A3

2024-04-01

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

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v purrr     1.0.2
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.2
```

```
##
## Attaching package: 'igraph'
##
## 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: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:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
```

```
##
##     union
```

```
library(gender)
library(lubridate)
library(dplyr)
library(gtsummary)
library(arrow)
```

```
##
## Attaching package: 'arrow'
##
## The following object is masked from 'package:lubridate':
##
##     duration
##
## The following object is masked from 'package:utils':
##
##     timestamp
```

```
library(tidyr)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
data_path <- "/Users/apple/Desktop/Network Analysis/A3/672_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.
```

```
#check missing values in name columns
sum(is.na(applications$examiner_name_first))
```

```
## [1] 0
```

```
sum(is.na(applications$examiner_name_last))
```

```
## [1] 0
```

```
# 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
## # i 2,585 more rows
```

```
#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 x 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
## # i 1,812 more rows
```

```
# 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  4620084 246.8    8247017 440.5         NA  4639586 247.8
## Vcells 49818248 380.1   93370866 712.4      16384 80134123 611.4
```

```
#Race
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
## # i 3,796 more rows
```

```
library(wru)
```

```
## Warning: package 'wru' was built under R version 4.3.2
```

```
##
## 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.
```

```
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 x 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
```

```
# pick the race category that has the highest probability for each last name
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       0.597   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
```

```
#Join back to application 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) limit (Mb) max used  (Mb)
## Ncells  4825709 257.8    8247017 440.5         NA  7541263 402.8
## Vcells 52201934 398.3   93370866 712.4      16384 92735908 707.6
```

```r
#Tenure
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
## # i 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
#identify the earliest and the latest date for each examiner and calculate the difference in days, whic
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
## # i 5,615 more rows
```

```r
#join back to application table
applications <- applications %>%
```

```
  left_join(examiner_dates, by = "examiner_id")

rm(examiner_dates)
gc()
```

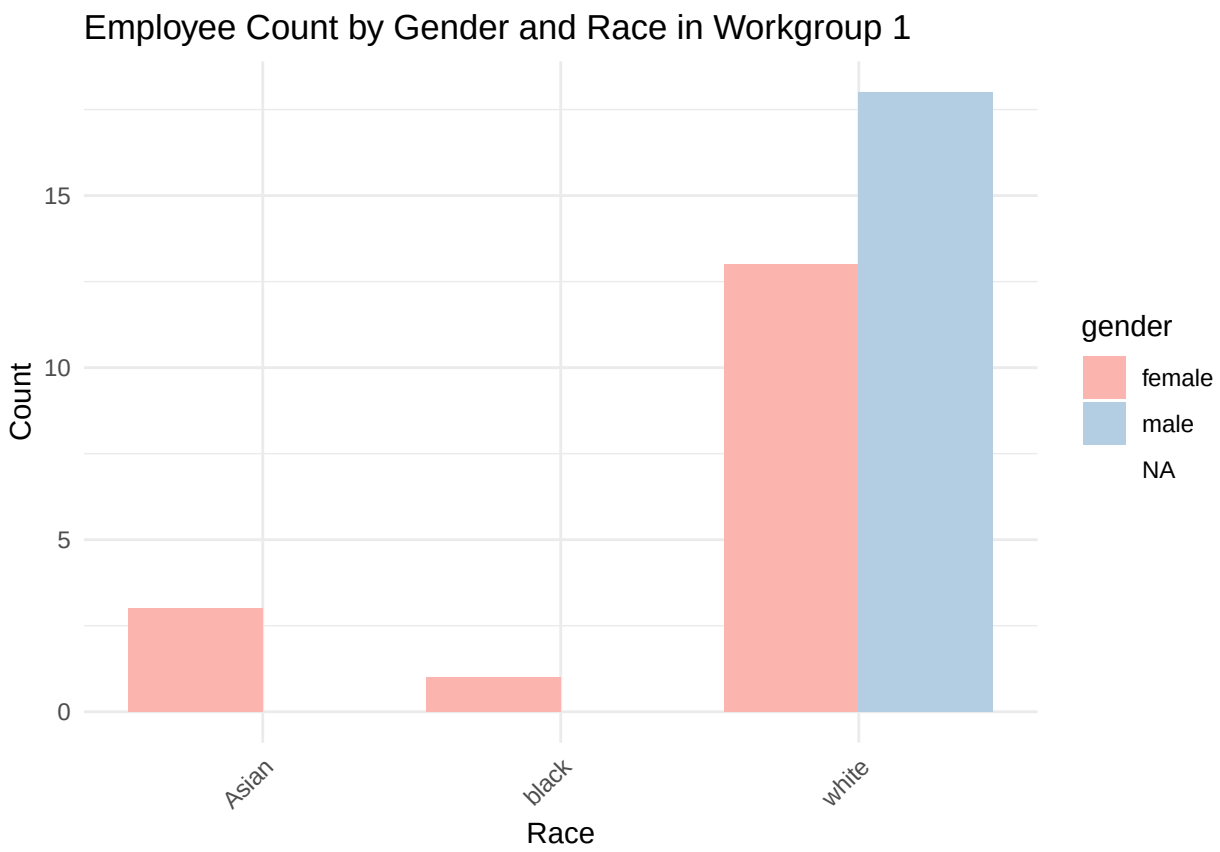
```
##            used  (Mb) gc trigger  (Mb) limit (Mb)  max used  (Mb)
## Ncells  4834214 258.2    8247017 440.5         NA   8247017 440.5
## Vcells 58276996 444.7  112125039 855.5      16384 111690702 852.2
```

