Homework 5: Self-Organizing Maps

MACS 30100: Perspectives on Computational Modeling
University of Chicago

Yile Chen, Jinfei Zhu, Xi Cheng, Boya Fu

The Task

Here are the prompts to guide your task. Again, **there is no single way this problem set should be executed**. Simply do your best, leveraging all tools and techniques we have covered, and most importantly defend ***all*** choices you make throughout the process so you can at least earn partial credit where appropriate

1. Read in the 2016 ANES data we have been using (`anes_2016.csv`), and create a subset of the data containing *at least* the main 14 questions/features (from above) as these are the core of the analysis.

```
library(tidyverse)
library(kohonen)
library(ggpubr)
library(corrplot)
library(tictoc)
library(amerika)
library(ppclust)
library(factoextra)

anes <- read_csv("anes_2016.csv")
anes_short <- anes %>%
  dplyr::select(pid3, vaccine, autism, birthright_b, forceblack,
                forcewhite, stopblack, stopwhite, freetrade, aa3,
                warmdo, finwell, childcare, healthspend, minwage)
```

2. Preprocess and clean the data.

```
#data preprocess

skimr::skim(anes_short)
```

Table 1: Data summary

Name	anes_short
Number of rows	1200
Number of columns	15
Column type frequency:	
numeric	15
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
pid3	0	1	2.07	0.99	1	1	2	3	5	
vaccine	0	1	2.30	1.74	1	1	1	3	7	
autism	0	1	4.48	1.56	1	3	5	6	6	
birthright_b	0	1	6.58	2.83	1	4	7	9	9	
forceblack	0	1	3.59	1.05	1	3	3	5	5	
forcewhite	0	1	2.68	0.86	1	2	3	3	5	
stopblack	0	1	3.66	1.03	1	3	4	5	5	
stopwhite	0	1	2.69	0.94	1	2	3	3	8	
freetrade	0	1	3.90	1.62	1	3	4	5	7	
aa3	0	1	6.71	2.67	1	4	7	9	9	
warmdo	0	1	3.20	2.05	1	1	3	4	8	
finwell	0	1	4.59	1.73	1	3	5	6	8	
childcare	0	1	3.38	1.79	1	2	3	4	7	
healthspend	0	1	3.26	1.92	1	2	3	4	8	
minwage	0	1	1.60	0.91	1	1	1	2	8	

```r
anes_missing <- anes %>%
  dplyr::select(pid3, vaccine, autism, birthright_b, forceblack,
                forcewhite, stopblack, stopwhite, freetrade, aa3,
                warmdo, finwell, childcare, healthspend, minwage) %>%
  mutate(vaccine = replace(vaccine, vaccine > 7, NA),
                autism = replace(autism, autism > 6, NA),
                birthright_b = replace(birthright_b, birthright_b > 7, NA),
                forceblack = replace(forceblack, forceblack > 5, NA),
                forcewhite = replace(forcewhite, forcewhite > 5, NA),
                stopblack = replace(stopblack, stopblack > 5, NA),
                stopwhite = replace(stopwhite, stopwhite > 5, NA),
                freetrade = replace(freetrade, freetrade > 7, NA),
                aa3 = replace(aa3, aa3 > 7, NA),
                warmdo = replace(warmdo, warmdo > 7, NA),
                finwell = replace(finwell, finwell > 7, NA),
                childcare = replace(childcare, childcare > 7, NA),
                healthspend = replace(healthspend, healthspend > 7, NA),
                minwage = replace(minwage, minwage > 4, NA),
                pid3 = replace(pid3, pid3 > 3, 3))
```

In this question, we find that the value of some variables are out of their expected scale. For example, the scale of variable `birthright` should be between 1 and 7. However, we have a lot of 9 here (from 75 percentile to 100 pencentile). After consulting the code book, we find that when the question is not asked, the responses are coded as 9, and there are 588 out of 1200 observations are recorded as 9. So if we simply mark them as NA and drop them, we will lose a lot of data. So an alternative way is to use the mean of the data to impute the missing value (though this is not a perfect solution either, it's better than simply dropping all missing values, which will result in around 300 observations left!)

```r
NA2mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))
anes_imputed <-replace(anes_missing, TRUE, lapply(anes_missing, NA2mean))

tibble(sum(is.na(anes_missing)), sum(is.na(anes_imputed)))
```

```
## # A tibble: 1 x 2
##   `sum(is.na(anes_missing))` `sum(is.na(anes_imputed))`
##                        <int>                      <int>
## 1                       1186                          0
```

We can find that before imputation, the missing value is 1186. After imputation, the missing value is 0. Then we can label the `pid3` variable.

```r
anes_short <-anes_imputed %>%
  mutate(pid3 = replace(pid3, pid3 >= 3, "others"),
         pid3 = replace(pid3,  pid3 == 2,  "republican"),
         pid3 = replace(pid3, pid3 == 1, "democrat"),
         pid3 = as.factor(pid3))

skimr::skim(anes_short)
```

Table 3: Data summary

Name	anes_short
Number of rows	1200
Number of columns	15
Column type frequency:	
factor	1
numeric	14
Group variables	None

Variable type: factor

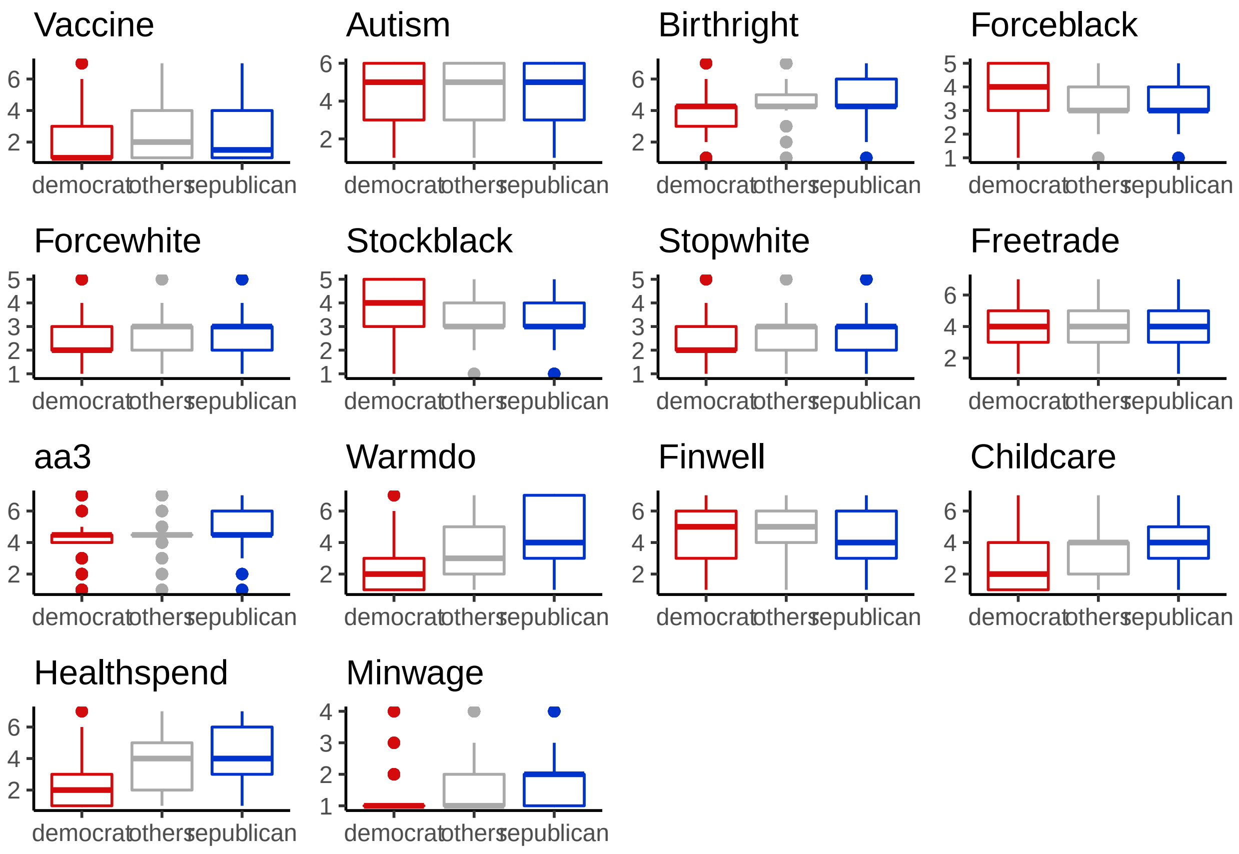
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
pid3	0	1	FALSE	3	oth: 461, dem: 459, rep: 280

