Ex3 - Advice network at USPTO

Jennifer Liu

2024-04-02

Part 1: Load data

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

```
library(arrow)
```

```
##
## Attaching package: 'arrow'
```

```
## The following object is masked from 'package:utils':
##
##     timestamp
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

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

```
library(gender)
library(wru)
```

```
##
## Please cite as:
##
## Khanna K, Bertelsen B, Olivella S, Rosenman E, Rossell Hayes A, Imai K
## (2024). _wru: Who are You? Bayesian Prediction of Racial Category Using
## Surname, First Name, Middle Name, and Geolocation_. R package version
## 3.0.1, <https://CRAN.R-project.org/package=wru>.
##
## Note that wru 2.0.0 uses 2020 census data by default.
## Use the argument `year = "2010"`, to replicate analyses produced with earlier package versions.
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:arrow':
##
##     duration
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(readr)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats 1.0.0     ✓ stringr 1.5.1
## ✓ ggplot2 3.5.0     ✓ tibble  3.2.1
## ✓ purrr   1.0.2     ✓ tidyr   1.3.1
## ── Conflicts ─────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ lubridate::duration() masks arrow::duration()
## ✖ dplyr::filter()       masks stats::filter()
## ✖ dplyr::lag()          masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
library(ggplot2)

data_path <- "C:\\Users\\Admin\\Downloads\\"

# Reading a Parquet file
applications <- read_parquet(paste0(data_path, "app_data_sample.parquet"))

# Reading a CSV file and suppressing the column type message
edges <- read_csv(paste0(data_path, "edges_sample.csv"), show_col_types = FALSE)
```

```
library(dplyr)
#install.packages("gender")
library(gender)
examiner_names <- applications %>% distinct(examiner_name_first)

head(examiner_names)
```

```
## # A tibble: 6 × 1
##   examiner_name_first
##   <chr>
## 1 JACQUELINE
## 2 BEKIR
## 3 CYNTHIA
## 4 MARY
## 5 MICHAEL
## 6 LINDA
```

Get gender for examiners

We'll get gender based on the first name of the examiner, which is recorded in the field `examiner_name_first`. We'll use library `gender` for that, relying on a modified version of their own example (https://cran.r-project.org/web/packages/gender/vignettes/predicting-gender.html).

Note that there are over 2 million records in the applications table – that's because there are many records for each examiner, as many as the number of applications that examiner worked on during this time frame. Our first step therefore is to get all *unique* names in a separate list `examiner_names`. We will then guess gender for each one and will join this table back to the original dataset. So, let's get names without repetition:

Now let's use function `gender()` as shown in the example for the package to attach a gender and probability to each name and put the results into the table `examiner_names_gender`. Note that the first time you run this code, you need to say "Yes" in the console to download the gender data.

```
# install.packages("tidyr")
library(tidyr)

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
)

head(examiner_names_gender)
```

```
## # A tibble: 6 × 3
##   examiner_name_first gender proportion_female
##   <chr>               <chr>              <dbl>
## 1 AARON               male              0.0082
## 2 ABDEL               male              0
## 3 ABDOU               male              0
## 4 ABDUL               male              0
## 5 ABDULHAKIM          male              0
## 6 ABDULLAH            male              0
```

Finally, let's join that table back to our original applications data and discard the temporary tables we have just created to reduce clutter in our environment.

```
# remove extra columns 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) max used  (Mb)
## Ncells  4708474 251.5    8055668 430.3  4728811 252.6
## Vcells 49957507 381.2   93537981 713.7 80273580 612.5
```

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.

