EDUC 423B/SOC 302B: Assignment 2

Caity McGinley

5-17-21 – OAE Extension

Hi Caity,

Nice work! At first, I was missing a lot of interpretation, but I was happy to find a lot of it in the discussion section, which I enjoyed reading! The PC part missed some crucial visualization, but you interpreted the PCs nicely. The cluster part was very well done! Please find more detailed comments below! Keep up the good work. The finish line is in sight and you are doing great!

Grade: A-

# Honor Code Statement

I strongly encourage students to form study groups and students may discuss and work on assignments in groups. I expect that each student understands their own submission. As such, students must write their submissions independently and clearly disclose the names of all other students who were part of their study group. Additionally, lifting code or solutions directly from the internet (e.g., Google, GitHub, Stack Overflow) is a violation of the [Stanford Honor Code](https://communitystandards.stanford.edu/policies-and-guidance/honor-code). I take academic honesty and Honor Code violations extremely seriously and expect the same of students. If you have questions about what may or may not constitute an Honor Code violation, please reach out to me and we will figure it out together.

**Email:** [caitym26@stanford.edu](mailto:caitym26@stanford.edu)

**SUID:** 06211809

**Study Group:** Victoria

I acknowledge and agree to abide by the Honor Code.

**Signed:** Caity McGinley

# Setup and data manipulation

library(tidyverse) library(tidymodels) library(corrplot) library(ggrepel) library(tidytext) library(dotwhisker) library(visdat) library(ggplot2) library(stringr) setwd(“C:/Users/cmcgi/Downloads/Soc 302A\_Lab1”)

characters <- read\_csv(“characters.csv”)

# Include all code required to load packages, import data, and manipulate data here.  
  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.0

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

## Warning: package 'stringr' was built under R version 4.0.3

## Warning: package 'forcats' was built under R version 4.0.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidymodels)

## Warning: package 'tidymodels' was built under R version 4.0.5

## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v broom 0.7.6 v recipes 0.1.15  
## v dials 0.0.9 v rsample 0.0.9   
## v infer 0.5.4 v tune 0.1.3   
## v modeldata 0.1.0 v workflows 0.2.2   
## v parsnip 0.1.5 v yardstick 0.0.8

## Warning: package 'dials' was built under R version 4.0.5

## Warning: package 'infer' was built under R version 4.0.5

## Warning: package 'modeldata' was built under R version 4.0.5

## Warning: package 'parsnip' was built under R version 4.0.5

## Warning: package 'recipes' was built under R version 4.0.5

## Warning: package 'rsample' was built under R version 4.0.5

## Warning: package 'tune' was built under R version 4.0.5

## Warning: package 'workflows' was built under R version 4.0.5

## Warning: package 'yardstick' was built under R version 4.0.5

## -- Conflicts ----------------------------------------- tidymodels\_conflicts() --  
## x scales::discard() masks purrr::discard()  
## x dplyr::filter() masks stats::filter()  
## x recipes::fixed() masks stringr::fixed()  
## x dplyr::lag() masks stats::lag()  
## x yardstick::spec() masks readr::spec()  
## x recipes::step() masks stats::step()

library(corrplot)

## Warning: package 'corrplot' was built under R version 4.0.5

## corrplot 0.88 loaded

library(ggrepel)

## Warning: package 'ggrepel' was built under R version 4.0.5

library(tidytext)

## Warning: package 'tidytext' was built under R version 4.0.3

library(dotwhisker)

## Warning: package 'dotwhisker' was built under R version 4.0.5

## Warning in checkMatrixPackageVersion(): Package version inconsistency detected.  
## TMB was built with Matrix version 1.3.2  
## Current Matrix version is 1.2.18  
## Please re-install 'TMB' from source using install.packages('TMB', type = 'source') or ask CRAN for a binary version of 'TMB' matching CRAN's 'Matrix' package

library(visdat)

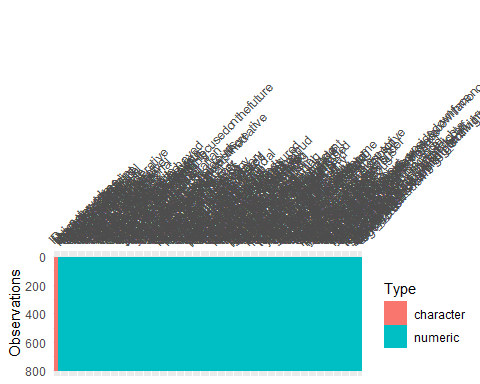
## Warning: package 'visdat' was built under R version 4.0.5

library(ggplot2)  
library(stringr)  
  
setwd("C:/Users/cmcgi/Downloads/Soc 302A\_Lab1")  
  
characters <- read\_csv("characters.csv")

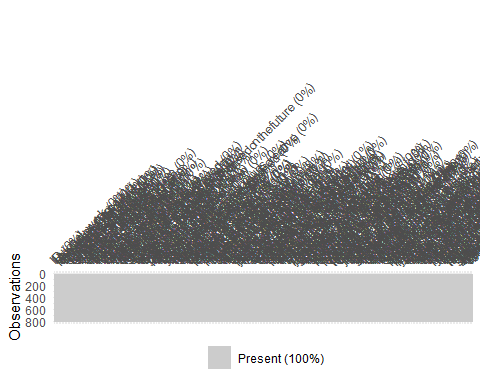
##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## ID = col\_character(),  
## fictional\_work = col\_character(),  
## character\_name = col\_character()  
## )  
## i Use `spec()` for the full column specifications.

