analysis_vi

Hunter York

11/22/2020

See how Adjusted Mutual Information Looks (Per Cheng and Park)

The following section compares the above classification system to Micro, Meso, and Macro occupation schedules using the 1950 occupation basis, per Siwei's crosswalk. This would ideally be updated to a more recent basis.

```
# cheng xwalk
cheng <- data.table(readstata13::read.dta13("../ref/occ1950_mc_xwalk_70.dta"))
cheng[, occ1950 := gsub("[^A-Za-z0-9]", "", tolower(occ1950))]

# occ50 recode
occ50_recode <- fread("../ref/occ1950_recode.csv")
occ50_recode <- occ50_recode[,1:2]
names(occ50_recode) <- unlist(occ50_recode[1, ])
occ50_recode <- occ50_recode[-1,]
occ50_recode <- occ50_recode[!is.na(as.numeric(occ1950_num))]
occ50_recode[, occ1950 := gsub("[^A-Za-z0-9]", "", tolower(occ1950))]

# merge
cheng <- merge(cheng, occ50_recode[,.(occ1950, occ1950_num)], all.x = T)

#fix the stragglers
fix <- cheng[is.na(occ1950_num)]
candidates <- occ50_recode[!occ1950 %in% cheng$occ1950]

for(c.fix in 1:nrow(fix)){
  goal <- substr(fix[c.fix, occ1950],1,7)
  new <- candidates[candidates$occ1950 %like% goal, occ1950_num]
  if(length(new) == 1){fix[c.fix, new_occ1950_num := new]}
}


fix[is.na(new_occ1950_num), new_occ1950_num := c(43, 34, 44, 603, 16, 46, 605,94, 19,48,69,84,26,23,27,

cheng <- merge(cheng, fix[,.(occ1950, new_occ1950_num)], by = "occ1950", all.x = T)
cheng[is.na(occ1950_num), occ1950_num := new_occ1950_num]
cheng[, occ1950_num := as.numeric(occ1950_num)]
cheng[, new_occ1950_num := NULL]
setnames(cheng, "occ1950_num", "OCC1950")
cheng[, occ1950 := NULL]

factorr <- function(x){ifelse(is.character(x), return(factor(x)), return(x))}
cheng[,names(cheng) := lapply(.SD, factorr), .SDcols = names(cheng)]
```

```
names(acs)[!names(acs) %like% "skl|knl|abl"] -> temp_vars
acs[, (temp_vars) := lapply(.SD, factorr), .SDcols = temp_vars]

#acs <- merge(acs, cheng, by.x = "OCC1950", by.y = "occ1950_num", all.x = T)
acs[,c("race", "hispan", "schlcoll","empstat","ahrsworkt",
       "wkswork1","uhrsworkly","classwly","workly",
       "HOURWAGE"):= NULL]

acs <- acs[cheng, on = "OCC1950"]

setnames(acs, c("mesoocc", "macroocc", "microocc"), paste0(c("mesoocc", "macroocc", "microocc"), "_curre

setnames(cheng, "OCC1950", "OCC50LY")

acs <- acs[cheng, on = "OCC50LY"]

setnames(acs, c("mesoocc", "macroocc", "microocc"), paste0(c("mesoocc", "macroocc", "microocc"), "_ly")]

# loop over scheme and calculate concordance
for(c.scheme in  c( "mesoocc", "macroocc", "microocc")){
  acs[,paste0(c.scheme, "_conc") := get(paste0(c.scheme, "_ly")) == get(paste0(c.scheme, "_current")) ]
}
```

```
# merge it all
acs <- merge(acs, skills_final,
             all.x = T,
             by.x = "OCC2010",
             by.y = "CPS Code")

#

# merge on both new and old jobs
setnames(acs, vars, paste0(vars, "_current"))
setnames(acs, "CPS Occupational Title","CPS Occupational Title_current")
# merge it all
acs[, OCC10LY := as.character(OCC10LY)]
acs <- merge(acs, skills_final,
             all.x = T,
             by.x = "OCC10LY",
             by.y = "CPS Code")

setnames(acs, vars, paste0(vars, "_ly"))
setnames(acs, "CPS Occupational Title","CPS Occupational Title_ly")

# subset to places where people have moved jobs
acs_moved <- acs[OCC2010 != OCC10LY]

# calculate flows
acs_flows <- acs[!is.na(`CPS Occupational Title_current`) &
                   !is.na(`CPS Occupational Title_ly`),.(mvmt = .N), by = .(OCC2010, OCC10LY, `CPS Occup

# graph top movement
temp <- acs_moved[, .N, by = OCC10LY] %>% .[order(N, decreasing = T)] %>% .[1:7, OCC10LY]
```

```r
acs_flows[,rank := frankv(mvmt, order = -1L, ties.method = "first"), by = OCC10LY]
acs_flows[, OCC_title_ly := paste0(str_sub(`CPS Occupational Title_ly`, 1, 15),
                                   "...")]
acs_flows[, OCC_title_current:= paste0(str_sub(`CPS Occupational Title_current`, 1, 15),
                                       "...")]

acs_flows[, OCC_title_current := factor(OCC_title_current)]

#
```

Create occupation categories based on skill first, see how well they track with flows

Overview

In the following section, I will attempt to create some sort of aggregation of occupational classifications based on skills alone. I aim to create 82 classes to compare with the micro-class scheme proffered by Grusky et al., but many decisions are fickle and subject to my own bias. As a first pass, I'm creating 80 classes using 5 quintiles of pc1, 4 quartiles of pc2, and 2 quantiles of pc3 and pc4. Each combination of these latent variables will correspond to a skills-based occupation category.

