Assignment_3

Ximena_Rodriguez

2023-03-28

Load data

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

```
data_path <- "C:/Users/ximen/OneDrive/MMA/4/Organizational Network/Advice_network/"
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 × 16
##    applicat…¹ filing_d…² exami…³ exami…⁴ exami…⁵ exami…⁶ exami…⁷ uspc_…⁸ uspc_…⁹
##    <chr>      <date>     <chr>   <chr>   <chr>     <dbl>   <dbl> <chr>   <chr>
##  1 08284457   2000-01-26 HOWARD  JACQUE… V         96082    1764 508     273000
##  2 08413193   2000-10-11 YILDIR… BEKIR   L         87678    1764 208     179000
##  3 08531853   2000-05-17 HAMILT… CYNTHIA <NA>      63213    1752 430     271100
##  4 08637752   2001-07-20 MOSHER  MARY    <NA>      73788    1648 530     388300
##  5 08682726   2000-04-10 BARR    MICHAEL E         77294    1762 427     430100
##  6 08687412   2000-04-28 GRAY    LINDA   LAMEY     68606    1734 156     204000
##  7 08716371   2004-01-26 MCMILL… KARA    RENITA    89557    1627 424     401000
##  8 08765941   2000-06-23 FORD    VANESSA L         97543    1645 424     001210
##  9 08776818   2000-02-04 STRZEL… TERESA  E         98714    1637 435     006000
## 10 08809677   2002-02-20 KIM     SUN     U         65530    1723 210     645000
## # … with 2,018,467 more rows, 7 more variables: patent_number <chr>,
## #   patent_issue_date <date>, abandon_date <date>, disposal_type <chr>,
## #   appl_status_code <dbl>, appl_status_date <chr>, tc <dbl>, and abbreviated
## #   variable names ¹application_number, ²filing_date, ³examiner_name_last,
## #   ⁴examiner_name_first, ⁵examiner_name_middle, ⁶examiner_id,
## #   ⁷examiner_art_unit, ⁸uspc_class, ⁹uspc_subclass
```

```
edges
```

```
## # A tibble: 32,906 × 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
```

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:

```
library(gender)
```

```
## Warning: package 'gender' was built under R version 4.2.3
```

```
# get a list of first names without repetitions
examiner_names <- applications %>%
  distinct(examiner_name_first)

examiner_names
```

```
## # A tibble: 2,595 × 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
```

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`

```
# 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 × 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
##  7 ABDULLAHI           male              0
##  8 ABIGAIL             female            0.998
##  9 ABIMBOLA            female            0.944
## 10 ABRAHAM             male              0.0031
## # … with 1,812 more rows
```

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 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) max used  (Mb)
## Ncells  4537828 242.4    7569784 404.3  5064867 270.5
## Vcells 49655810 378.9   95598063 729.4 79971583 610.2
```

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)
```

```
## Warning: package 'wru' was built under R version 4.2.3
```

```
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
## # … with 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()
```

```
## 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
## # … with 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
## # … with 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  4690142 250.5    7569784 404.3  7569784 404.3
## Vcells 54018001 412.2   95598063 729.4 94210055 718.8
```

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.

```
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
## # … with 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)))
```

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
## # … with 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  4696581 250.9   13592681  726.0  13592681  726.0
## Vcells 64282501 490.5  137837210 1051.7 137572263 1049.6
```

Exploratory Data Analysis

```
wg1 <- applications %>%
  filter(str_starts(examiner_art_unit, "176")) %>%
  arrange(application_number)

wg2 <- applications %>%
  filter(str_starts(examiner_art_unit, "212")) %>%
  arrange(application_number)
```

In the first group, we can see there is a majority of the white race, more men than women and also we can observe some outliers in tenure.

