Assignment 3

Emery Dittmer

2023-03-28

#1:Load data

Load the following data: + applications from app\_data\_sample.parquet + edges from edges\_sample.csv

# change to your own path!  
data\_path <- "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 × 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 ¹​application\_number, ²​filing\_date, ³​examiner\_name\_last,  
## # ⁴​examiner\_name\_first, ⁵​examiner\_name\_middle, ⁶​examiner\_id,  
## # ⁷​examiner\_art\_unit, ⁸​uspc\_class, ⁹​uspc\_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)  
#install\_genderdata\_package() # only run this line the first time you use the package, to get data for it  
# 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 4662387 249.0 8222411 439.2 5106678 272.8  
## Vcells 49868824 380.5 93037893 709.9 80184624 611.8

## 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 × 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>.

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

write.csv(examiner\_race, "examiner\_race.csv", row.names=FALSE)

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/>. And this one for case\_when() function: <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 4797340 256.3 8222411 439.2 8222411 439.2  
## Vcells 54202889 413.6 111725471 852.4 92949736 709.2

## 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 4803510 256.6 14934374 797.6 14934374 797.6  
## Vcells 64466725 491.9 134150565 1023.5 134137413 1023.4

#2. Focus on Art Unit:Descriptive Stats ## Work Unit Breakdown of people

We will compare genders and ethnicity across all work units within the US Patent office. First let’s do some descriptive statistics on the overall population.

Lets keep only one observation per person for the data since once person could count twice for a work group

person\_level\_data <- applications %>%   
 group\_by(examiner\_id) %>%   
 summarise(  
 art\_unit = min(examiner\_art\_unit, na.rm = TRUE),  
 gender = min(gender, na.rm = TRUE),  
 race = min(race,na.rm=TRUE),  
 start\_year = min(year(earliest\_date), na.rm = TRUE),  
 latest\_date = max(latest\_date, na.rm = TRUE))%>%   
 mutate(  
 tc = floor(art\_unit/100)\*100,  
 work\_group = as.factor(floor(art\_unit/10)\*10)  
 ) %>%   
 filter(!is.na(gender) & !is.na(race)) # dropping all records where we don't know the gender  
person\_level\_data

## # A tibble: 4,849 × 8  
## examiner\_id art\_unit gender race start\_year latest\_date tc work\_group  
## <dbl> <dbl> <chr> <chr> <dbl> <date> <dbl> <fct>   
## 1 59012 1716 male white 2004 2015-07-24 1700 1710   
## 2 59025 2465 male Asian 2009 2017-05-18 2400 2460   
## 3 59040 1724 female Asian 2007 2017-05-23 1700 1720   
## 4 59052 2138 male Asian 2001 2007-02-28 2100 2130   
## 5 59055 2165 male Asian 2004 2007-12-26 2100 2160   
## 6 59056 2124 male Asian 2000 2017-05-22 2100 2120   
## 7 59081 2489 male Asian 2011 2017-05-19 2400 2480   
## 8 59086 2487 female white 2010 2017-05-18 2400 2480   
## 9 59096 1612 male white 2000 2015-11-20 1600 1610   
## 10 59117 2439 male white 2009 2011-09-02 2400 2430   
## # … with 4,839 more rows

#grouping by work unit  
work\_unit\_level\_data <-person\_level\_data %>%  
 group\_by(work\_group,race,gender) %>%  
 summarize(  
 n=n()  
 )

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

work\_unit\_level\_data

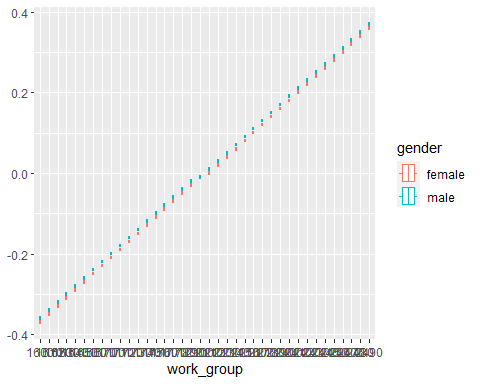
## # A tibble: 263 × 4  
## # Groups: work\_group, race [146]  
## work\_group race gender n  
## <fct> <chr> <chr> <int>  
## 1 1600 Asian female 3  
## 2 1600 black female 1  
## 3 1600 white female 13  
## 4 1600 white male 18  
## 5 1610 Asian female 18  
## 6 1610 Asian male 15  
## 7 1610 black female 4  
## 8 1610 black male 2  
## 9 1610 Hispanic female 2  
## 10 1610 Hispanic male 3  
## # … with 253 more rows