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
vaccine	0	1	2.30	1.74	1	1	1.00	3.00	7	
autism	0	1	4.48	1.56	1	3	5.00	6.00	6	
birthright_b	0	1	4.26	1.55	1	4	4.26	4.26	7	
forceblack	0	1	3.59	1.05	1	3	3.00	5.00	5	
forcewhite	0	1	2.68	0.86	1	2	3.00	3.00	5	
stopblack	0	1	3.66	1.03	1	3	4.00	5.00	5	
stopwhite	0	1	2.69	0.92	1	2	3.00	3.00	5	
freetrade	0	1	3.90	1.62	1	3	4.00	5.00	7	
aa3	0	1	4.48	1.42	1	4	4.48	4.48	7	
warmdo	0	1	3.20	2.05	1	1	3.00	4.00	7	
finwell	0	1	4.59	1.73	1	3	5.00	6.00	7	
childcare	0	1	3.38	1.79	1	2	3.00	4.00	7	
healthspend	0	1	3.25	1.91	1	2	3.00	4.00	7	
minwage	0	1	1.60	0.89	1	1	1.00	2.00	4	

3. Explore the data using any approach(es) or tool(s) you think best, such as feature-level correlations, boxplots, scatterplots, density plots, etc.

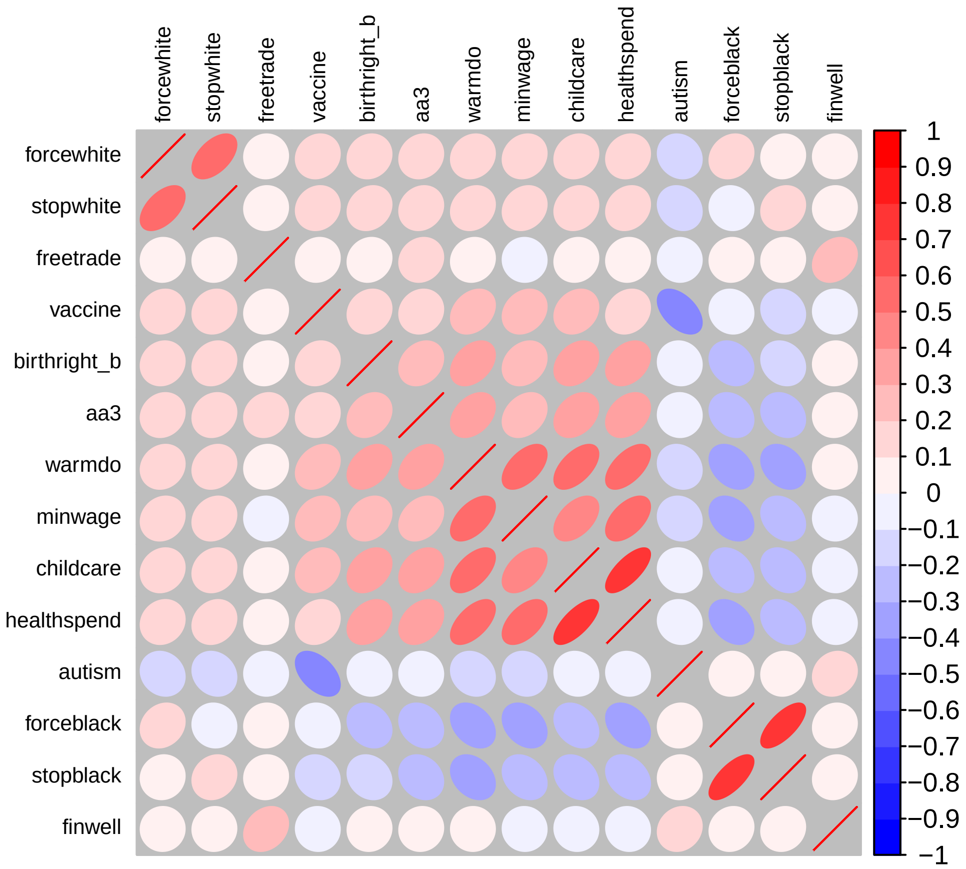
```r
#Creat box plot for each variable
box_plot <- function(n, t) {ggplot(data = anes_short, aes(pid3, n, color = pid3)) +
    geom_boxplot(alpha = 1, show.legend = FALSE) + labs(x = NULL, y = NULL, title = t) +
    scale_color_manual(values=c(amerika_palettes$Republican[2], "darkgrey",
                  amerika_palettes$Democrat[2])) + theme_classic()}

ggarrange(box_plot(anes_short$vaccine, "Vaccine"),
          box_plot(anes_short$autism, "Autism"),
          box_plot(anes_short$birthright_b, "Birthright"),
          box_plot(anes_short$forceblack, "Forceblack"),
          box_plot(anes_short$forcewhite, "Forcewhite"),
          box_plot(anes_short$stopblack, "Stockblack"),
          box_plot(anes_short$stopwhite, "Stopwhite"),
          box_plot(anes_short$freetrade, "Freetrade"),
          box_plot(anes_short$aa3, "aa3"),
          box_plot(anes_short$warmdo, "Warmdo"),
          box_plot(anes_short$finwell, "Finwell"),
          box_plot(anes_short$childcare, "Childcare"),
          box_plot(anes_short$healthspend, "Healthspend"),
          box_plot(anes_short$minwage, "Minwage"),
          ncol = 4, nrow = 4)
```



From the boxplots, we can observe the difference among different parties of each feature. Opinions of toward some variables do not have notible partisan differences: such as `autism`, `forcewhite`, `stopwhite`,`freetrade`. However, people with different party affiliations have very different perspectives toward topics such as `vaccine`, `warmdo` (climate change), `birthright`, `childcare`, `minwage` and `healthspend`. In general, democrats are more supportive to topics related to racial justice, climate change, health care and (raising) minimal wage.

Draw the correlation plots:

```r
#Explore feature-level correlations
cor <- cor(anes_short[,2:15])

palette <- colorRampPalette(c("blue", "white", "red")) (20)
corrplot(cor, tl.cex = 0.7,  tl.col = "black", method='ellipse', bg='grey', col = palette, order = "AOE
```



The above matrix shows the correlation between variable pairs. We can see that there are a strong positive correlation between `stopblack` and `forceblack`, and `healthspend` and `childcare` - meaning that, e.g., people who think police officers use more force than is necessary to BLACK people also tend to believe that police officers stop BLACK people on the street without a good reason. People who favor an increase in government spending to help working parents pay for CHILD CARE when they can't pay for it all themselves also tend to favor an increase in government spending to help people pay for HEALTH INSURANCE. Meanwhile, there is a notable negative correlation between `autism` and `vaccine`, which suggests that those who believe that vaccines can cause autism, tend to oppose requiring children to be vaccinated in order to attend public schools.