```
#install.packages("wru")
library(wru)

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

examiner_surnames
```

```
## # A tibble: 3,806 × 1
##    surname
##    <chr>
##  1 HOWARD
##  2 YILDIRIM
##  3 HAMILTON
##  4 MOSHER
##  5 BARR
##  6 GRAY
##  7 MCMILLIAN
##  8 FORD
##  9 STRZELECKA
## 10 KIM
## # ℹ 3,796 more rows
```

We'll follow the instructions for the package outlined here https://github.com/kosukeimai/wru (https://github.com/kosukeimai/wru).

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

```
## Predicting race for 2020
```

```
## Warning: Unknown or uninitialised column: `state`.
```

```
## Proceeding with last name predictions...
```

```
## ℹ All local files already up-to-date!
```

```
## 701 (18.4%) individuals' last names were not matched.
```

```
examiner_race
```

```
## # A tibble: 3,806 × 6
##    surname    pred.whi pred.bla pred.his pred.asi pred.oth
##    <chr>         <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
##  1 HOWARD       0.597   0.295    0.0275   0.00690   0.0741
##  2 YILDIRIM     0.807   0.0273   0.0694   0.0165    0.0798
##  3 HAMILTON     0.656   0.239    0.0286   0.00750   0.0692
##  4 MOSHER       0.915   0.00425  0.0291   0.00917   0.0427
##  5 BARR         0.784   0.120    0.0268   0.00830   0.0615
##  6 GRAY         0.640   0.252    0.0281   0.00748   0.0724
##  7 MCMILLIAN    0.322   0.554    0.0212   0.00340   0.0995
##  8 FORD         0.576   0.320    0.0275   0.00621   0.0697
##  9 STRZELECKA   0.472   0.171    0.220    0.0825    0.0543
## 10 KIM          0.0169  0.00282  0.00546  0.943     0.0319
## # ℹ 3,796 more rows
```

As you can see, we get probabilities across five broad US Census categories: white, black, Hispanic, Asian and other. (Some of you may correctly point out that Hispanic is not a race category in the US Census, but these are the limitations of this package.)

Our final step here is to pick the race category that has the highest probability for each last name and then join the table back to the main applications table. See this example for comparing values across columns: https://www.tidyverse.org/blog/2020/04/dplyr-1-0-0-rowwise/ (https://www.tidyverse.org/blog/2020/04/dplyr-1-0-0-rowwise/). And this one for `case_when()` function: https://dplyr.tidyverse.org/reference/case_when.html (https://dplyr.tidyverse.org/reference/case_when.html).

```
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 × 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       0.597   0.295    0.0275   0.00690   0.0741      0.597 white
##  2 YILDIRIM     0.807   0.0273   0.0694   0.0165    0.0798      0.807 white
##  3 HAMILTON     0.656   0.239    0.0286   0.00750   0.0692      0.656 white
##  4 MOSHER       0.915   0.00425  0.0291   0.00917   0.0427      0.915 white
##  5 BARR         0.784   0.120    0.0268   0.00830   0.0615      0.784 white
##  6 GRAY         0.640   0.252    0.0281   0.00748   0.0724      0.640 white
##  7 MCMILLIAN    0.322   0.554    0.0212   0.00340   0.0995      0.554 black
##  8 FORD         0.576   0.320    0.0275   0.00621   0.0697      0.576 white
##  9 STRZELECKA   0.472   0.171    0.220    0.0825    0.0543      0.472 white
## 10 KIM          0.0169  0.00282  0.00546  0.943     0.0319      0.943 Asian
## # ℹ 3,796 more rows
```

Let’s join the data back to the applications table.

```
# 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) max used  (Mb)
## Ncells  4787044 255.7    8055668 430.3  6633894 354.3
## Vcells 52133505 397.8   93537981 713.7 93464061 713.1
```

Examiner's tenure

To figure out the timespan for which we observe each examiner in the applications data, let's find the first and the last observed date for each examiner. We'll first get examiner IDs and application dates in a separate table, for ease of manipulation. We'll keep examiner ID (the field `examiner_id`), and earliest and latest dates for each application (`filing_date` and `appl_status_date` respectively). We'll use functions in package `lubridate` to work with date and time values.

