analysis_vi

Hunter York

11/22/2020

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.

Run the code

```
# subset to only acs for where we have skill data
acs <- acs[!is.na(pc1_current)]
# first establish cutpoints for each of the latent variables
lat_var_cps <- list()
for(i in 1:4){
  if(i == 1){q <-3}else if(i == 2){q <- 3}else{q <-3}
  lat_var_cps[[i]]<- quantile(acs[,get(paste0("pc", i, "_current"))],
                              probs = seq(0,1,length.out = q + 1), na.rm = T)
}

# use the cutpoints to bin each individual into the quantiles
for(i in 1:4){
  for(q in 1:(length(lat_var_cps[[i]]) - 1)){
    print(paste0(i,q))
    if(q == 1){
      acs[get(paste0("pc", i, "_current")) >= lat_var_cps[[i]][q] &
            get(paste0("pc", i,  "_current")) <= lat_var_cps[[i]][q + 1], paste0("pc", i, "_binned") :=
    }else{
      acs[get(paste0("pc", i, "_current")) > lat_var_cps[[i]][q] &
            get(paste0("pc", i,  "_current")) <= lat_var_cps[[i]][q + 1], paste0("pc", i, "_binned") :=
    }
  }
}
```

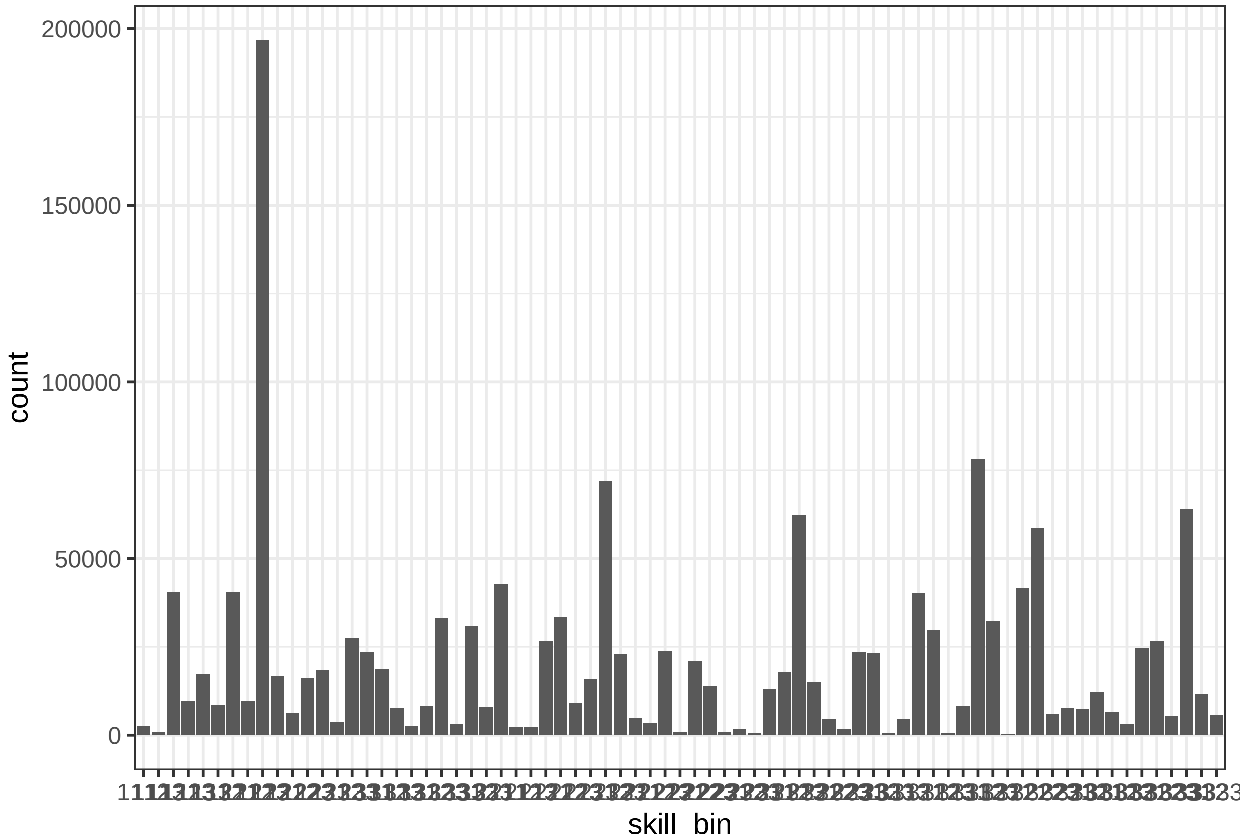
```
## [1] "11"
## [1] "12"
## [1] "13"
## [1] "21"
## [1] "22"
```

```
## [1] "23"
## [1] "31"
## [1] "32"
## [1] "33"
## [1] "41"
## [1] "42"
## [1] "43"
```

```
# now construct them into one categorical variable
acs[,skill_bin := paste0(pc1_binned, pc2_binned, pc3_binned, pc4_binned)]
```

See how the distribution looks

```
ggplot(acs) + geom_bar(aes(x = skill_bin))
```



Not bad, but also not great There's a few intersections of the bins that have no observations.

See what the groupings of jobs are for each cluster of skills

```
# creata dataset of skills groupings and children occupations
skills_occs_list <- acs[, .N, by = .(OCC2010,`CPS Occupational Title_current`, skill_bin)]
skills_occs_list[, grand_N := sum(N), by = skill_bin]

# show top 10
unique(skills_occs_list[order(grand_N, decreasing = T),.(skill_bin, grand_N)][, skill_bin])[1:5] -> temp
skills_occs_list[skill_bin %in% temp] %>% .[order(grand_N, decreasing = T)]
```

```
##      OCC2010
```

```
##  1:     20
##  2:    230
##  3:    410
##  4:    430
##  5:   4700
##  6:    160
##  7:   5000
##  8:   3710
##  9:    330
## 10:   4230
## 11:   5550
## 12:   5860
## 13:   5110
## 14:   5260
## 15:   5810
## 16:   5820
## 17:   4130
## 18:   5850
## 19:   5840
## 20:   4900
## 21:   5540
## 22:   9640
## 23:   4400
## 24:   3940
## 25:   6040
## 26:   8330
## 27:   5400
## 28:   5700
## 29:    540
## 30:    930
## 31:   5350
## 32:   5140
## 33:    940
## 34:   5200
## 35:   7330
## 36:   8220
## 37:   8965
## 38:   7950
## 39:   8030
## 40:   8200
## 41:   8740
## 42:   7740
## 43:   8140
## 44:   7350
## 45:   7840
## 46:   8930
## 47:   9650
## 48:   8720
## 49:   7240
## 50:   7730
## 51:   8130
## 52:   7940
## 53:   8410
## 54:   7850
```

```
## 55:    8940
## 56:    8010
## 57:    7710
## 58:     205
## 59:    6200
## 60:    6230
## 61:    4000
## 62:    4200
## 63:    6460
## 64:    6120
##     OCC2010
##                                                                                                              CPS
##  1:                                                                                                          Gene
##  2:
##  3:                                                                                  Property, real estate, and com
##  4:
##  5:                                                                                    First-line supervisors/manag
##  6:                                                                                         Transportation, storage
##  7:                                                              First-line supervisors/managers of office and adm
##  8:                                                                                   First-line supervisors/manage
##  9:
## 10:                                                                                                          Maid
## 11:                                                                                                             H
## 12:
## 13:                                                                                             Billing and posting c
## 14:
## 15:
## 16:
## 17: Food preparation and serving related workers, all other including dining room and cafeteria atten
## 18:                                                                            Mail clerks and mail machine opera
## 19:                                                                                            Insurance claims a
## 20:                                                                                             Models, demonstra
## 21:
## 22:
## 23:
## 24:
## 25:                                                                                                Graders and sor
## 26:                                                                                                    Shoe and le
## 27:                                                                                                      Receptior
## 28:                                                                                                 Secretaries ar
## 29:                                                                               Claims adjusters, appraisers, e
## 30:                                                                                            Tax examiners, col
## 31:                                                                                                    Corresponde
## 32:                                                                                                            Pay
## 33:
## 34:
## 35:                                                                                            Industrial and ref
## 36:                                                                                              Metalworkers and
## 37:                        Production workers, including semiconductor processors and cooling and f
## 38:                                   Cutting, punching, and press machine setters, operators, and
## 39:
## 40:                                                            Plating and coating machine setters, operators, and
## 41:                                                                                      Inspectors, testers, sort
## 42:                                                                                                Structural me
```

```
## 43:                                                                   Welding, sol
## 44:                                                                            Ma
## 45:
## 46:                                                           Paper goods machine set
## 47:
## 48:                                    Extruding, forming, pressing, and compacting machine set
## 49:
## 50:                                                                        Engine
## 51:
## 52:      Rolling machine setters, operators, and tenders and forging machine setters, operators, an
## 53:                                                    Textile knitting and weaving machine set
## 54:                                                                     Food cooking m
## 55:
## 56:                                      Lathe and turning machine tool setters, operators, an
## 57:                                                       Aircraft structure, surfaces, rig
## 58:                                                              Farmers, ranchers, and
## 59:                                         First-line supervisors/managers of construction t
## 60:
## 61:
## 62:                                              First-line supervisors/managers of houseke
## 63:
## 64:                                                                           For
##                                                                               CP
##     skill_bin     N grand_N
##  1:      1213  9707  196701
##  2:      1213 12963  196701
##  3:      1213  7199  196701
##  4:      1213 83031  196701
##  5:      1213 61305  196701
##  6:      1213  2471  196701
##  7:      1213 17934  196701
##  8:      1213  1894  196701
##  9:      1213   197  196701
## 10:      3132 18543   78024
## 11:      3132  5590   78024
## 12:      3132 12160   78024
## 13:      3132  5765   78024
## 14:      3132  3275   78024
## 15:      3132  6269   78024
## 16:      3132  4871   78024
## 17:      3132  6001   78024
## 18:      3132  1684   78024
## 19:      3132  2435   78024
## 20:      3132   459   78024
## 21:      3132  3401   78024
## 22:      3132  4899   78024
## 23:      3132  1039   78024
## 24:      3132   519   78024
## 25:      3132   977   78024
## 26:      3132   137   78024
## 27:      2132 13473   72030
## 28:      2132 46938   72030
## 29:      2132  5419   72030
## 30:      2132   686   72030
```

```
## 31:        2132  2310   72030
## 32:        2132  2400   72030
## 33:        2132   746   72030
## 34:        2132    58   72030
## 35:        3331  7444   64006
## 36:        3331  4157   64006
## 37:        3331 16813   64006
## 38:        3331  1411   64006
## 39:        3331  6490   64006
## 40:        3331   331   64006
## 41:        3331 11473   64006
## 42:        3331   373   64006
## 43:        3331  8545   64006
## 44:        3331   446   64006
## 45:        3331   961   64006
## 46:        3331   473   64006
## 47:        3331   897   64006
## 48:        3331   640   64006
## 49:        3331   640   64006
## 50:        3331   166   64006
## 51:        3331  1359   64006
## 52:        3331   199   64006
## 53:        3331   310   64006
## 54:        3331   107   64006
## 55:        3331   145   64006
## 56:        3331   526   64006
## 57:        3331   100   64006
## 58:        2313  6525   62366
## 59:        2313 10775   62366
## 60:        2313 14905   62366
## 61:        2313 25915   62366
## 62:        2313  3524   62366
## 63:        2313   489   62366
## 64:        2313   233   62366
##     skill_bin     N grand_N
```