4. Construct and present a self-organizing map (SOM) of the *question space*. **Think carefully about the scale of response categories, as these vary across questions.** Also, remember to tune the relevant hyperparameters appropriately. You might consider the `kohonen` package in R (though there are many others), or the `minisom` package in Python. A response to this may include creating grids, fitting models, and creating (multiple) visualizations of the results.

```r
#Scale variables - except pid3
anes_scale <- scale(anes_short[,2:15])
#anes_short <- cbind(anes_short[,1], scale(anes_short[,2:15]))
head(anes_scale)
```

```
##          vaccine     autism birthright_b forceblack forcewhite  stopblack
## [1,] -0.7449753  0.9760944   -1.4559963  0.3881195 -0.7998299  0.3255877
## [2,] -0.7449753  0.3339270    0.0000000  0.3881195  1.5382843  0.3255877
## [3,]  0.4040869 -1.5925751    0.0000000 -1.5167613 -0.7998299 -0.6415044
## [4,] -0.7449753  0.9760944    0.0000000  0.3881195  0.3692272  0.3255877
## [5,] -0.7449753  0.9760944    0.4741187 -0.5643209  0.3692272 -0.6415044
## [6,]  2.7022114 -0.9504077    0.0000000  1.3405599  2.7073415 -0.6415044
##        stopwhite   freetrade       aa3    warmdo    finwell childcare
## [1,] -0.7427138  0.06466994  0.000000 -1.073036  0.8150347 -1.331864
## [2,]  1.4213540  0.06466994  0.000000  0.392306  0.2367941 -0.772257
## [3,]  0.3393201  0.06466994  1.776195  1.857648  1.3932753  1.466169
## [4,] -0.7427138  0.68057411 -1.750672 -1.073036  0.8150347 -1.331864
## [5,]  0.3393201  0.06466994  0.000000  1.857648 -2.0761683  2.025776
## [6,]  0.3393201 -1.78304257  0.000000  1.857648  1.3932753  1.466169
##      healthspend    minwage
## [1,]  -1.1826770 -0.6765981
## [2,]  -0.6581131 -0.6765981
## [3,]   1.4401424  2.7082744
## [4,]  -1.1826770 -0.6765981
## [5,]   1.9647063  2.7082744
## [6,]   1.9647063  2.7082744
```

For the 14 social issue questions, we have different scales, for example,

- `vaccine` has a scale of 7 from 1 (favor vaccine) to 7(oppose vaccine)
- `autism` has a scale of 6 from 1(extremely likely that vaccine caused autism) to 6 (extremely unlikely that vaccine caused autism)
- `forceblack` has a scale of 5 from 1 (Never) to 5 (Often)

In this condition, as with other clustering and dimension reduction techniques, features need to be normalized (de-meaned) and standardized (scaled by the SD), to prevent any single feature from taking over the process.

```r
# create the structure of the output layer
set.seed(1234)
search_grid <- somgrid(xdim = 10,
                      ydim = 10,
                      topo = "rectangular",
                      neighbourhood.fct = "gaussian")

search_grid2 <- somgrid(xdim = 5,
                      ydim = 5,
                      topo = "rectangular",
                      neighbourhood.fct = "gaussian")

search_grid3 <- somgrid(xdim = 15,
                      ydim = 15,
                      topo = "rectangular",
                      neighbourhood.fct = "gaussian")

# fit
{
  tic()
som_fit <- som(anes_scale,
               grid = search_grid,
               alpha = c(0.1, 0.001),
               radius = 1,
               rlen = 500,
               dist.fcts = "euclidean",
               mode = "batch")
  toc()
}
```

```
## 3.54 sec elapsed
```

```r
{
  tic()
som_fit2 <- som(anes_scale,
               grid = search_grid2,
               alpha = c(0.1, 0.001),
               radius = 1,
               rlen = 500,
               dist.fcts = "euclidean",
               mode = "batch")
  toc()
}
```

```
## 0.877 sec elapsed
```

```r
{
  tic()
som_fit3 <- som(anes_scale,
               grid = search_grid3,
               alpha = c(0.1, 0.001),
               radius = 1,
               rlen = 500,
               dist.fcts = "euclidean",
               mode = "batch")
  toc()
}
```

```
## 6.517 sec elapsed
```

```r
summary(som_fit)
```

```
## SOM of size 10x10 with a rectangular topology and a gaussian neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: euclidean.
## Training data included: 1200 objects.
## Mean distance to the closest unit in the map: 2.025.
```

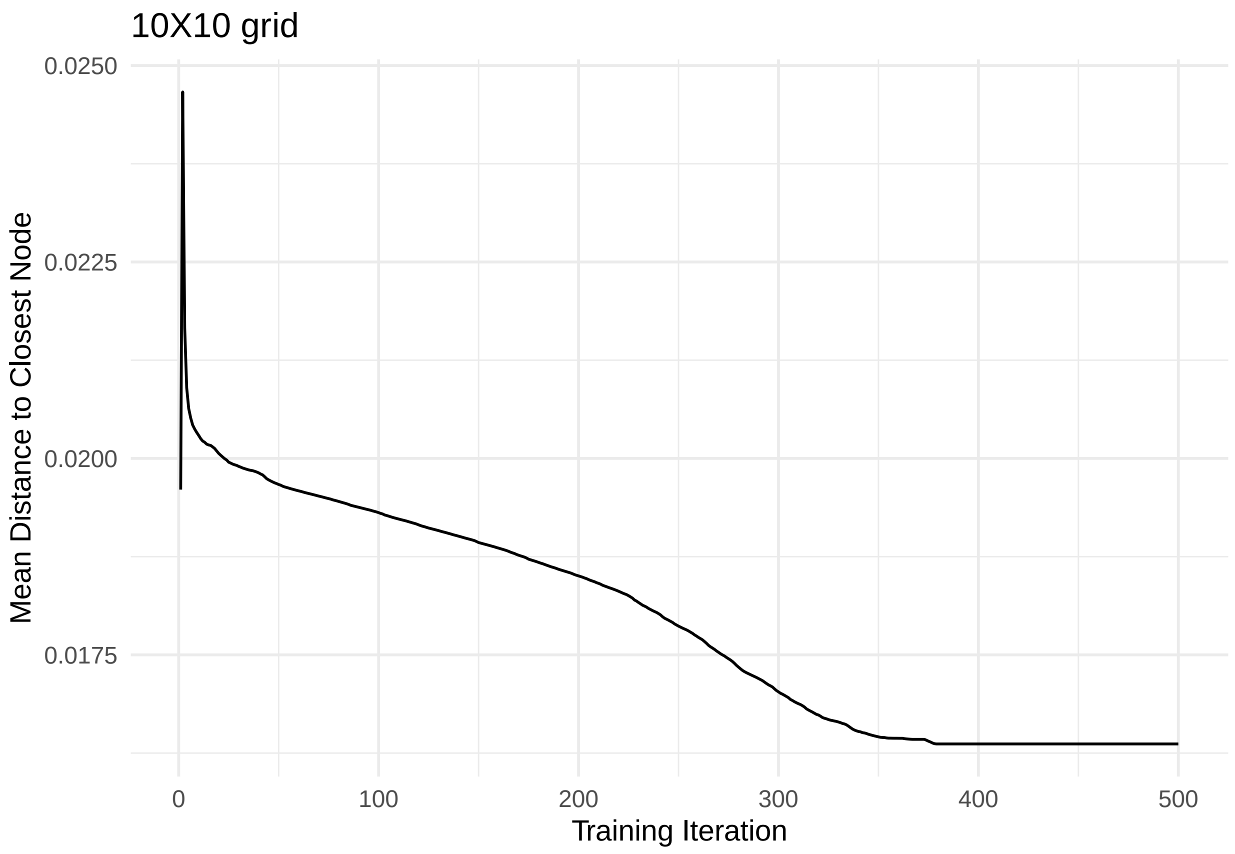
```r
summary(som_fit2)
```

```
## SOM of size 5x5 with a rectangular topology and a gaussian neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: euclidean.
## Training data included: 1200 objects.
## Mean distance to the closest unit in the map: 2.464.
```