```
#install.packages("lubridate")
library(lubridate) # to work with dates

examiner_dates <- applications %>%
  select(examiner_id, filing_date, appl_status_date)

examiner_dates
```

```
## # A tibble: 2,018,477 × 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
## # ℹ 2,018,467 more rows
```

The dates look inconsistent in terms of formatting. Let's make them consistent. We'll create new variables `start_date` and `end_date` .

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

```
head(examiner_dates)
```

```
## # A tibble: 6 × 5
##   examiner_id filing_date appl_status_date   start_date end_date
##         <dbl> <date>      <chr>              <date>     <date>
## 1       96082 2000-01-26  30jan2003 00:00:00 2000-01-26 2003-01-30
## 2       87678 2000-10-11  27sep2010 00:00:00 2000-10-11 2010-09-27
## 3       63213 2000-05-17  30mar2009 00:00:00 2000-05-17 2009-03-30
## 4       73788 2001-07-20  07sep2009 00:00:00 2001-07-20 2009-09-07
## 5       77294 2000-04-10  19apr2001 00:00:00 2000-04-10 2001-04-19
## 6       68606 2000-04-28  16jul2001 00:00:00 2000-04-28 2001-07-16
```

Let's now identify the earliest and the latest date for each examiner and calculate the difference in days, which is their tenure in the organization.

```
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 × 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
## # ℹ 5,615 more rows
```

Joining back to the applications data.

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

rm(examiner_dates)
gc()
```

```
##            used  (Mb) gc trigger  (Mb)  max used  (Mb)
## Ncells  4795560 256.2    8055668 430.3   8055668 430.3
## Vcells 58208639 444.1  112325577 857.0 111576798 851.3
```

In order to keep these steps saved, I will save the file and reload the data back into R.

```
write.csv(applications, "applications_updated.csv")

applications_updated = read.csv("applications_updated.csv")
```

Part 2:

Pick two workgroups you want to focus on (remember that a workgroup is represented by the first 3 digits of `examiner_art_unit` value)

How do they compare on examiners' demographics? Show summary statistics and plots.

```
#Identify workgroups
attach(applications_updated)
applications_updated$workgroup = substr(applications_updated$examiner_art_unit, 1, 3)
```

```
selected_workgroups = applications_updated %>% filter(workgroup %in% c("176", "213"))

head(selected_workgroups)
```

```
##    X application_number filing_date examiner_name_last examiner_name_first
## 1  1            8284457  2000-01-26             HOWARD          JACQUELINE
## 2  2            8413193  2000-10-11           YILDIRIM               BEKIR
## 3  5            8682726  2000-04-10               BARR             MICHAEL
## 4 18            8974843  2000-01-11              PEESO              THOMAS
## 5 41            9101427  2000-09-16               BARR             MICHAEL
## 6 71            9180120  2000-04-05               WONG              LESLIE
##   examiner_name_middle examiner_id examiner_art_unit uspc_class uspc_subclass
## 1                    V       96082              1764        508        273000
## 2                    L       87678              1764        208        179000
## 3                    E       77294              1762        427        430100
## 4                    R       77284              2132        713        168000
## 5                    E       77294              1762        427        008000
## 6                    A       60400              1761        426        637000
##   patent_number patent_issue_date abandon_date disposal_type appl_status_code
## 1       6521570        2003-02-18         <NA>           ISS              150
## 2       6440298        2002-08-27         <NA>           ISS              250
## 3          <NA>              <NA>   2000-12-27           ABN              161
## 4          <NA>              <NA>   2003-01-13           ABN              168
## 5       6455097        2002-09-24         <NA>           ISS              250
## 6          <NA>              <NA>   2001-06-15           ABN              161
##     appl_status_date   tc gender  race earliest_date latest_date tenure_days
## 1 30jan2003 00:00:00 1700 female white    2000-01-10  2016-04-01        5926
## 2 27sep2010 00:00:00 1700   <NA> white    2000-01-04  2016-09-09        6093
## 3 19apr2001 00:00:00 1700   male white    2000-01-03  2017-05-05        6332
## 4 23jan2003 00:00:00 2100   male white    2000-01-03  2017-04-28        6325
## 5 25oct2010 00:00:00 1700   male white    2000-01-03  2017-05-05        6332
## 6 21sep2001 00:00:00 1700 female Asian    2000-01-14  2017-05-22        6338
##   workgroup
## 1       176
## 2       176
## 3       176
## 4       213
## 5       176
## 6       176
```

```
if("workgroup" %in% colnames(selected_workgroups)) {
  grouped_data <- selected_workgroups %>%
    group_by(workgroup) %>%
    summarize(count = n()) # Example of counting the number of rows in each group

  print(grouped_data)
} else {
  print("Column 'workgroup' not found in 'selected_workgroups'")
}
```

