Exercise 3

2023-05-18

Open data file

I’m going to load the data I saved after the last exercise.

```
library(arrow)
applications <- read_feather("C:/Users/vrind/OneDrive/Desktop/Course terms/Summer Term/People
Analytics/Week 2/app_data_starter.feather")

# lists all variables (columns) in the data
applications |> tbl_vars()
```

```
## <dplyr:::vars>
##  [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"
```

Look at examiners demographics

The first thing to note here is that our unit of interest is an *examiner*, but our data is at the level of a *patent application*. Examiners work with many patent applications during their tenure at the USPTO. Those who have longer tenure in our sample will have worked on more applications, and so if we count the number of *records* with attributes `male` or `female`, we will overcount those who have worked there longer.

We may be better off creating a separate table—a.k.a. a *dataframe*—where there is only one record per examiner. In other words, we need to “collapse” the applications data, with multiple records per examiner, to examiner-level data, where we only have one record per individual.

```
library(dplyr)
applications %>%
  distinct(examiner_id) %>%
  count()
```

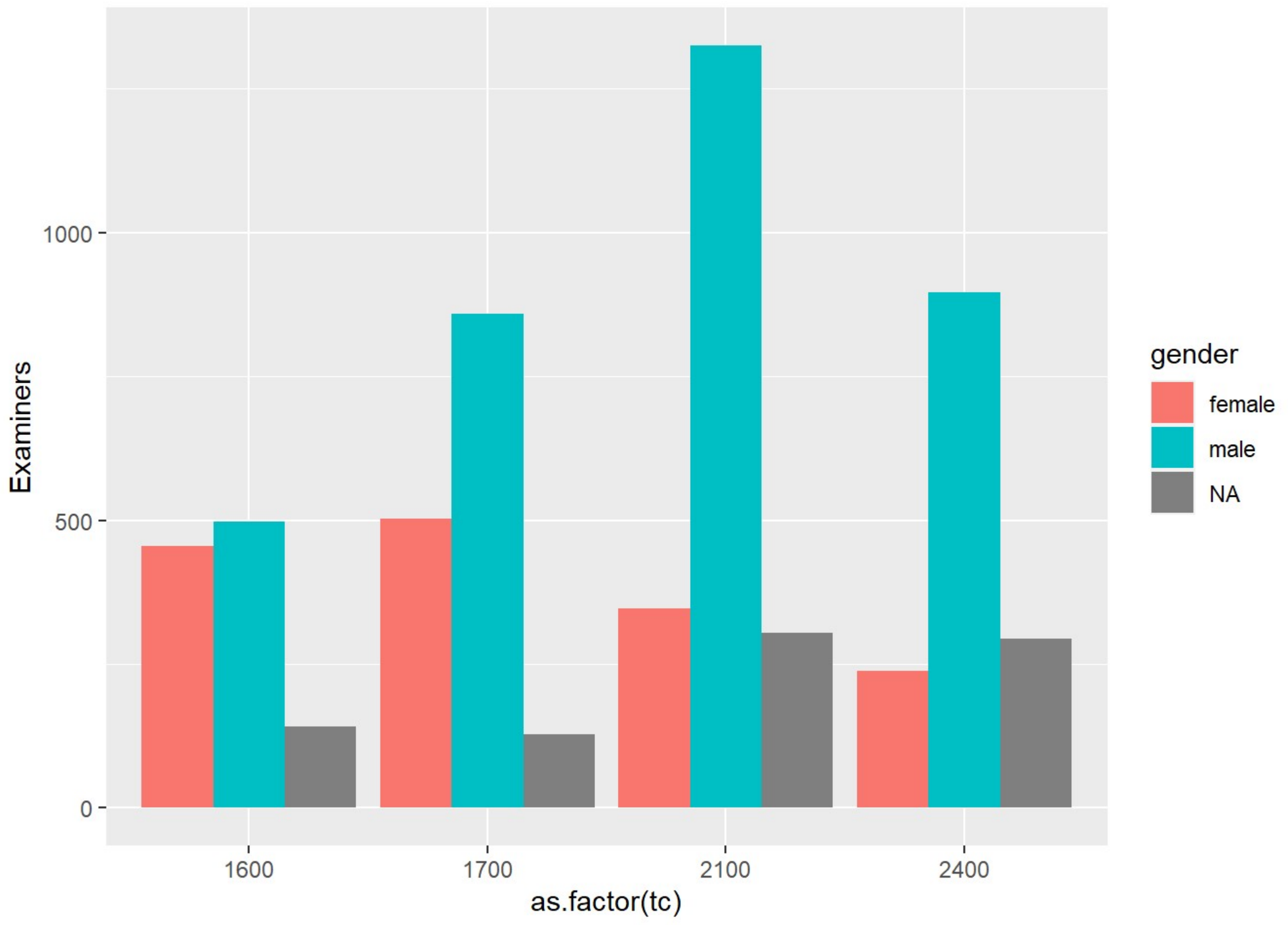
```
## # A tibble: 1 × 1
##       n
##   <int>
## 1  5649
```