```
# Distributions for Workgroup 1
library(ggplot2)

# Color palette for plots
my_colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b")

# Plot race
p1 <- wg1 %>%
  group_by(race) %>%
  summarise(n_examiners = n_distinct(examiner_id)) %>%
  ggplot(aes(x = race, y = n_examiners, fill = race)) +
  geom_bar(stat ='identity') +
  scale_fill_manual(values = my_colors)

# Plot gender
p2 <- wg1 %>%
  group_by(gender) %>%
  summarise(n_examiners = n_distinct(examiner_id)) %>%
  ggplot(aes(x = gender, y = n_examiners, fill = gender)) +
  geom_bar(stat ='identity') +
  scale_fill_manual(values = c("#9467bd", "#8c564b", "#9b9b9b"))

# plot hist tenure
p3 <- ggplot(wg1, aes(x = tenure_days, fill = "Tenure")) +
  geom_histogram() +
  scale_fill_manual(values = my_colors)

library(gridExtra)
```

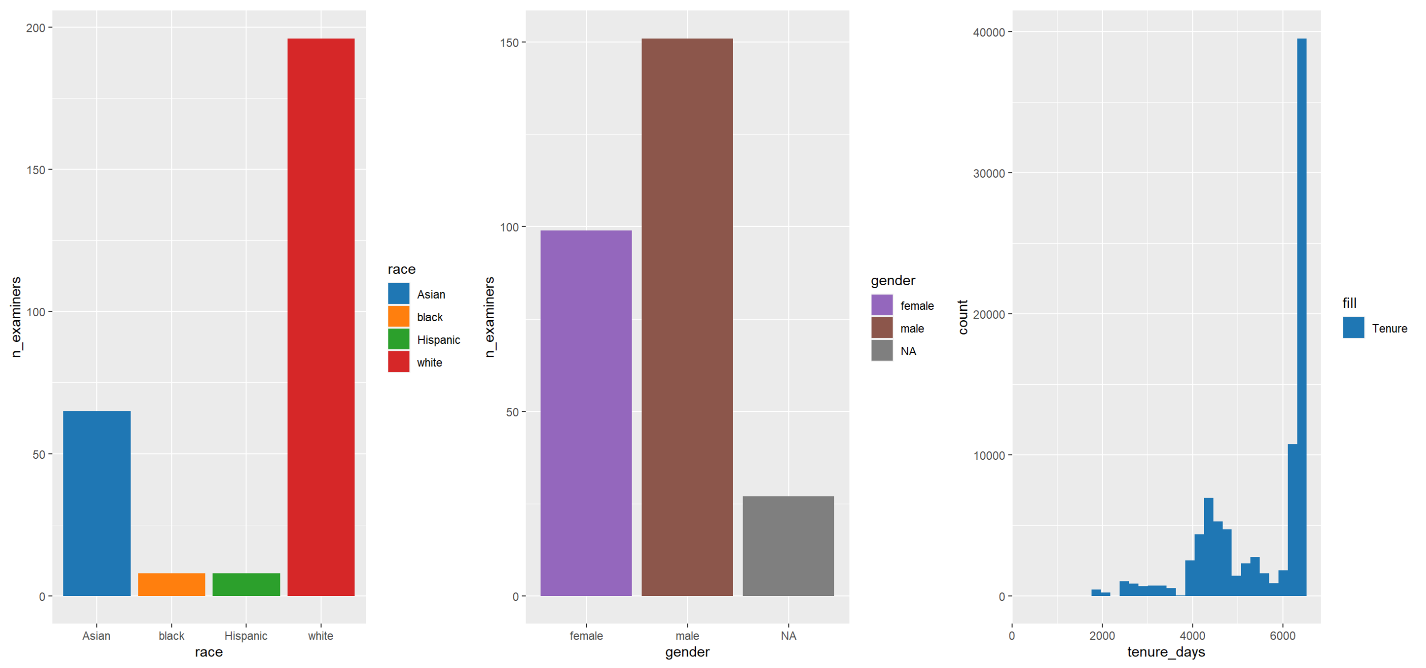
```
##
## Attaching package: 'gridExtra'
```

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

```
# Combine plots into a grid
grid.arrange(p1, p2, p3, ncol = 3)
```

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