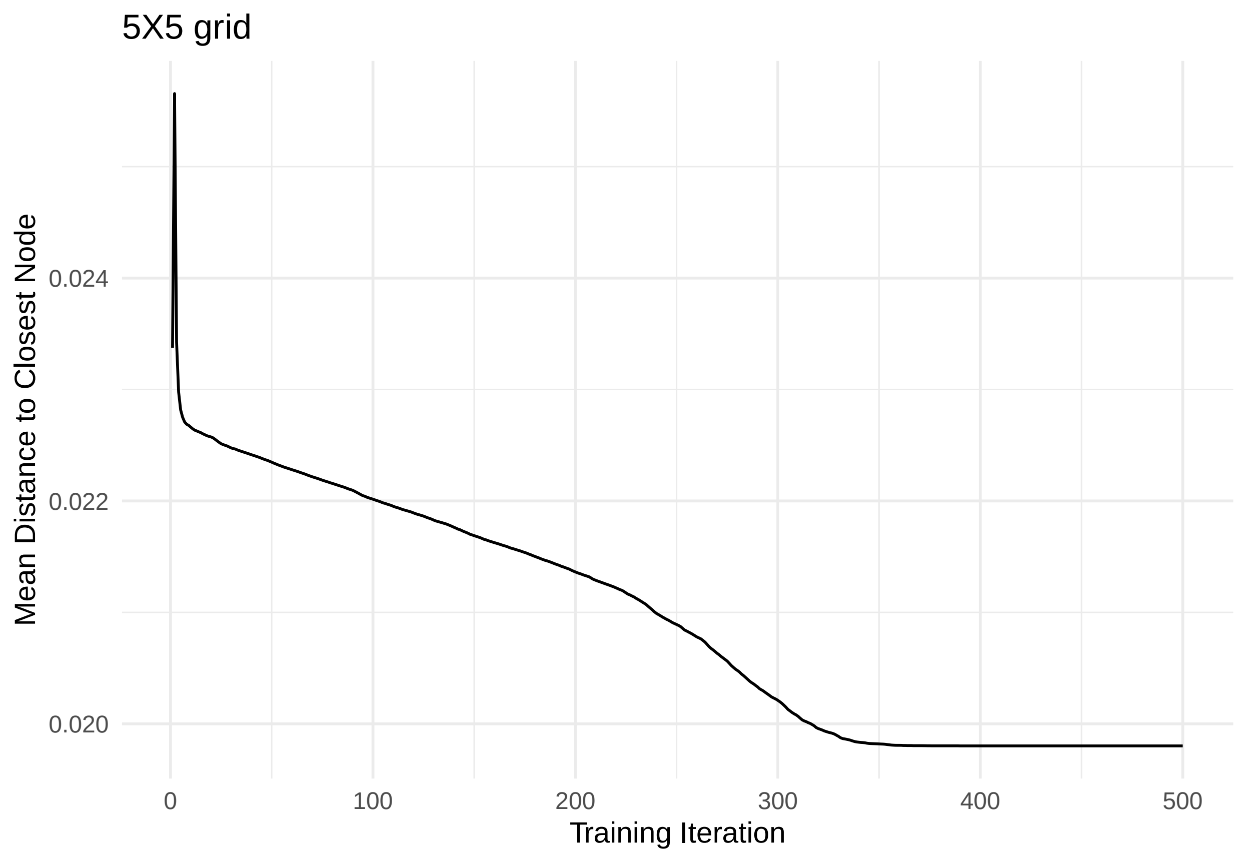
```r
summary(som_fit3)
```

```
## SOM of size 15x15 with a rectangular topology and a gaussian neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: euclidean.
## Training data included: 1200 objects.
## Mean distance to the closest unit in the map: 1.724.
```

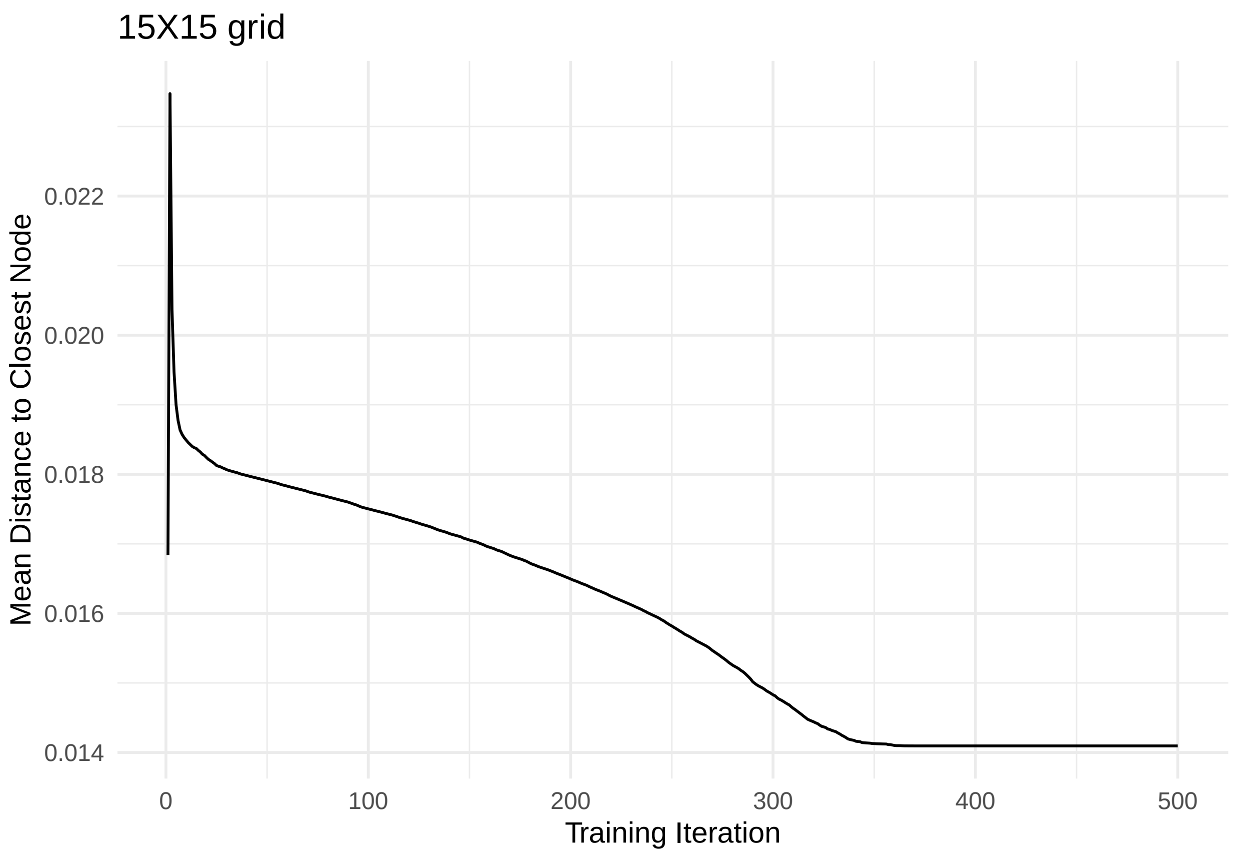
```r
# plot
par(mfrow = c(1,2))
som_fit$changes %>%
  as_tibble() %>%
  mutate(changes = V1,
         iteration = seq(1:length(changes))) %>%
  ggplot(aes(iteration, changes)) +
  geom_line() +
  labs(x = "Training Iteration",
       y = "Mean Distance to Closest Node", title = "10X10 grid") +
  theme_minimal()
```