#we will also need to aggregated by total number of people in work\_unit  
work\_unit\_aggregated <- work\_unit\_level\_data %>%  
 group\_by(work\_group) %>%  
 summarize(  
 n=sum(n)  
 ) %>%  
 arrange (desc(n))  
work\_unit\_aggregated

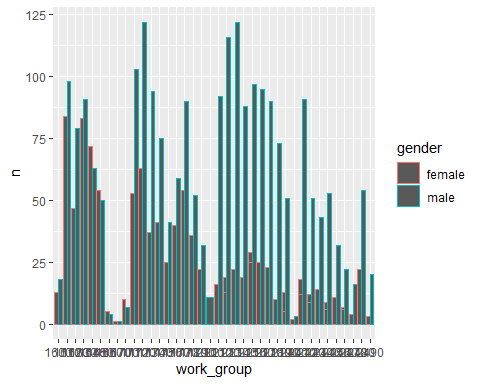
## # A tibble: 38 × 2  
## work\_group n  
## <fct> <int>  
## 1 2130 237  
## 2 1610 226  
## 3 2150 226  
## 4 1720 225  
## 5 2120 210  
## 6 1710 208  
## 7 1630 207  
## 8 2410 203  
## 9 2160 197  
## 10 1770 189  
## # … with 28 more rows

Let’s plot the race, and gender as a function of workgroup. First looking at counts then distributions

library(ggplot2)  
ggplot(work\_unit\_level\_data) +  
 geom\_boxplot(aes(x = work\_group, color = gender))

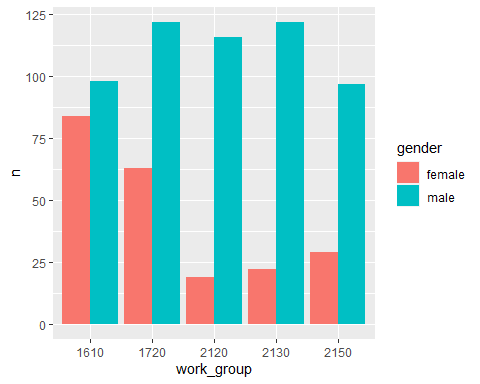


ggplot(work\_unit\_level\_data,aes(x = work\_group, color = gender, y=n)) +  
 geom\_bar(stat="identity", position=position\_dodge())

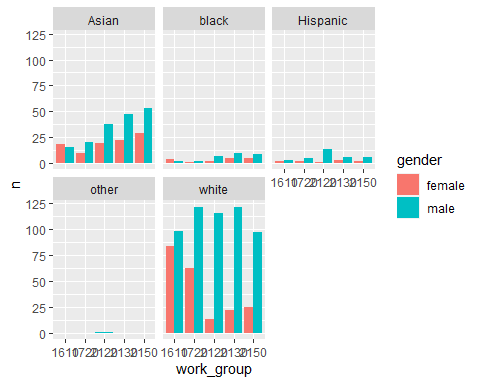


Let’s plot for the top 5 work groups to make it easier to read. First we will look at the number (counts) then we will look at the distributions using box plots.