#Exploring

vis\_dat(characters) #ugly, too much data



vis\_miss(character)#no missingness



Really good to be exploring the data like this before you get started. Do remember that this is aggregate data, so we can’t really assess how much missing values existed before aggregation. Did the data owner average only for respondents who completed the entire activity? Or any rating? How often do respondents complete the entire activity? I bet is lower than we’d like :P

# PCA

Apply Principal Component Analysis to reduce the number of variables.

* Decide how many principal components (PCs) you will extract.
* Examine your chosen number of PCs and the variables that contribute most to them. Interpret the PCs. What do they represent according to you? Try to come up with labels (fun allowed).
* Create a dataset with the fictional characters’ score on the number of PCs.

# Include all code required to run PCA here  
  
#Recipe  
char\_pca\_recipe <- recipe(~ ., data = characters) %>%  
 update\_role(ID, fictional\_work, character\_name, new\_role = "id") %>% #what does this do?  
 step\_normalize(all\_predictors()) %>%  
 step\_naomit(all\_predictors()) %>% #not necessary  
 step\_pca(all\_predictors())  
  
  
# Now we can prep the recipe and take a look at it.   
char\_pca\_prep <- prep(char\_pca\_recipe)  
char\_pca\_prep

## Data Recipe  
##   
## Inputs:  
##   
## role #variables  
## id 3  
## predictor 268  
##   
## Training data contained 800 data points and no missing data.  
##   
## Operations:  
##   
## Centering and scaling for playful\_serious, shy\_bold, ... [trained]  
## Removing rows with NA values in all\_predictors()  
## PCA extraction with playful\_serious, shy\_bold, ... [trained]

#ID = 3   
  
# predictors = 268  
  
#PCA  
char\_pca <- tidy(char\_pca\_prep, 3) #hat is the significance of 3rd step? To get only the pca results. The first step is step\_normalize() the second step is step\_na.omit() and the third step in the recipe is the step\_pca()  
char\_pca

## # A tibble: 71,824 x 4  
## terms value component id   
## <chr> <dbl> <chr> <chr>   
## 1 playful\_serious -0.0393 PC1 pca\_jXfEW  
## 2 shy\_bold 0.0490 PC1 pca\_jXfEW  
## 3 cheery\_sorrowful 0.0198 PC1 pca\_jXfEW  
## 4 masculine\_feminine -0.0116 PC1 pca\_jXfEW  
## 5 charming\_awkward 0.0202 PC1 pca\_jXfEW  
## 6 lewd\_tasteful -0.109 PC1 pca\_jXfEW  
## 7 intellectual\_physical 0.0568 PC1 pca\_jXfEW  
## 8 strict\_lenient 0.0177 PC1 pca\_jXfEW  
## 9 refined\_rugged 0.0484 PC1 pca\_jXfEW  
## 10 trusting\_suspicious 0.0675 PC1 pca\_jXfEW  
## # ... with 71,814 more rows

#counting  
# or "coefficients" or "coordinates".  
count(char\_pca, component) #268 components?

## # A tibble: 268 x 2  
## component n  
## <chr> <int>  
## 1 PC1 268  
## 2 PC10 268  
## 3 PC100 268  
## 4 PC101 268  
## 5 PC102 268  
## 6 PC103 268  
## 7 PC104 268  
## 8 PC105 268  
## 9 PC106 268  
## 10 PC107 268  
## # ... with 258 more rows

char\_variance <-   
 tidy(char\_pca\_prep, type = "variance", 3)  
char\_variance

## # A tibble: 1,072 x 4  
## terms value component id   
## <chr> <dbl> <int> <chr>   
## 1 variance 67.8 1 pca\_jXfEW  
## 2 variance 51.3 2 pca\_jXfEW  
## 3 variance 38.4 3 pca\_jXfEW  
## 4 variance 19.2 4 pca\_jXfEW  
## 5 variance 13.5 5 pca\_jXfEW  
## 6 variance 7.31 6 pca\_jXfEW  
## 7 variance 7.12 7 pca\_jXfEW  
## 8 variance 6.80 8 pca\_jXfEW  
## 9 variance 3.78 9 pca\_jXfEW  
## 10 variance 3.25 10 pca\_jXfEW  
## # ... with 1,062 more rows

count(char\_variance, terms)

## # A tibble: 4 x 2  
## terms n  
## <chr> <int>  
## 1 cumulative percent variance 268  
## 2 cumulative variance 268  
## 3 percent variance 268  
## 4 variance 268

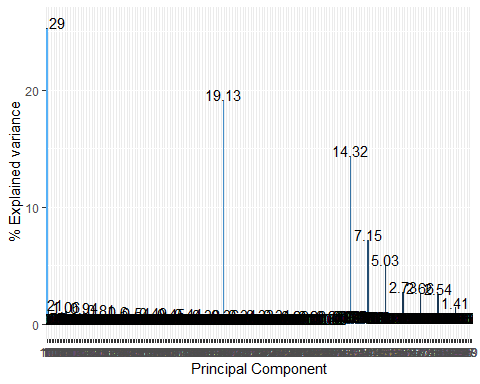
#Remove PCs from column so its only numbers, then filter based on number range   
  
char\_variance$component <- gsub("PC", "", char\_variance$component ) #Removal  
char\_variance$component <- as.numeric(as.character(char\_variance$component )) #conversion  
class(char\_variance$component)

## [1] "numeric"

#filtering for only certain components  
  
 new\_char\_frame\_variance <- char\_variance %>% filter(component %in% (1:5) )  
new\_char\_frame\_variance