Compare TCs by gender graphically

This is what chatGPT gave us:

```
library(dplyr)
library(ggplot2)
applications %>%
  group_by(tc, gender) %>%
  #filter(!is.na(gender)) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  ggplot(aes(x = as.factor(tc), y = n, fill = gender)) +
  geom_col(position = "dodge") +
  ylab("Examiners")
```

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

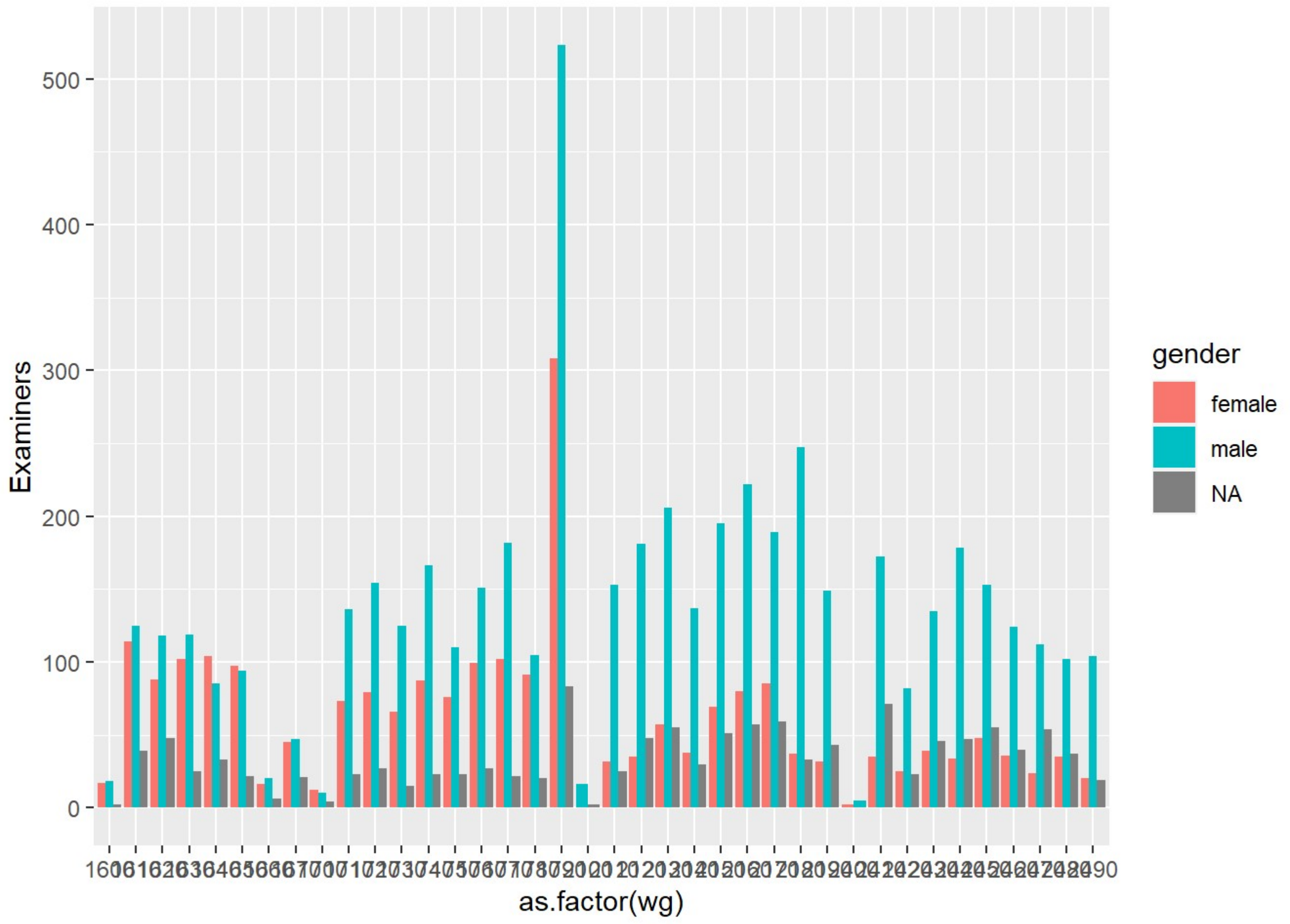


Compare WGs by gender graphically