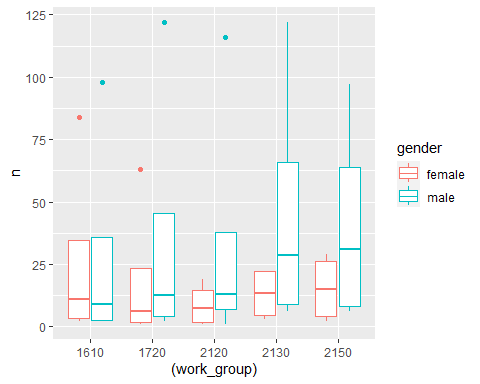
work\_unit\_level\_data\_top5 <- work\_unit\_level\_data %>%  
 filter(work\_group %in% head(work\_unit\_aggregated$work\_group,5))  
  
ggplot(work\_unit\_level\_data\_top5,aes(x = work\_group, fill = gender, y=n)) +  
 geom\_bar(stat="identity", position=position\_dodge())



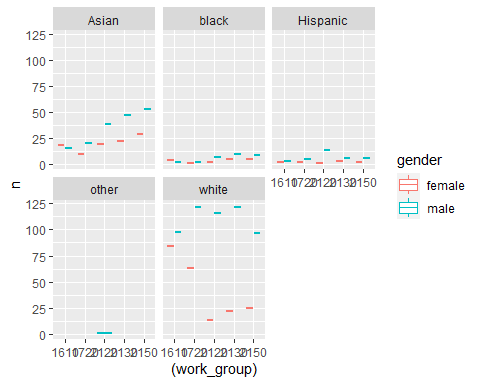
ggplot(work\_unit\_level\_data\_top5,aes(x = work\_group, fill = gender, y=n)) +  
 geom\_bar(stat="identity", position=position\_dodge())+  
 facet\_wrap(~race)



ggplot(work\_unit\_level\_data\_top5) +  
 geom\_boxplot(aes(x = (work\_group),y=n, color = gender))



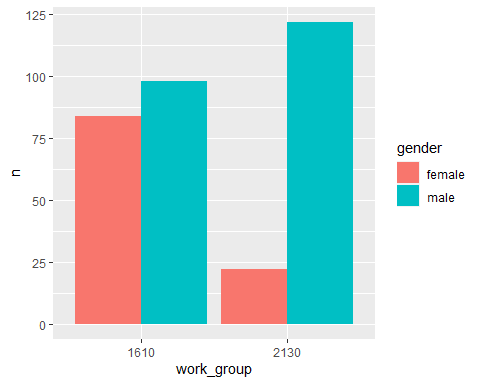
ggplot(work\_unit\_level\_data\_top5) +  
 geom\_boxplot(aes(x = (work\_group),y=n, color = gender))+  
 facet\_wrap(~race)



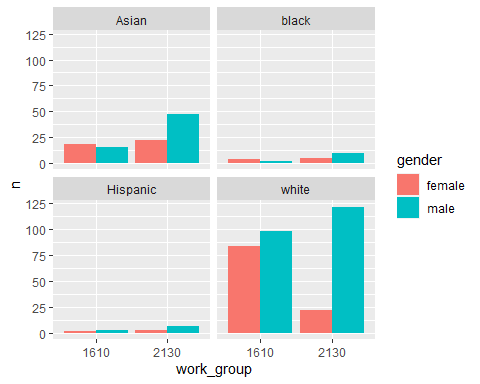
remove(work\_unit\_level\_data\_top5)

Even the top 5 is alot of data. For the remaining analysis we will focus on the top 2 work\_units: 2130 and 1610. Since we are only using 2 art units the ditribution is not as relenvant to plot at the moment.

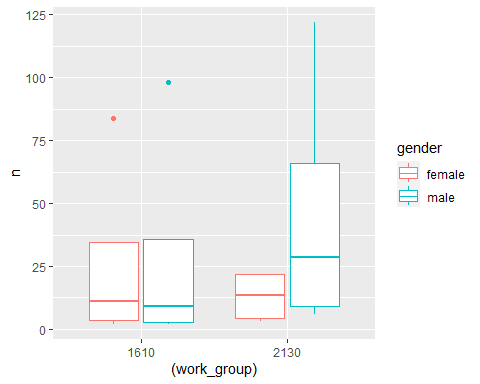
work\_unit\_level\_data\_top2 <- work\_unit\_level\_data %>%  
 filter(work\_group %in% head(work\_unit\_aggregated$work\_group,2))  
  