## # A tibble: 20 x 4  
## terms value component id   
## <chr> <dbl> <dbl> <chr>   
## 1 variance 67.8 1 pca\_jXfEW  
## 2 variance 51.3 2 pca\_jXfEW  
## 3 variance 38.4 3 pca\_jXfEW  
## 4 variance 19.2 4 pca\_jXfEW  
## 5 variance 13.5 5 pca\_jXfEW  
## 6 cumulative variance 67.8 1 pca\_jXfEW  
## 7 cumulative variance 119. 2 pca\_jXfEW  
## 8 cumulative variance 157. 3 pca\_jXfEW  
## 9 cumulative variance 177. 4 pca\_jXfEW  
## 10 cumulative variance 190. 5 pca\_jXfEW  
## 11 percent variance 25.3 1 pca\_jXfEW  
## 12 percent variance 19.1 2 pca\_jXfEW  
## 13 percent variance 14.3 3 pca\_jXfEW  
## 14 percent variance 7.15 4 pca\_jXfEW  
## 15 percent variance 5.03 5 pca\_jXfEW  
## 16 cumulative percent variance 25.3 1 pca\_jXfEW  
## 17 cumulative percent variance 44.4 2 pca\_jXfEW  
## 18 cumulative percent variance 58.7 3 pca\_jXfEW  
## 19 cumulative percent variance 65.9 4 pca\_jXfEW  
## 20 cumulative percent variance 70.9 5 pca\_jXfEW

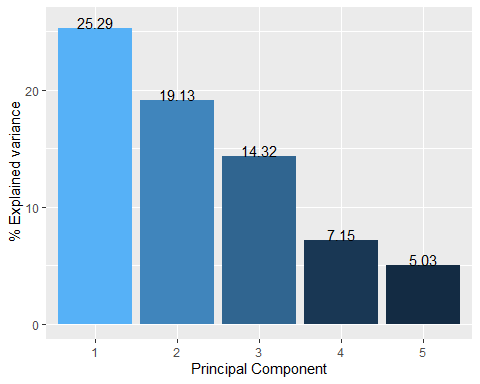
# Let's plot them!   
char\_variance %>%  
 filter(terms == "percent variance") %>%  
 mutate(component = as.character(component)) %>%  
 ggplot(aes(x = component,   
 y = value,  
 fill = value)) +  
 geom\_col() +  
 geom\_text(aes(label = round(value, 2)),  
 hjust = 0.5,  
 nudge\_y = 0.5) +  
 labs(x = "Principal Component", y = " % Explained variance") +  
 guides(fill = FALSE)



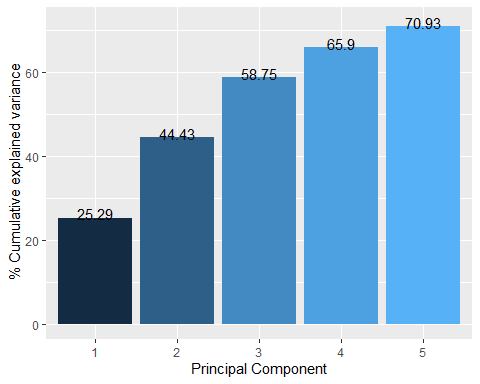
This figure is hard to interpret because you are using the dataframe with all 268 components. The PC variable is also character so the order is PC1 PC100 PC101 PC102 etc ,

Here I would filter the data on the first 50 components or so, knowing that it probably would be enough to make an assessment of the explained variance and make sure the PC is a numeric variable

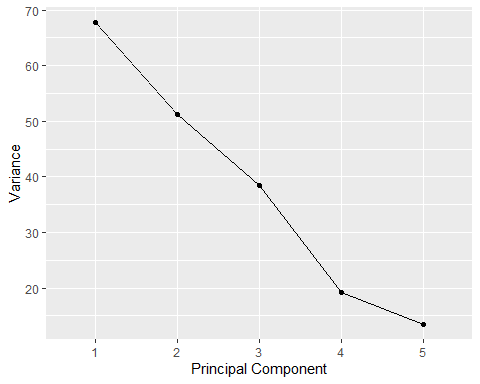
#New data, less components, 5 components, 70% variance explained  
#For example, if you have a 100 variables, but you could explain 75% with just 5 components,  
# you'd definitely get enough bang for your buck! That is, you are trying to   
# simplify the data, but still have most of your explanatory power intact. Yep ☺   
new\_char\_frame\_variance %>%   
 filter(terms == "percent variance") %>%  
 mutate(component = as.character(component)) %>%  
 ggplot(aes(x = component,   
 y = value,  
 fill = value)) +  
 geom\_col() +  
 geom\_text(aes(label = round(value, 2)),  
 hjust = 0.5,  
 nudge\_y = 0.5) +  
 labs(x = "Principal Component", y = " % Explained variance") +  
 guides(fill = FALSE)

Ahh this Is much better than the previous plot!

#plotting  
new\_char\_frame\_variance %>%  
 filter(terms == "cumulative percent variance") %>%  
 mutate(component = as.character(component)) %>%  
 ggplot(aes(x = component,   
 y = value,  
 fill = value)) +  
 geom\_col() +  
 geom\_text(aes(label = round(value, 2)),  
 hjust = 0.5,  
 nudge\_y = 1) +  
 labs(x = "Principal Component", y = " % Cumulative explained variance") +  
 guides(fill = FALSE)

So with 5 you have only 71% of the variance explained. You could have looked at a few more PCs. Going from 268 to let’s say 10 would be a big win, especially if those 10 PCs would explain 95% instead of the 71% that the 5PCs explain

#Scree plot  
# We can now also plot the eigenvalues by making a scree plot.  
new\_char\_frame\_variance %>%  
 filter(terms == "variance") %>%  
 mutate(component = as.character(component)) %>%  
 ggplot(aes(x = component,   
 y = value)) +   
 geom\_point() +  
 geom\_line(group = 1) +  
 labs(x = "Principal Component", y = "Variance")

You can see they are far above 1 (rule of thumb for deciding how many PCs to keep. In fact, the first 19 PCs have an eigenvalue over 1.