```
## Warning: Removed 1017 rows containing non-finite values (`stat_bin()`).
```

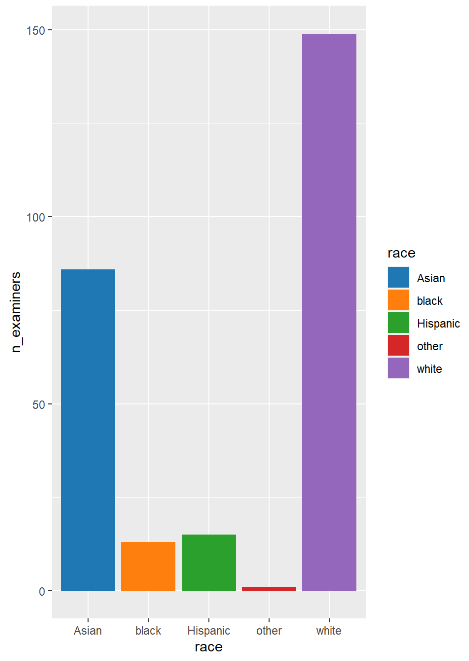


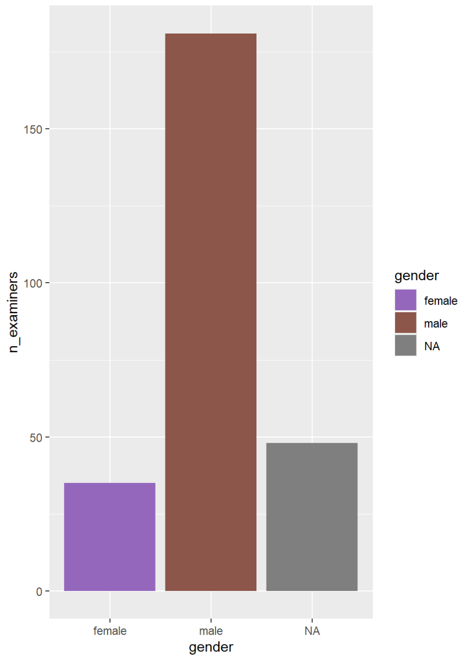
In the second group we can see there is also a majority of the white race, more men than women and also we can observe some outliers in tenure, the tenure graph is left-skewed.

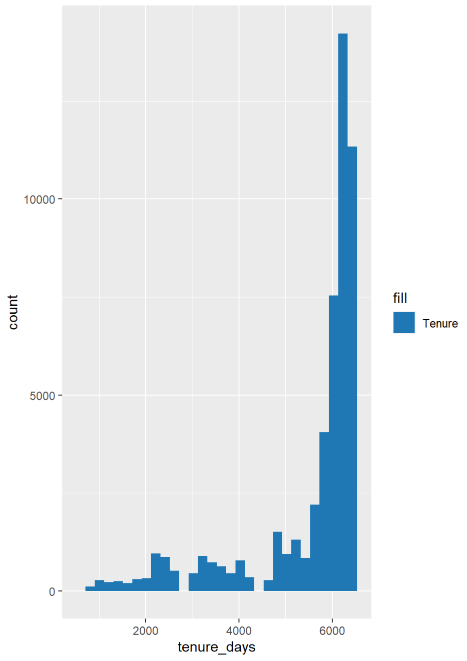
```
# Distributions for Workgroup 2

# Color palette for plots
my_colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b")

# Plot race
p1 <- wg2 %>%
  group_by(race) %>%
  summarise(n_examiners = n_distinct(examiner_id)) %>%
  ggplot(aes(x = race, y = n_examiners, fill = race)) +
  geom_bar(stat ='identity') +
  scale_fill_manual(values = my_colors)

# Plot gender
p2 <- wg2 %>%
  group_by(gender) %>%
  summarise(n_examiners = n_distinct(examiner_id)) %>%
  ggplot(aes(x = gender, y = n_examiners, fill = gender)) +
  geom_bar(stat ='identity') +
  scale_fill_manual(values = c("#9467bd", "#8c564b", "#9b9b9b"))

# plot hist tenure
p3 <- ggplot(wg2, aes(x = tenure_days, fill = "Tenure")) +
  geom_histogram() +
  scale_fill_manual(values = my_colors)

library(gridExtra)

# Combine plots into a grid
grid.arrange(p1, p2, p3, ncol = 3)
```

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