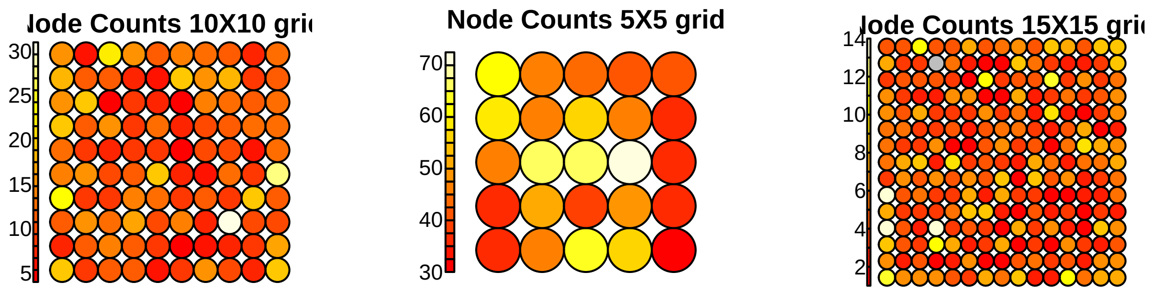
```r
som_fit2$changes %>%
  as_tibble() %>%
  mutate(changes = V1,
         iteration = seq(1:length(changes))) %>%
  ggplot(aes(iteration, changes)) +
  geom_line() +
  labs(x = "Training Iteration",
       y = "Mean Distance to Closest Node", title = "5X5 grid") +
  theme_minimal()
```



```
som_fit3$changes %>%
  as_tibble() %>%
  mutate(changes = V1,
         iteration = seq(1:length(changes))) %>%
  ggplot(aes(iteration, changes)) +
  geom_line() +
  labs(x = "Training Iteration",
       y = "Mean Distance to Closest Node", title = "15X15 grid") +
  theme_minimal()
```



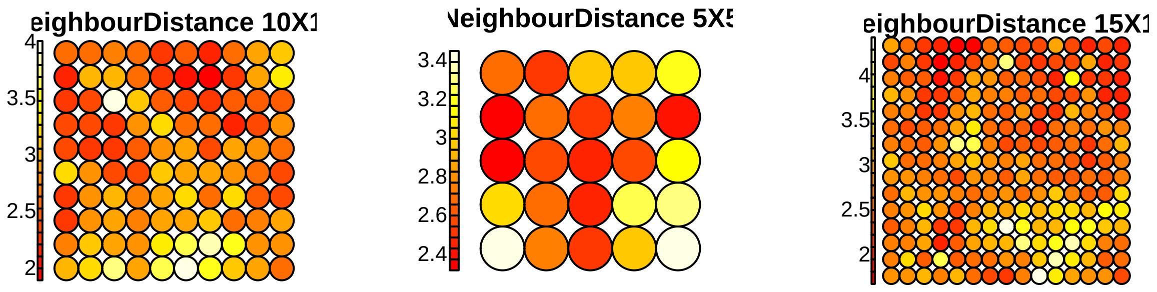
```r
par(mfrow = c(1,3))
plot(som_fit, type="count", main="Node Counts 10X10 grid")

plot(som_fit2, type="count", main="Node Counts 5X5 grid")
plot(som_fit3, type="count", main="Node Counts 15X15 grid")
```



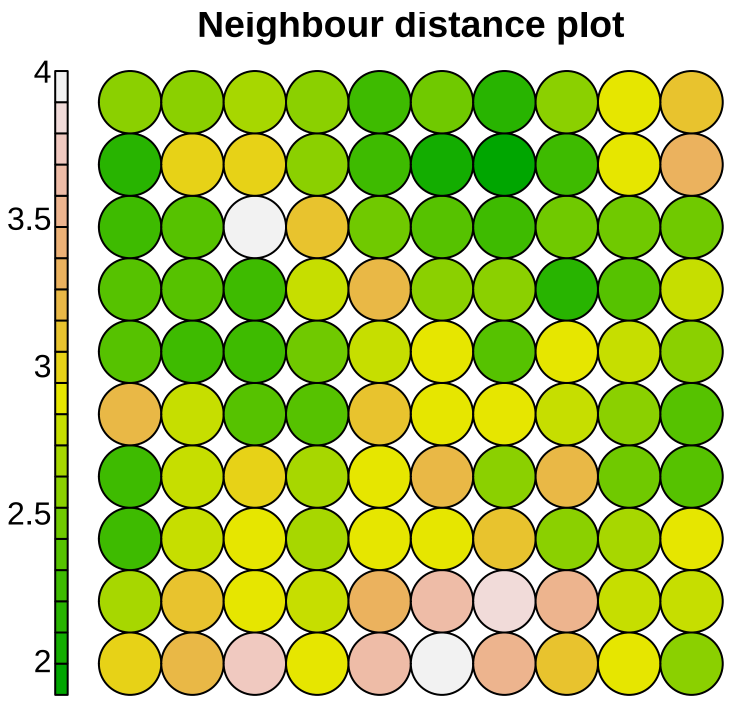
In fitting the model, we tuned three grid size (10X10, 5X5, 15X15). We find that 1. the greater the grid size, the longer the running time, and the smaller the mean distance to closest node (better learning progress). As shown in the three node counts plots, the 5X5 grid only provides 25 area units and has several units which contains more than 60 samples, suggesting a larger grid is needed. Whereas in the 15X15 grid, there are many nodes with no observations, suggesting the grid is too large. Therefore, we decide to choose the 10X10 grid to fit our SOM model.

As the SOM training iterations progress, the distance from each node's weights to the samples represented by that node is reduced. Ideally, this distance should reach a minimum plateau. The 10X10 Mean Distance to Closest Node plot above shows the progress over time. We can see the curve turns steady after 400, so no more iterations are required.

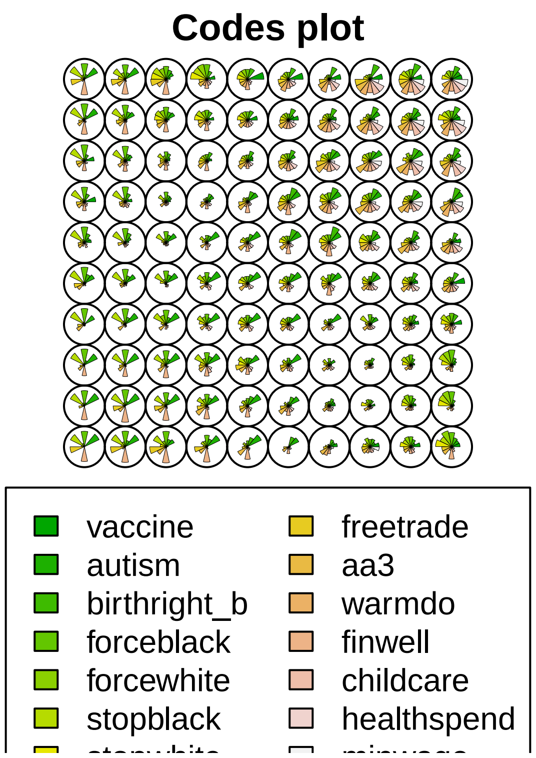
```r
par(mfrow = c(1,3))
plot(som_fit, type="dist.neighbours", main="NeighbourDistance 10X10")
plot(som_fit2, type="dist.neighbours", main="NeighbourDistance 5X5")
plot(som_fit3, type="dist.neighbours", main="NeighbourDistance 15X15")
```

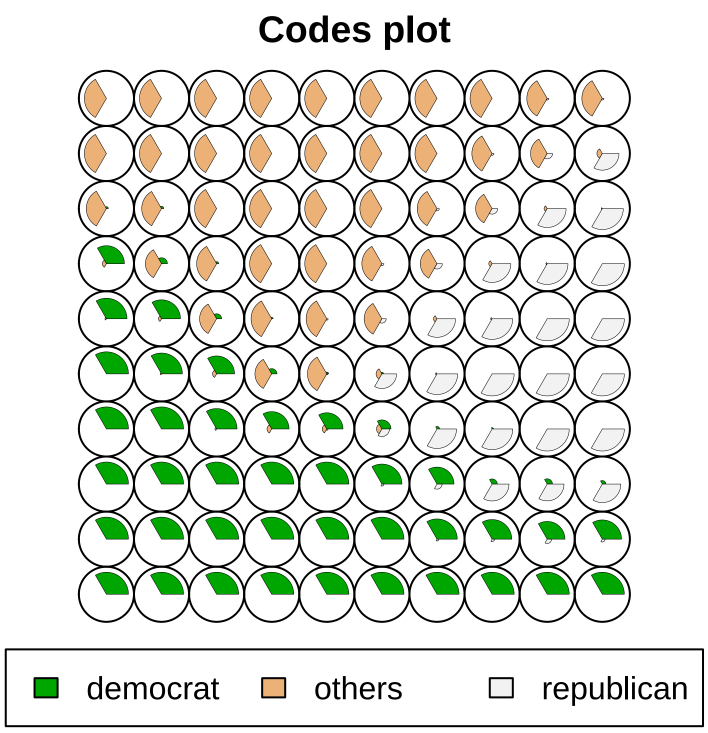


```r
set.seed(2021)
plot(som_fit, type = "dist.neighbours", palette.name = terrain.colors)
```



```r
#inspect partisan clustering
som_map <- xyf(anes_scale,
               classvec2classmat(factor(anes_short$pid3)),
               grid = search_grid,
               rlen = 500)

plot(som_map)
```