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:

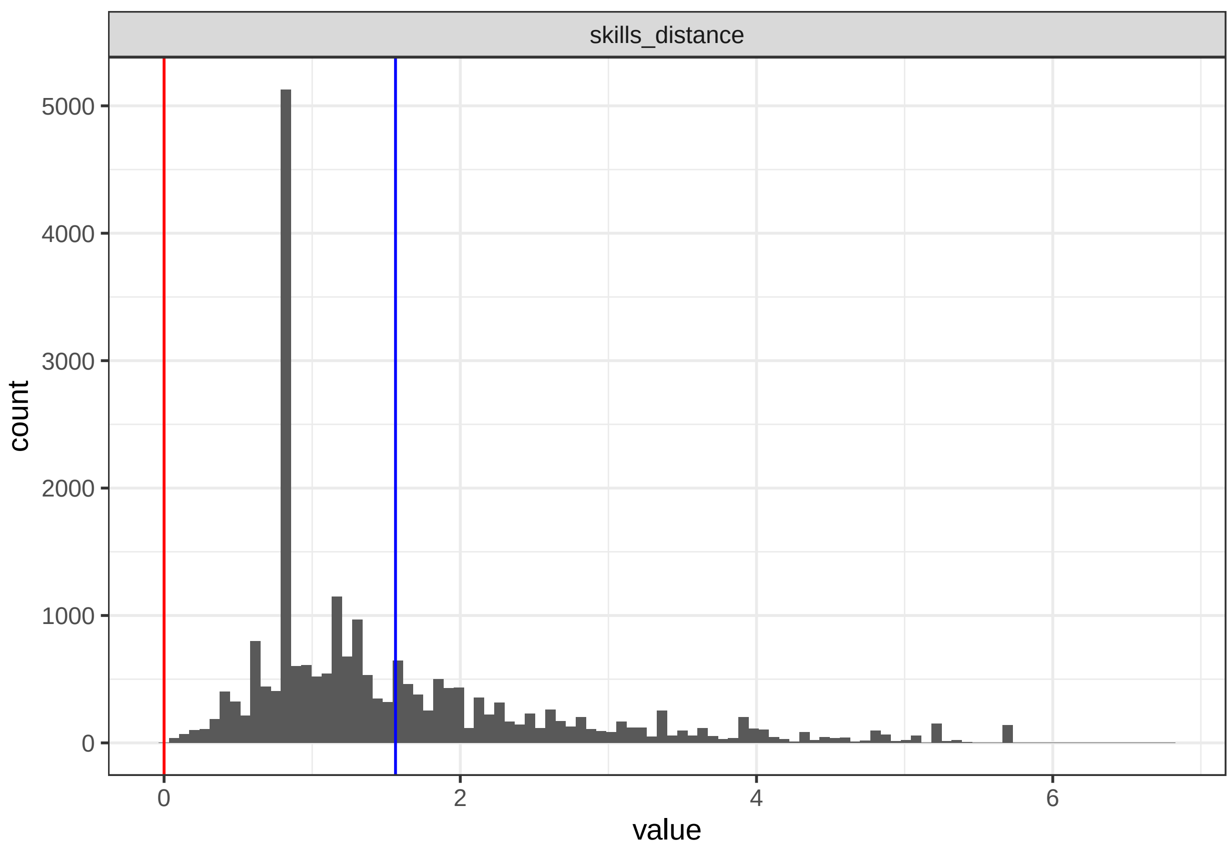
$$D_{i,j} = \prod_{n}^{4} |PC_n^{i,j} - PC_n^{i,j}|$$

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

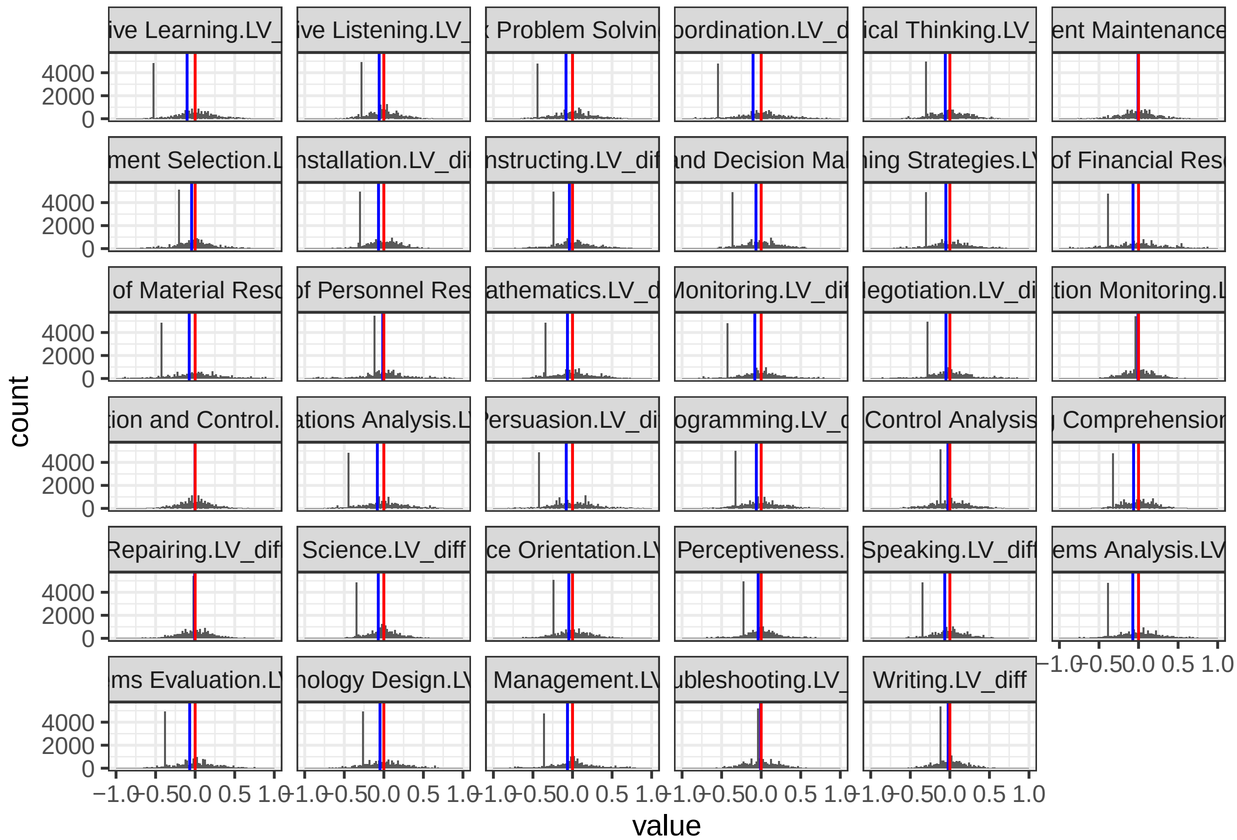
The following graph shows the distribution of skill distance for all occupational movements.

```
acs[, skills_distance := (abs((pc1_current -pc1_ly)*(pc2_current - pc2_ly)*(pc3_current - pc3_ly)*(pc4_

for(i in 1:4){
  acs[, paste0("pc", i, "_distance") := abs(get(paste0('pc', i, '_current')) - get(paste0('pc', i, '_ly
  acs[, paste0("pc", i, "_diff") := (get(paste0('pc', i, '_current')) - get(paste0('pc', i, '_ly')))]
}
```

