Gender Pay Gap Graph

Olivia Beck

2022-08-12

This vignette is to describe the workflow for creating the gender pay gap graph and to discuss the issues that currently exist with it.

Getting the inputs

Load in the global.R file and get a list of inputs for graphing. This possible list of inputs are:

- form.year : a list of IRS filing years to include in the graph
- state : a list of states to include
- major.group : a list of broad categories to include
- ntee : a list of major groups to include
- ntee.cc : a list of common codes to include (this is not currently operational)
- hosp : to include hospitals, no hospitals, or only hospitals
- univ : to include universities, no universities, or only universities
- tot.expenses : range of total expenses to include
- tot.employee : range of total employees to include

Here is an example:

```
source("global.R")

filter.gender.1 <- list(form.year = 2019, #input$filter.year,
                     state = c(state.abb, "DC", "PR"),  #  input$filter.gender.State,
                     major.group = c(1,2,3,4,5,6,7),#input$filter.gender.MajorGroup,
                     ntee = NA, #need to add input for this
                     ntee.cc = NA, #need to add input for this
                     hosp = NA, #input$filter.gender.HOSP,
                     univ = NA, #input$filter.gender.UNIV,
                     tot.expense = c(0, Inf), #input$filter.gender.TotalExpenses,
                     tot.employee = c(0, Inf) #input$filter.gender.TotalExpenses)
)
    # Change null to na for fail safe purposes
    if(is.null(filter.gender.1$form.year)){filter.gender.1$form.year <- NA}
    if(is.null(filter.gender.1$state)){filter.gender.1$state <- NA}
    if(is.null(filter.gender.1$major.group)){filter.gender.1$major.group <- NA}
```

```r
    if(is.null(filter.gender.1$ntee)){filter.gender.1$ntee <- NA}
    if(is.null(filter.gender.1$ntee.cc)){filter.gender.1$ntee.cc <- NA}
    if(is.null(filter.gender.1$hosp)){filter.gender.1$hosp <- NA}
    if(is.null(filter.gender.1$univ)){filter.gender.1$univ <- NA}
    if(is.null(filter.gender.1$tot.expense)){filter.gender.1$tot.expense <- c(-Inf, Inf)}
    if(is.null(filter.gender.1$tot.employee)){filter.gender.1$tot.employee <- c(0, Inf)}

gender.graph.filters <- filter.gender.1
```

The user has 2 plotting options.

- y-axis : options are MajorGroup, NTEE, or NTEE.CC (this will need to be changed to Braod Category, Major Group, Common Code language in the final product)
- x-axis : mean or median

Here are example plotting inputs:

```r
#yaxis options: "MajorGroup", "NTEE", "NTEE.CC"
y.axis <- "MajorGroup" # input$filter.gender.graph.yaxis
#stat options: Median, Mean
s <- "Median" #input$filter.gender.graphs.method
```

Get the filtered data

Use the dat_filtering function from funcs/applying-filters-funcs.R file.

```r
#get filtered data
    dat.filterd <- dat_filtering(form.year = gender.graph.filters$form.year,
                                 state = gender.graph.filters$state,
                                 major.group =  gender.graph.filters$major.group,
                                 ntee = gender.graph.filters$ntee,
                                 ntee.cc = gender.graph.filters$ntee.cc,
                                 hosp = gender.graph.filters$hosp,
                                 univ = gender.graph.filters$univ,
                                 tot.expense =  gender.graph.filters$tot.expense,
                                 tot.employee =  gender.graph.filters$tot.employee
    )
```

```
## Rows: 12847 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): FormType, Name, State, NTEE, NTEE.CC, Gender, ZIP5, LocationType, FIPS
## dbl (8): EIN, FormYr, MajorGroup, TotalExpense, TotalEmployee, GrossReceipts...
## lgl (4): UNIV, HOSP, TransitionYr, TRANS.D
##
## 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.
```

This can later be updated to use the funcs/dat-filtering-hard.R file by putting the inputs in the search and hard format in the pretend-shiny vignette by setting the users input to be in the search list and setting every option in the hard list to TRUE. This way it is more efficent as we do not need more than one function for the same purpose. I wrote this graph in the shiny app before I wrote the compensation appraisal tool and I did not have enough time to change the function structures, but it definitely can be done.