This is what chatGPT gave us:

```r
library(dplyr)
library(ggplot2)
applications %>%
  mutate(wg = floor(examiner_art_unit/10)*10) %>%
  group_by(wg, gender) %>%
  #filter(!is.na(gender)) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  ggplot(aes(x = as.factor(wg), y = n, fill = gender)) +
  geom_col(position = "dodge") +
  ylab("Examiners")
```

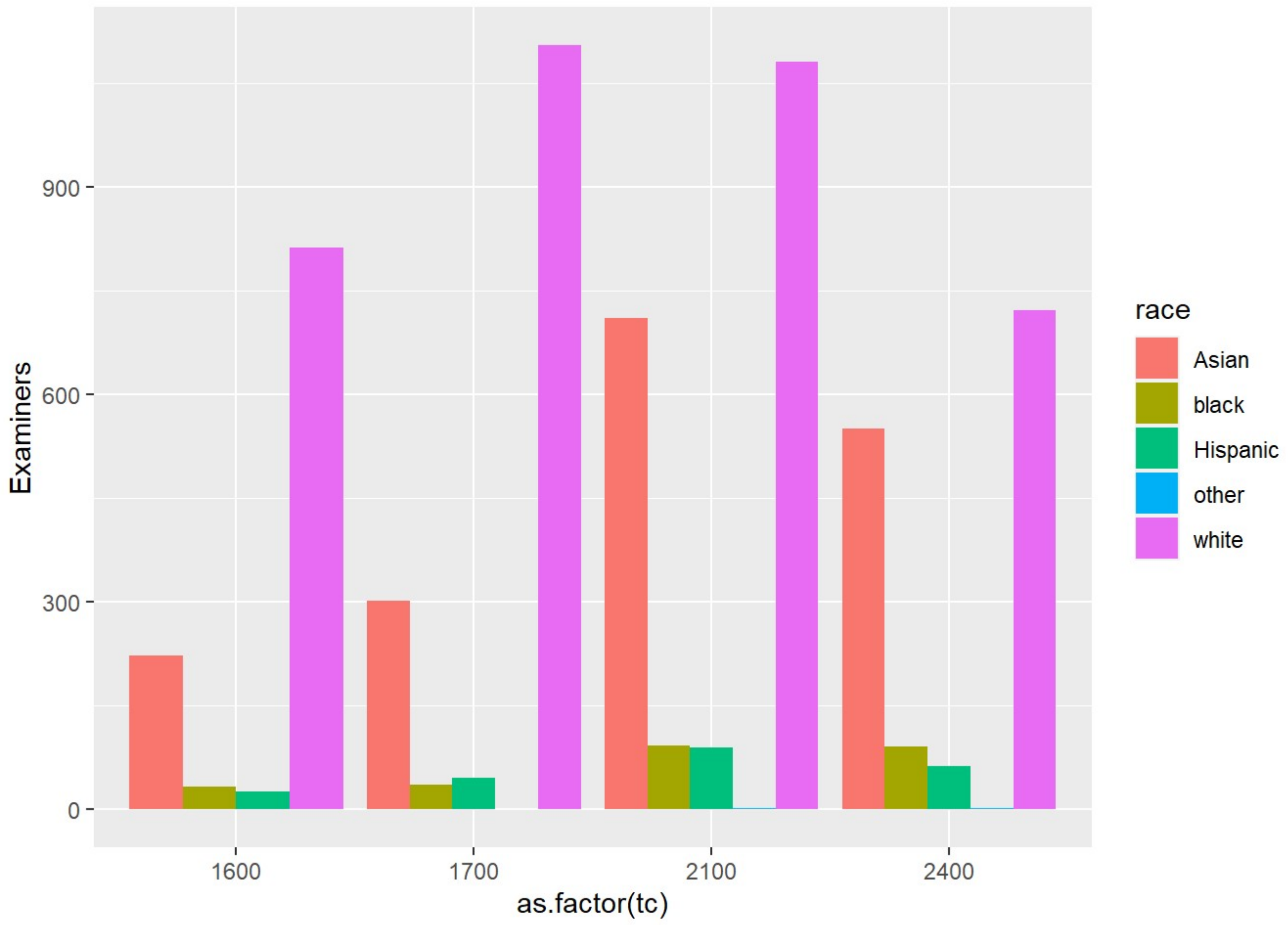
```
## `summarise()` has grouped output by 'wg'. You can override using the `.groups`
## argument.
```