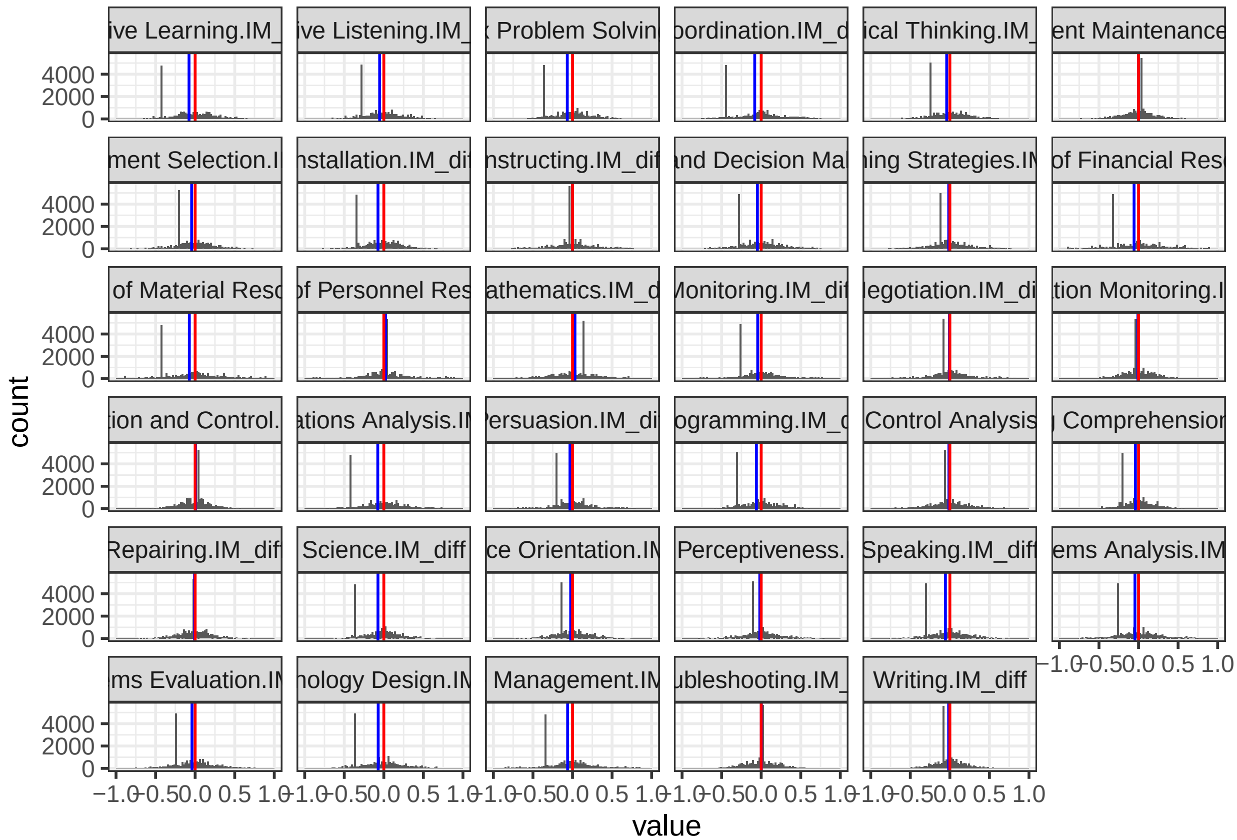
```
for(c.skill in vars){
  acs[, paste0(c.skill, "_distance") := abs(get(paste0(c.skill, '_current')) - get(paste0(c.skill, '_ly
  acs[, paste0(c.skill, "_diff") := (get(paste0(c.skill, '_current')) - get(paste0(c.skill, '_ly')))]

}

# 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 = "skills_distance"] %>% melt()) +
  geom_histogram(aes(x = value), bins = 100) +
  geom_vline(aes(xintercept = mean, group = variable),

             data = skills_flows[ind1990 == ind90ly,.SD, .SDcols = "skills_distance"] %>% melt() %>% .[

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

  geom_vline(xintercept = 0, color = "red")
```



```
ggplot(skills_flows[ind1990 == ind90ly,.SD, .SDcols = paste0(vars[!vars %like% "skills|pc|tech|IM"], "_d
  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|t

             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|LV"], "_d
  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|t

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

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



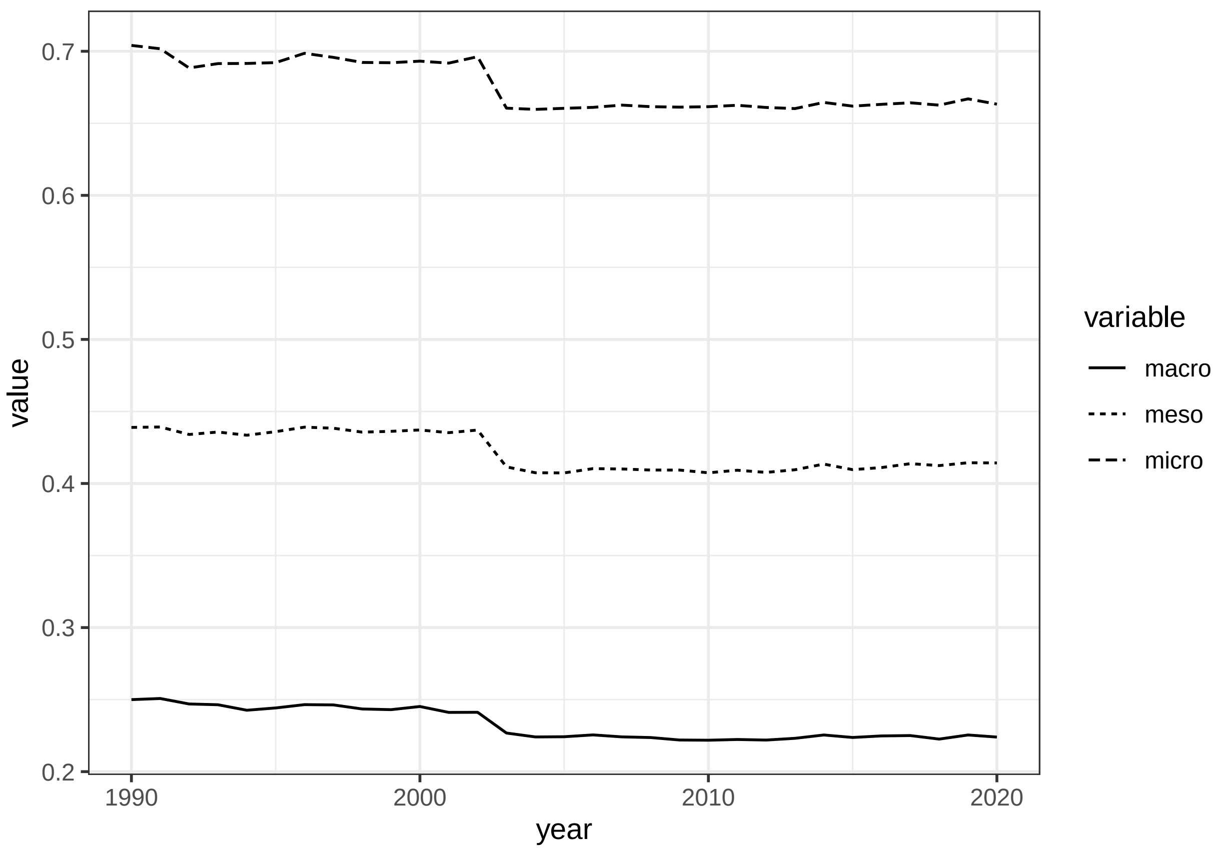
Right skewed = good! More people are shifting between jobs with similar skillsets than dissimilar skill sets.

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.

```r
# 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)){
```

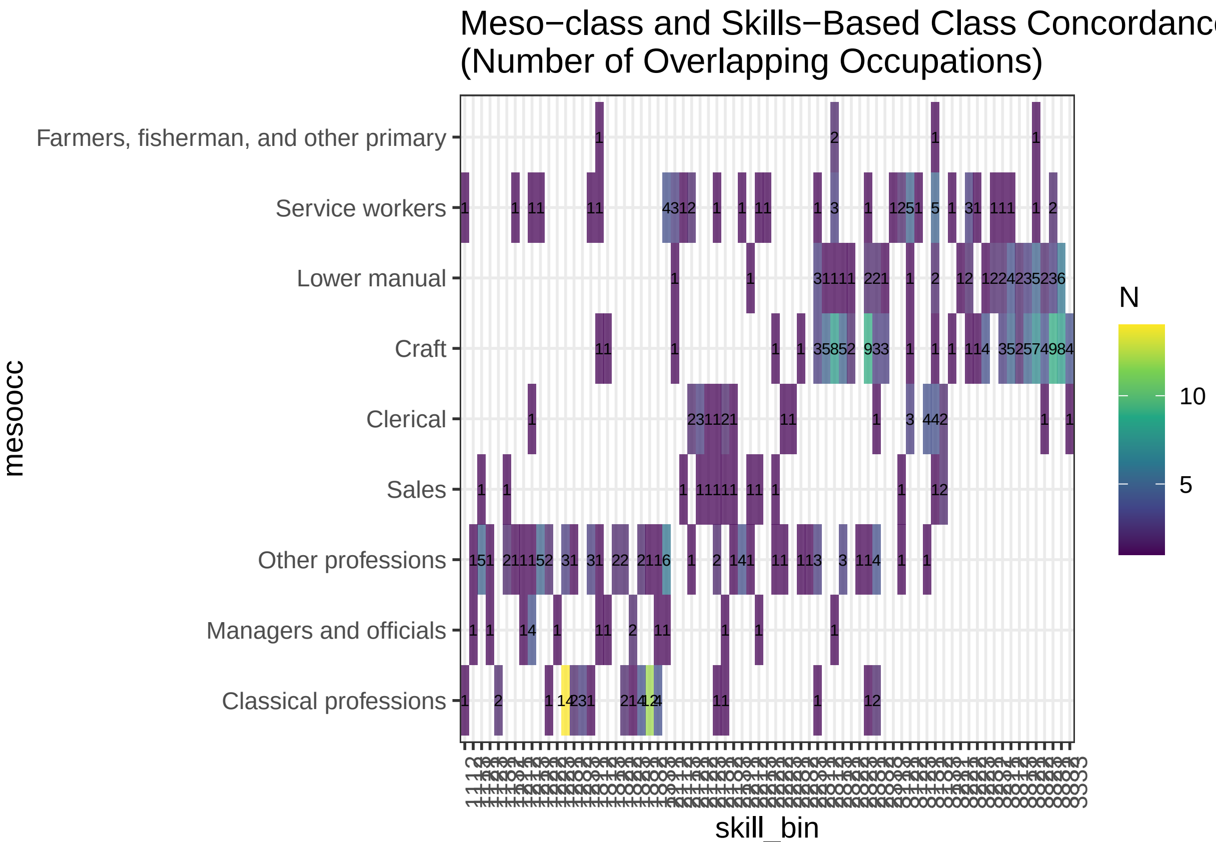
```
  goal <- substr(fix[c.fix, occ1950],1,7)
  new <- candidates[candidates$occ1950 %like% goal, occ1950_num]
  if(length(new) == 1){fix[c.fix, new_occ1950_num := new]}
}