Compute skills distance between all job movements

For this exercise, I'm defining distance in skills as the geometric mean of distance between each variable. Thus:

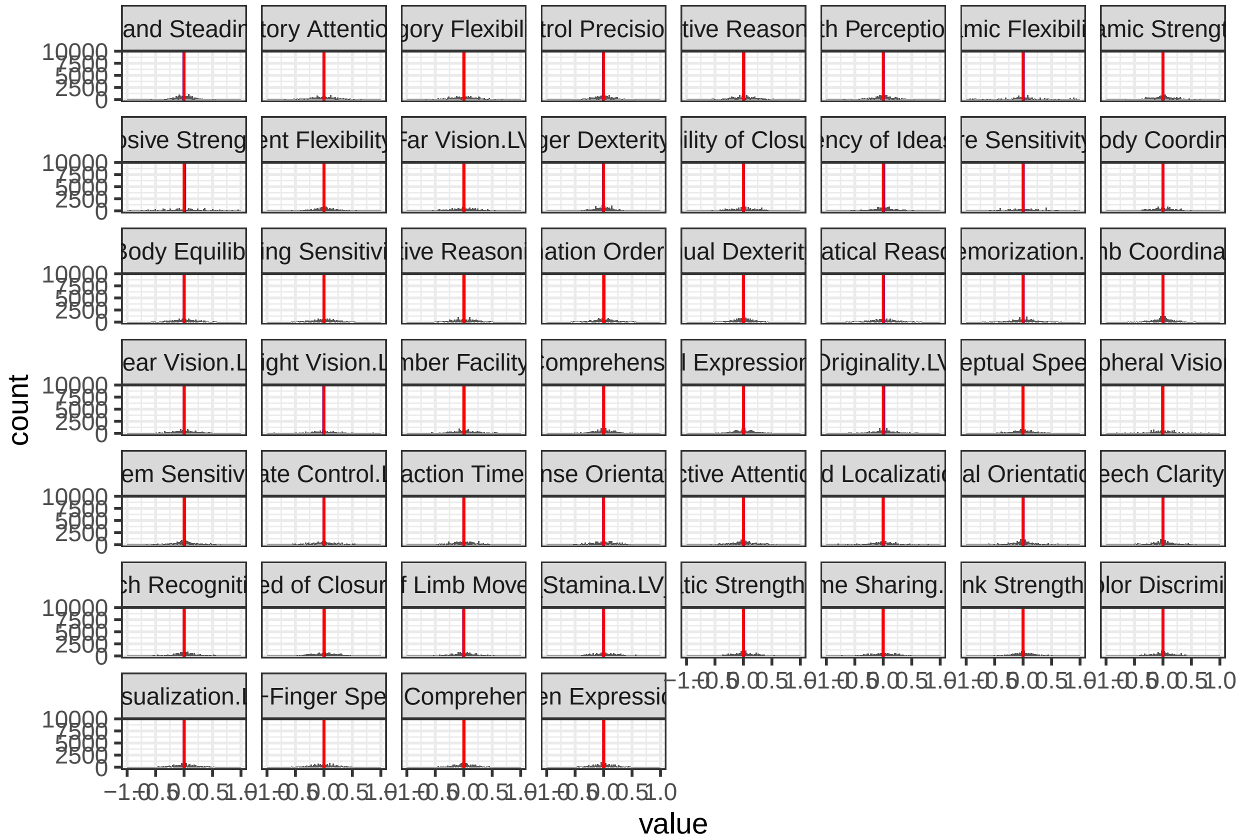
$$D_{i,j} = \prod_{n}^{4} |PC_n^{i,j} - PC_n^{i,j}|$$

Where: $D_{i,j}$ is the skill distance from job i to job j.

