## #   uspc_class <chr>, uspc_subclass <chr>, patent_number <chr>,
## #   patent_issue_date <date>, abandon_date <date>, disposal_type <chr>,
## #   appl_status_code <dbl>, appl_status_date <chr>, tc <dbl>
```

```
edges
```

```
## # A tibble: 32,906 x 4
##    application_number advice_date ego_examiner_id alter_examiner_id
##    <chr>              <date>                <dbl>             <dbl>
##  1 09402488           2008-11-17            84356             66266
##  2 09402488           2008-11-17            84356             63519
##  3 09402488           2008-11-17            84356             98531
##  4 09445135           2008-08-21            92953             71313
##  5 09445135           2008-08-21            92953             93865
##  6 09445135           2008-08-21            92953             91818
##  7 09479304           2008-12-15            61767             69277
##  8 09479304           2008-12-15            61767             92446
##  9 09479304           2008-12-15            61767             66805
## 10 09479304           2008-12-15            61767             70919
## # ... with 32,896 more rows
```

Determine the gender for each examiner

```
library(gender)
#install_genderdata_package() # only run this line the first time you use the package, to get data for
# get a list of first names without repetitions
examiner_names <- applications %>%
  distinct(examiner_name_first)
examiner_names
```

```
## # A tibble: 2,595 x 1
##    examiner_name_first
##    <chr>
##  1 JACQUELINE
##  2 BEKIR
##  3 CYNTHIA
##  4 MARY
##  5 MICHAEL
##  6 LINDA
##  7 KARA
##  8 VANESSA
##  9 TERESA
## 10 SUN
## # ... with 2,585 more rows
```

```
# get a table of names and gender
examiner_names_gender <- examiner_names %>%
  do(results = gender(.$examiner_name_first, method = "ssa")) %>%
  unnest(cols = c(results), keep_empty = TRUE) %>%
  select(
    examiner_name_first = name,
    gender,
    proportion_female
  )
examiner_names_gender
```

```
## # A tibble: 1,822 x 3
##    examiner_name_first gender proportion_female
##    <chr>               <chr>              <dbl>
```

```
##   1 AABON               male              0.0082
##   2 ABDEL               male              0
##   3 ABDOU               male              0
##   4 ABDUL               male              0
##   5 ABDULHAKIM          male              0
##   6 ABDULLAH            male              0
##   7 ABDULLAHI           male              0
##   8 ABIGAIL             female            0.998
##   9 ABIMBOLA            female            0.944
##  10 ABRAHAM             male              0.0031
## # ... with 1,812 more rows
```

final step in determining gender by name…

```
# remove extra colums from the gender table
examiner_names_gender <- examiner_names_gender %>%
  select(examiner_name_first, gender)
# joining gender back to the dataset
applications <- applications %>%
  left_join(examiner_names_gender, by = "examiner_name_first")
# cleaning up
rm(examiner_names)
rm(examiner_names_gender)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells  4777409 255.2    8350785 446.0         NA  5178903 276.6
## Vcells 50056660 382.0   96079046 733.1      16384 80372405 613.2
```

### Guess the examiner's race

We'll now use package `wru` to estimate likely race of an examiner. Just like with gender, we'll get a list of unique names first, only now we are using surnames.

```
library(wru)
examiner_surnames <- applications %>%
  select(surname = examiner_name_last) %>%
  distinct()
examiner_surnames
```

```
## # A tibble: 3,806 x 1
##    surname
##    <chr>
##  1 HOWARD
##  2 YILDIRIM
##  3 HAMILTON
##  4 MOSHER
##  5 BARR
##  6 GRAY
##  7 MCMILLIAN
##  8 FORD
##  9 STRZELECKA
```

```
## 10 KIM
## # ... with 3,796 more rows
```

```r
examiner_race <- predict_race(voter.file = examiner_surnames, surname.only = T) %>%
  as_tibble()
```

```
## [1] "Proceeding with surname-only predictions..."
```

```
## Warning in merge_surnames(voter.file): Probabilities were imputed for 698
## surnames that could not be matched to Census list.
```

```r
examiner_race
```

```
## # A tibble: 3,806 x 6
##    surname    pred.whi pred.bla pred.his pred.asi pred.oth
##    <chr>         <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
##  1 HOWARD       0.643   0.295    0.0237   0.005    0.0333
##  2 YILDIRIM     0.861   0.0271   0.0609   0.0135   0.0372
##  3 HAMILTON     0.702   0.237    0.0245   0.0054   0.0309
##  4 MOSHER       0.947   0.00410  0.0241   0.00640  0.0185
##  5 BARR         0.827   0.117    0.0226   0.00590  0.0271
##  6 GRAY         0.687   0.251    0.0241   0.0054   0.0324
##  7 MCMILLIAN    0.359   0.574    0.0189   0.00260  0.0463
##  8 FORD         0.620   0.32     0.0237   0.0045   0.0313
##  9 STRZELECKA   0.666   0.0853   0.137    0.0797   0.0318
## 10 KIM          0.0252  0.00390  0.00650  0.945    0.0198
## # ... with 3,796 more rows
```