```
# Filter the data for workgroup 1 and workgroup 2
workgroup1 <- applications %>%
  filter(substr(examiner_art_unit, 1, 3) == "160")

workgroup2 <- applications %>%
  filter(substr(examiner_art_unit, 1, 3) == "170")
#155 observations in group 1, 45 obs in group 2.
```

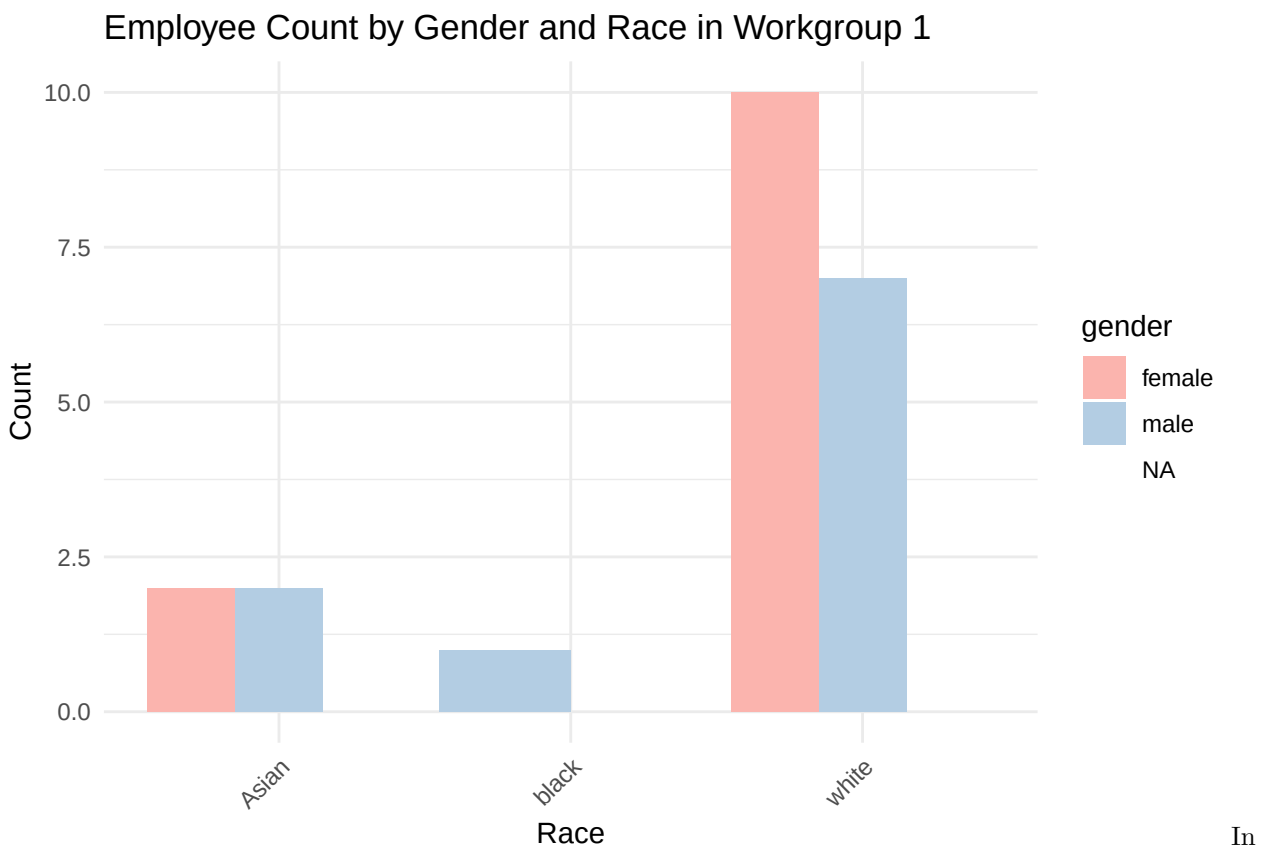
```
names(workgroup1)
```

```
##  [1] "application_number"   "filing_date"          "examiner_name_last"
##  [4] "examiner_name_first"  "examiner_name_middle" "examiner_id"
##  [7] "examiner_art_unit"    "uspc_class"           "uspc_subclass"
## [10] "patent_number"        "patent_issue_date"    "abandon_date"
## [13] "disposal_type"        "appl_status_code"     "appl_status_date"
## [16] "tc"                   "gender"               "race"
## [19] "earliest_date"        "latest_date"          "tenure_days"
```

```
# Count the number of employees in workgroup 1, grouped by gender and race
employee_count <- workgroup1 %>%
  group_by(gender, race) %>%
  summarise(count = n_distinct(examiner_id), .groups = 'drop')
ggplot(employee_count, aes(x = race, y = count, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Employee Count by Gender and Race in Workgroup 1",
       x = "Race",
       y = "Count") +
  scale_fill_brewer(palette = "Pastel1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Count the number of employees in workgroup 1, grouped by gender and race
employee_count_2 <- workgroup2 %>%
  group_by(gender, race) %>%
  summarise(count = n_distinct(examiner_id), .groups = 'drop')
ggplot(employee_count_2, aes(x = race, y = count, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Employee Count by Gender and Race in Workgroup 1",
       x = "Race",
       y = "Count") +
  scale_fill_brewer(palette = "Pastel1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