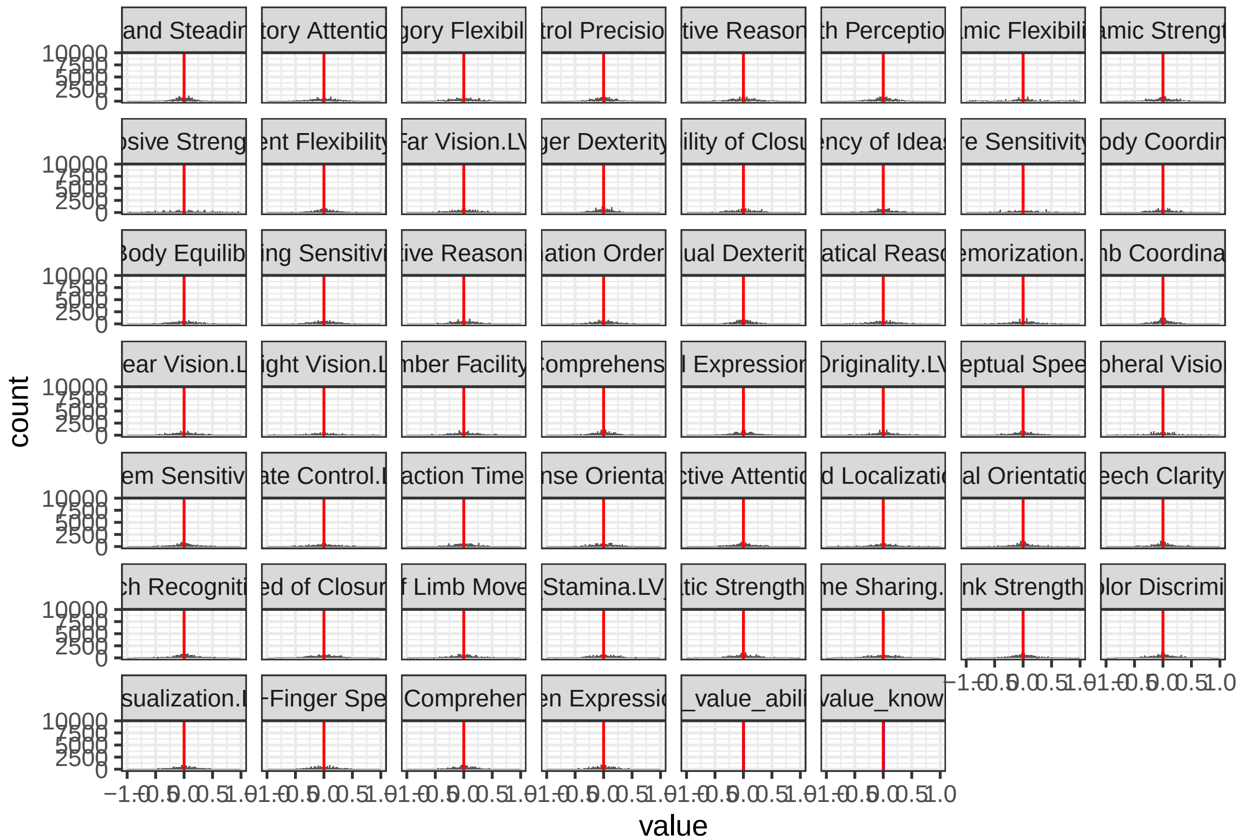
The following graph shows the distribution of skill distance for all occupational movements.

```r
for(c.skill in vars){
  #acs[, paste0(c.skill, "_distance") := abs(get(paste0(c.skill, '_current')) - get(paste0(c.skill, '_ly
  acs[, paste0(c.skill, "_diff") := (get(paste0(c.skill, '_current')) - get(paste0(c.skill, '_ly')))]

}

# subset to flows
skills_flows <- acs[OCC10LY != OCC2010]

# # graph distribution
# ggplot(skills_flows[!(OCC2010 == 4760 & OCC10LY == 4850),.SD, .SDcols = paste0(vars[!vars %like% "ski
#   geom_histogram(aes(x = value)) +
#   facet_wrap(~variable) +
#   xlim(0,1) +
#   geom_abline(intercept = 2831.867, slope = -2831.867)


ggplot(skills_flows[ind1990 == ind90ly,.SD,
                    .SDcols = paste0(vars[!vars %like% "skills|pc|tech|IM|average|skl|knl"], "_diff")] %
  geom_histogram(aes(x = value), bins = 100) +
```

```
  geom_vline(aes(xintercept = mean, group = variable),

             data = skills_flows[ind1990 == ind90ly,.SD, .SDcols = paste0(vars[!vars %like%"skills|pc|te
               melt() %>% .[,.(mean = mean(value, na.rm = T)), by = variable],

             color = "blue" )+
  facet_wrap(~variable) +

  xlim(-1,1)    +
  geom_vline(xintercept = 0, color = "red")
```




```
ggplot(skills_flows[ind1990 == ind90ly,.SD,
                    .SDcols = paste0(vars[!vars %like% "skills|pc|tech|IM|average|skl|knl"], "_diff")] %
  geom_histogram(aes(x = value), bins = 100) +
  geom_vline(aes(xintercept = mean, group = variable),

             data = skills_flows[ind1990 == ind90ly,.SD, .SDcols = paste0(vars[!vars %like% "skills|pc|

             color = "blue" )+
  facet_wrap(~variable) +

  xlim(-1,1)  +
  geom_vline(xintercept = 0, color = "red")
```




```
acs[,paste0(vars, "_diff") := NULL]
```

Right skewed = good! More people are shifting between jobs with similar skillsets than dissimilar skill sets.

Calculate % of job moves that remain within class for each class

This is obviously biased by the number of classes. I think this can be standardzied mathematically, but I haven't tried to figure that out.