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)]

acs <- merge(acs, cheng, by.x = "OCC1950", by.y = "occ1950_num", all.x = T)

# now compute AMI
ami_scores <- acs[complete.cases(acs[,.(skill_bin, macroocc)]),.(macro = aricode::AMI(macroocc, skill_bi
                                                                 meso = aricode::AMI(mesoocc, skill_bin
                                                                 micro = aricode::AMI(microocc, skill_bi

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



Recreate Table II from Siwei's paper

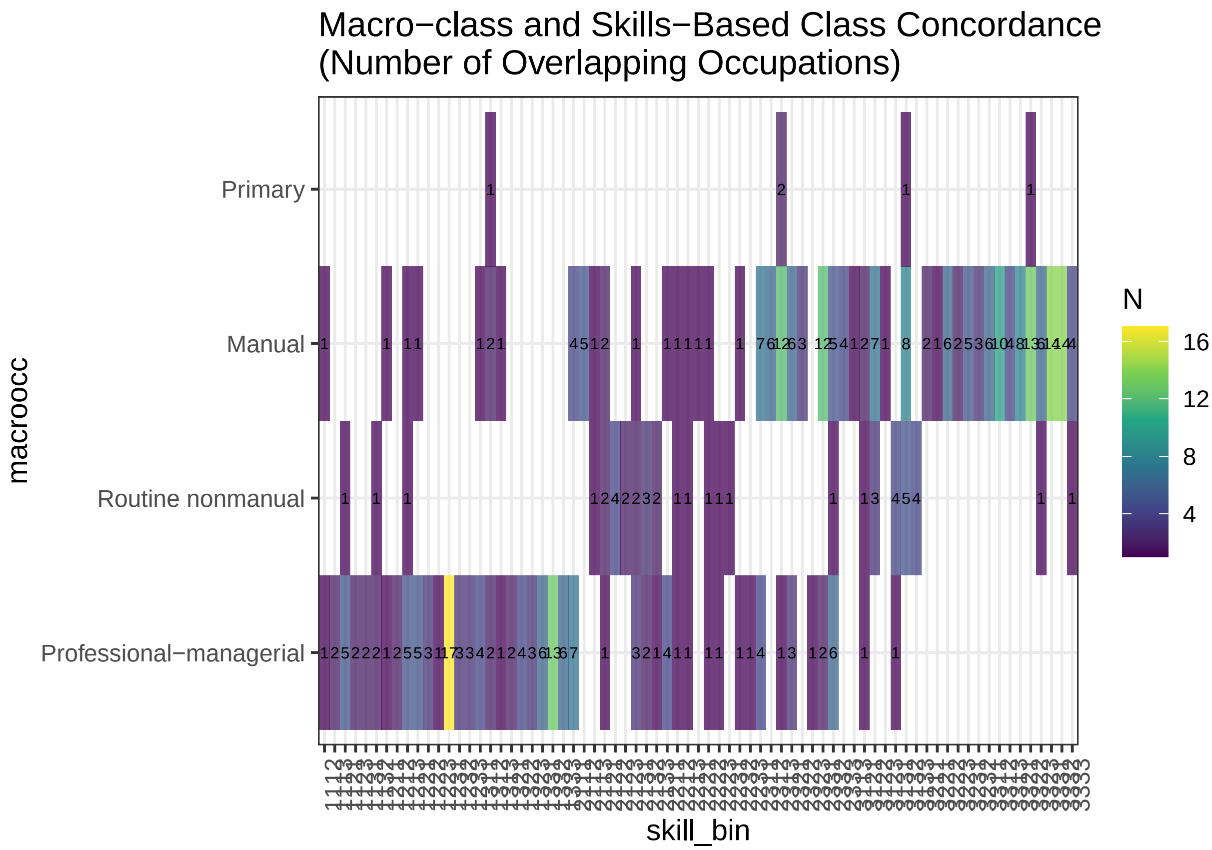
This is the number of overlap between each category for each scheme

```r
overlap <- acs[,.(skill_bin, mesoocc, macroocc, microocc, occ1950)] %>% unique()
overlap_meso_count <- overlap[,.N, by = .(mesoocc, skill_bin)]
ggplot(overlap_meso_count) +
  geom_tile(aes(y = mesoocc, x = skill_bin, fill = N), alpha = .75) +
  geom_text(aes(y = mesoocc, x = skill_bin, label = N), size = 2) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Meso-class and Skills-Based Class Concordance\n(Number of Overlapping Occupations)") +
  scale_fill_viridis_c()
```




```r
overlap_micro_count <- overlap[,.N, by = .(microocc, skill_bin)]
ggplot(overlap_micro_count) +
  geom_tile(aes(y = microocc, x = skill_bin, fill = N), alpha = .75) +
  geom_text(aes(y = microocc, x = skill_bin, 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()
```


```r
overlap_macro_count <- overlap[,.N, by = .(macroocc, skill_bin)]
ggplot(overlap_macro_count) +
  geom_tile(aes(y = macroocc, x = skill_bin, fill = N), alpha = .75) +
  geom_text(aes(y = macroocc, x = skill_bin, label = N), size = 2) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Macro-class and Skills-Based Class Concordance\n(Number of Overlapping Occupations)")+
  scale_fill_viridis_c()
```



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

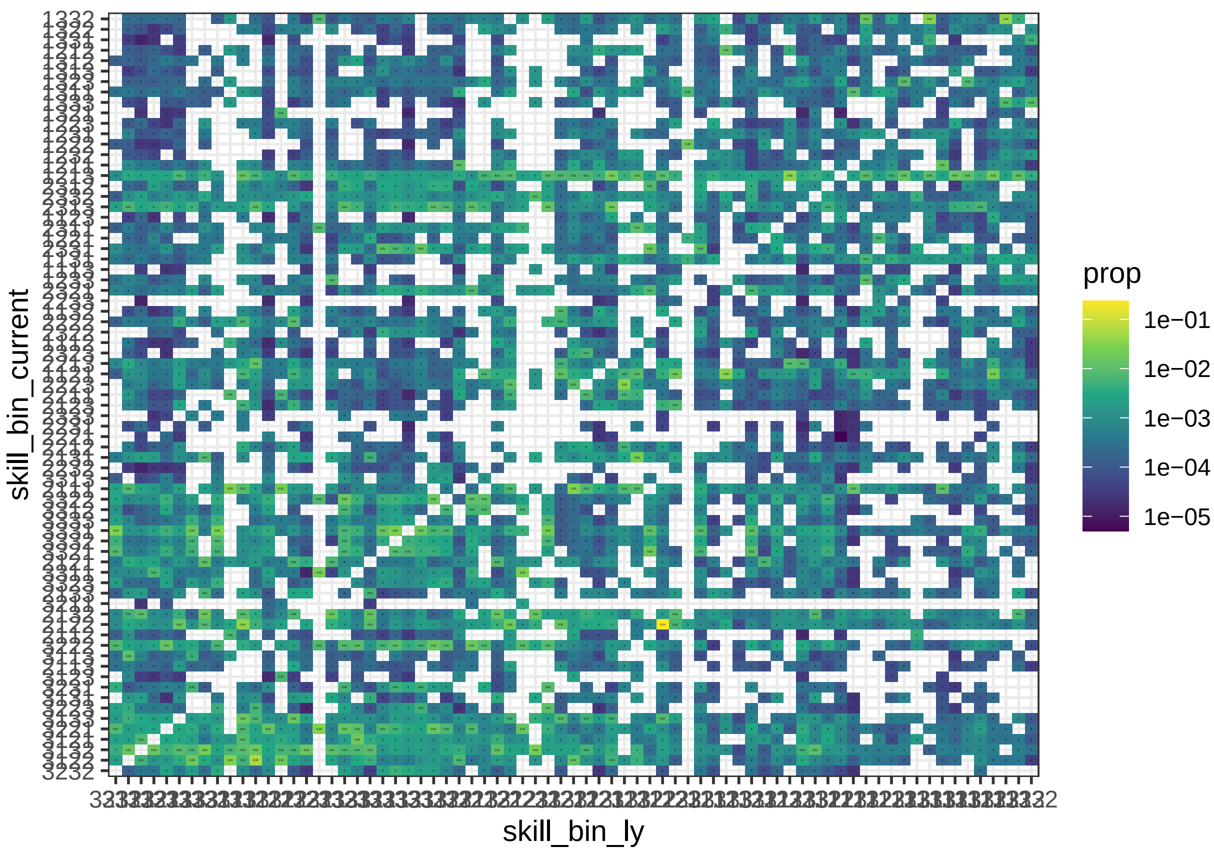
This is obviously biased by the number of classes. I think this can be standardzied mathematically, but I haven't tried to figure that out.

```
setnames(acs, c("skill_bin", "mesoocc", "macroocc", "microocc"), paste0(c("skill_bin", "mesoocc", "macro

acs <- merge(acs, cheng, by.x = "OCC50LY", by.y = "occ1950_num", all.x = T)
acs <- merge(acs, skills_occs_list[,.(OCC2010, skill_bin)], by.x = "OCC10LY",  by.y = "OCC2010", all.x =
setnames(acs, c("skill_bin", "mesoocc", "macroocc", "microocc"), paste0(c("skill_bin", "mesoocc", "macro

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

# melt and tabulate
num_correct <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(ski
total <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skill_bin_
num_correct <- num_correct/total
num_cats <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skill_b
```