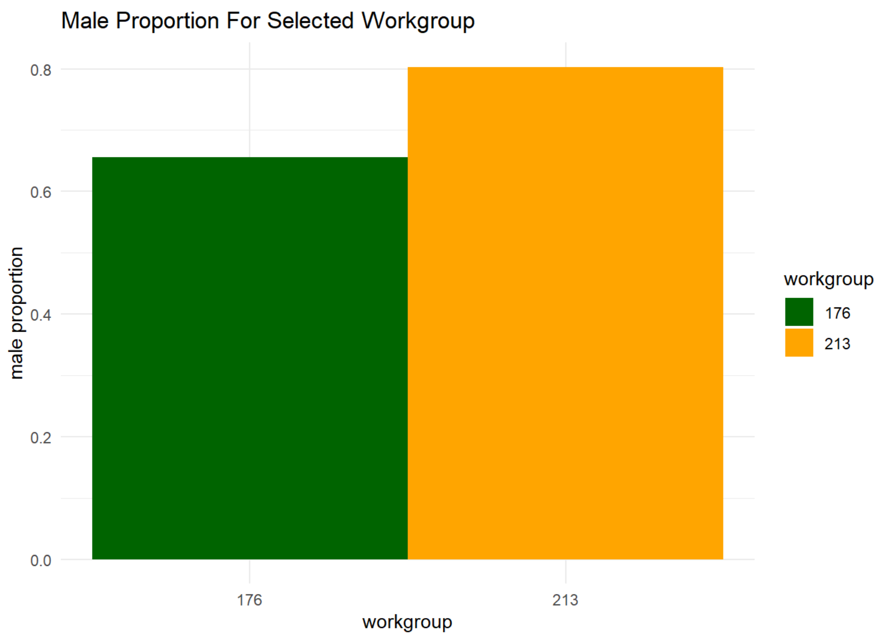
```
## # A tibble: 2 × 2
##   workgroup count
##   <chr>     <int>
## 1 176       91376
## 2 213       30257
```

```
demographic <- selected_workgroups %>%
  group_by(workgroup) %>%
  summarize(
    count = n(),
    proportion_white = mean(selected_workgroups$race.x=="white", na.rm = TRUE),
    proportion_female = mean(gender=="female", na.rm = TRUE),
    avg_tenure = mean(tenure_days, na.rm = TRUE)
  )

print(demographic)
```

```
## # A tibble: 2 × 5
##   workgroup count proportion_white proportion_female avg_tenure
##   <chr>     <int>            <dbl>             <dbl>      <dbl>
## 1 176       91376              NaN             0.344      5501.
## 2 213       30257              NaN             0.197      5371.
```

```
proportion_male = selected_workgroups %>% group_by(workgroup) %>% summarise(proportion_male = mean(gender == 'male', na.rm =
TRUE)) %>% ungroup()
```

```
#install.packages("ggplot2")
library(ggplot2)
ggplot(proportion_male, aes(x=workgroup, y=proportion_male, fill=workgroup)) + geom_bar(stat = "identity", width=1) + scale_
fill_manual(values=c("darkgreen","orange")) + labs(title = "Male Proportion For Selected Workgroup", x="workgroup",y="male p
roportion") + theme_minimal()
```