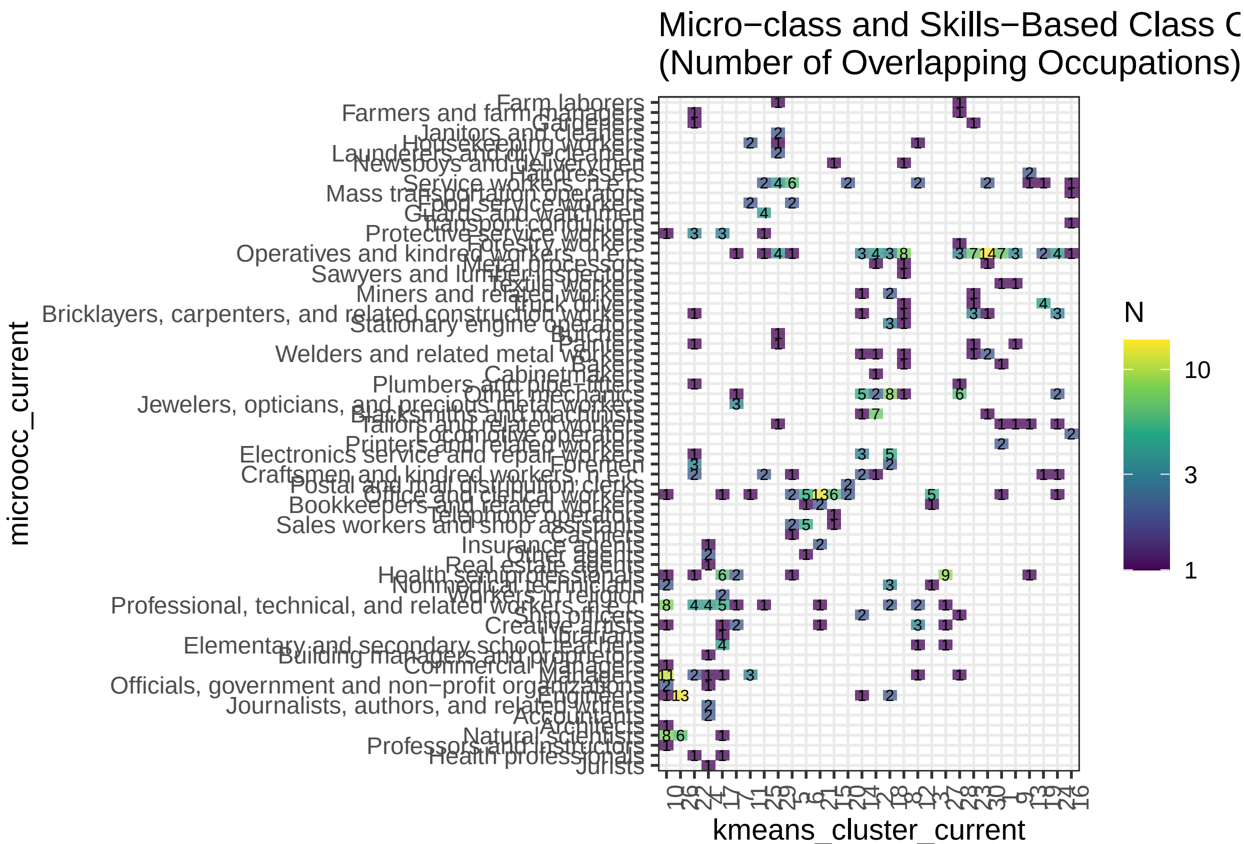
See the extent to which interoccupation mobility is defined by upwards mobility in terms of skill