Compare TCs by race graphically

```
library(dplyr)
library(ggplot2)
applications %>%
  group_by(tc, race) %>%
  #filter(!is.na(race)) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  ggplot(aes(x = as.factor(tc), y = n, fill = race)) +
  geom_col(position = "dodge") +
  ylab("Examiners")
```

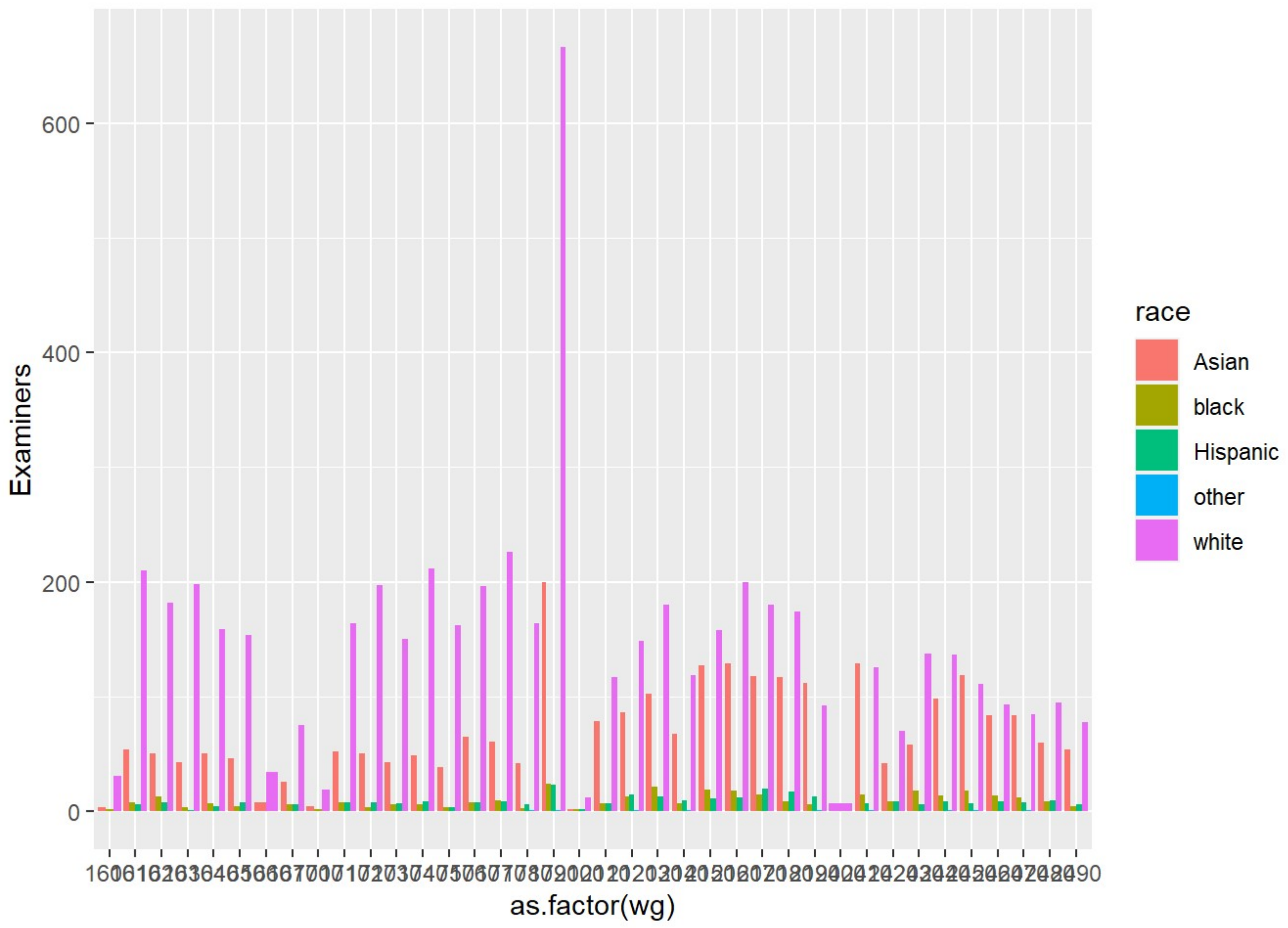
```
## `summarise()` has grouped output by 'tc'. You can override using the `.groups`
## argument.
```