#Examine your chosen number of PCs and the variables that contribute most to them. Interpret the PCs. What do they represent according to you? Try to come up with labels (fun allowed).   
#Create a dataset with the fictional characters’ score on the number of PCs.I I would also examine how the variables land on these PC dimensions instead of looking at the individual cases right away  
  
#Creating new data set  
new\_dataset<- juice(char\_pca\_prep) #juice is getting each individual person their scores on those PCA components  
  
  
new\_dataset %>%  
 ggplot() +  
 geom\_point(aes(x = PC1,   
 y = PC2,  
 color = character\_name),  
 size = 2) +  
 labs(title = "Individual scores on the first two dimensions")



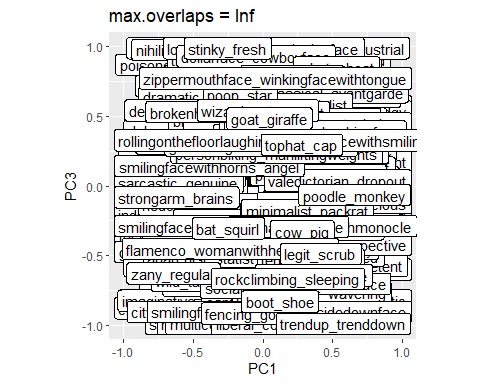
#This is ugly   
  
  
  
#Cleaning ok so now we are going back to the variable – pc dataset right?  
  
char\_pca$component <- gsub("PC", "", char\_pca$component ) #Removal  
char\_pca$component <- as.numeric(as.character(char\_pca$component)) #conversion  
class(char\_pca$component)

## [1] "numeric"

new\_char\_pca <- char\_pca %>% filter(component %in% (1:5) )  
new\_char\_pca

## # A tibble: 1,340 x 4  
## terms value component id   
## <chr> <dbl> <dbl> <chr>   
## 1 playful\_serious -0.0393 1 pca\_jXfEW  
## 2 shy\_bold 0.0490 1 pca\_jXfEW  
## 3 cheery\_sorrowful 0.0198 1 pca\_jXfEW  
## 4 masculine\_feminine -0.0116 1 pca\_jXfEW  
## 5 charming\_awkward 0.0202 1 pca\_jXfEW  
## 6 lewd\_tasteful -0.109 1 pca\_jXfEW  
## 7 intellectual\_physical 0.0568 1 pca\_jXfEW  
## 8 strict\_lenient 0.0177 1 pca\_jXfEW  
## 9 refined\_rugged 0.0484 1 pca\_jXfEW  
## 10 trusting\_suspicious 0.0675 1 pca\_jXfEW  
## # ... with 1,330 more rows

new\_char\_pca$component <-sub("^","PC", new\_char\_pca$component)  
  
#Just a big dot? #i want character names here?   
  
# Set it globally:  
options(ggrepel.max.overlaps = Inf)  
  
new\_char\_pca %>%  
 filter(component == "PC1" | component == "PC3") %>%  
 pivot\_wider(names\_from = "component",   
 values\_from = "value") %>%  
 ggplot(aes(PC1, PC3)) +  
 geom\_segment(xend = 0,   
 yend = 0,  
 arrow = arrow(ends = "first",  
 type = "closed")) +  
 geom\_label\_repel(aes(label = terms)) +  
 xlim(-1, 1) +   
 ylim(-1, 1) +  
 coord\_fixed() +  
 labs(title = "max.overlaps = Inf")

Yeah this too much. Why not let the ggrepel

#Very crowded   
  
#Examine your chosen number of PCs and the variables that contribute most to them. Interpret the PCs. What do they represent according to you? Try to come up with labels (fun allowed).   
#Create a dataset with the fictional characters’ score on the number of PCs.  
  
  
#Interpreting   
pca\_interpret <- new\_char\_pca %>%   
  
 group\_by(terms) %>%  
  
 filter(value == max(value)) %>%  
  
 arrange(value, terms)  
  
#checking  
pca1\_interpret <- pca\_interpret %>%   
 filter(component == "PC1")  
  
pca2\_interpret <- pca\_interpret %>%   
 filter(component == "PC2")  
  
pca3\_interpret <- pca\_interpret %>%   
 filter(component == "PC3")  
  
pca4\_interpret <- pca\_interpret %>%   
 filter(component == "PC4")  
pca5\_interpret <- pca\_interpret %>%   
 filter(component == "PC5")  
  
#Interpretation  
#Definitely good vs evil theme, sweet and sour (strict/lenient, nice/grumpy, hard/soft), stick in the mud/interesting, urban vs country, nerd vs jock,   
#These will be PCs I look at

Good start! I would have like to see a better visualization of the variables on the PCS and a more elaborate interpretation. From your labels I think you did interpret the PCs, but didn’t write your thought process down.

# Clustering

Apply a K means clustering algorithm to examine a typology of fictional characters using the PCs.

* Decide how many clusters you will retain.
* Describe and interpret the clusters
* Select 5 fictional works to plot the characters, a selection of principal components and the clusters to visualize. Discuss why you chose these works and answer the main question (What kind of fictional characters exist?)