Formatting the graph

If we are including hospitals and/or universities, seperate them out at the MajorGroup stage. This effectively allows the user to go between the 10 and 12 Broad Categories groupings without explicitly telling them so.

```
    #if we want to include hospitals, add a major group for hospitals
    dat.filterd$MajorGroup[which(dat.filterd$HOSP == T)] <- 11

    #if we want to include universities, add a major group for hospitals
    dat.filterd$MajorGroup[which(dat.filterd$UNIV == T)] <- 12
```

Format the data to keep only the information we need in the plot.

```
#format data for plotting
dat.plot <- dat.filterd %>%
  #remove any data with unknown gender
  dplyr::filter(Gender != "U") %>%
  #keep only the data we need to plot
  dplyr::select(CEOCompensation, Gender, paste(y.axis)) %>%
  #assign the Yaxis to be the user's plot input
  dplyr::rename(Yaxis = paste(y.axis))
```

Next we aggregate by gender and the select y-axis criteria.

```
dat.plot <- dat.plot %>%
  #group by everything but CEOCompensation
  dplyr::group_by_at(vars(-CEOCompensation)) %>%
  #summarize in each group
  dplyr::summarise(Value =  ifelse(s == "Median",
                                   median(CEOCompensation),
                                   mean(CEOCompensation)),
                   n = n(),
                   .groups = "keep") %>%
  dplyr::ungroup()
```

More formatting…

```
    #rename major groups to something readable
    if(y.axis == "MajorGroup"){
      dat.plot <- dat.plot %>%
        dplyr::mutate(Yaxis = case_when(Yaxis == 1 ~ "Arts, Culture, and Humanities",
                                        Yaxis == 2 ~ "Education",
                                        Yaxis == 3 ~ "Environment and Animals",
                                        Yaxis == 4 ~ "Health",
                                        Yaxis == 5 ~ "Human Services",
                                        Yaxis == 6 ~ "International, Foreign Affairs",
                                        Yaxis == 7 ~ "Public, Societal Benefit",
                                        Yaxis == 8 ~ "Religion Related",
                                        Yaxis == 9 ~ "Mutual/Membership Benefit",
                                        Yaxis == 10 ~ "Unknown/Unclassified",
                                        Yaxis == 11 ~ "Hosptial",
                                        Yaxis == 12 ~ "University"))
    }
    #recode as factors
    dat.plot$Gender <- as.factor(dat.plot$Gender)
    dat.plot$Yaxis <- as.factor(dat.plot$Yaxis)

    #format
    x.label <- paste0(toupper(strsplit(s, "")[[1]][1]), paste0(strsplit(s, "")[[1]][-1], collapse = "")

    #a little more formatting
    dat.diff <- dat.plot %>%
      tidyr::pivot_wider(names_from = Gender, values_from = c(Value, n)) %>%
      dplyr::mutate(diff = abs(Value_F - Value_M)) %>%
      dplyr::mutate(pois = (Value_F + Value_M) / 2) %>%
      dplyr::select(c(Yaxis, pois, diff)) %>%
      base::merge(dat.plot)
```

In the final product, there will need to be a similar thing for the NTEE variable. I think there is a more efficient way of reassigning these axis labels using the data set in data-raw/ntee.csv, but again, I did not have time to implement in.