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

```
grid_mvmt <- acs[,.N, by = .(skill_bin_ly, skill_bin_current)]
grid_mvmt[,total_per_source := sum(N), by = skill_bin_ly]
```

```
grid_mvmt[, prop := N/ total_per_source]

acs[,.(skill_bin_ly, average_value_skills_ly)] %>%.[,.(average_value_skills_ly = mean(average_value_ski
  .[,skill_bin_ly] -> temp
grid_mvmt[, skill_bin_current := factor(skill_bin_current, levels = temp)]
grid_mvmt[, skill_bin_ly := factor(skill_bin_ly, levels = temp)]

###
ggplot(grid_mvmt[skill_bin_ly != skill_bin_current]) +
  geom_tile(aes(x = skill_bin_ly, y = skill_bin_current, fill = prop)) +
  geom_text(aes(x = skill_bin_ly, y = skill_bin_current, label = round(prop, 2)), size = .5) +
  scale_fill_viridis_c(trans = "log10")
```



Do a grid search to find optimal number of skill bins

```
# subset to only acs for where we have skill data
# first establish cutpoints for each of the latent variables

grid_searchr <- function(qlist){
  qs <- strsplit(qlist, ",")
  q1 <- qs[[1]][1]
  q2 <- qs[[1]][2]
  q3 <- qs[[1]][3]
  q4 <- qs[[1]][4]

  lat_var_cps <- list()
```

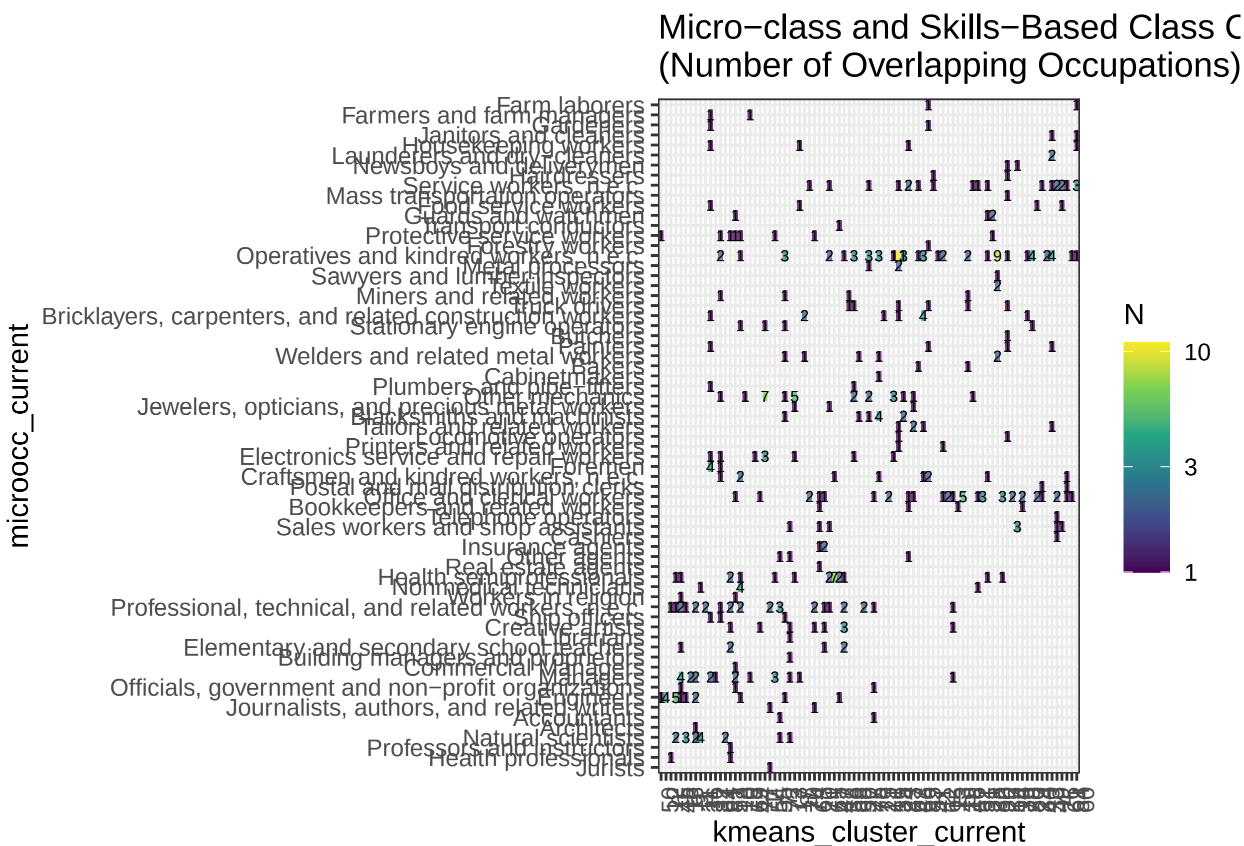
```
for(i in 1:4){
  if(i == 1){q <-q1}else if(i == 2){q <- q2}else if(i == 3){q <-q3}else if(i == 4){q <- q4}
  q <- as.numeric(q)
  lat_var_cps[[i]]<- quantile(acs[,get(paste0("pc", i, "_current"))],
                              probs = seq(0,1,length.out = q + 1), na.rm = T)
}

# use the cutpoints to bin each individual into the quantiles
for(i in 1:4){
  for(q in 1:(length(lat_var_cps[[i]]) - 1)){
    print(paste0(i,q))
    if(i == 1){
      acs[get(paste0("pc", i, "_current")) >= lat_var_cps[[i]][q] &
            get(paste0("pc", i,  "_current")) <= lat_var_cps[[i]][q + 1], paste0("pc", i, "_binned")
    }else{
      acs[get(paste0("pc", i, "_current")) > lat_var_cps[[i]][q] &
            get(paste0("pc", i,  "_current")) <= lat_var_cps[[i]][q + 1], paste0("pc", i, "_binned")
    }
  }
}

# now construct them into one categorical variable
acs[,skill_bin := paste0(pc1_binned, pc2_binned, pc3_binned, pc4_binned)]

# creata dataset of skills groupings and children occupations
skills_occs_list <- acs[, .N, by = .(OCC2010,`CPS Occupational Title_current`, skill_bin)]
skills_occs_list[, grand_N := sum(N), by = skill_bin]

acs[, skill_bin_current := NULL]
setnames(acs, "skill_bin", "skill_bin_current")

acs <- merge(acs, skills_occs_list[,.(OCC2010, skill_bin)], by.x = "OCC10LY",  by.y = "OCC2010", all.
setnames(acs, c("skill_bin"), paste0(c("skill_bin"), "_ly"))

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

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

# melt and tabulate
num_correct <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(sl
total <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skill_bi
num_correct <- num_correct/total
num_cats <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skil

acs[, skill_bin := NULL]
acs[, skill_bin_current := NULL]
acs[, skill_bin_ly := NULL]
```

```r
  return(data.table(q1 = q1, q2 = q2, q3= q3, q4= q4, skill_bin_cats = num_cats$skill_bin_ly, skill_bin
}

# ## establish list of parameters to optimize over
# expand.grid(q1 = seq(2,10,2), q2 = seq(2,10,2), q3 = seq(2,10,2), q4 = seq(2,10,2)) %>% data.table ->
# candidate_grids[, qs := paste(q1, q2, q3, q4, sep = ",")]
# candidate_grids[, cats := q1*q2*q3*q4]
# candidate_grids <- candidate_grids[cats < 150 & cats >= 40]
# ## loop over
# acs$skill_bin <- NULL
# acs$skill_bin_current <- NULL
# acs$skill_bin_ly <- NULL
#
# grid_out <- lapply(candidate_grids$qs, grid_searchr) %>% rbindlist()
#
# # graph
# grid_out[, q1q2 := paste0(q1,q2)]
# grid_out[, q3q4 := paste0(q3,q4)]
# grid_out[, cats2 := prod(unlist(lapply(.SD, as.numeric))),  .SDcols = paste0('q', 1:4), by = .(q1q2,q
# grid_out[, coef := skill_bin_correct*skill_bin_cats]
#
# ggplot(grid_out) +
#   geom_tile(aes(x = q1q2, y = q3q4, fill = coef))
```