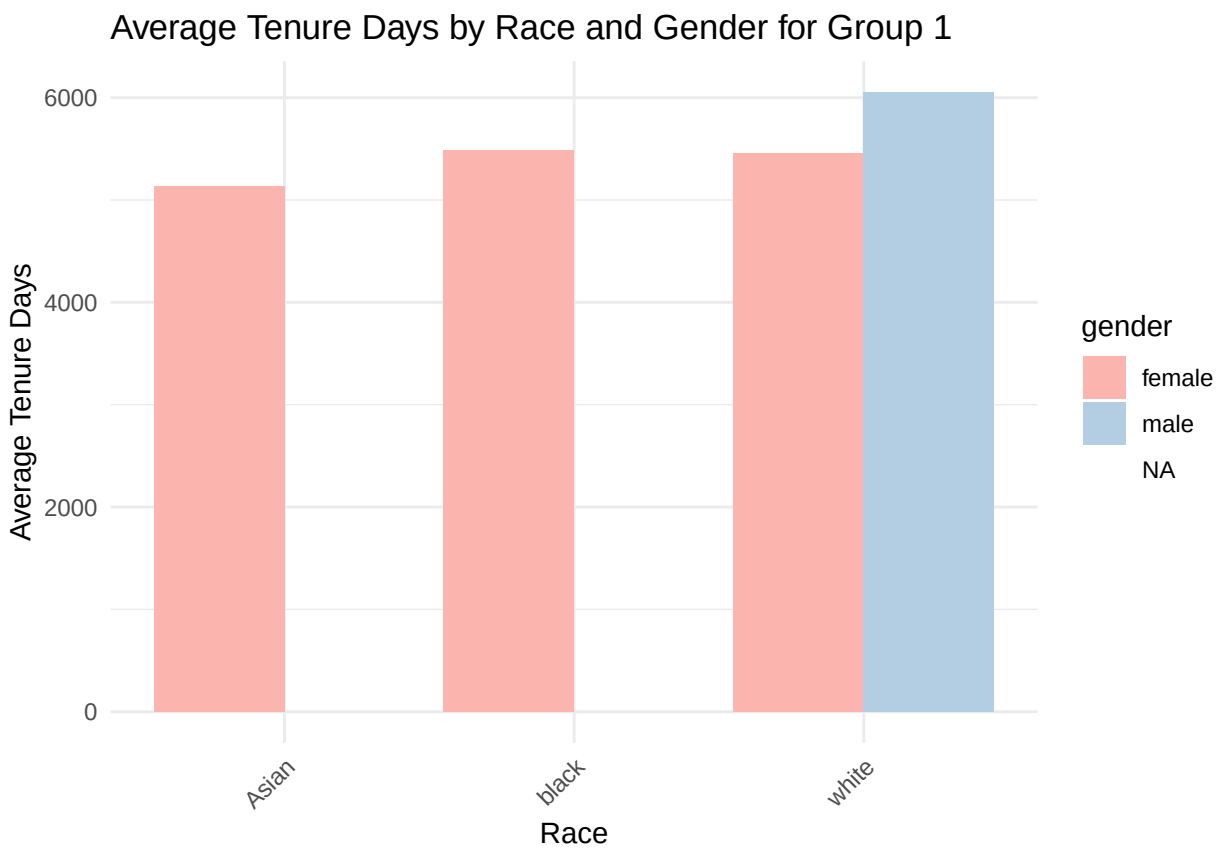


In every group, the number of male and female employee is sumilar, and the majority of employees are white. This shows a trend of white being the dominant demographic across thses two workgroups.

```
# Calculate average tenure days by race and gender
average_tenure_by_race_gender <- workgroup1 %>%
  group_by(race, gender) %>%
  summarise(average_tenure_days = mean(tenure_days, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'race'. You can override using the
## `.groups` argument.
```