Do a grid search to find optimal number of skill bins

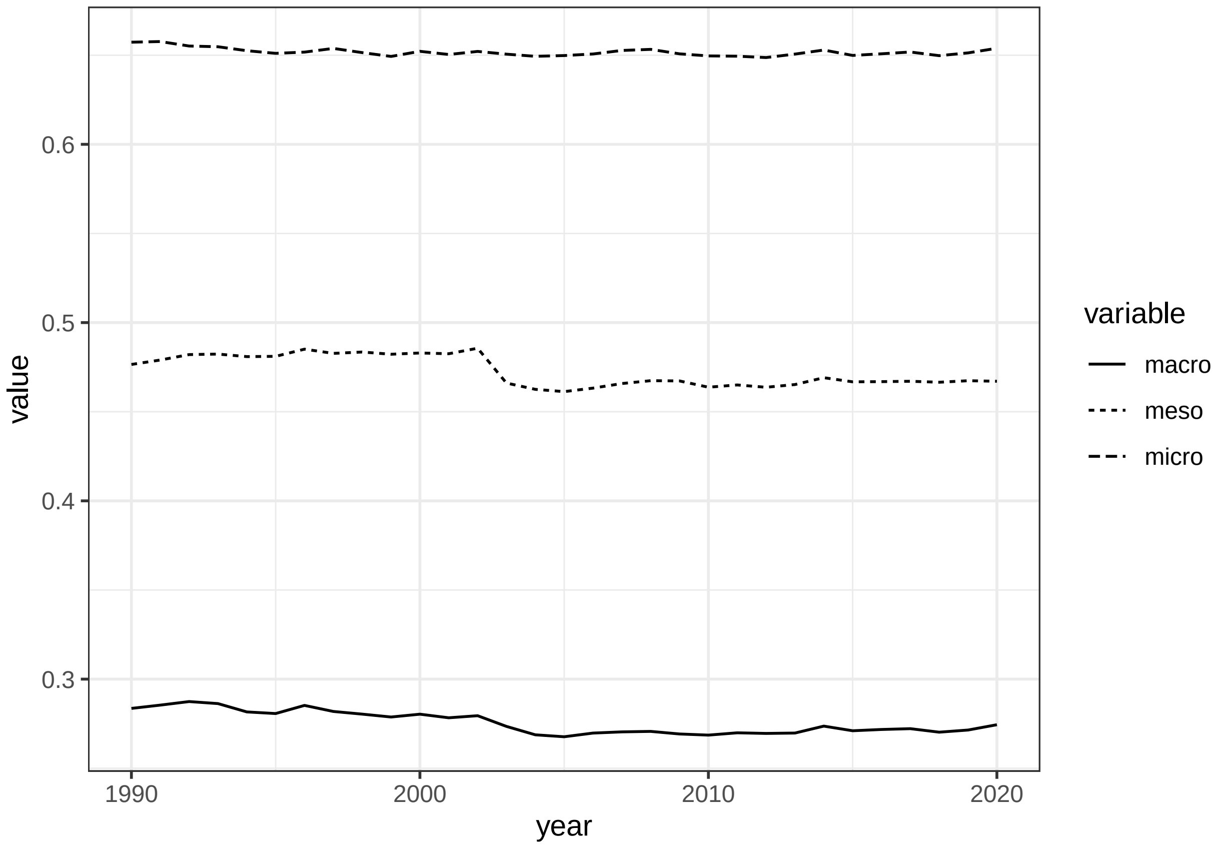
try network

```
# temp <- estimateNetwork(data = acs[,.SD, .SDcols = names(acs)[names(acs) %like% "LV_current" & !names
#                          default = "EBICglasso")
# plot(temp, layout = "spring", labels = colnames(temp))
# plot(temp, layout = "spring")

rm(list = ls()[!ls() %in% c("acs", "vars", "weighted.var")])
##################
acs <- copy(acs[complete.cases(acs[,.SD, .SDcols = names(acs)[names(acs) %like% "LV" & !names(acs) %lik

## check out missing cases above!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
# df <- data.table(rbind(acs[, .SD, .SDcols = names(acs)[names(acs) %like% "LV_current" & !names(acs) %
```

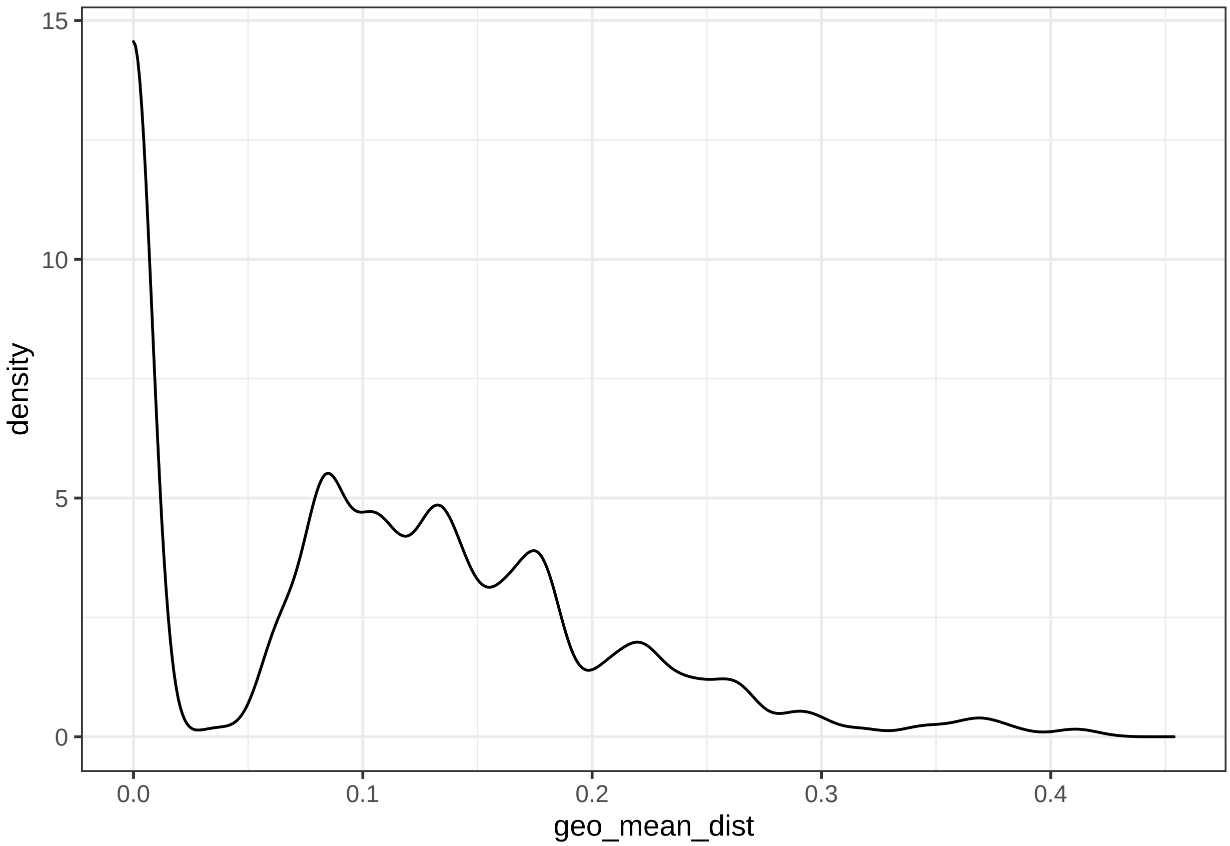
```r
df <- acs[!duplicated(acs$OCC2010), .SD, .SDcols = names(acs)[names(acs) %like% "skl|knl|abl|OCC2010" &
df_occ <- df[,OCC2010]
df[, OCC2010 := NULL]

stats::kmeans(df, centers = 30) -> temp

df_temp <- data.table(OCC2010 = df_occ, kmeans_cluster = temp$cluster)

acs <- merge(acs, df_temp, by = "OCC2010")
setnames(acs, "kmeans_cluster", "kmeans_cluster_current")

df_temp <- data.table(OCC10LY = df_occ, kmeans_cluster = temp$cluster)
acs <- merge(acs, df_temp, by = "OCC10LY")
setnames(acs, "kmeans_cluster", "kmeans_cluster_ly")

# optimize k
optr <- function(k){
  stats::kmeans(df, centers = k) -> temp2
  return(temp2$tot.withinss)
}

#lapply(seq(5,105,10), optr) -> out


overlap_micro_count <- acs[,.(N = length(unique(OCC2010))), by = .(microocc_current, kmeans_cluster_cur
overlap_micro_count[,kmeans_cluster_current := factor(kmeans_cluster_current,
                                                      levels = acs[,.(mean = mean(average_value_skills_
ggplot(overlap_micro_count) +
  geom_tile(aes(y = microocc_current, x = kmeans_cluster_current, fill = N), alpha = .75) +
  geom_text(aes(y = microocc_current, x = kmeans_cluster_current, label = N), size = 2) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Micro-class and Skills-Based Class Concordance\n(Number of Overlapping Occupations)")+
  scale_fill_viridis_c( trans = "log10")
```