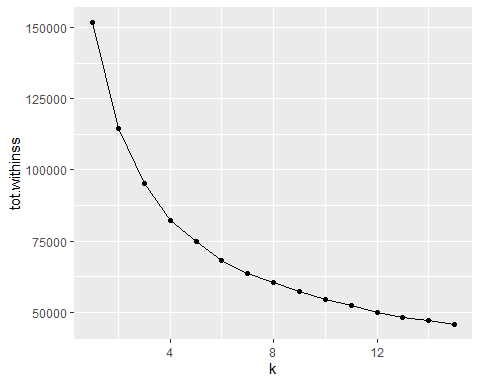
#clustering is reducing number of rows (character genre)  
#cluster analysis on 5 PCs will differ in PC contribution to the characters  
#further explore typology  
#might not have distinctive elbow, don't want to have too many types   
#combine PCs and clusters at individuals from the 5 fictional works (cluster membership on first two PCs)  
  
  
#Filter Data set by 5 fictional works   
#rename PCs  
#Filter PCs for kmeans  
  
PCs\_4\_cluster <- new\_dataset %>%   
 select (PC1, PC2, PC3, PC4, PC5)  
  
#good vs evil theme, sweet and sour (strict/lenient, nice/grumpy, hard/soft), stick in the mud/interesting, urban vs country, nerd vs jock  
  
PCS\_4\_cluster <- PCs\_4\_cluster %>%   
 rename(  
 good\_evil = PC1,  
 sweet\_sour = PC2,  
 stickinmud\_interesting = PC3,  
 urban\_country = PC4,   
 nerd\_jock = PC5  
 )  
  
  
  
#The characters with their PCA values   
set.seed(1234)  
character\_clusters <- kmeans(PCS\_4\_cluster,   
 centers = 5,   
 nstart = 25,  
 iter.max = 20)  
character\_clusters

## K-means clustering with 5 clusters of sizes 162, 102, 164, 218, 154  
##   
## Cluster means:  
## good\_evil sweet\_sour stickinmud\_interesting urban\_country nerd\_jock  
## 1 -4.653496 5.753487 -5.892390 0.4978960 -0.8336385  
## 2 12.289879 1.408462 -4.063778 0.3156962 0.4798270  
## 3 4.700999 5.038211 6.200357 -0.6171094 0.1476295  
## 4 -8.025870 -2.379163 2.501483 0.4102436 0.5237155  
## 5 3.110225 -8.982709 -1.253956 -0.6564112 -0.3394423  
##   
## Clustering vector:  
## [1] 4 3 3 4 1 1 4 2 1 2 1 4 1 4 4 2 5 5 3 5 2 1 3 5 5 3 1 2 1 1 1 2 4 4 4 1 3  
## [38] 4 3 1 1 4 4 4 3 3 4 2 3 4 3 3 5 4 4 4 3 5 4 4 4 1 4 4 4 4 3 1 1 3 3 1 4 1  
## [75] 4 3 3 3 1 1 3 1 3 1 4 3 5 4 5 5 5 1 5 5 5 5 3 4 1 1 2 2 3 3 1 3 1 1 5 1 3  
## [112] 5 5 3 4 3 5 2 3 1 4 4 3 3 5 4 1 4 1 1 4 1 4 1 2 3 5 4 2 4 4 3 3 5 4 3 1 2  
## [149] 2 2 3 4 4 3 4 4 2 2 1 2 3 5 5 1 5 3 1 4 4 3 3 5 4 4 5 3 3 1 4 4 5 4 4 4 4  
## [186] 4 5 4 4 3 1 1 3 4 5 1 3 3 3 4 3 1 5 3 2 1 4 3 5 4 1 1 3 4 3 4 4 5 4 3 2 4  
## [223] 5 1 4 5 1 1 3 5 4 1 2 5 2 2 2 1 2 3 3 4 4 5 3 4 1 4 1 2 3 5 4 4 1 1 3 5 3  
## [260] 4 4 1 2 1 1 2 1 2 5 4 4 4 1 2 4 5 4 4 5 1 4 1 5 4 3 4 4 1 4 5 4 4 4 3 1 2  
## [297] 2 4 4 2 3 5 1 4 4 5 3 4 3 2 5 5 5 2 4 3 1 1 2 3 2 2 2 5 3 1 3 1 4 5 3 4 5  
## [334] 2 2 5 2 1 2 3 1 4 1 5 1 1 2 2 2 2 1 2 1 1 5 2 3 3 3 1 4 3 1 5 4 2 5 4 3 2  
## [371] 1 1 4 3 4 4 1 3 3 5 5 1 1 3 1 3 2 3 4 4 4 4 5 4 4 1 4 4 3 5 4 1 1 1 3 4 3  
## [408] 3 5 4 3 1 3 5 5 4 3 3 4 3 4 3 2 4 4 5 3 5 4 4 1 4 3 1 2 1 3 2 3 5 3 1 2 1  
## [445] 4 4 3 4 2 5 4 5 1 4 1 4 1 1 3 5 4 3 5 2 5 5 1 5 1 3 4 3 3 1 1 5 4 4 3 4 5  
## [482] 4 4 4 2 5 4 2 2 2 2 1 2 1 5 3 2 1 5 4 4 1 1 3 3 5 5 3 1 1 2 1 5 1 5 4 1 5  
## [519] 2 3 5 4 3 3 5 4 5 3 4 4 5 2 5 1 1 1 5 1 3 1 4 3 5 1 2 1 3 4 1 3 3 2 3 3 1  
## [556] 1 5 4 3 4 5 4 4 5 5 2 3 4 5 3 3 5 1 1 4 1 3 1 1 3 5 1 5 2 5 5 4 3 1 4 4 5  
## [593] 5 4 4 1 2 4 1 4 5 4 4 4 1 4 2 4 5 5 5 5 4 1 2 1 2 5 2 5 3 4 3 2 3 3 4 3 4  
## [630] 4 4 2 3 3 5 1 4 3 5 3 5 5 5 5 4 4 5 1 4 1 2 2 5 5 2 2 4 4 5 1 4 2 4 4 1 2  
## [667] 5 3 4 1 1 1 2 5 5 1 3 3 5 3 1 2 2 5 5 1 2 5 1 3 1 2 5 2 1 4 4 1 5 5 3 1 2  
## [704] 2 1 3 1 2 2 2 5 4 4 4 4 4 4 3 5 3 1 3 1 2 1 5 5 4 4 4 5 2 4 4 4 2 4 5 3 4  
## [741] 3 3 4 3 3 1 4 2 5 4 4 1 4 5 5 4 5 3 4 3 5 2 5 5 3 5 3 3 5 5 4 4 5 5 5 1 4  
## [778] 5 3 1 4 3 4 3 2 3 4 4 1 4 3 5 4 4 5 5 4 3 4 3  
##   
## Within cluster sum of squares by cluster:  
## [1] 14237.21 11265.61 17875.71 16457.37 14933.62  
## (between\_SS / total\_SS = 50.8 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