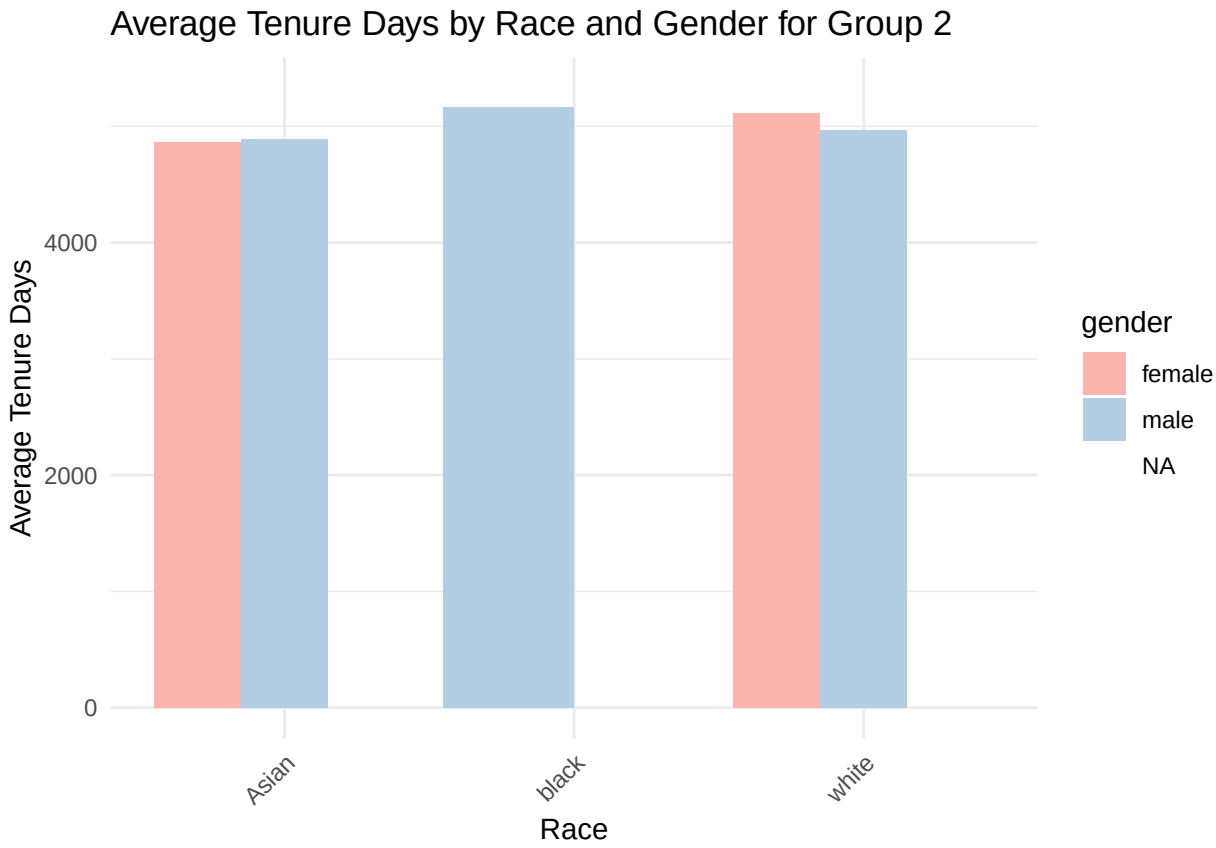
```
# Plotting
ggplot(average_tenure_by_race_gender, aes(x = race, y = average_tenure_days, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Average Tenure Days by Race and Gender for Group 1",
       x = "Race",
       y = "Average Tenure Days") +
  scale_fill_brewer(palette = "Pastel1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Calculate average tenure days by race and gender
average_tenure_by_race_gender_2 <- workgroup2 %>%
  group_by(race, gender) %>%
  summarise(average_tenure_days = mean(tenure_days, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'race'. You can override using the
## `.groups` argument.
```

```
# Plotting
ggplot(average_tenure_by_race_gender_2, aes(x = race, y = average_tenure_days, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Average Tenure Days by Race and Gender for Group 2",
       x = "Race",
       y = "Average Tenure Days") +
  scale_fill_brewer(palette = "Pastel1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
average_tenure_by_race_gender
```

```
## # A tibble: 6 x 3
##   race  gender average_tenure_days
##   <chr> <chr>                <dbl>
## 1 Asian female               5134.
## 2 Asian <NA>                 5196
## 3 black female               5490
## 4 black <NA>                 4919
## 5 white female               5459.
## 6 white male                 6051.
```

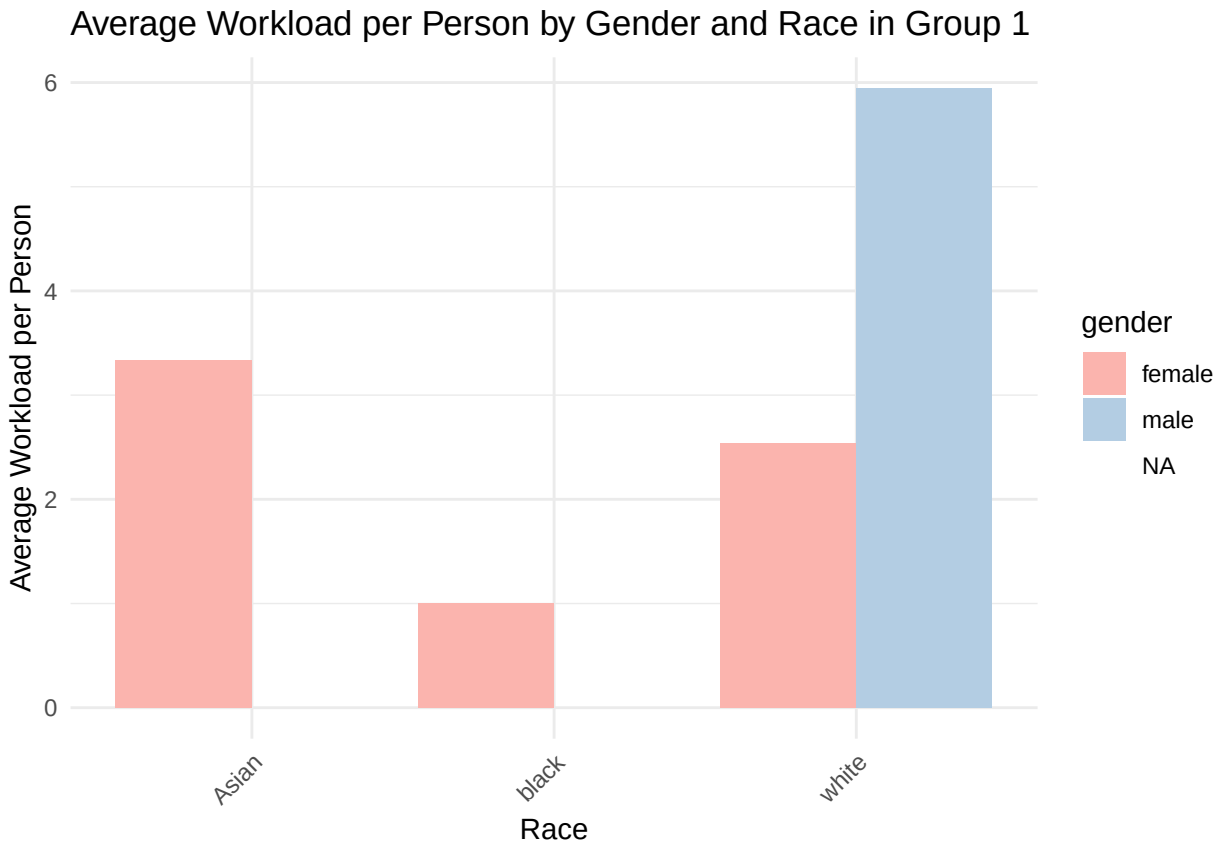
```
average_tenure_by_race_gender_2
```

```
## # A tibble: 8 x 3
##   race  gender average_tenure_days
##   <chr> <chr>                <dbl>
## 1 Asian female               4865.
## 2 Asian male                 4886.
## 3 Asian <NA>                 4902
## 4 black male                 5161
## 5 black <NA>                 5326
## 6 white female               5112.
## 7 white male                 4967.
## 8 white <NA>                 5042
```