### Compare WGs by race graphically

```
library(dplyr)
library(ggplot2)
applications %>%
  mutate(wg = floor(examiner_art_unit/10)*10) %>%
  group_by(wg, race) %>%
  #filter(!is.na(gender)) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  ggplot(aes(x = as.factor(wg), y = n, fill = race)) +
  geom_col(position = "dodge") +
  ylab("Examiners")
```

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

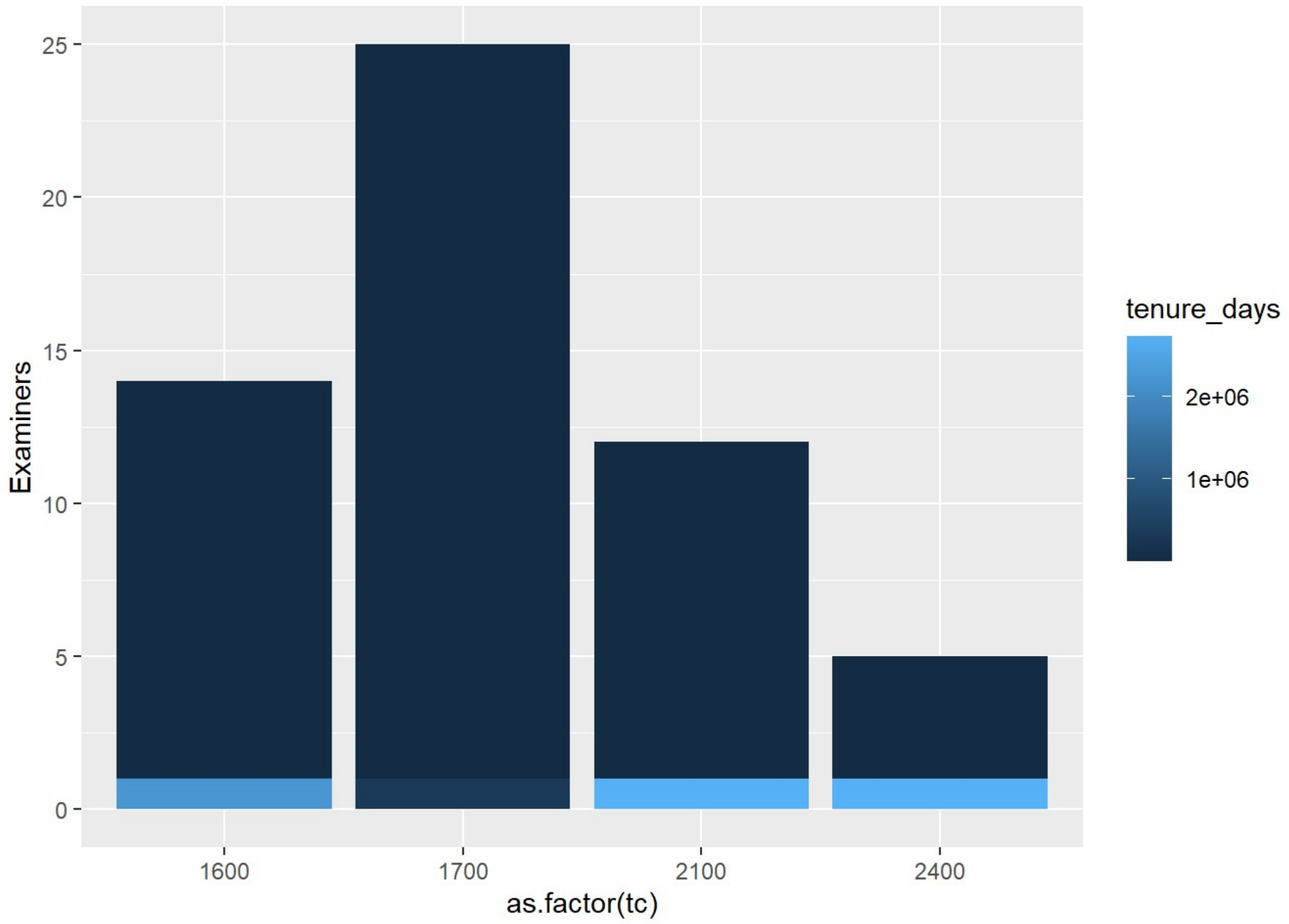


### Compare TCs by tenure graphically

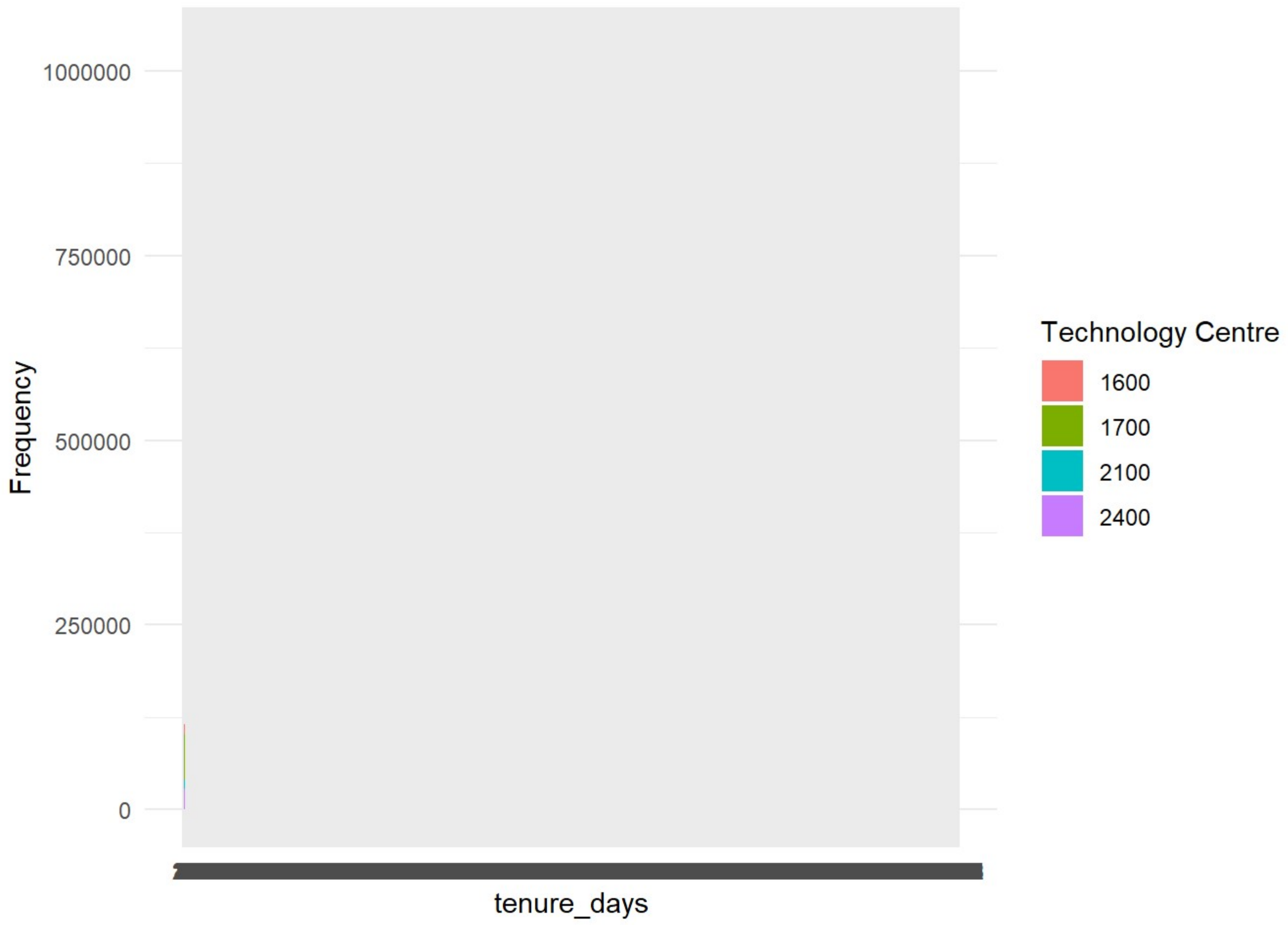
```
library(dplyr)
library(ggplot2)
applications %>%
  group_by(tc, tenure_days) %>%
  #filter(!is.na(tenure_days)) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  ggplot(aes(x = as.factor(tc), y = n, fill = tenure_days)) +

  geom_col(position = "dodge") +
  ylab("Examiners")
```

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