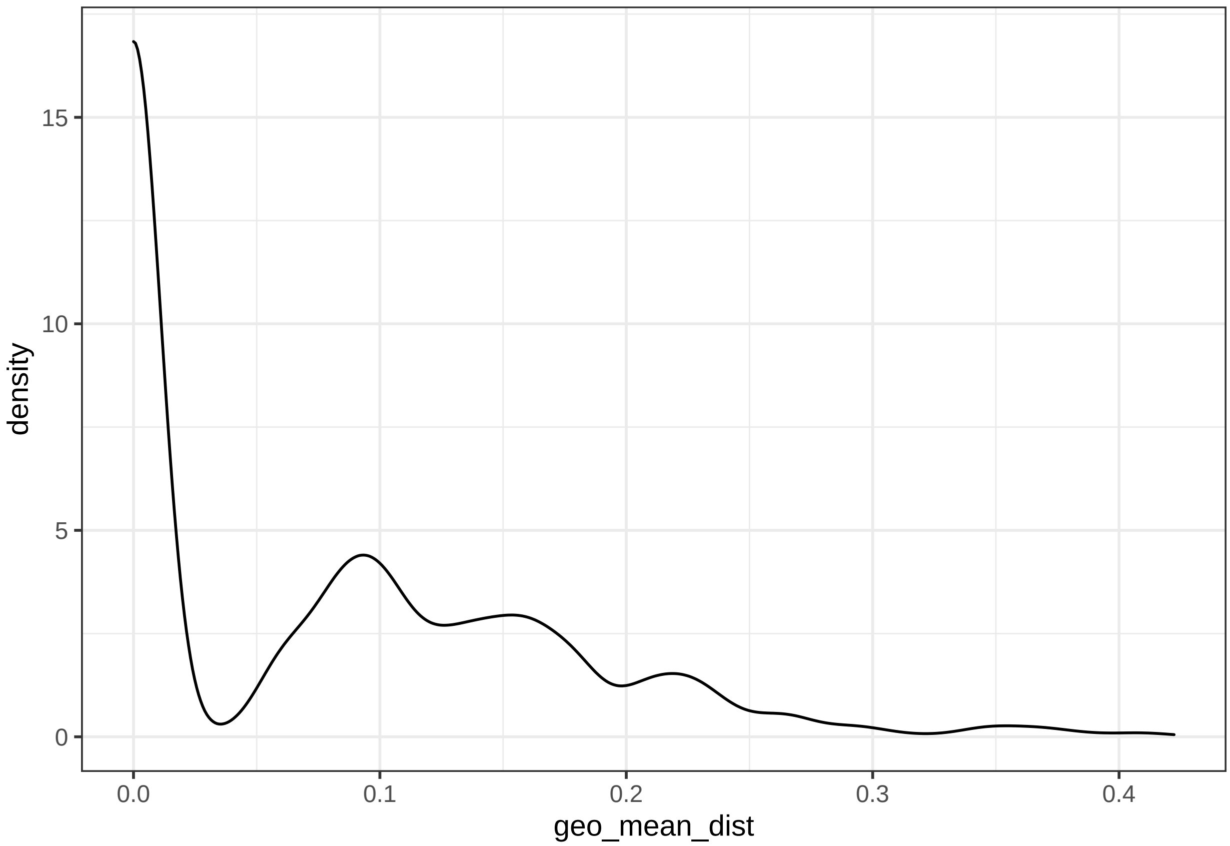

```r
# now compute AMI
ami_scores <- acs[complete.cases(acs[,.(kmeans_cluster_current, macroocc_current)]),
                  .(macro = aricode::AMI(macroocc_current, kmeans_cluster_current),
                    meso = aricode::AMI(mesoocc_current, kmeans_cluster_current),
                    micro = aricode::AMI(microocc_current, kmeans_cluster_current)), by = .(year)]

ami_scores %>%
  melt(., id.var = "year") %>%
  ggplot(.) +
  geom_line(aes(x = year, y = value, linetype = variable))
```