```r
examiner_race <- examiner_race %>%
  mutate(max_race_p = pmax(pred.asi, pred.bla, pred.his, pred.oth, pred.whi)) %>%
  mutate(race = case_when(
    max_race_p == pred.asi ~ "Asian",
    max_race_p == pred.bla ~ "black",
    max_race_p == pred.his ~ "Hispanic",
    max_race_p == pred.oth ~ "other",
    max_race_p == pred.whi ~ "white",
    TRUE ~ NA_character_
  ))
examiner_race
```

```
## # A tibble: 3,806 x 8
##    surname    pred.whi pred.bla pred.his pred.asi pred.oth max_race_p race
##    <chr>         <dbl>    <dbl>    <dbl>    <dbl>    <dbl>      <dbl> <chr>
##  1 HOWARD       0.643   0.295    0.0237   0.005    0.0333       0.643 white
##  2 YILDIRIM     0.861   0.0271   0.0609   0.0135   0.0372       0.861 white
##  3 HAMILTON     0.702   0.237    0.0245   0.0054   0.0309       0.702 white
##  4 MOSHER       0.947   0.00410  0.0241   0.00640  0.0185       0.947 white
##  5 BARR         0.827   0.117    0.0226   0.00590  0.0271       0.827 white
##  6 GRAY         0.687   0.251    0.0241   0.0054   0.0324       0.687 white
##  7 MCMILLIAN    0.359   0.574    0.0189   0.00260  0.0463       0.574 black
##  8 FORD         0.620   0.32     0.0237   0.0045   0.0313       0.620 white
##  9 STRZELECKA   0.666   0.0853   0.137    0.0797   0.0318       0.666 white
## 10 KIM          0.0252  0.00390  0.00650  0.945    0.0198       0.945 Asian
## # ... with 3,796 more rows
```

```r
# removing extra columns
examiner_race <- examiner_race %>%
  select(surname,race)
applications <- applications %>%
  left_join(examiner_race, by = c("examiner_name_last" = "surname"))
rm(examiner_race)
rm(examiner_surnames)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells  5117595 273.4    8350785 446.0         NA  6196527 331.0
## Vcells 53743521 410.1   96079046 733.1      16384 94805400 723.4
```

Determine tenure as well as a number of days…

```r
library(lubridate) # to work with dates
examiner_dates <- applications %>%
  select(examiner_id, filing_date, appl_status_date)
examiner_dates
```

```
## # A tibble: 2,018,477 x 3
##    examiner_id filing_date appl_status_date
##          <dbl> <date>      <chr>
##  1       96082 2000-01-26  30jan2003 00:00:00
##  2       87678 2000-10-11  27sep2010 00:00:00
##  3       63213 2000-05-17  30mar2009 00:00:00
##  4       73788 2001-07-20  07sep2009 00:00:00
##  5       77294 2000-04-10  19apr2001 00:00:00
##  6       68606 2000-04-28  16jul2001 00:00:00
##  7       89557 2004-01-26  15may2017 00:00:00
##  8       97543 2000-06-23  03apr2002 00:00:00
##  9       98714 2000-02-04  27nov2002 00:00:00
## 10       65530 2002-02-20  23mar2009 00:00:00
## # ... with 2,018,467 more rows
```

```r
examiner_dates <- examiner_dates %>%
  mutate(start_date = ymd(filing_date), end_date = as_date(dmy_hms(appl_status_date)))
```

```r
examiner_dates <- examiner_dates %>%
  group_by(examiner_id) %>%
  summarise(
    earliest_date = min(start_date, na.rm = TRUE),
    latest_date = max(end_date, na.rm = TRUE),
    tenure_days = interval(earliest_date, latest_date) %/% days(1)
    ) %>%
  filter(year(latest_date)<2018)
examiner_dates
```

```
## # A tibble: 5,625 x 4
##    examiner_id earliest_date latest_date tenure_days
##          <dbl> <date>        <date>            <dbl>
```

```
## 1        59012 2004-07-28    2015-07-24         4013
## 2        59025 2009-10-26    2017-05-18         2761
## 3        59030 2005-12-12    2017-05-22         4179
## 4        59040 2007-09-11    2017-05-23         3542
## 5        59052 2001-08-21    2007-02-28         2017
## 6        59054 2000-11-10    2016-12-23         5887
## 7        59055 2004-11-02    2007-12-26         1149
## 8        59056 2000-03-24    2017-05-22         6268
## 9        59074 2000-01-31    2017-03-17         6255
## 10       59081 2011-04-21    2017-05-19         2220
## # ... with 5,615 more rows
```