```
## Warning: Removed 107 rows containing non-finite values (`stat_bin()`).
```

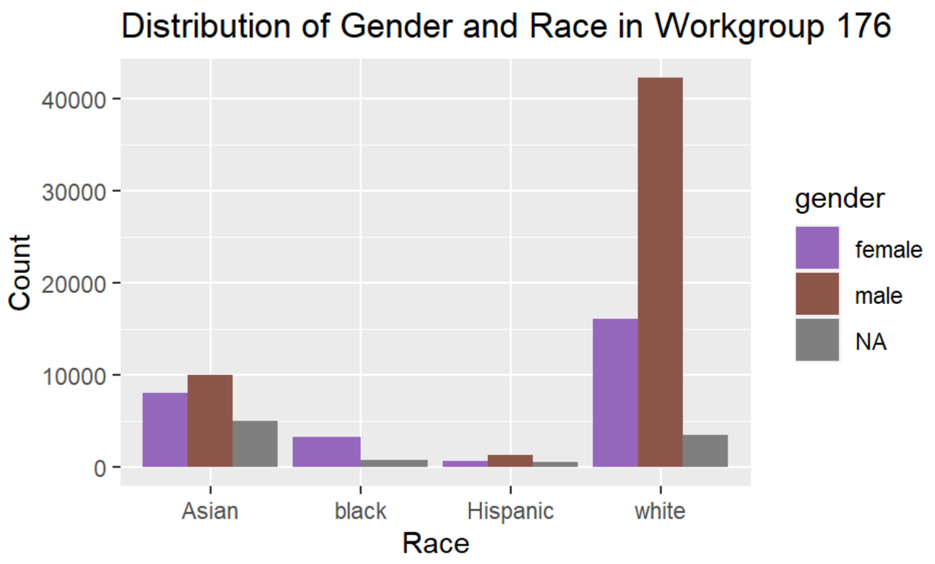




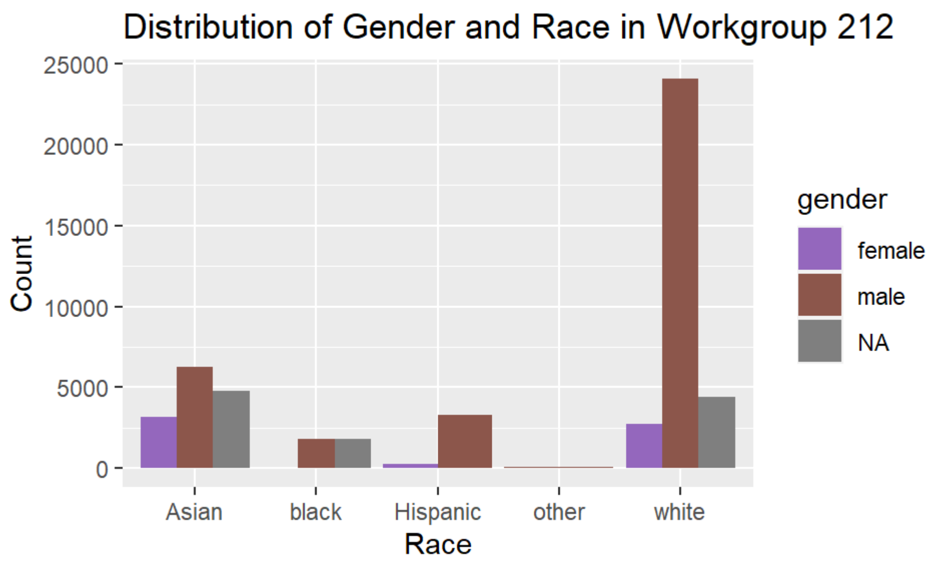


Let's see the distribution of gender and race at the same time:

```
ggplot(wg1, aes(x = race, fill = gender)) +
  geom_bar(position = "dodge") +
  xlab("Race") +
  ylab("Count") +
  ggtitle("Distribution of Gender and Race in Workgroup 176")+
  scale_fill_manual(values = c("#9467bd", "#8c564b", "#9b9b9b"))
```