# We now use the broom package to get our results. This shows 1 row for each  
# cluster that we asked for (5) and the variables that we are using in the  
# columns.   
# A negative number means that the cluster scores below the average for all  
# teens and a positive number means that the cluster scores above the average  
# for all teens.  
 tidy(character\_clusters)

## # A tibble: 5 x 8  
## good\_evil sweet\_sour stickinmud\_intere~ urban\_country nerd\_jock size withinss  
## <dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl>  
## 1 -4.65 5.75 -5.89 0.498 -0.834 162 14237.  
## 2 12.3 1.41 -4.06 0.316 0.480 102 11266.  
## 3 4.70 5.04 6.20 -0.617 0.148 164 17876.  
## 4 -8.03 -2.38 2.50 0.410 0.524 218 16457.  
## 5 3.11 -8.98 -1.25 -0.656 -0.339 154 14934.  
## # ... with 1 more variable: cluster <fct>

# Deciding k.  
# How do you know know how many subgroups you should ask for? One way to  
# determine k is to look at a scree plot. We would need to run a k means model  
# for different k's (e.g., 1 to 15) and collect the total within-cluster sum of   
# squares (wss). This takes a while  
set.seed(1234)  
tibble(k = 1:15) %>%  
 mutate(kmeansmod = map(k, ~ kmeans(PCS\_4\_cluster,   
 .x,  
 nstart = 25,  
 iter.max = 20)),  
 glanced = map(kmeansmod, glance)) %>%  
 unnest(cols = c(glanced)) %>%  
 ggplot(aes(k, tot.withinss)) +  
 geom\_line() +  
 geom\_point()

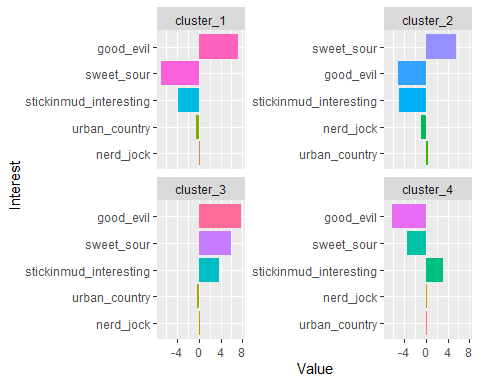


#No elbow, maybe choose 4 Yeah this one is a hard one!  
#redoing  
set.seed(1234)  
character\_clustersv2 <- kmeans(PCS\_4\_cluster,   
 centers = 4,   
 nstart = 25,  
 iter.max = 20)  
character\_clustersv2

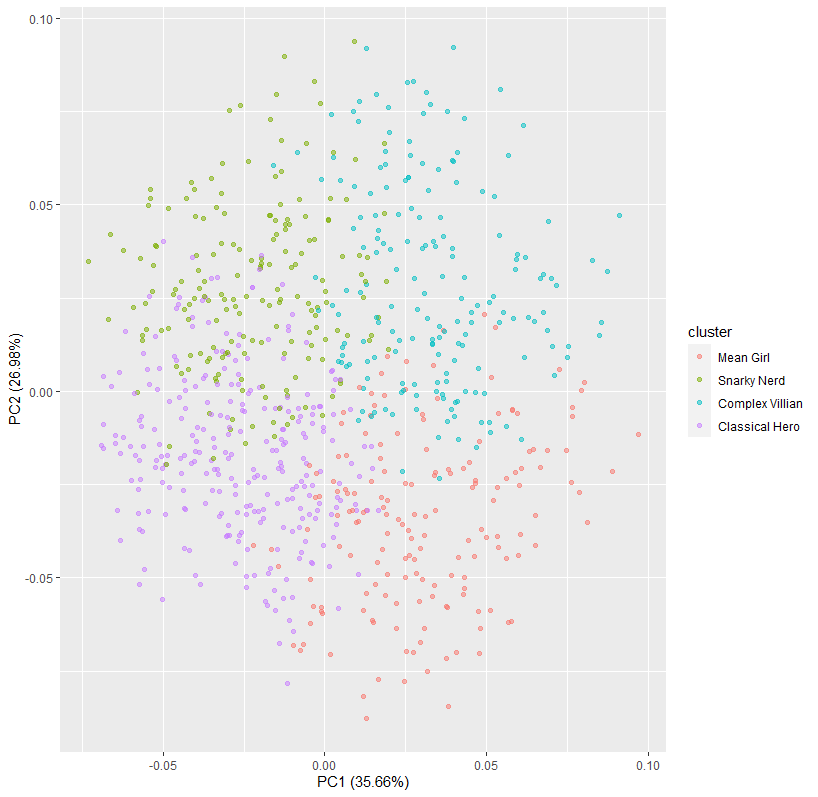
# We now use the broom package to get our results. This shows 1 row for each  
# cluster that we asked for (5) and the variables that we are using in the  
# columns.   
# A negative number means that the cluster scores below the average for all  
# teens and a positive number means that the cluster scores above the average  
# for all teens.  
tidy(character\_clustersv2)