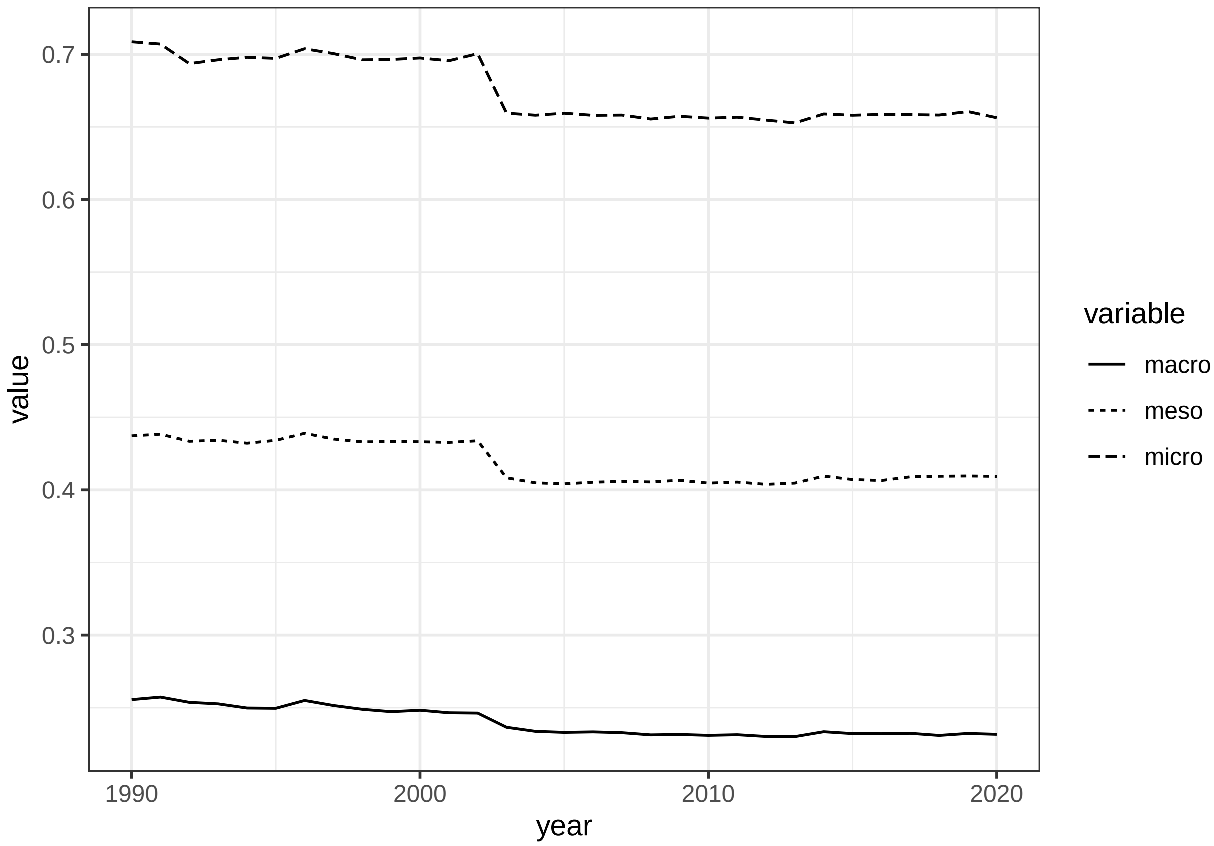
try network

```r
# 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")


##################
acs <- acs[!is.na(`Time Management.IM_ly` & !is.na(`Time Management.IM_current`))]
df <- data.table(rbind(acs[, .SD, .SDcols = names(acs)[names(acs) %like% "LV_current" & !names(acs) %li

stats::kmeans(df, centers = 85) -> temp

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

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

# assign clusters
acs[, kmeans_cluster_current := temp$cluster[1:nrow(acs)]]
acs[, kmeans_cluster_ly := temp$cluster[(nrow(acs) + 1):(nrow(acs)*2)]]

#
```

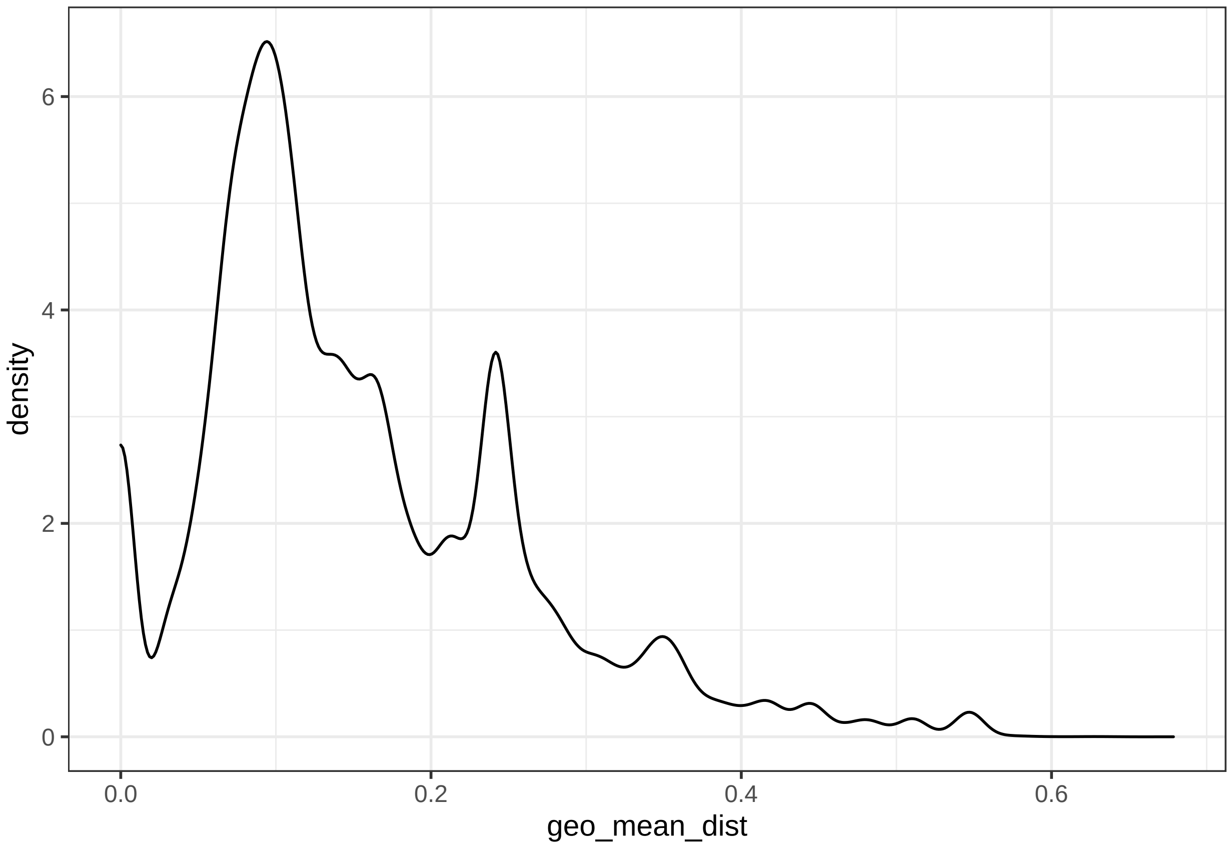
```r
overlap_micro_count <- acs[,.(N = length(unique(OCC2010))), by = .(microocc_current, kmeans_cluster_cur
overlap_micro_count[,kmeans_cluster_current := factor(kmeans_cluster_current,
                                                     levels = acs[,.(mean = mean(average_value_skills_
ggplot(overlap_micro_count) +
  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")
```