```
ggplot(wg2, aes(x = race, fill = gender)) +
  geom_bar(position = "dodge") +
  xlab("Race") +
  ylab("Count") +
  ggtitle("Distribution of Gender and Race in Workgroup 212")+
  scale_fill_manual(values = c("#9467bd", "#8c564b", "#9b9b9b"))
```



Advice Network

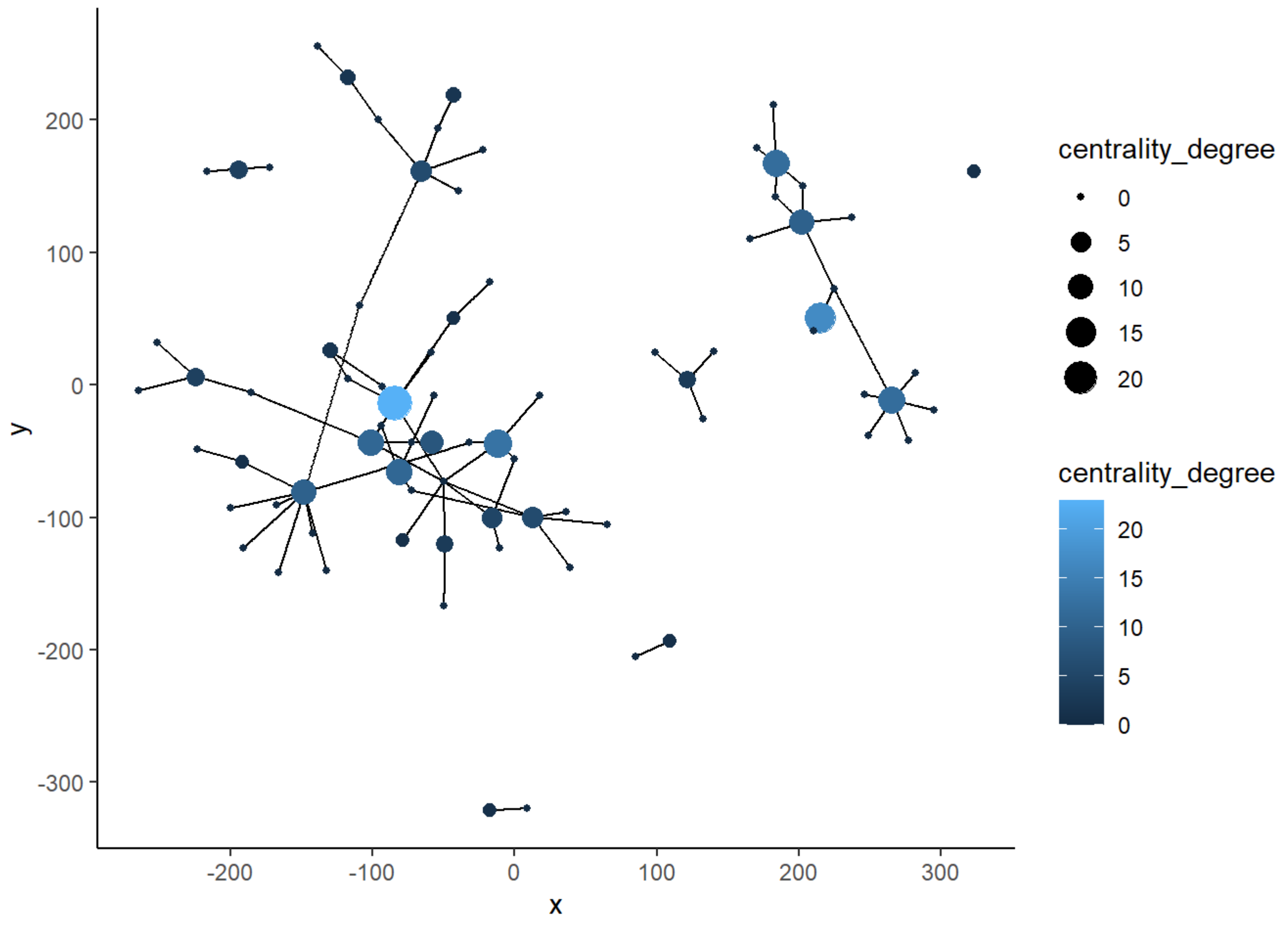
```
library(tidygraph)
```

```
##
## Attaching package: 'tidygraph'
```

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

```r
library(ggraph)
library(tidyverse)

#finding edges of first workgroup. We keep the edges of applications that belong to wg1
edges_wg1 <- edges %>% drop_na() %>% inner_join(wg1[c('application_number')], by = 'application_
number')



# Renaming columns 'alter_examiner_id' and 'ego_examiner_id' to 'to' and 'from'
edges_wg1 = edges_wg1 %>% rename(to = alter_examiner_id ,
                     from = ego_examiner_id )