## # A tibble: 4 x 8  
## good\_evil sweet\_sour stickinmud\_intere~ urban\_country nerd\_jock size withinss  
## <dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl>  
## 1 7.16 -6.91 -3.73 -0.410 0.208 167 19967.  
## 2 -5.12 5.63 -4.95 0.462 -0.801 181 16395.  
## 3 7.79 5.85 3.70 -0.348 0.284 180 21544.  
## 4 -6.14 -3.38 3.13 0.174 0.217 272 24423.  
## # ... with 1 more variable: cluster <fct>

# We now plot the most important variables (large negative and positive values)  
# to interpret the clusters.  
# We will create an importance variable that reflects the absolute size by   
# squaring the k means value and then we group our data by cluster and take  
# the largest 10 values based on the importance variable.  
# Then, we need to use the reorder\_within() function from the tidytext package   
# for our multi-cluster visualization.   
tidy(character\_clustersv2) %>%  
 mutate(cluster = str\_c("cluster\_", 1:4)) %>%  
 pivot\_longer(cols = -c(cluster, withinss, size),  
 names\_to = "Interest",  
 values\_to = "Value") %>%  
 mutate(importance = Value \* Value) %>%  
 group\_by(cluster) %>%  
 slice\_max(importance, n = 10) %>%  
 ungroup() %>%  
 mutate(cluster = as.factor(cluster),  
 Interest = reorder\_within(Interest,  
 by = importance,  
 within = cluster)) %>%  
 ggplot() +  
 geom\_col(aes(y = Value,   
 x = Interest,  
 fill = Interest)) +  
 facet\_wrap(~ cluster,   
 nrow = 2,  
 scales = "free\_y") +   
 guides(fill = FALSE) +  
 scale\_x\_reordered() +   
 coord\_flip()



# Let's try to interpret the values  
# Cluster 1: scaled evil, sweet, stick in mud - Mean Girl?   
# Cluster 2: scaled sour, good, stick in mud, nerd - Snarky Nerd?  
# Cluster 3: scaled evil, sour, interesting - Complex Villain?  
# Cluster 4: good, sweet, interesting - Classical hero?  
  
  
# Another visualization is to plot clusters on principal components. You could either first run a PCA, juice the object, and join the PCA loadings to your data (see lab last week) or run autoplot(). For a k means cluster object, autoplot will return the clusters plotted on the first two principal components. This is helpful if you have many variables   
and the clusters are way up there in high dimensional space.  
# autoplot(character\_clustersv2,   
# PCS\_4\_cluster,   
# alpha = 0.5) +  
# scale\_color\_discrete(breaks=c("1", "2", "3", "4"),  
# labels=c("Mean Girl", "Snarky Nerd", "Complex Villain",   
# "Classical Hero"))  
This wanted commented out because it would not knit but I pasted the graph in below that is ok for me. Wonder why it didn’t want to knit!



#Snarky Nerd and Classical Hero seem to have some overlap while the Complex Villain and Mean Girl have overlap. Classical hero and complex Villain don't mix much, understandably. Snarky Nerd and Mean Girl also seem to not mix much. I like my clusters! They make sense. :D   
  
  
# Select 5 fictional works to plot the characters, a selection of principal components, and the clusters to visualize. Discuss why you chose these works and answer the main question (What kind of fictional characters exist?)  
  
  
#Need to combine cluster info, PCs, characters, and 5 shows   
#Shows: Mean Girls, Pirates of the Caribbean, Schitt's Creek, Star Wars, Big Bang

#adding clusters  
clust <- mutate(new\_dataset, cluster = character\_clustersv2$cluster)  
  
#Filtering 5 datasets  
works <- c("Mean Girls", "Pirates of the Caribbean", "Schitt's Creek", "Star Wars", "Big Bang", "The Big Bang Theory")  
  
filteredworks <- filter(clust, fictional\_work %in% works)  
  
# We make five dummy variables based on cluster membership.  
#"Mean Girl", "Snarky Nerd", "Complex Villain", "Classical Hero"  
  
filteredworks$cluster <- as.numeric(filteredworks$cluster)  
  
  
filtereddata <- filteredworks %>%  
mutate(cluster = case\_when(  
 cluster == 1 ~ "Mean Girl (Or Guy)",  
 cluster == 2 ~ "Snarky Nerd",  
cluster == 3 ~ "Complex Villain",  
 cluster == 4 ~ "Classical Hero"))  
  
   
#Plotting   
ggplot(filtereddata) +  
 geom\_text(aes(PC1, PC2, label =character\_name, color = cluster)) +  
 # guides(color= TRUE) +  
 facet\_wrap(~ fictional\_work)

****

Good figure!