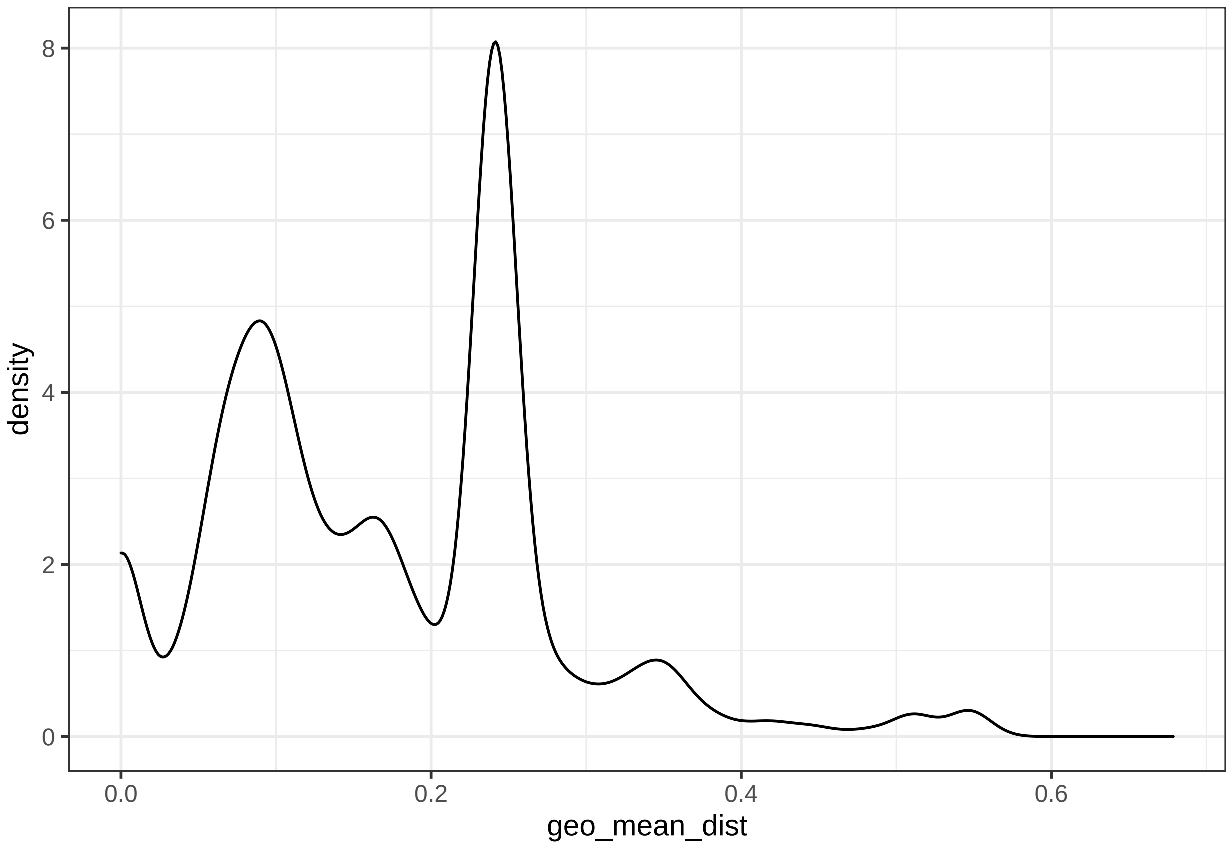

```r
# now compute AMI
ami_scores <- acs[complete.cases(acs[,.(kmeans_cluster_current, macroocc_current)]),
                  .(macro = aricode::AMI(macroocc_current, kmeans_cluster_current),
                    meso = aricode::AMI(mesoocc_current, kmeans_cluster_current),
                    micro =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



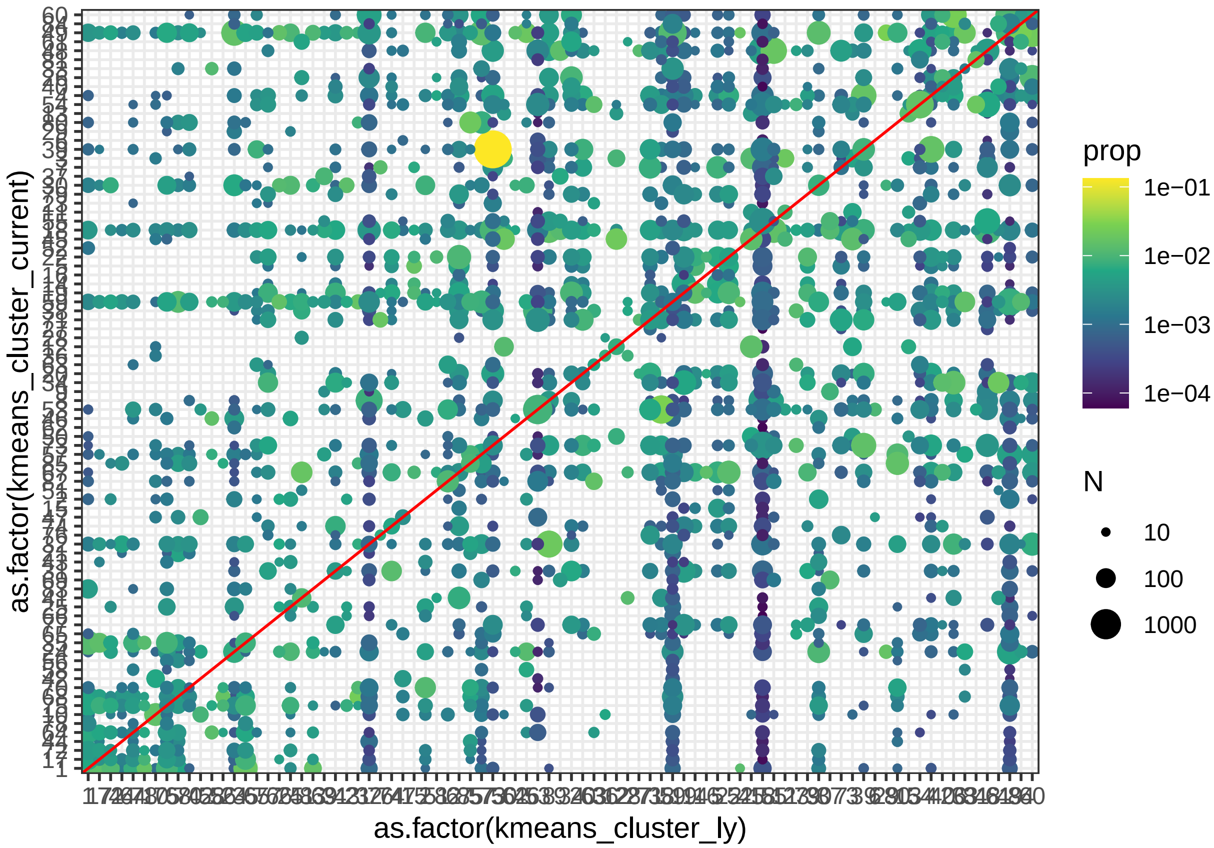

```
# judge concordance
acs[, kmeans_cluster_conc := ifelse(kmeans_cluster_current == kmeans_cluster_ly, 1,0)]
num_correct <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(ski
total <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skill_bin_
num_correct <- num_correct/total
num_cats <- acs[OCC1950 != OCC50LY & !is.na(skill_bin_conc) & !is.na(skill_bin_current) & !is.na(skill_

# 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")
```

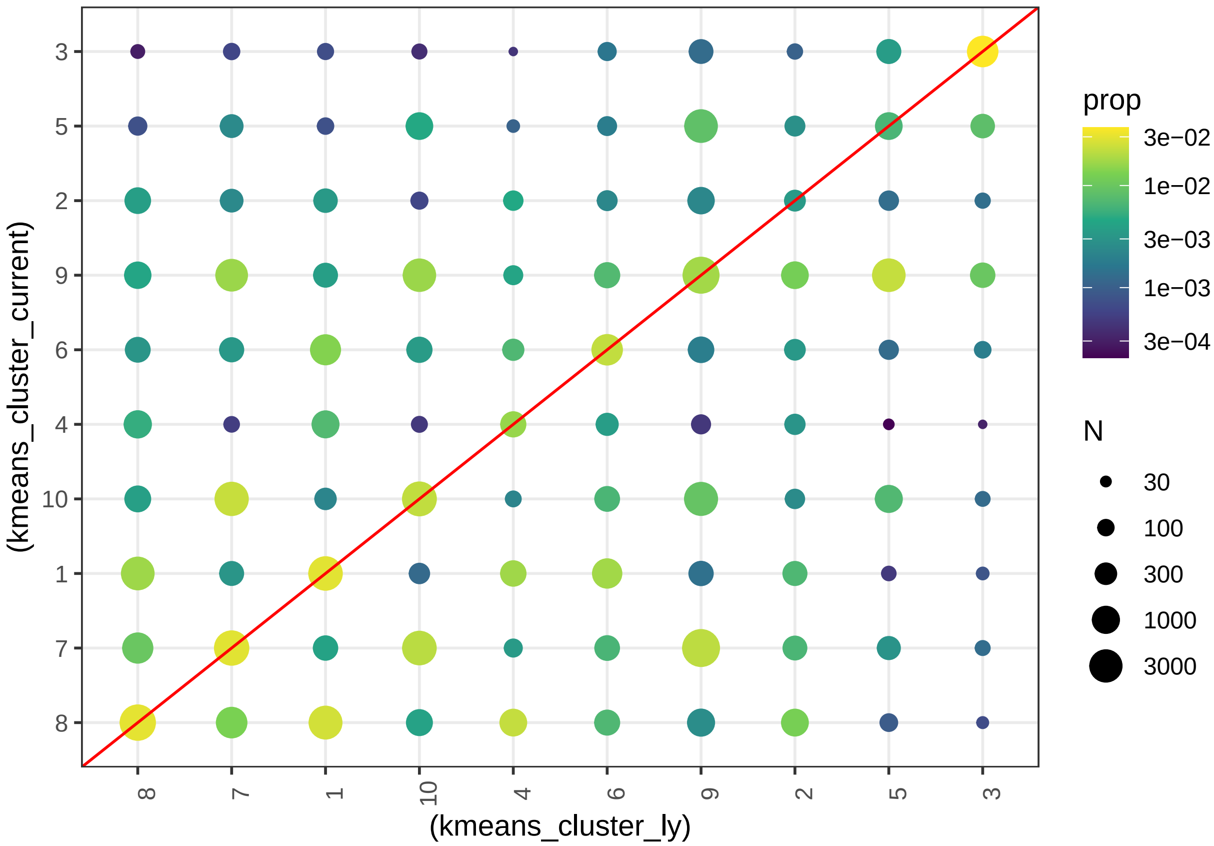


loop over k

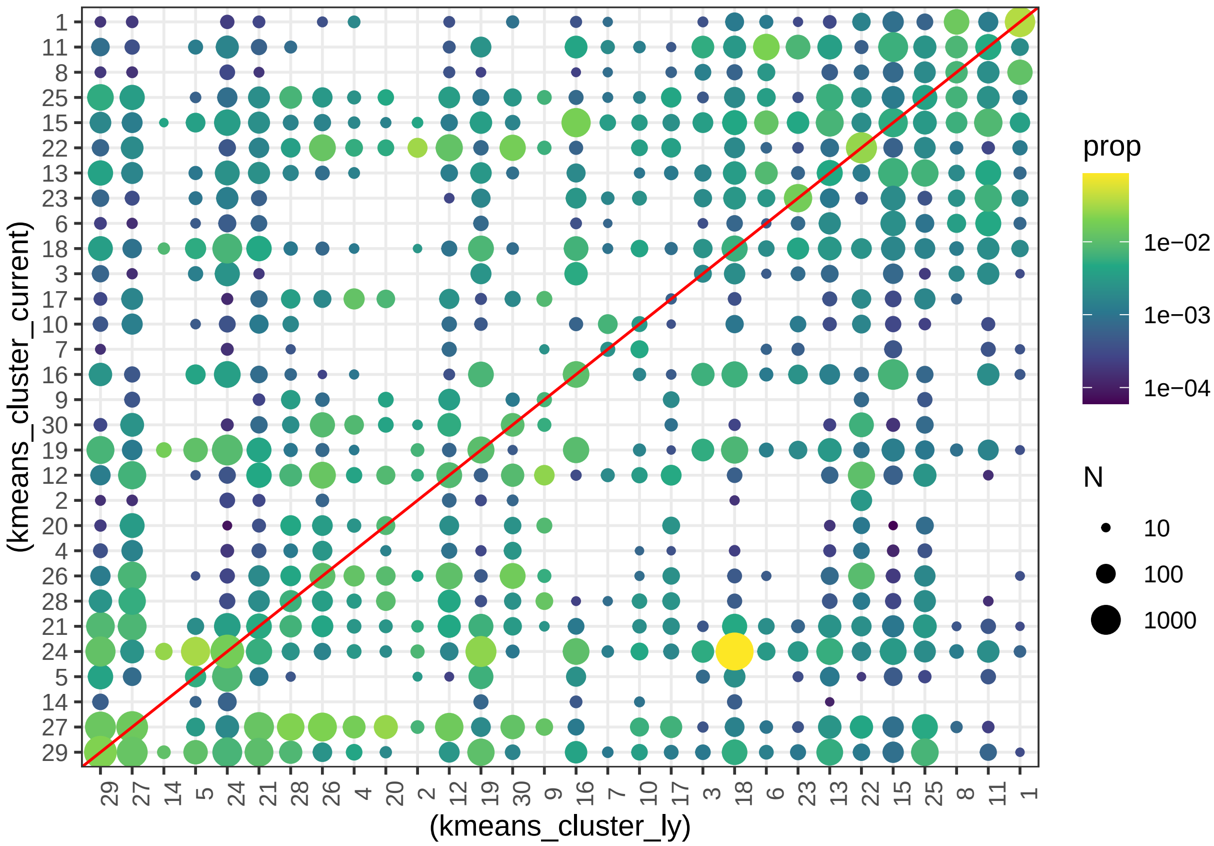
```
gg_list <- list()
q <- 0
for(k in seq(10,110,20)){
  q <- q + 1
  print(k)
  df <- data.table(rbind(acs[, .SD, .SDcols = names(acs)[names(acs) %like% "LV_current" & !names(acs) %
  stats::kmeans(df, centers = k) -> temp