ggplot(work\_unit\_level\_data\_top2,aes(x = work\_group, fill = gender, y=n)) +  
 geom\_bar(stat="identity", position=position\_dodge())



ggplot(work\_unit\_level\_data\_top2,aes(x = work\_group, fill = gender, y=n)) +  
 geom\_bar(stat="identity", position=position\_dodge())+  
 facet\_wrap(~race)



ggplot(work\_unit\_level\_data\_top2) +  
 geom\_boxplot(aes(x = (work\_group),y=n, color = gender))



subset\_app\_data <- person\_level\_data %>%   
 #here we make sure on ly the top 2 work groups are picked  
 filter(work\_group %in% head(work\_unit\_aggregated$work\_group,2)) %>%   
 mutate(race = race, gender =gender) %>%   
 select(gender, race, work\_group)

## Gender

let’s investigate gender, first accros borht work groups then within the workgroup

subset\_app\_data %>%   
 count(gender) %>%   
 mutate(pct = n/sum(n))

## # A tibble: 2 × 3  
## gender n pct  
## <chr> <int> <dbl>  
## 1 female 160 0.346  
## 2 male 303 0.654

subset\_app\_data %>%   
 group\_by(work\_group) %>%  
 count(gender) %>%   
 mutate(pct = n/sum(n))

## # A tibble: 4 × 4  
## # Groups: work\_group [2]  
## work\_group gender n pct  
## <fct> <chr> <int> <dbl>  
## 1 1610 female 108 0.478  
## 2 1610 male 118 0.522  
## 3 2130 female 52 0.219  
## 4 2130 male 185 0.781

## Race

let’s investigate race with the same process as above, first accros borht work groups then within the workgroup

subset\_app\_data %>%  
 group\_by(work\_group) %>%  
 count(race) %>%   
 mutate(pct = n/sum(n))

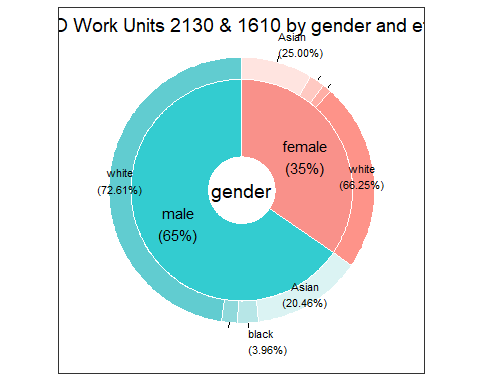
## # A tibble: 8 × 4  
## # Groups: work\_group [2]  
## work\_group race n pct  
## <fct> <chr> <int> <dbl>  
## 1 1610 Asian 33 0.146   
## 2 1610 black 6 0.0265  
## 3 1610 Hispanic 5 0.0221  
## 4 1610 white 182 0.805   
## 5 2130 Asian 69 0.291   
## 6 2130 black 15 0.0633  
## 7 2130 Hispanic 9 0.0380  
## 8 2130 white 144 0.608

## Puttin it together

Let’s investgate both at the same time

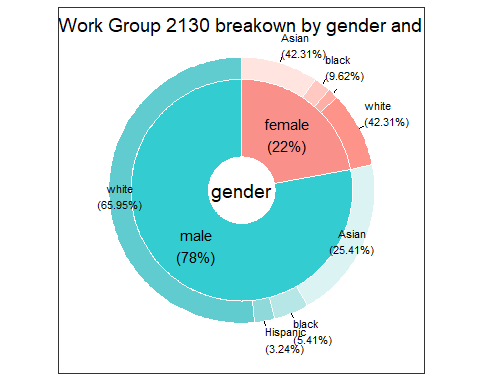
library(webr)  
PieDonut(subset\_app\_data, aes(gender,race), title = "USPTO Work Units 2130 & 1610 by gender and ethnicity")

## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as  
## of ggplot2 3.3.4.  
## ℹ The deprecated feature was likely used in the webr package.  
## Please report the issue at <]8;;https://github.com/cardiomoon/webr/issueshttps://github.com/cardiomoon/webr/issues]8;;>.