From the chart, workgroup 213 has a higher proportion of male members compared to workgroup 176, as evidenced by the height of the orange bar (for workgroup 213) relative to the green bar (for workgroup 176). There is a notable difference as shown in the visual comparison above.

```
{ggplot(selected_workgroups, aes(x = tenure_days, fill = workgroup)) + geom_histogram(position = "dodge", binwidth = 500) + labs(title = "Tenure
```

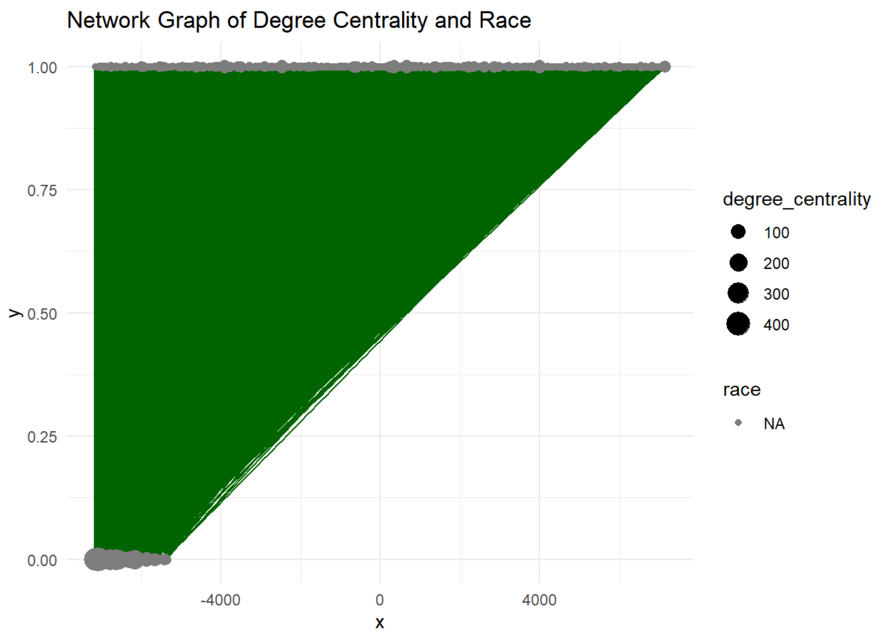
Part 3

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

```
filter_edges = edges %>% filter(application_number %in% applications_updated$application_number)

library(igraph)
network <- graph_from_data_frame(filter_edges, directed = TRUE)
degree_centrality = degree(network)
between_centrality = betweenness(network)
close_centrality = closeness(network)

applications_updated$degree_centrality <- degree_centrality[match(applications_updated$examiner_id, V(network)$name)]

applications_updated$between_centrality <- between_centrality[match(applications_updated$examiner_id, V(network)$name)]

applications_updated$close_centrality <- close_centrality[match(applications_updated$examiner_id, V(network)$name)]
```

Recall that we select workgroups “176”, “213”

```r
library(dplyr)
library(igraph)
library(ggraph)

# Filter the applications for selected workgroups
selected_examiners = applications_updated %>% filter(workgroup %in% c("176", "213"))

# Calculate degree centrality
V(network)$degree_centrality = degree(network)

# Create a Race data frame with unique examiner_id and race.x values
Race = applications_updated %>%
  select(examiner_id, race) %>%
  distinct()

# Convert examiner_id to character to ensure matching
Race$examiner_id = as.character(Race$examiner_id)

# Assign race to each vertex in the network
for (i in seq_along(V(network))) {
  vertex_name <- V(network)$name[i]
  V(network)$race[i] <- Race$race[Race$examiner_id == vertex_name]
}

# Visualize the network
ggraph(network, layout = "tree") +
  geom_edge_link(color = "darkgreen") +
  geom_node_point(aes(size = degree_centrality, color = race)) +
  scale_color_manual(values = c("Category1" = "red", "Category2" = "blue", "Category3" = "green")) +
  theme_minimal() +
  labs(title = "Network Graph of Degree Centrality and Race")
```


Network Graph of Degree Centrality and Race
y 0.50
x
degree_centrality
race
NA