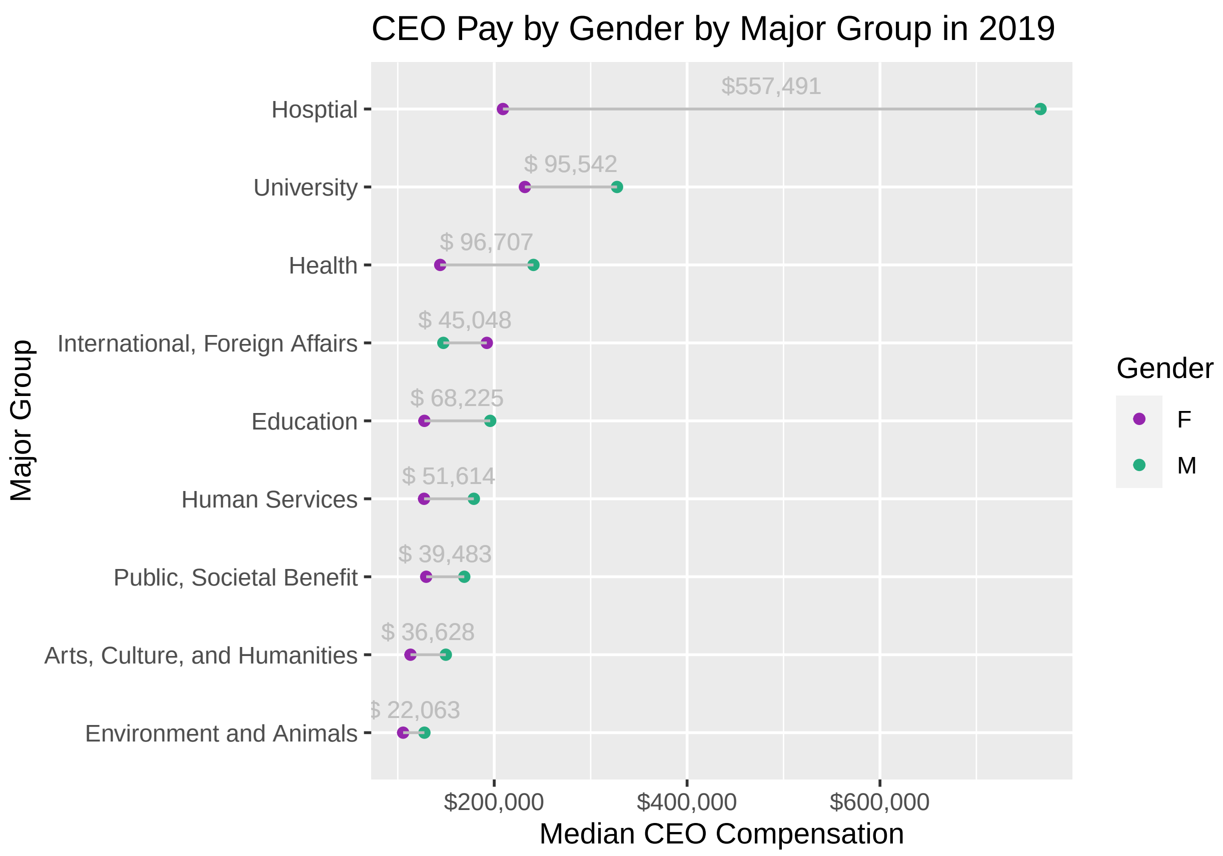
Making the ggplot

```
#make the ggplot
p <- dat.diff %>%
  #ggplot object
  ggplot(aes(x = Value,
             y = reorder(Yaxis,Value),
             color = Gender,
             text = paste("Classification:", Yaxis),
             n = n)) +
  # plot points for each gender within each y-axis category
  geom_point() +
  # plot line between two points within each y-axis category
  geom_line(aes(group = Yaxis), col = "gray" ) +
  #add text to hover boxes - to be used in plotly object
  geom_text(aes(x = pois, y = reorder(Yaxis,Value) ,
                label = dollarize (round(diff))),
            nudge_y = 0.3,
            color = "grey",
            size = 3)+
  #gender colors: green for male and purple for female
  scale_color_manual(values=c("#9525AD", "#25AD80")) +
  #make x-axis in dollar format
  scale_x_continuous(labels=scales::dollar_format())+
  # make the title and labels
  ggtitle(paste("CEO Pay by Gender by", case_when(y.axis == "MajorGroup" ~ "Major Group",
                                          y.axis == "NTEE" ~ "NTEE Code",
                                            y.axis == "NTEE.CC" ~ "NTEE-CC Code"),
                "in 2019")) +
  xlab(paste(x.label, "CEO Compensation")) +
  ylab(paste(case_when(y.axis == "MajorGroup" ~ "Major Group",
                       y.axis == "NTEE" ~ "NTEE Code",
                       y.axis == "NTEE.CC" ~ "NTEE-CC Code")))
```

Output a plotly object. This can't be outputed in a pdf document so I'm just outputting the regular ggplot here.

```
# ggplotly(p,
#          #hard code in what is in the hover box
#          tooltip = c( "text", "n", "Gender", "Value"))  %>%
#          #suppress the hover over the difference line
#          style(p, hoverinfo = "none", traces = c(4))

p
```


CEO Pay by Gender by Major Group in 2019
Hosptial
University
Health
International, Foreign Affairs
Education
Human Services
Public, Societal Benefit
Arts, Culture, and Humanities
Environment and Animals
$557,491
$ 95,542
$ 96,707
$ 45,048
$ 68,225
$ 51,614
$ 39,483
$ 36,628
$ 22,063
Major Group
$200,000
$400,000
$600,000
Median CEO Compensation
Gender
F
M