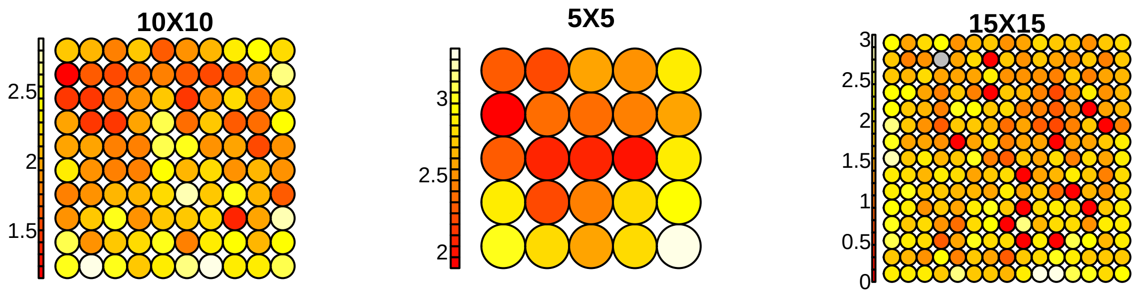




The neighbor distance plot visualizes the distance between each node and its neighbours. Areas of low neighbour distance indicate groups of nodes that are similar. Areas with large distances indicate the nodes are much more dissimilar – and indicate natural boundaries between node clusters. So here we use this plot to identify clusters.

The first codes plot shows patterns in the distribution of samples and variables by visualising the weight vectors across the map. The fan diagram, where individual fan representations of the magnitude of each variable in the weight vector is shown for each node. The second codes plot demonstrates clusters identified by the model. Together wtih the neighbor distance plot, it can be seen that there is partisan differences.

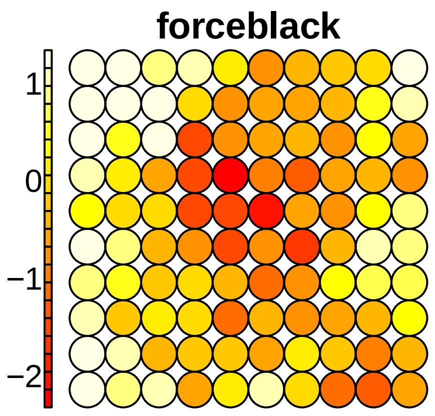
```r
par(mfrow = c(1,3))
plot(som_fit, type = "quality", main = "10X10")
plot(som_fit2, type = "quality", main = "5X5")
plot(som_fit3, type = "quality", main = "15X15")
```

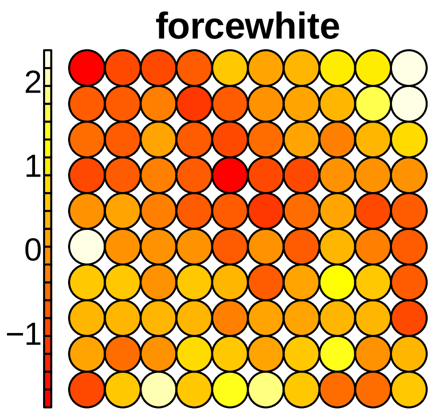


The quality plots also show that 10X10 grid is better than the other two. Large values in several areas of the 5X5 grid suggests that a larger map would be benificial. Empty nodes in 15X15 indicate that the map size is too big for the number of samples.

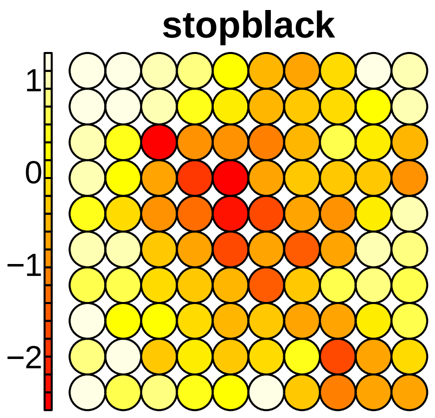
5. Comment on the results thus far as it relates to the main goal of the task. In other words, did you uncover evidence of partisan differences in the data? Did you not? Regardless, why do you think you got the results you did? What are some other substantive patterns you detected?

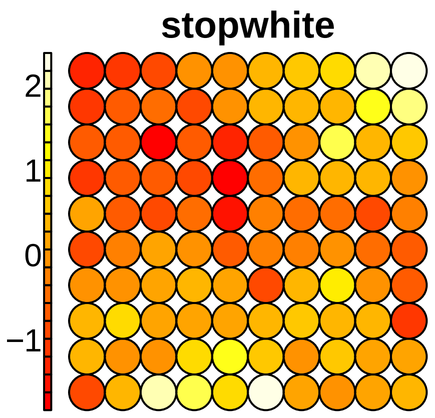
```r
set.seed("2021")

#forceblack/forcewhite
par(mfrow = c(1,2))
plot(som_fit, type = "property", property = getCodes(som_fit)[,4], main=colnames(getCodes(som_fit))[4])
plot(som_fit, type = "property", property = getCodes(som_fit)[,5], main=colnames(getCodes(som_fit))[5])
```





```r
#stopblack/stopwhite
par(mfrow = c(1,2))
plot(som_fit, type = "property", property = getCodes(som_fit)[,6], main=colnames(getCodes(som_fit))[6])
plot(som_fit, type = "property", property = getCodes(som_fit)[,7], main=colnames(getCodes(som_fit))[7])
```





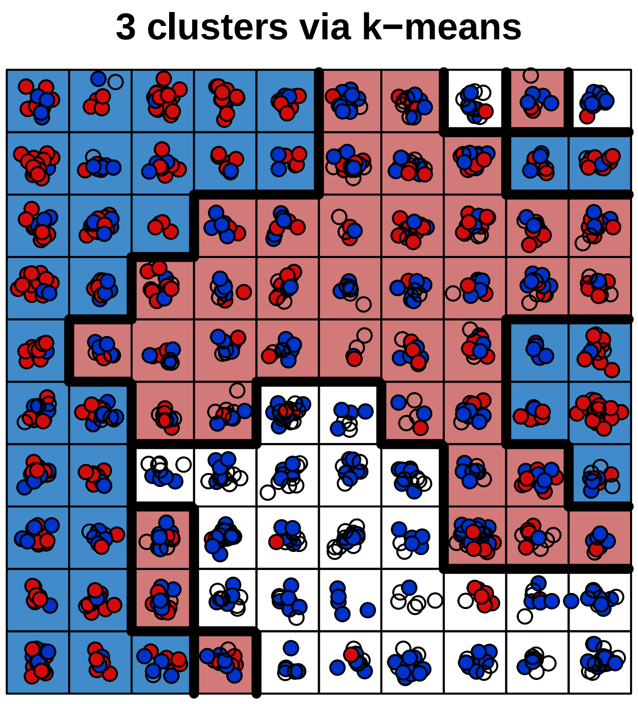
In addition to what has been discussed in Task4, the descriptive statistics in Task 3 are suggesting partisan differences in public opinions, given the differences of opinion on the 14 salient social issues. For example, the boxplots show that democrats are more likely, on average, to claim unnecessary racial disparities in policing (with the highest mean in stopblack and the lowest mean in stopwhite). The feature-level correlations also indicate partisan differences and similarities among these features and thus make the partisan differences in general worth researching.