The analysis of tenure between these two workgroups reveals a consistent pattern across different demographics, with the average tenure approximating 5000 days. This suggests that factors such as gender and race do not significantly impact the length of tenure within these groups. Furthermore, an examination within each

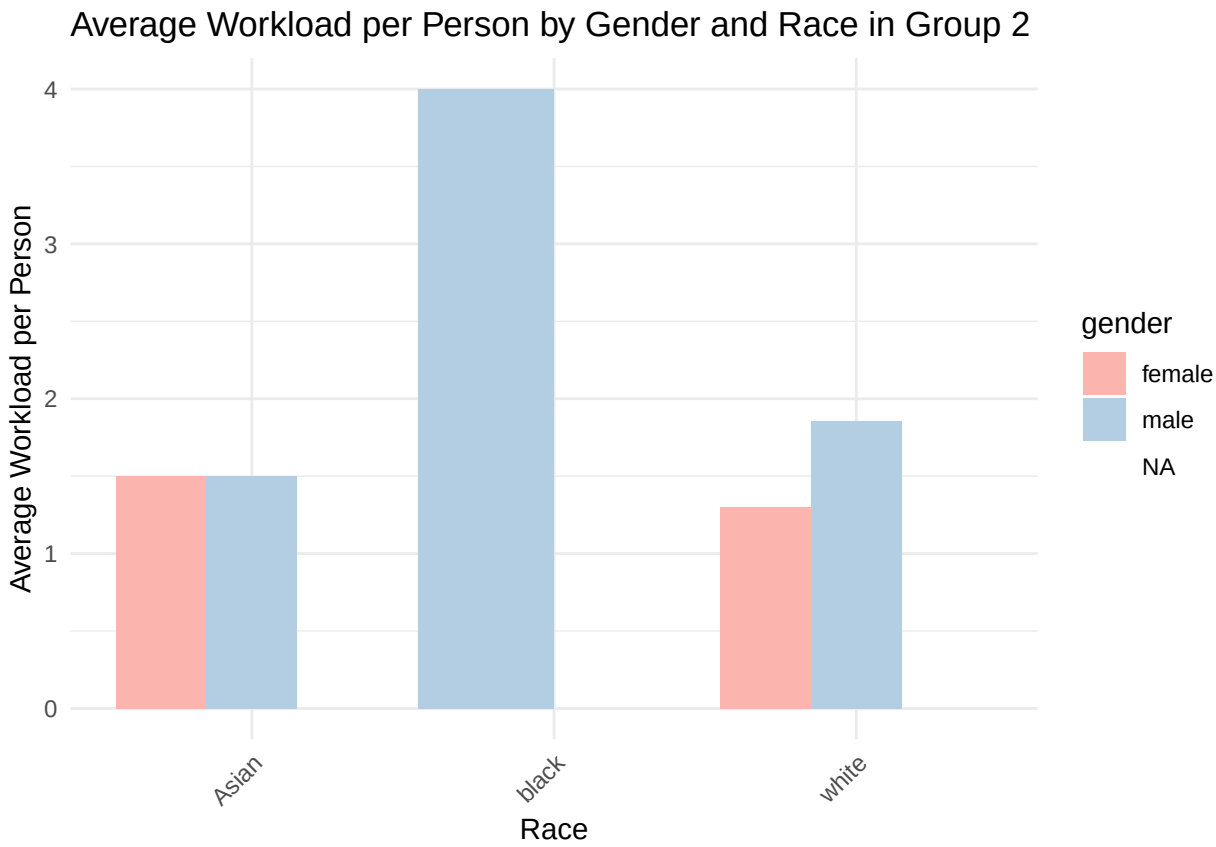
group highlights a similarity in average tenure, reinforcing the conclusion that these demographic variables do not influence tenure durations in a meaningful way.

```
# Calculating the total number of applications and the number of people per gender and race
workload_summary <- workgroup1 %>%
  group_by(gender, race) %>%
  summarise(total_applications = n_distinct(application_number),
            total_people = n_distinct(examiner_id), .groups = 'drop') %>%
  mutate(average_workload_per_person = total_applications / total_people)

# Plotting the average workload per person by gender and race
ggplot(workload_summary, aes(x = race, y = average_workload_per_person, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_brewer(palette = "Pastel1") +
  theme_minimal() +
  labs(title = "Average Workload per Person by Gender and Race in Group 1",
       x = "Race",
       y = "Average Workload per Person") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Calculating the total number of applications and the number of people per gender and race
workload_summary_2 <- workgroup2 %>%
  group_by(gender, race) %>%
  summarise(total_applications = n_distinct(application_number),
            total_people = n_distinct(examiner_id), .groups = 'drop') %>%
  mutate(average_workload_per_person = total_applications / total_people)

# Plotting the average workload per person by gender and race
```

```
ggplot(workload_summary_2, aes(x = race, y = average_workload_per_person, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_brewer(palette = "Pastel1") +
  theme_minimal() +
  labs(title = "Average Workload per Person by Gender and Race in Group 2",
       x = "Race",
       y = "Average Workload per Person") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The distribution of average workloads reveals notable disparities between groups. In Group 1, white males bear the highest average workload, each managing an average of 6 applications. Conversely, Group 2 sees black males shouldering the most substantial workload with an average of 4 applications per person.