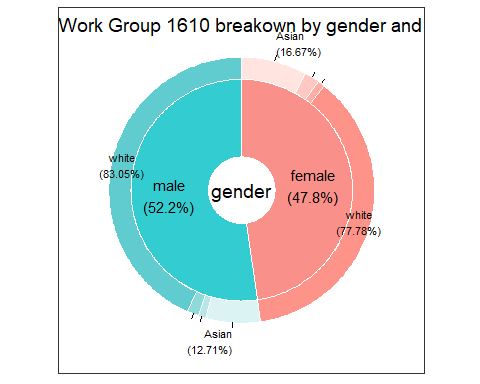
subset\_app\_data1 <- subset\_app\_data %>% filter(work\_group==2130)  
subset\_app\_data2 <- subset\_app\_data %>% filter(work\_group==1610)  
  
PieDonut(subset\_app\_data1, aes(gender,race), title = "USPTO Work Group 2130 breakown by gender and ethnicity", explodeDonut=TRUE)

## Warning in geom\_arc\_bar(aes\_string(x0 = "x", y0 = "y", r0 = as.character(r1), :  
## Ignoring unknown aesthetics: explode



PieDonut(subset\_app\_data2, aes(gender,race), title = "USPTO Work Group 1610 breakown by gender and ethnicity", explodeDonut=TRUE)

## Warning in geom\_arc\_bar(aes\_string(x0 = "x", y0 = "y", r0 = as.character(r1), :  
## Ignoring unknown aesthetics: explode



remove(subset\_app\_data1, subset\_app\_data2)

#3: Advice Network ##Nodes & Edges First we need to subset the data and remove the examiners who are not in the work groups we are looking at

#copy data in case  
edges\_full <- edges  
edges <- edges\_full  
  
subset\_exam\_id <- person\_level\_data %>%  
 filter(work\_group %in% head(work\_unit\_aggregated$work\_group,2)) %>%  
 select(examiner\_id,work\_group) %>%  
 drop\_na()  
  
#crete the edges  
edges <- edges %>%  
 filter(ego\_examiner\_id %in% subset\_exam\_id$examiner\_id)%>%  
 drop\_na() %>%  
 mutate(from=ego\_examiner\_id,to=alter\_examiner\_id) %>%  
 select(from, to)  
  
#create the nodes  
#many issues with nodes will try pulling from edges list  
# nodes\_all <- unique(select(edges\_full, ego\_examiner\_id)) %>%  
# mutate(id=ego\_examiner\_id, verticies =ego\_examiner\_id) %>%  
# select(id,verticies) %>%  
# drop\_na  
  
nodes\_all <-as.data.frame(do.call(rbind,append(as.list(edges$from),as.list(edges$to))))  
  
nodes\_all <- nodes\_all %>%  
 mutate(id=V1) %>%  
 select(id) %>%  
 distinct(id) %>%  
 drop\_na()  
nodes <- nodes\_all  
# nodes <- nodes\_all %>%  
# mutate(label=as.character(ego\_examiner\_id)) %>%  
# filter(id %in% edges$from | id %in% edges$to ) %>%  
# drop\_na() %>%  
# select(id,label)

library(visNetwork)  
visNetwork(nodes, edges)%>%  
 visLegend() %>%  
 visEdges(arrows ="to")%>%  
 visEdges(arrows ="from")



### 3.1 Degree Centrality

The count of the number of links each node has to other nodes. For instance, seat A(labelled as 3 above) has a degree centrality of 3 since it is connected to 3 other nodes: 2, B & C (B labelled as 4 and C labelled as 5 above)

We can validate this with the igraph package wich has a built in functionality for centrality degree

library(igraph)