## Tenure distribution across technology centres



Correlations

We have tenure for each person and we want to know whether tenure is predicted by gender.

```
library(dplyr)
examiners <- applications %>%
  group_by(examiner_id) %>%
  summarise(
    tenure = first(tenure_days),
    gender = first(gender),
    race = first(race),
    tc = first(tc)
    )

library(broom)
fit1 <- lm(tenure ~ gender + race, data = examiners)
tidy(fit1)
```

```
## # A tibble: 6 × 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)     3658.     1838.    1.99   0.0466
## 2 gendermale      1016.     1587.    0.640  0.522
## 3 raceblack      13461.     4209.    3.20   0.00139
## 4 raceHispanic    -399.     3873.   -0.103  0.918
## 5 raceother        894.    36030.    0.0248 0.980
## 6 racewhite       1176.     1721.    0.683  0.494
```

```
fit2 <- lm(tenure ~ gender + race + tc, data = examiners)
tidy(fit2)
```

```
## # A tibble: 7 × 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  10649.     5342.     1.99    0.0463
## 2 gendermale    1526.     1629.     0.937   0.349
## 3 raceblack    13508.     4209.     3.21    0.00134
## 4 raceHispanic  -411.     3873.    -0.106   0.916
## 5 raceother      207.    36030.     0.00574 0.995
## 6 racewhite      658.     1760.     0.374   0.708
## 7 tc              -3.59      2.57  -1.39    0.163
```

Explaining turnover by gender

```
applications <- applications %>%
  mutate(appl_status_date = dmy_hms(appl_status_date))

applications <- applications %>%
  mutate(year = year(appl_status_date))
```

```
turnover <- applications %>%
  group_by(examiner_id) %>%
  summarize(min_year = min(year), max_year = max(year), tc = first(tc), gender = first(gende
r), race = first(race)) %>%
  mutate(year_left = if_else(max_year<2017, max_year+1, NA_real_))
```