**Discussion**

**Critically discuss the data and algorithms in relation to the research question. Do you have a suggestion for an improved analysis?**

* **Decide how many principal components (PCs) you will extract.**
* **Examine your chosen number of PCs and the variables that contribute most to them. Interpret the PCs. What do they represent according to you? Try to come up with labels (fun allowed).**
* **Create a dataset with the fictional characters’ score on the number of PCs.**
* **Decide how many clusters you will retain.**
* **Describe and interpret the clusters**
* **Select 5 fictional works to plot the characters, a selection of principal components and the clusters to visualize. Discuss why you chose these works and answer the main question (What kind of fictional characters exist?)**

This assignment looks at both PCA and clustering. PCA is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. It works by increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Clustering, on the other hand, is another unsupervised machine learning, process that consists of the grouping of similar data points into a cluster. Given a finite set of data points, clustering aims to find homogeneous subgroups of data points with similar characteristics. The data for this assignment consisted of a file that contains the average ratings of 800 fictional characters made by 1,665,664 users of the on-line test at https://openpsychometrics.org/tests/characters/ for 268 different "traits" defined by a bipolar adjective pair.

For the first part of the assignment, I decided to retain 5 PCs. This is because with just 5 components, 70% of the variance is explained. The variables the contributed the most the 5 PCs could be grouped into what I deem the following with the highest eigenvalues: good vs evil theme, sweet and sour (strict/lenient, nice/grumpy, hard/soft), stick in the mud/interesting, urban vs country, and nerd vs jock. For the second section, focused on clustering, I decided to retain four clusters. This was motivated because looking at the scree plot, the distance seemed larger amongst the first four points, indicating the most distinction. There was no elbow in the plot so I visually evaluated the points. Looking at the scaling of the values for the five variable characteristics I determined, I noticed a couple of patterns. Using the bar chart that showed the directionality and weight of the eigenvector of the PC to the cluster, I came up with the following.

Cluster 1 was scaled evil, sweet, and stick in mud, which gave me the connotation of the quintessential mean girl in the movie, who is mean, fake, and perhaps sickly sweet. Regina George from Mean Girls was classified as this so I was pretty happy about that, but Darth Vader was also lumped into this cluster so I think it needs a bit more work. Cluster 2 was scaled sour, good, stick in mud, and nerdy – I called this the “Snarky Nerd” but likeable nerdy character that’s kind of brooding and sarcastic. Leonard and Raj from Big Bang Theory got sorted here as well Stevie Budd (sarcastic queen) and C-P30. Yay! But so did Cady and Aaron from Mean Girls and Luke Skywalker. Not quite the stereotype there. For Cluster 3, it was scaled evil, sour, interesting aka the Complex Villain? Who doesn’t love a complicated villain origin story. Jack Sparrow is here fittingly. Han Solo the smuggler is also here. Janis from Mean Girls too. But Penny and Howard somehow also ended up in cluster three and they just don’t git the genre. Lastly, Cluster 4 was scaled good, sweet, interesting—obviously the Classical hero! Johnny Rose, the protagonist in Schitt’s Creek, falls here. So do Will Turner and Elizabeth Swan, Princess Leia, and Amy Farah Fowler. This one didn’t seem to mischaracterize someone, but it didn’t get all the characters that it could have. I chose Mean Girls, Schitt’s Creek, The Big Bang Theory, Pirates of the Caribbean, and Star Wars because I was familiar with the content, but also because I thought that each work had a specific character that fit into the five main PCs with the highest eigenvalues: good vs evil theme, sweet and sour (strict/lenient, nice/grumpy, hard/soft), stick in the mud/interesting, urban vs country, and nerd vs jock, and the four clusters: Mean Girl (or Guy), Snarky Nerd, Complex Villain, and Classical Hero. I tried to also choose popular shows that followed some stereotypes, but also provided characters with complex, well-rounded personalities. Lastly, the works that I chose also provided a chance to see if my clusters were appropriately named as there were characters that were similar and also different—there were certain characters I was worried might be misclassified that didn’t and others I didn’t think would be that were. Good work here. I like that you summarize the algorithms and lay out your thought process here. Justification of fictional works is good!

Overall, I think my distillation of what kinds of characters exist, Mean Girls (or Guys), Snarky Nerd, Complex Villain, and Classical Hero fit the bill for the types of characters normally seen in works of fiction. However, I think if it was broken down into target audience, like YA, moms (Hallmark movie/Lifetime-type stuff, maybe soap operas), you would see even clearer lines between the types of characters that exist. I think another thing that also needs to be improved is the identification of characters that are not necessarily evil, but perhaps drive the plot of the work through mishap. For example, Moira, Daniel, and Alexis from Schitt’s Creek. They aren’t complex villains, but rather, complex characters whose faults and mishaps drive the plot forward. Perhaps the same from Penny and Howard. Furthermore, the evilness of Regina George is much different than Dark Vader, but both serve as the antagonist of their respective fictional work. If I would do it over, I might separate the good vs evil into protagonist and antagonist instead of Mean Girl and Classical hero and Snarky Nerd. Perhaps I would also had supporting character. From the autoplot, you can see that the data has well separated clusters. That means that the performance of k-means depends completely on the goodness of the initialization. Initialization, perhaps the set of centroids to be used for initialization, needs to be improved.

For this test in the waters of PCA and Clustering, I believe this data to be appropriate. However, I think including characters from so many different shows made the process more difficult as there were many works I was not familiar with, and this will often be the case! Good to try to understand the patterns in machine learning and get familiar with the data. Hence zooming in on 5 works you know therefore, I think sorting the data to be only shows I knew would have been a better move earlier, so I could recognize patterns faster. I’d like to also follow up and look at a dataset that also had the number of lines a character says in the work. This is known as the Bechdel–Wallace test. The test is a measure of the representation of women in fiction. It asks whether a work features at least two women who talk to each other about something other than a man. Yes it will be very interesting to examine gender bias in how people perceive fictional characters. Nice discussion. Other things you could have mentioned would be the non-random volunteer sample