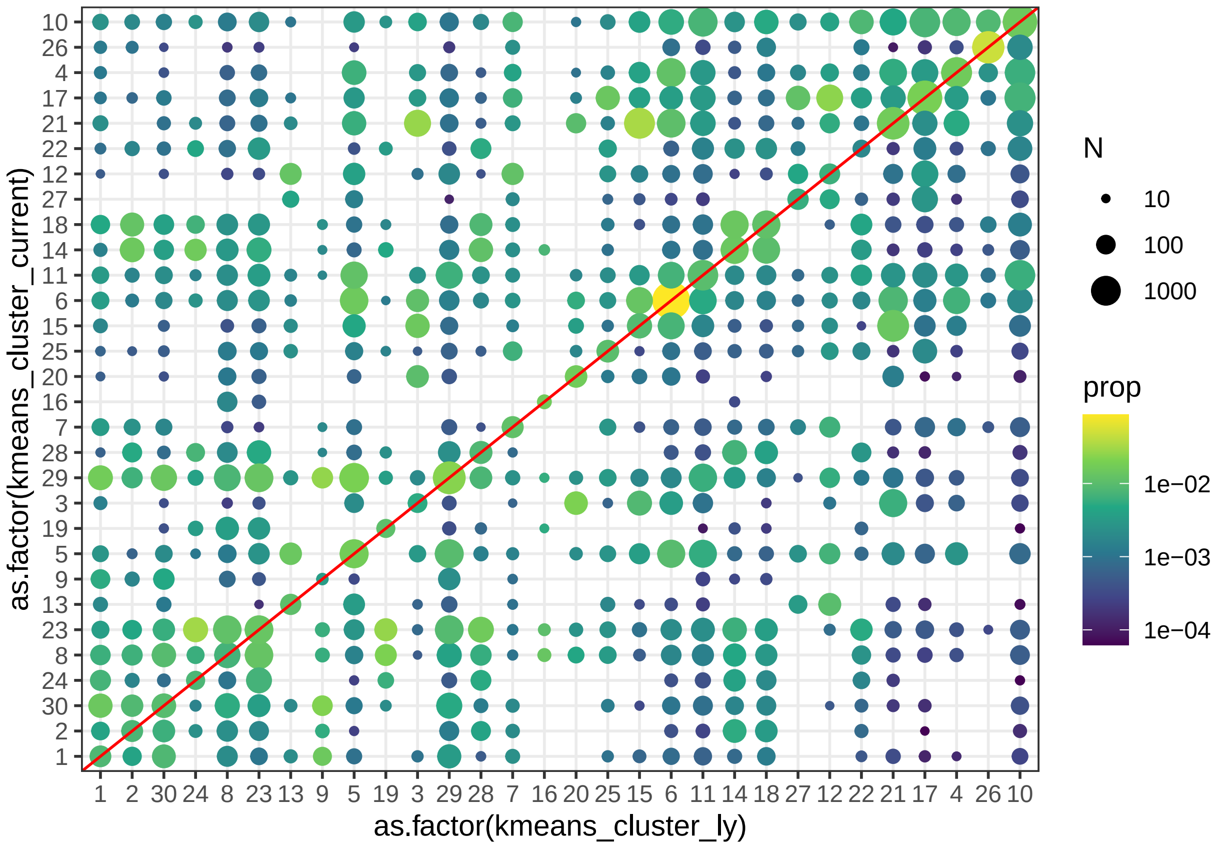

```
# judge concordance
acs[, kmeans_cluster_conc := ifelse(kmeans_cluster_current == kmeans_cluster_ly, 1,0)]
num_correct <- acs[OCC1950 != OCC50LY, colSums(.SD, na.rm = T), .SDcols = names(acs)[names(acs) %like%
total <- acs[OCC1950 != OCC50LY , nrow(.SD)]
num_correct <- num_correct/total
num_cats <- acs[OCC1950 != OCC50LY , lapply(.SD, FUN = function(x){length(unique(x))}), .SDcols = names

# get matrix of means and compute distances between each cluster
# geometric means
centers <- temp$centers
centers_long <- melt(centers)
centers_long <- merge(centers_long, centers_long, by = "Var2", allow.Cartesian = T)
centers_dist <- data.table(centers_long)[,.(geo_mean_dist = prod(abs(value.x - value.y)) ^ (1/length(va
                                          by = .(Var1.x, Var1.y)]
setnames(centers_dist, c("kmeans_cluster_current", "kmeans_cluster_ly", "geo_mean_dist"))

#
acs <- merge(acs, centers_dist, by = c("kmeans_cluster_current", "kmeans_cluster_ly"))

#graph dist
ggplot(acs[OCC2010 != OCC10LY]) +
  geom_density(aes(x = geo_mean_dist))
```




```r
ggplot(acs[OCC2010 != OCC10LY & ind == indly]) +
  geom_density(aes(x = geo_mean_dist))
```




```
#
acs[, source_kmeans_N := .N, by = kmeans_cluster_ly]

acs[, kmeans_cluster_current := factor(kmeans_cluster_current,
                                       levels = centers_dist[kmeans_cluster_current == 1] %>%
                                         .[order(geo_mean_dist)] %>% .[,kmeans_cluster_ly])]
acs[, kmeans_cluster_ly := factor(kmeans_cluster_ly,
                                  levels = centers_dist[kmeans_cluster_current == 1] %>%
                                    .[order(geo_mean_dist)] %>% .[,kmeans_cluster_ly])]


acs[OCC10LY != OCC2010,.(N = .N, source_kmeans_N = unique(source_kmeans_N)),
    by= .(kmeans_cluster_current, kmeans_cluster_ly)] %>%
  .[, total := sum(N), by = kmeans_cluster_ly] %>%
  .[, prop := N/source_kmeans_N] %>%
  .[N >= 10] %>%
  ggplot() +
  geom_point(aes(y = as.factor(kmeans_cluster_current), x = as.factor(kmeans_cluster_ly), color = prop,
  scale_color_viridis_c(trans = "log10") +
  scale_radius(trans = "log10")+
  geom_abline(yintercept = 0, slope = 1, color= "red")
```