#picking 2013 for analysis year

```
regression_data <- turnover %>%
  dplyr::filter(min_year <= 2013, year_left >= 2014 | is.na(year_left)) %>%
  mutate(left = if_else(year_left != 2014 | is.na(year_left),0,1)) %>%
   drop_na(gender)
```

```
regression_data %>%
  count(gender, left) %>%
  group_by(gender) %>%
  mutate(pct = n/sum(n))
```

```
## # A tibble: 4 × 4
## # Groups:   gender [2]
##   gender  left     n    pct
##   <chr>  <dbl> <int>  <dbl>
## 1 female     0   732 0.968
## 2 female     1    24 0.0317
## 3 male       0  1656 0.966
## 4 male       1    59 0.0344
```

```
# Assuming your dataset is named "data"
set.seed(123)  # Set seed for reproducibility

# Calculate the number of rows for the training set and holdout set
total_rows <- nrow(regression_data)
train_rows <- round(0.85 * total_rows)

# Split the data into training and holdout sets
training_data <- slice_sample(regression_data, n = train_rows)
# Create a vector of randomly selected row indices for the training set
train_indices <- sample(total_rows, train_rows)
holdout_data <- regression_data[-train_indices, ]
```

```
model_training_data <- lm(data = training_data, left ~ gender + as.factor(tc))
tidy(model_training_data)
```

```
## # A tibble: 5 × 5
##   term               estimate std.error statistic  p.value
##   <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        0.0378     0.0106     3.55   0.000393
## 2 gendermale         0.000529   0.00903    0.0586 0.953
## 3 as.factor(tc)1700 -0.00953    0.0121    -0.789  0.430
## 4 as.factor(tc)2100 -0.000148   0.0118    -0.0126 0.990
## 5 as.factor(tc)2400  0.00181    0.0136     0.133  0.894
```

```
summary(model_training_data)
```

```
##
## Call:
## lm(formula = left ~ gender + as.factor(tc), data = training_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04011 -0.03830 -0.03816 -0.02878  0.97175
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.0377743  0.0106402   3.550 0.000393 ***
## gendermale         0.0005292  0.0090290   0.059 0.953272
## as.factor(tc)1700 -0.0095283  0.0120742  -0.789 0.430116
## as.factor(tc)2100 -0.0001478  0.0117748  -0.013 0.989988
## as.factor(tc)2400  0.0018081  0.0135579   0.133 0.893923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1857 on 2095 degrees of freedom
## Multiple R-squared:  0.0006021,  Adjusted R-squared:  -0.001306
## F-statistic: 0.3155 on 4 and 2095 DF,  p-value: 0.8677
```

The sum of squares of residuals (ss_residual) is calculated as the sum of squared differences between the actual values and the predicted values. The total sum of squares (ss_total) is computed as the sum of squared differences between the actual values and the mean of the actual values.

Finally, the R-squared value is calculated by subtracting the ratio of the sum of squares of residuals to the total sum of squares from 1. The resulting rsquared variable will contain the R-squared value, which indicates the goodness-of-fit of the linear regression model on the holdout data.

```
# Assuming you have the linear regression model stored in 'model'
# and the holdout data stored in 'holdout_data'

# Extract the dependent variable from the holdout data
holdout_actual <- holdout_data$left

# Use the model to predict values for the holdout data
holdout_predicted <- predict(model_training_data, newdata = holdout_data)

# Calculate the R-squared value
ss_residual <- sum((holdout_actual - holdout_predicted)^2)
ss_total <- sum((holdout_actual - mean(holdout_actual))^2)
rsquared <- 1 - (ss_residual / ss_total)
```

An R-squared value of 0.0014967026476731 is a very small value to correctly predict the relation between turnover rate with gender and tc. This implies that the model does not effectively capture the relationship between the independent variables and the dependent variable in the holdout data.

The model may not be able to accurately capture the underlying patterns or relationships in the data, leading to poor performance in predicting or explaining the dependent variable. SO maybe the turnover rate is not linearly dependent on gender and race.