# Creating a tbl_graph from the edges_wg1 data frame
graph <- as_tbl_graph(x = edges_wg1[c('to', 'from')])

# Adding degree centrality to the nodes
graph <- graph %>%
  activate(nodes) %>%
  mutate(centrality_degree = centrality_degree())

# Plotting the graph using ggraph
ggraph(graph, layout = 'graphopt') +
  geom_edge_link() +
  geom_node_point(aes(size = centrality_degree, colour = centrality_degree)) +
  theme_classic()
```

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` in the `default_aes` field and elsewhere instead.
```



Degree Centrality

```
# Compute degree centrality for each node in the graph
deg_centrality <- graph %>%
  activate(nodes) %>%
  mutate(centrality_degree = centrality_degree())

# Convert the resulting tbl_graph to a data frame
nodes_df <- as.data.frame(deg_centrality)

# Print summary statistics for the degree centrality scores
summary(nodes_df$centrality_degree)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.000   0.000   2.152   1.500  23.000
```

Betweeness Centrality

```
# Compute degree centrality and betweenness centrality for each node in the graph
bet_centrality <- graph %>%
  activate(nodes) %>%
  mutate(centrality_betweenness = centrality_betweenness())

# Convert the resulting tbl_graph to a data frame
nodes_df <- as.data.frame(bet_centrality)

# Print summary statistics for the centrality scores

summary(nodes_df$centrality_betweenness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.02532 0.00000 1.00000
```

Degree Centrality by Race

```
# Convert examiner_id to character type in workgroup 1
wg1$examiner_id = as.character(wg1$examiner_id)

# Calculate centrality degree for workgroup 1 nodes and convert to a data frame
nodes_df = graph %>%
  activate(nodes) %>%
  mutate(centrality_degree = centrality_degree()) %>%
  as.data.frame()

# Join workgroup 1 data with centrality degree data and calculate mean centrality degree by race
wg1_race_centrality = nodes_df %>%
  left_join(wg1, by = c('name' = 'examiner_id')) %>%
  select(name, centrality_degree, race) %>%
  distinct() %>%
  group_by(race) %>%
  summarise(mean_degree = mean(centrality_degree), n_examiners = n())
```

```
## Warning in left_join(., wg1, by = c(name = "examiner_id")): Each row in `x` is expected to ma
tch at most 1 row in `y`.
## ℹ Row 1 of `x` matches multiple rows.
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

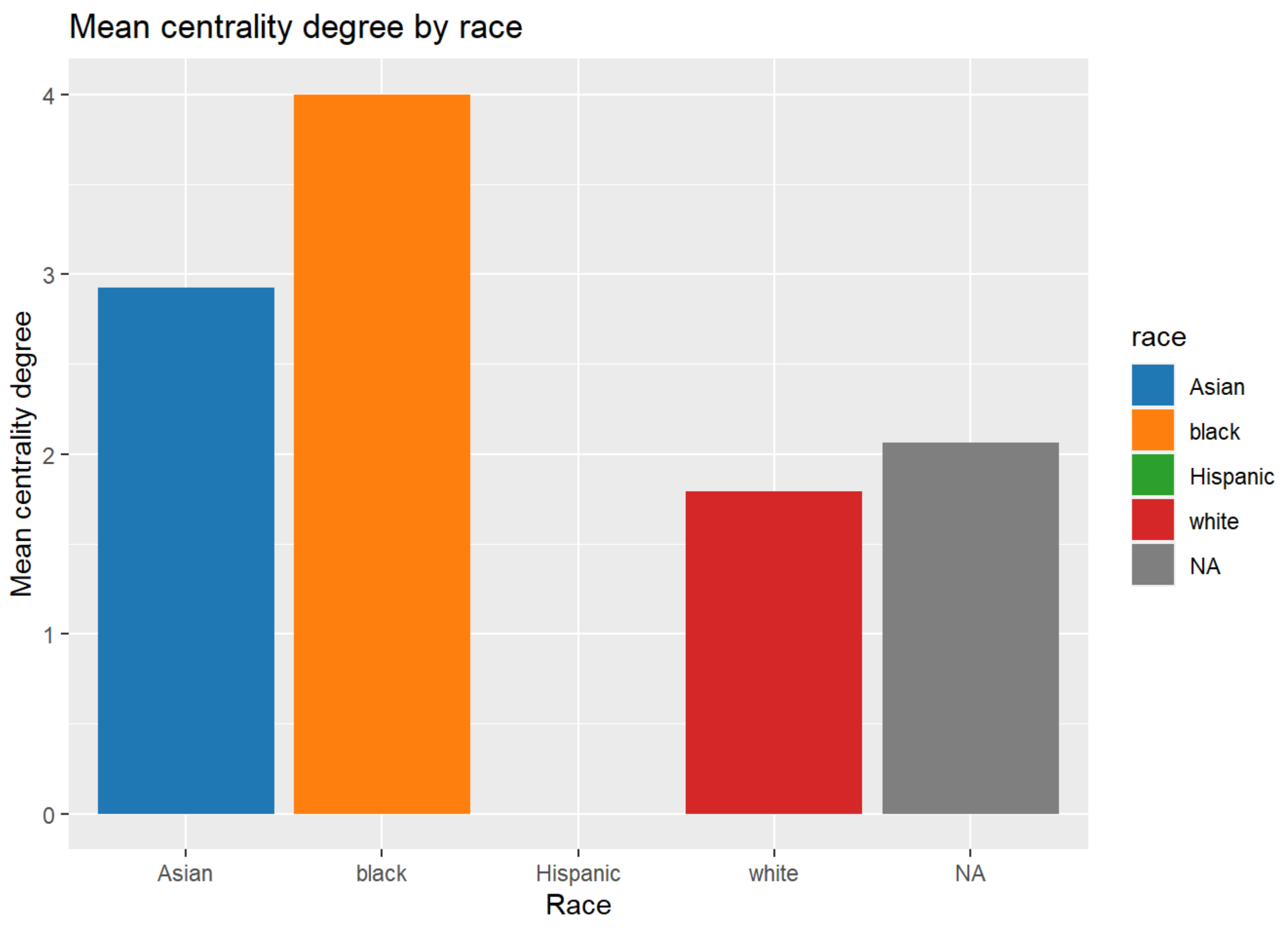
```
# View the resulting data frame
wg1_race_centrality
```