Similarly, the partisan differences in certain public opinions can be detected through *heatmaps* (displayed above). As is shown in the heatmap for stopblack/stopwhite and forceblack/forcewhite, deeper color on the heatmap indicates a higher tendency to answer “Very often” to these questions. The areas with different colors show some sort of clustering.

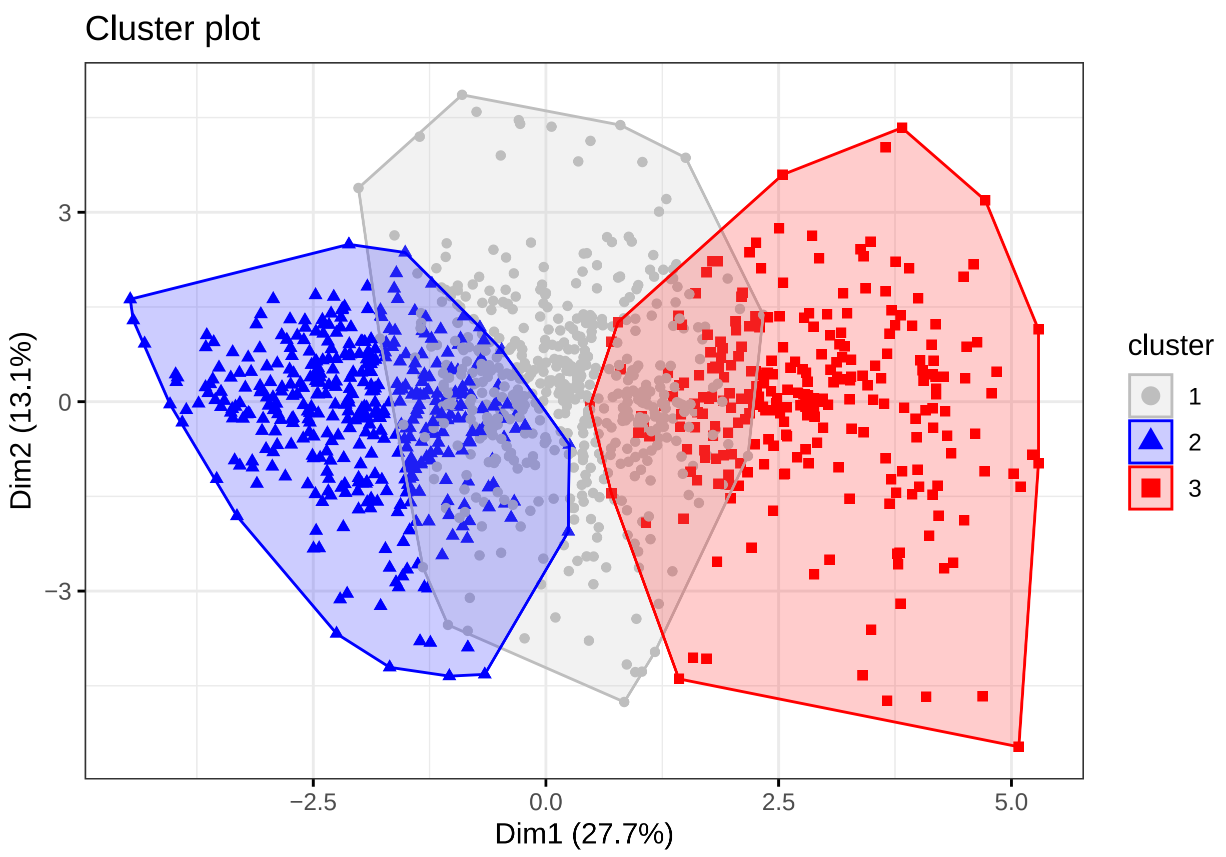
We might be able to detect interesting patterns when reading these heatmaps. For example, in race-related questions, groups with polarized attitudes towards one racial group may also have polarized but opposite attitudes toward the other racial group. This pattern can be detected by the distributions of colors on contrary heatmaps with colors that complement each other.

6. To validate the SOM results, fit a k-means algorithm to the data and plot respondents' political party affiliations as well as their cluster assignments from the k-means fit. Discuss the results. E.g., Do you see evidence of partisan differences across the groups? Do you not? How do you know?

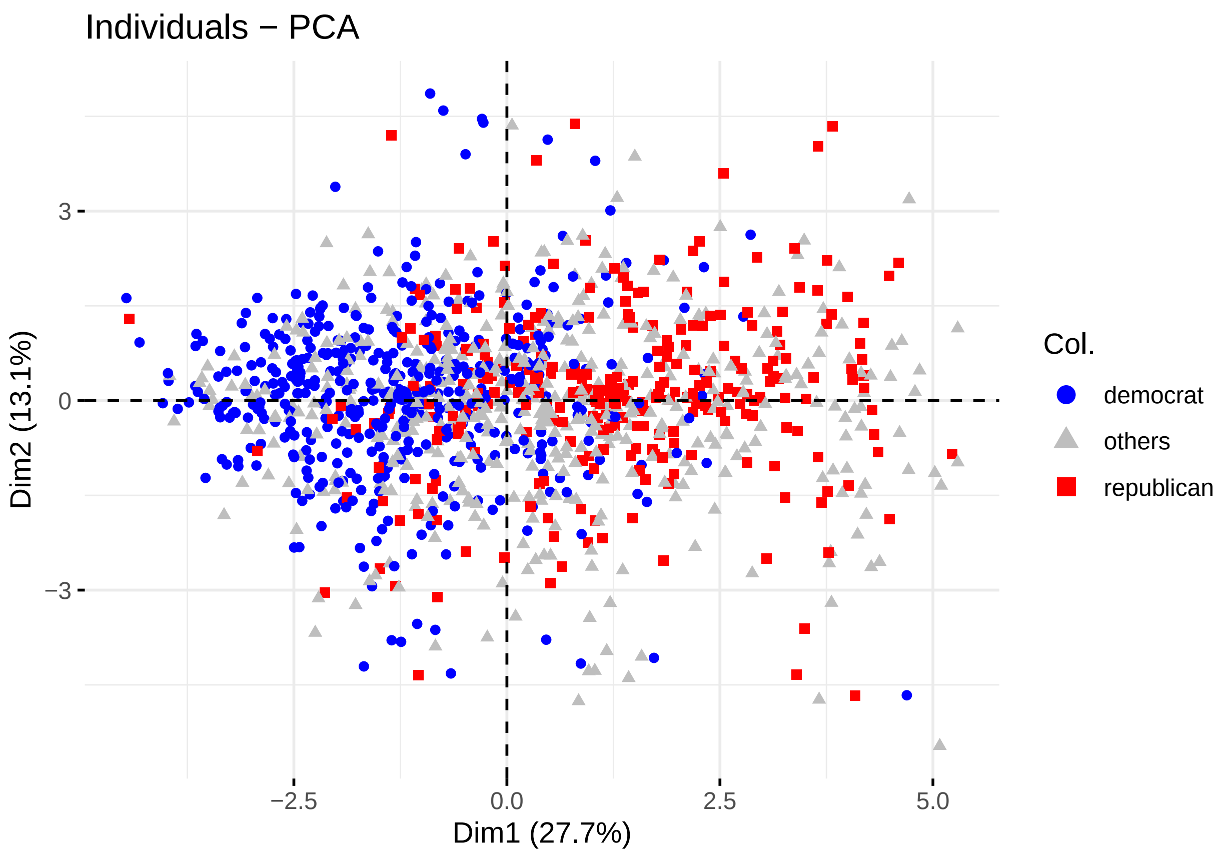
```r
## k-means
set.seed(2021)
point_colors <- c(amerika_palettes$Republican[2],
                  amerika_palettes$Democrat[2])

neuron_colors <- c(amerika_palettes$Republican[3],
                   amerika_palettes$Democrat[3])


kmeans_clusters <- som_fit$codes[[1]] %>%
  kmeans(., centers = 3)

class_assign_km <- map_dbl(kmeans_clusters$cluster, ~{
  if(. == 1) 3
  else if(. == 2) 2
  else 1
}
)

plot(som_fit,
     type = "mapping",
     pch = 21,
     bg = point_colors[as.factor(anes_short$pid3)],
     shape = "straight",
     bgcol = neuron_colors[as.integer(class_assign_km)],
     main = "3 clusters via k-means"); add.cluster.boundaries(x = som_fit, clustering = class_assign_km
                                                              lwd = 5, lty = 5)
```



```r
set.seed(1234)
km <- kmeans(anes_scale,3)

fviz_cluster(km, data = anes_scale,
              palette = c("grey", "blue", "red"),
              geom = "point",
              ellipse.type = "convex",
              ggtheme = theme_bw()
              )
```



```r
fviz_pca_ind(prcomp(anes_scale),
                      axes = c(1,2),
                      geom = c('point'),
                      col.ind = factor(anes_short$pid3), palette = c("blue", "grey", "red"))
```