loop over k

```
# gg_list <- list()
# q <- 0
# for(k in seq(10,110,20)){
#   q <- q + 1
#   print(k)
#   df <- data.table(rbind(acs[, .SD, .SDcols = names(acs)[names(acs) %like% "LV_current" & !names(acs)
#   stats::kmeans(df, centers = k) -> temp
#
#   # assign clusters
#   acs[, kmeans_cluster_current := temp$cluster[1:nrow(acs)]]
#   acs[, kmeans_cluster_ly := temp$cluster[(nrow(acs) + 1):(nrow(acs)*2)]]
#
#   #
#   # get matrix of means and compute distances between each cluster
#   # geometric means
#   centers <- temp$centers
#   centers_long <- melt(centers)
#   centers_long <- merge(centers_long, centers_long, by = "Var2", allow.Cartesian = T)
#   centers_dist <- data.table(centers_long)[,.(geo_mean_dist = prod(abs(value.x - value.y)) ^ (1/lengt
#                                           by = .(Var1.x, Var1.y)]
#   setnames(centers_dist, c("kmeans_cluster_current", "kmeans_cluster_ly", "geo_mean_dist"))
#
#   #
#   acs[, geo_mean_dist := NULL]
```

```r
#   acs <- merge(acs, centers_dist, by = c("kmeans_cluster_current", "kmeans_cluster_ly"))
#
#   #
#   acs[, source_kmeans_N := .N, by = kmeans_cluster_ly]
#   centers <- data.table(centers)
#   centers[,total_skills := rowSums(.SD), .SDcols = names(centers)]
#
#   acs[, kmeans_cluster_current := factor(kmeans_cluster_current,
#                                          levels = centers[, order(total_skills)])]
#   acs[, kmeans_cluster_ly := factor(kmeans_cluster_ly,
#                                     levels = centers[, order(total_skills)])]
#
#
#   gg <- acs[OCC10LY != OCC2010,.(N = .N, source_kmeans_N = unique(source_kmeans_N)),
#             by= .(kmeans_cluster_current, kmeans_cluster_ly)] %>%
#     .[, total := sum(N), by = kmeans_cluster_ly] %>%
#     .[, prop := N/source_kmeans_N] %>%
#     .[N >= 10] %>%
#     ggplot()+
#       geom_point(aes(y = (kmeans_cluster_current), x = (kmeans_cluster_ly), color = prop, size = N))
#   scale_color_viridis_c(trans = "log10") +
#   scale_radius(trans = "log10")+
#   geom_abline(yintercept = 0, slope = 1, color= "red") +
#   scale_x_discrete(labels = centers[, order(total_skills)],
#                    breaks = centers[, order(total_skills)], drop = F)+
#   scale_y_discrete(labels = centers[, order(total_skills)],
#                    breaks = centers[, order(total_skills)], drop = F)+
#   theme(axis.text.x = element_text(angle = 90))
#
#   print(gg)
#
# }
```

See how well each scheme predicts log earnings

```r
mod <- lm(log_incwage ~ as.factor(kmeans_cluster_current), acs[year == 1999])
mod2 <- lm(log_incwage ~ as.factor(mesoocc_current), acs[year == 1999])
stats::AIC(mod2)
```

```
## [1] 79947.1
```

```r
stats::AIC(mod)
```

```
## [1] 77968.11
```