##   
## 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(tidygraph)

##   
## Attaching package: 'tidygraph'

## The following object is masked from 'package:igraph':  
##   
## groups

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

library(tidyverse)  
  
g <- igraph::graph\_from\_data\_frame(edges, vertices = nodes) %>% as\_tbl\_graph(directed=TRUE)  
#not sure why this isnt working  
#g = tbl\_graph(nodes = nodes, edges = edges, directed = FALSE)  
g <- g %>%   
 activate(nodes) %>%   
 mutate(degree = centrality\_degree()) %>%   
 activate(edges)  
  
tg\_nodes <-  
 g %>%  
 activate(nodes) %>%  
 data.frame() %>%  
 arrange(desc(degree)) %>%  
 rename(Centrality\_Degree=degree) %>%  
 mutate(name=as.integer(name))  
  
nodes\_all <- nodes\_all %>%  
 left\_join(tg\_nodes,by=c("id"="name"))   
  
remove(g,tg\_nodes)

There is agreement between our calculations and the calculations for the package therefore we can use them!

### 3.2 Closeness centrality

A measure that calculates the ability to spread information efficiently via the edges the node is connected to. It is calculated as the inverse of the average shortest path between nodes.

For instance, for node A (labelled 3), the closeness is 1/((1+2+1+1+2+2+2+2+3))=0.0625. The higher the number, the closer the node is to the center based on distance. See appendix For details

g <- igraph::graph\_from\_data\_frame(edges, vertices = nodes) %>% as\_tbl\_graph(directed=TRUE)  
  
g <- g %>%   
 activate(nodes) %>%   
 mutate(degree = centrality\_closeness()) %>%   
 activate(edges)  
  
tg\_nodes <-  
 g %>%  
 activate(nodes) %>%  
 data.frame() %>%  
 arrange(desc(degree)) %>%  
 rename(Centrality\_Closeness=degree) %>%  
 mutate(name=as.integer(name))  
  
nodes\_all <- nodes\_all %>%  
 left\_join(tg\_nodes,by=c("id"="name"))   
remove(g,tg\_nodes)

### 3.3 Betweenness centrality

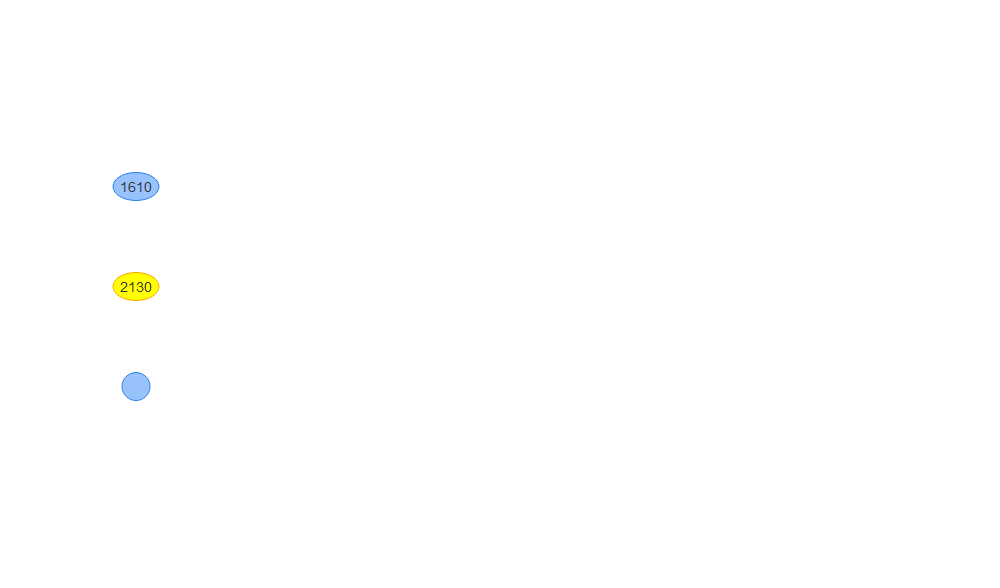
A measure that detects a node’s influence over the flow of information within a graph. This is the sum of the shortest paths between two points i and j divided by the number of shortest paths that pass-through node v.

g <- igraph::graph\_from\_data\_frame(edges, vertices = nodes) %>% as\_tbl\_graph(directed=TRUE)  
  
g <- g %>%   
 activate(nodes) %>%   
 mutate(degree = centrality\_betweenness()) %>%   
 activate(edges)  
  
tg\_nodes <-  
 g %>%  
 activate(nodes) %>%  
 data.frame() %>%  
 arrange(desc(degree)) %>%  
 rename(Centrality\_Betweenness=degree) %>%  
 mutate(name=as.integer(name))  
  
nodes\_all <- nodes\_all %>%  
 left\_join(tg\_nodes,by=c("id"="name"))   
remove(g,tg\_nodes)

## Visualize all together

LEt’s put all the data together now!