```
names(edges)
```

```
## [1] "application_number" "advice_date"        "ego_examiner_id"
## [4] "alter_examiner_id"
```

```
#add the gender, race, and examiner_art_unit for both the ego and alter examiners from the applications

# Assuming examiner_id is unique within applications, we do not need to group by tenure_days
applications_profile <- applications %>%
  select(examiner_id, gender, race, examiner_art_unit) %>%
  distinct()

# Join for ego examiner info
edges_info <- edges %>%
  inner_join(applications_profile, by = c("ego_examiner_id" = "examiner_id")) %>%
  rename(
```

```
    ego_gender = gender,
    ego_race = race,
    ego_examiner_art_unit = examiner_art_unit
  )
```

```
## Warning in inner_join(., applications_profile, by = c(ego_examiner_id = "examiner_id")): Detected an
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 341 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Now for the alter examiner info. Ensure to use unique names for the second join to avoid overwriting
edges_info <- edges_info %>%
  inner_join(applications_profile, by = c("alter_examiner_id" = "examiner_id")) %>%
  rename(
    alter_gender = gender,
    alter_race = race,
    alter_examiner_art_unit = examiner_art_unit
  )
```

```
## Warning in inner_join(., applications_profile, by = c(alter_examiner_id = "examiner_id")): Detected a
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1801 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Finally, truncate the art unit codes to their first three characters
edges_info <- edges_info %>%
  mutate(
    ego_examiner_art_unit = substr(ego_examiner_art_unit, 1, 3),
    alter_examiner_art_unit = substr(alter_examiner_art_unit, 1, 3)
  )
```

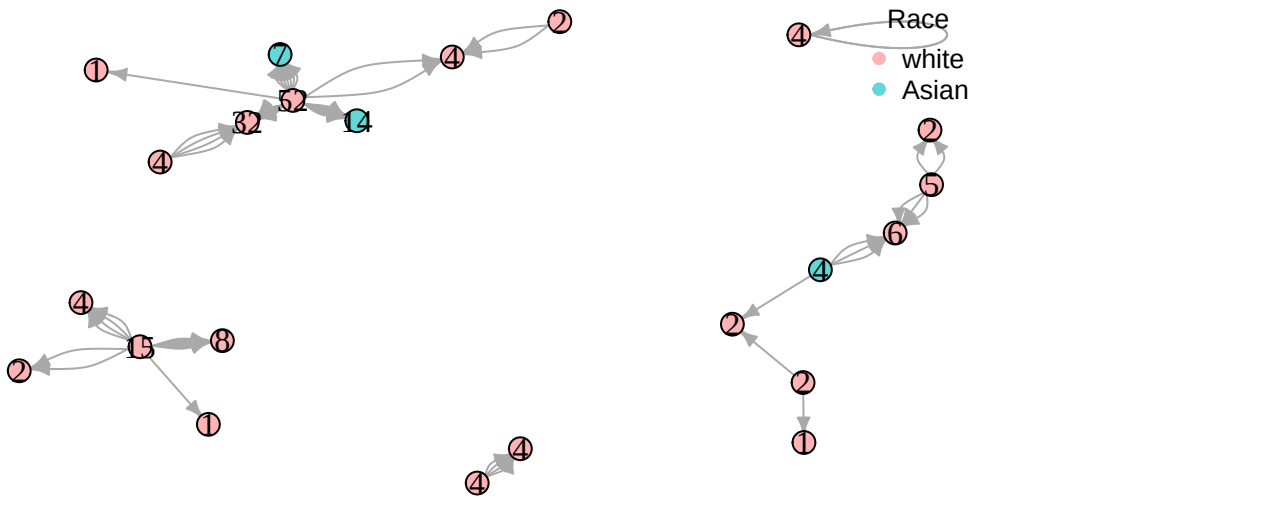
```
# Filter edges_info for ego_examiner_art_unit equal to "160"
edges_info_group1 <- edges_info %>%
  filter(ego_examiner_art_unit == "160")
```