```
applications <- applications %>%
  left_join(examiner_dates, by = "examiner_id")
rm(examiner_dates)
gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells  5131070 274.1   15087248  805.8         NA  15087248  805.8
## Vcells 66121129 504.5  138529826 1056.9      16384 137965366 1052.6
```

Application Processing Time section (new for ex4)

```
applications$appl_end_date <- paste(applications$patent_issue_date, applications$abandon_date, sep=',')
# clean  the column by removing instances of commas and NA's
applications$appl_end_date <- gsub('NA', "", as.character(applications$appl_end_date))
applications$appl_end_date <- gsub(',', "", as.character(applications$appl_end_date))
# make date format  consistent
applications$appl_end_date <- as.Date(applications$appl_end_date, format="%Y-%m-%d")
applications$filing_date <- as.Date(applications$filing_date, format="%Y-%m-%d")
# calculate the difference in days between the application end date and the filing date
applications$appl_proc_days <- as.numeric(difftime(applications$appl_end_date, applications$filing_date
# Remove data points where the filing date is after the issue/abandon dates, this is not possible
applications <- applications %>% filter(appl_proc_days >=0 | appl_proc_days != NA)
```

want only unique instances

```
vars <- c("gender","race","tenure_days","appl_proc_days")
applications = drop_na(applications,any_of(vars))
```

make my group selection

```
group_161 <- applications[substr(applications$examiner_art_unit, 1, 3) == 161,]
group_161 <-group_161[row.names(unique(group_161[,"examiner_id"])),]


group_162 <- applications[substr(applications$examiner_art_unit, 1, 3) == 162,]
group_162 <-group_162[row.names(unique(group_162[,"examiner_id"])),]
```

Create advice networks from `edges_sample` and calculate centrality scores for examiners in your selected workgroups

```r
#create distinct subset of examiners with only the art unit and examiner id to be able to re join onto
examiner_dis = distinct(subset(applications, select = -c(filing_date, abandon_date, earliest_date, appl
examiner_dis$group = substr(examiner_dis$examiner_art_unit, 1,3)
#get rid of all examiners except those in group 161 or 162
examiner_dis = examiner_dis[examiner_dis$group==161 | examiner_dis$group==162,]
```

Now that we have a list of the examiners who are part of work groups 161 and 162 we can combine (merge) it with the edge list (edges) this will allow us to form our subset network.

```r
#edges_examiner = merge(x=edges, y=examiner_dis, by.x="ego_examiner_id", by.y="examiner_id", all.x=TRUE
#edges_examiner = edges_examiner %>% rename(ego_au=examiner_art_unit, ego_group=group)

#edges_examiner = merge(x=edges_examiner, y=examiner_dis, by.x="alter_examiner_id", by.y="examiner_id",
#edges_examiner = edges_examiner %>% rename(alter_au=examiner_art_unit, alter_group=group)

#edges_examiner = drop_na(edges_examiner) #drop all na, ie values not in the selected workgroups.





##^^^ ERROR HAPPENING HERE I HAVE COMMENTED IT OUT SO THAT I CAN KNIT IT.
```

Now the above data set has the edges of alter and ego examiners. we can next create the list of nodes for both the ego and alter examiners.

```r
#Ego_nodes = subset(edges_examiner, select=c(ego_examiner_id,ego_au, ego_group)) %>% rename(examiner_id

#Alter_nodes = subset(edges_examiner, select=c(alter_examiner_id,alter_au, alter_group))%>% rename(exam

#Nodes_full = distinct(rbind(Ego_nodes, Alter_nodes))

#Nodes_full is a list of all the distinct examiners involved in the advice network
```

Now we can create the graph network.

```r
#ran into error where i had duplicated vertex names, to fix take the first instances only in the nodes

#Nodes_full = Nodes_full %>% group_by(examiner_id) %>% summarise(examiner_id=first(examiner_id), art_un

#network <- graph_from_data_frame(d=edges_examiner, vertices=Nodes_full, directed=TRUE)
```

```r
#Degree <- degree(network)
#Closeness <- closeness(network)
#merge back into the dataframe...
#comp <- data.frame(Nodes_full, Degree, Closeness)
#applications_final <- merge(x=applications, y=comp, by='examiner_id', all.x=TRUE)
```

Note: ran into an error above: Vector memory exhausted, not sure how to approach this differently to avoid having such a big data set being merged. note that the lm() work would look something like the following but the code does not run.

lm <- lm(appl_proc_days~ Degree + Closeness+ gender + tenure_days, data=applications_final) summary(lm1)

by observing the estimated coefficients from this step we could make some observations on what variables are most affecting application processing time (days)

we could then next create an interaction variable to discover the interaction between some demographic data and centrality (degree, closeness) this would look like the following. note that we could swap our gender for race.

lm <- lm(appl_proc_days~ Degree + Closeness+ gender + tenure_days +Degree*gender + Closeness*gender, data=applications_final) summary(lm1)

the findings from this could help us discover if there is some subgroup that is better at processing applications faster.