Based on the k-means clustering, there are partisan differences across the groups. We can see clear groupings from the above plots while there are still some overlapping spaces between groups (maybe due to the data processing procedure that we use mean values to impute the missing value). In the two dimensional cluster plot and PCA plot, in general, democrats are to the left and republicans are to the right.

7. Taken with the SOM results, do the k-means results show similar or different patterns? Or is it unclear? Discuss both models in relation to each other for a full, well-rounded validation. *Note*: you are encouraged to think of and implement other ways to validate your results. You are welcome to go back to class notes, Google, etc.

The k-means results show similar patterns, but it's not very clean because we have some overlaps.

In the plot, points are the true class labels which we can see how they are grouping in the projection space. The background colors–the light blue and light red are results from k-means, indicating the nodes. They are the correlations that exist across each one of the feature and where they are projecting on the space.

The k-means clustering are overlaid to the weight vectors of SOM algorithm. The output space of SOM (i.e. `som_fit$codes`) showing how each features mapping onto each one of the neuron.

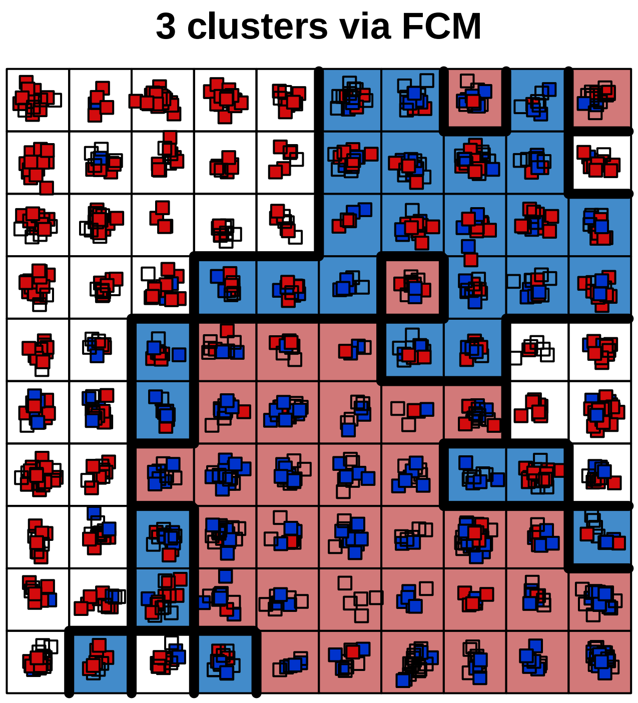
The dark lines are the hard partitioning, grouping among the best matching unit. We can see decent grouping toward the edges. However, near the middle of space, we can see a lot of overlapping. This is a good thing because the space is well-defined, so changing algorithm doesn't affect the grouping lines a lot.

We can uncover evidence of partisan differences across the groups. Since the data we are using don't include anything of party affiliation, but we can actually conclude the partisan feature, which means respondents from the same party seem to think in similar way, and think differently compared to other parties.

We can try to use soft clustering instead. In a soft clustering, the strict assignment and no overlapping properties are relaxed, and an observation can belong to more than one cluster (and even all of the clusters), with varying degrees.

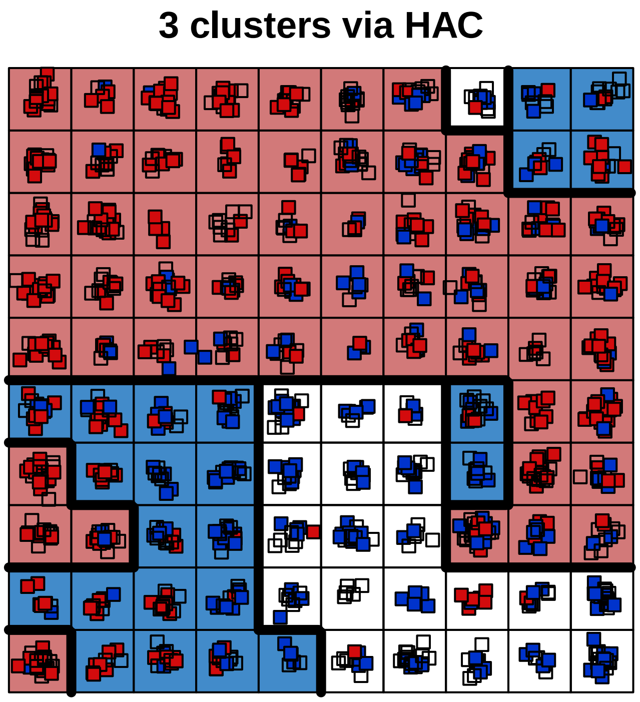
In the following part, we use other clustering algorithm including FCM and HAC to try to pull apart natural separation that exists in the projection space such as soft partitioning clustering algorithm.

```r
# FCM
set.seed(2021)
fcm_clusters <- som_fit$codes[[1]] %>%
  ppclust::fcm(., centers = 3)

plot(som_fit,
     type = "mapping",
     pch = 22,
     bg = point_colors[as.factor(anes$pid3)],
     shape = "straight",
     bgcol = neuron_colors[as.integer(fcm_clusters$cluster)],
     main = "3 clusters via FCM"); add.cluster.boundaries(x = som_fit, clustering = fcm_clusters$cluste
                                                          lwd = 5, lty = 5)
```



FCM clustering algorithm groups the observations into three clusters, where white (other party) appear on both left and right side, unlike with k-means where the white clusters clusters are most at the lower part of the grid with only two at the upper part. There are white cell inside red cell and between blue and white cell, as well as blue cell inside red cell, which means the clusters also have some overlapping. Besides, the number of cells of three clusters are almost evenly distributed, similar to the k-means clusters.

```r
## HAC
set.seed(2021)
hac_clusters <- som_fit$codes[[1]] %>%
  dist() %>%
  hclust() %>%
  cutree(., k = 3)

plot(som_fit, type = "mapping", pch = 22,
     bg = point_colors[as.factor(anes$pid3)],
     shape = "straight",
     bgcol = neuron_colors[as.integer(hac_clusters)],
     main = "3 clusters via HAC"); add.cluster.boundaries(x = som_fit,
                                                           clustering = hac_clusters,
                                                           lwd = 5,
                                                           lty = 5)
```



HAC clusters show a different result–the clusters of republican is siginificantly larger than the rest, which means a lot of data points belong to the republican clusters, however, as we can see, there are a lot of blue points in the red clusters. Similar to k-means, in the white cell, there are many data points are in fact blue (democrat).

By comparing the clustering result of k-means HAC and FCM clustering, we find that the grouping of clusters change a lot, which means the data is very sensitive to algorithm. The reason could be that there are many missing values and we should not simply impute them with the mean value, or simply because the difference of algorithms.