```
names(edges_info_group1)
```

```
##  [1] "application_number"      "advice_date"
##  [3] "ego_examiner_id"         "alter_examiner_id"
##  [5] "ego_gender"              "ego_race"
##  [7] "ego_examiner_art_unit"   "alter_gender"
##  [9] "alter_race"              "alter_examiner_art_unit"
```

```
# Create a dataframe of unique examiner IDs with their race
ego_info <- edges_info_group1 %>%
  select(ego_examiner_id, ego_race) %>%
  distinct() %>%
  rename(examiner_id = ego_examiner_id, race = ego_race)

alter_info <- edges_info_group1 %>%
  select(alter_examiner_id, alter_race) %>%
  distinct() %>%
  rename(examiner_id = alter_examiner_id, race = alter_race)

# Combine ego and alter information, then remove duplicates
```

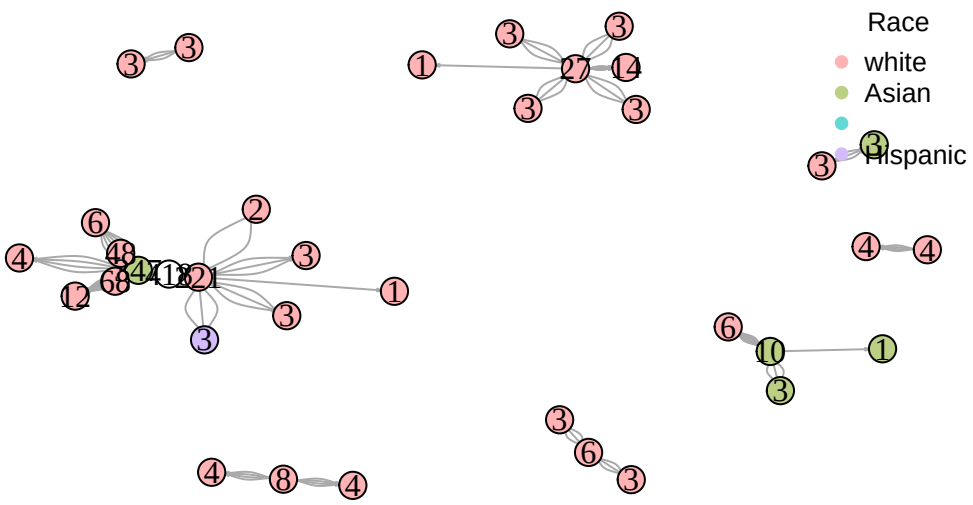
```r
unique_nodes_1 <- bind_rows(ego_info, alter_info) %>%
  distinct()

# Create the edge list for graph construction
edge_list_1 <- edges_info_group1 %>%
  select(ego_examiner_id, alter_examiner_id)

# Create the graph
g_1 <- graph_from_data_frame(d = edge_list_1, directed = TRUE)

# Extract race information for nodes in the graph
node_races <- unique_nodes_1$race[match(V(g_1)$name, unique_nodes_1$examiner_id)]

# Define colors for each unique race in the graph
unique_races <- unique(node_races)
race_colors <- setNames(hcl.colors(length(unique_races), "Set3"), unique_races)

# Ensure each node's race is set correctly
V(g_1)$race <- node_races

# Calculate degree centrality for each node
degree_centrality <- degree(g_1, mode = "all")

# Add the degree centrality as a label for each node
V(g_1)$label <- degree_centrality

# Plot the graph with labels and race-based colors
plot(g_1, vertex.color = race_colors[V(g_1)$race], asp = FALSE,
     main = "Network of Group 1 Examiners by Race", vertex.size = 5,
     edge.arrow.size = 0.5, vertex.label = V(g_1)$label, vertex.label.color = "black")

# Add a legend to the graph
legend("topright", legend = names(race_colors), col = race_colors, pch = 19,
       title = "Race", cex = 0.8, bty = "n")
```



```r
# Filter edges_info for ego_examiner_art_unit equal to "160"
edges_info_group2 <- edges_info %>%
  filter(ego_examiner_art_unit == "170")
# Create a dataframe of unique examiner IDs with their race
ego_info_2 <- edges_info_group2 %>%
  select(ego_examiner_id, ego_race) %>%
  distinct() %>%
  rename(examiner_id = ego_examiner_id, race = ego_race)

alter_info_2<- edges_info_group2 %>%
  select(alter_examiner_id, alter_race) %>%
  distinct() %>%
  rename(examiner_id = alter_examiner_id, race = alter_race)

# Combine ego and alter information, then remove duplicates
unique_nodes_2 <- bind_rows(ego_info_2, alter_info_2) %>%
  distinct()

# Create the edge list for graph construction
edge_list_2 <- edges_info_group2 %>%
  select(ego_examiner_id, alter_examiner_id)