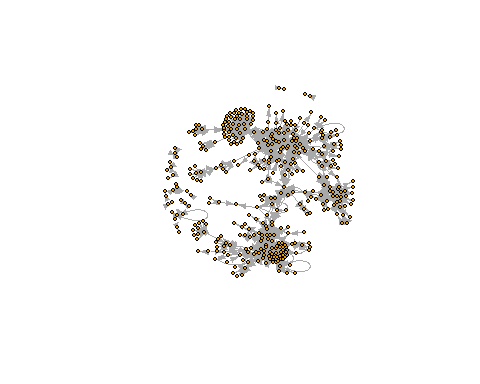
nodes <- nodes\_all %>%   
 left\_join(subset\_exam\_id,by=c("id"="examiner\_id")) %>%  
 mutate(label = paste("Examiner:",id,"\n",  
 "Centrality Degre:",format(Centrality\_Degree, digits = 2),"\n",  
 "Closenness:",format(Centrality\_Closeness, digits = 2),"\n",  
 "Betweenness:",format(Centrality\_Betweenness, digits = 2),"\n",  
 sep = " "),  
 group=work\_group) %>%  
 mutate(font.size = 12)   
  
visNetwork(nodes, edges)%>%  
 visLegend() %>%  
 visEdges(arrows ="to")%>%  
 visEdges(arrows ="from")



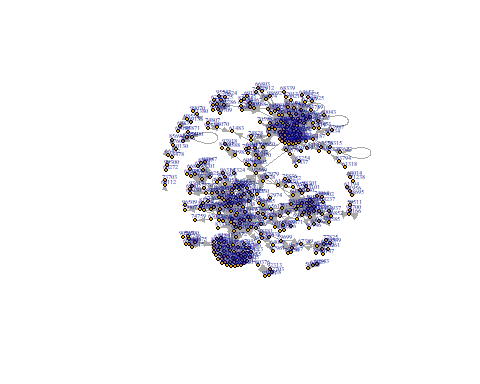
## Igraph version

labels must be removed for igraph or else it does not work well.

net <- igraph::graph\_from\_data\_frame(edges, vertices = nodes\_all) %>% as\_tbl\_graph(directed=TRUE)  
plot(net, edge.arrow.size=.4,vertex.label=NA,vertex.size=4)



plot(net, edge.arrow.size=.4,vertex.label.cex=.4,vertex.label.dist=1,vertex.size=4)



## appendix

testing to make sure examiners in edges data

test<-merge(edges,person\_level\_data,by.x="to",by.y="examiner\_id")  
test %>%  
 group\_by(work\_group) %>%  
 count(work\_group) %>%  
 arrange(desc(n))

## # A tibble: 27 × 2  
## # Groups: work\_group [27]  
## work\_group n  
## <fct> <int>  
## 1 2130 223  
## 2 1610 214  
## 3 2110 66  
## 4 2180 27  
## 5 2400 26  
## 6 1710 16  
## 7 2120 14  
## 8 1630 13  
## 9 2150 12  
## 10 1600 11  
## # … with 17 more rows

Nodes and edges mismatch solving

test <- edges %>%  
 filter(from %in% nodes$id)  
  
test <- edges %>%  
 filter(from %in% nodes$id | to %in% nodes$id)  
  
test <- nodes %>%  
 filter(id %in% edges$to)  
  
  
edges[(!edges$from %in% nodes\_all$id) ,]

## # A tibble: 0 × 2  
## # … with 2 variables: from <dbl>, to <dbl>

edges[(!edges$to %in% nodes\_all$id) ,]

## # A tibble: 0 × 2  
## # … with 2 variables: from <dbl>, to <dbl>