```
## # A tibble: 5 × 3
##   race     mean_degree n_examiners
##   <chr>          <dbl>       <int>
## 1 Asian           2.92          13
## 2 Hispanic        0              1
## 3 black           4              3
## 4 white           1.79          29
## 5 <NA>            2.06          33
```

We can see that even when the majority of employees is white, the ones that have higher centrality are black and asian.

```
ggplot(wg1_race_centrality, aes(x = race, y = mean_degree, fill = race)) +
  geom_col() +
  scale_fill_brewer(type = 'qual', palette = 'Set1') +
  labs(title = "Mean centrality degree by race",
       x = "Race",
       y = "Mean centrality degree") +
  scale_fill_manual(values = my_colors)
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



Degree Centrality by Gender

```
# Join workgroup 1 data with centrality degree data and calculate mean centrality degree by race
wg1_gen_centrality = nodes_df %>%
  left_join(wg1, by = c('name' = 'examiner_id')) %>%
  select(name, centrality_degree, gender) %>%
  distinct() %>%
  group_by(gender) %>%
  summarise(mean_degree = mean(centrality_degree), n_examiners = n())
```

```
## Warning in left_join(., wg1, by = c(name = "examiner_id")): Each row in `x` is expected to ma
tch at most 1 row in `y`.
## ℹ Row 1 of `x` matches multiple rows.
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

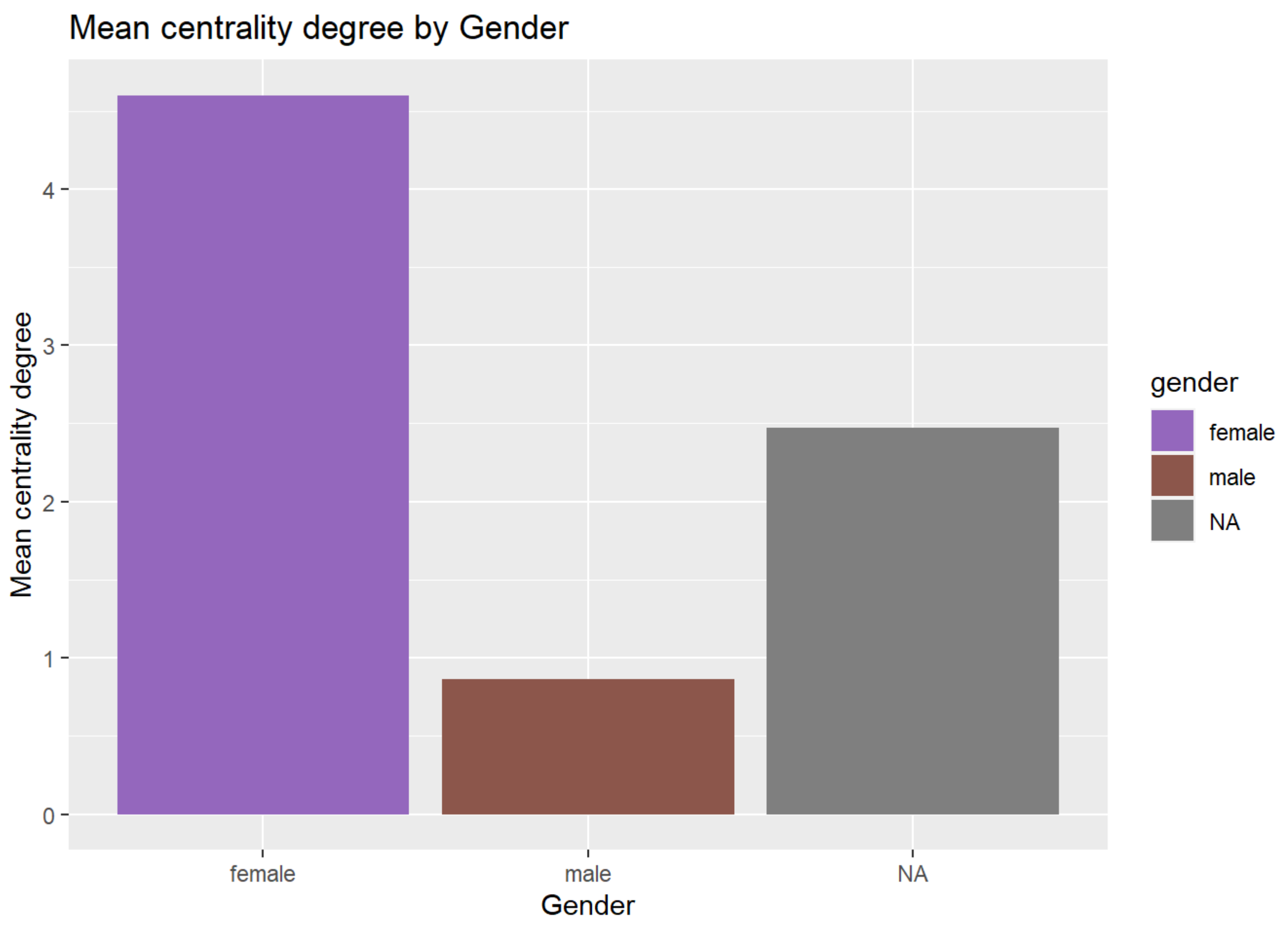
```
# View the resulting data frame
wg1_gen_centrality
```

```
## # A tibble: 3 × 3
##   gender mean_degree n_examiners
##   <chr>        <dbl>       <int>
## 1 female       4.6            10
## 2 male         0.862          29
## 3 <NA>         2.48           40
```

In the same way, as we saw in the race category, when analyzing gender one can see that women tend to ask for more advice even though the males are the vast majority of this sample.

```
ggplot(wg1_gen_centrality, aes(x = gender, y = mean_degree, fill = gender)) +
  geom_col() +
  scale_fill_brewer(type = 'qual', palette = 'Set1') +
  labs(title = "Mean centrality degree by Gender",
       x = "Gender",
       y = "Mean centrality degree") +
  scale_fill_manual(values = c("#9467bd", "#8c564b", "#9b9b9b"))
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```


Mean centrality degree by Gender
Mean centrality degree
0
1
2
3
4
female
male
NA
Gender
gender
female
male
NA