# Create the graph
g_2 <- graph_from_data_frame(d = edge_list_2, directed = TRUE)
```

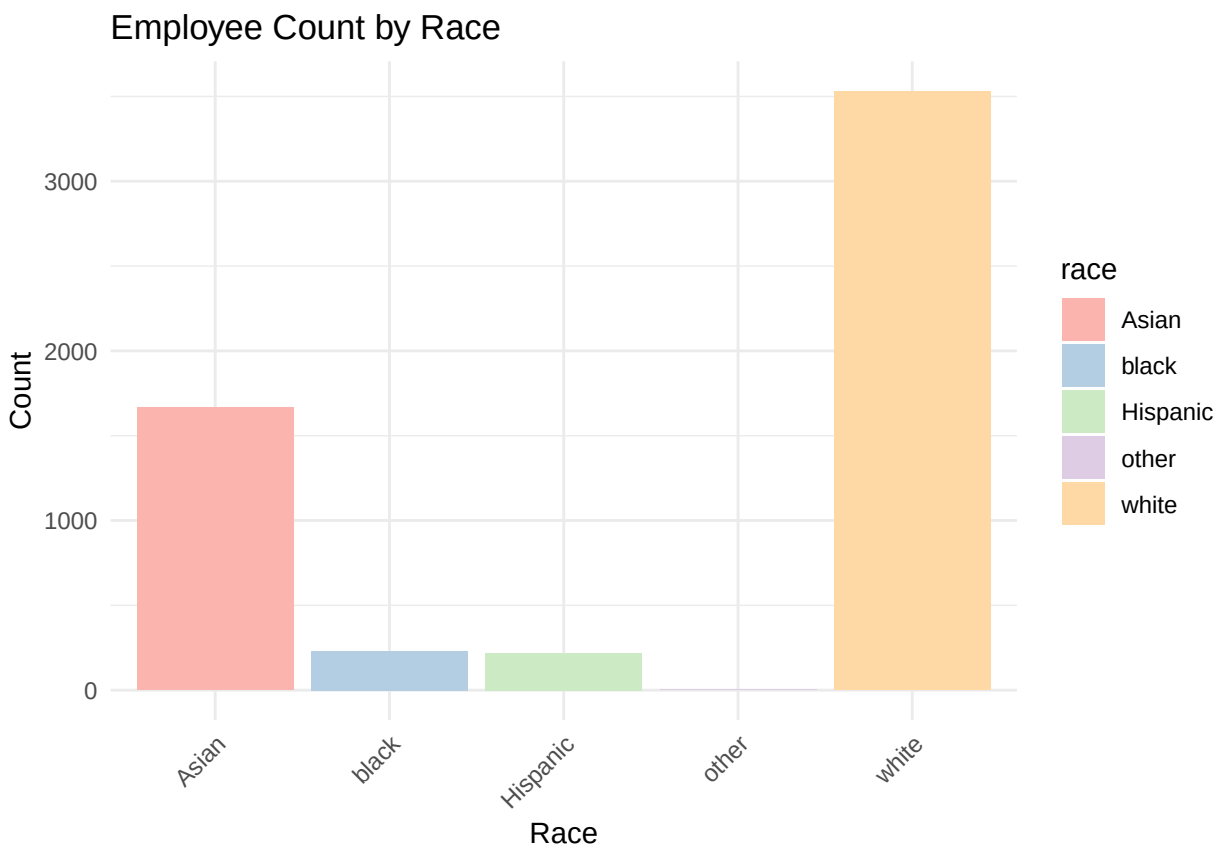
```
## Warning in graph_from_data_frame(d = edge_list_2, directed = TRUE): In `d' `NA'
## elements were replaced with string "NA"
```

```r
# Extract race information for nodes in the graph
node_races_2 <- unique_nodes_2$race[match(V(g_2)$name, unique_nodes_2$examiner_id)]

# Define colors for each unique race in the graph
unique_races_2 <- unique(node_races_2)
race_colors_2 <- setNames(hcl.colors(length(unique_races_2), "Set3"), unique_races_2)

# Ensure each node's race is set correctly
V(g_2)$race <- node_races_2

# Calculate degree centrality for each node
degree_centrality_2 <- degree(g_2, mode = "all")

# Add the degree centrality as a label for each node
V(g_2)$label <- degree_centrality_2

# Plot the graph with the new layout
plot(g_2, vertex.color = race_colors_2[V(g_2)$race], asp = FALSE,
     main = "Network of Group 2 Examiners by Race", vertex.size = 6,
     edge.arrow.size = 0.1, vertex.label = V(g_2)$label, vertex.label.color = "black")

# Add a legend to the graph
legend("topright", legend = names(race_colors_2), col = race_colors_2, pch = 19,
       title = "Race", cex = 0.8, bty = "n")
```



In the network graphs, each node's color shows the race of the examiner, and the number on the node indicates how many connections, or 'links', they have to others—this is their 'degree centrality'. The higher the number, the more connections they have. We're looking to see if examiners with a lot of connections tend to work more with people of their own race. While ego and alter examiners often come from different art units,it seems that examiners tend to connect with others of the same race based on the network graph, especially among white examiners. However, this observation alone isn't enough to conclude a preference for same-race interactions.For instance, in the second group's network graph, there's an Asian examiner with a high number of connections—a centrality score of 341—but most of their connections are with white examiners.Given that white examiners make up a larger portion of the USPTO workforce, as the employee count graph indicates,this might be why we see white examiners linked more often. It's not just about who prefers to work with whom; it's also about how many people there are of each race.

```
# Count the number of total employees , grouped by race
employee_totalcount <- applications %>%
  group_by(race) %>%
  summarise(count = n_distinct(examiner_id), .groups = 'drop')
ggplot(employee_totalcount, aes(x = race, y = count, fill = race)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Employee Count by Race ",
       x = "Race",
       y = "Count") +
  scale_fill_brewer(palette = "Pastel1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


Employee Count by Race
Count
3000
2000
1000
0
Asian
black
Hispanic
other
white
Race
race
Asian
black
Hispanic
other
white