  # 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/length(
                                         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)

}
```

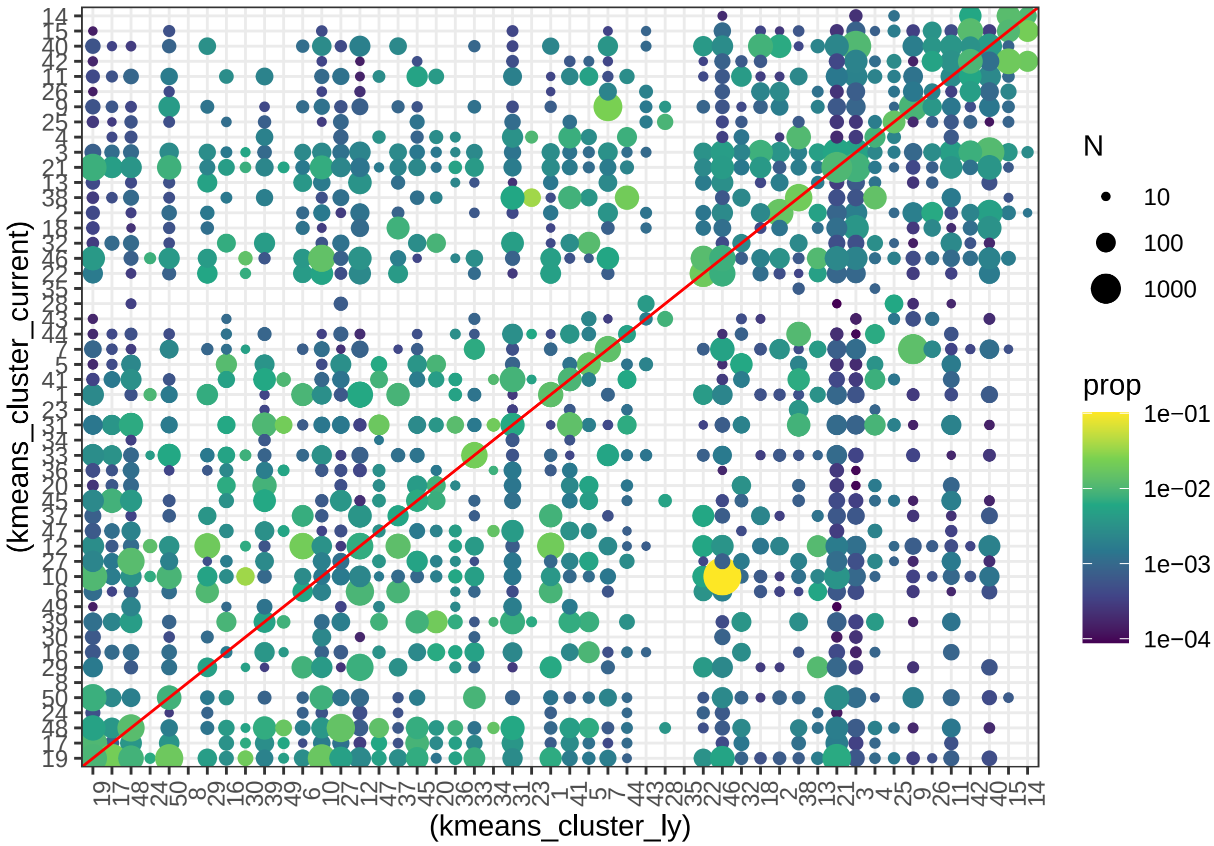
```
## [1] 10
```


prop
3e−02
1e−02
3e−03
1e−03
3e−04
N
30
100
300
1000
3000
(kmeans_cluster_current)
3
5
2
9
6
4
10
1
7
8
(kmeans_cluster_ly)
8
7
1
10
4
6
9
2
5
3

```
## [1] 30
```

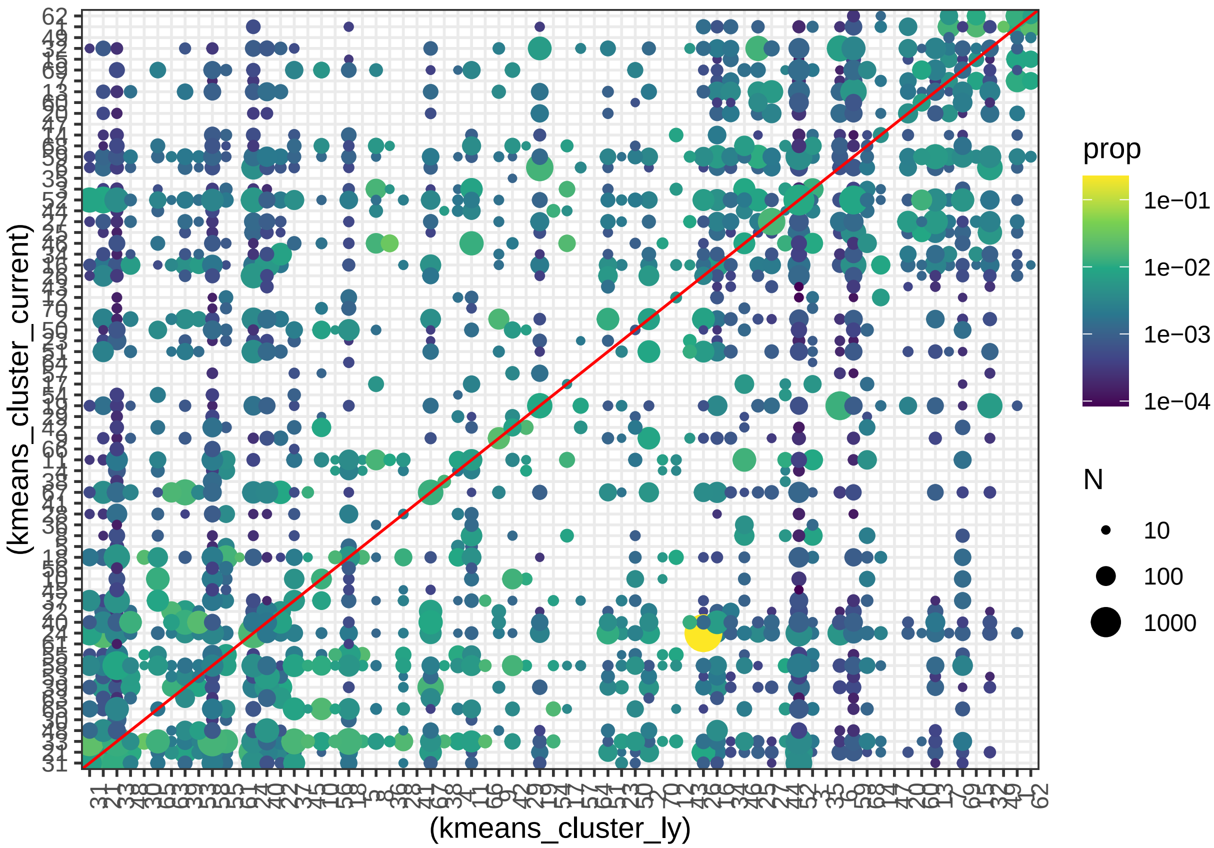

prop
1e−02
1e−03
1e−04
N
10
100
1000
(kmeans_cluster_current)
(kmeans_cluster_ly)


```
## [1] 50
```

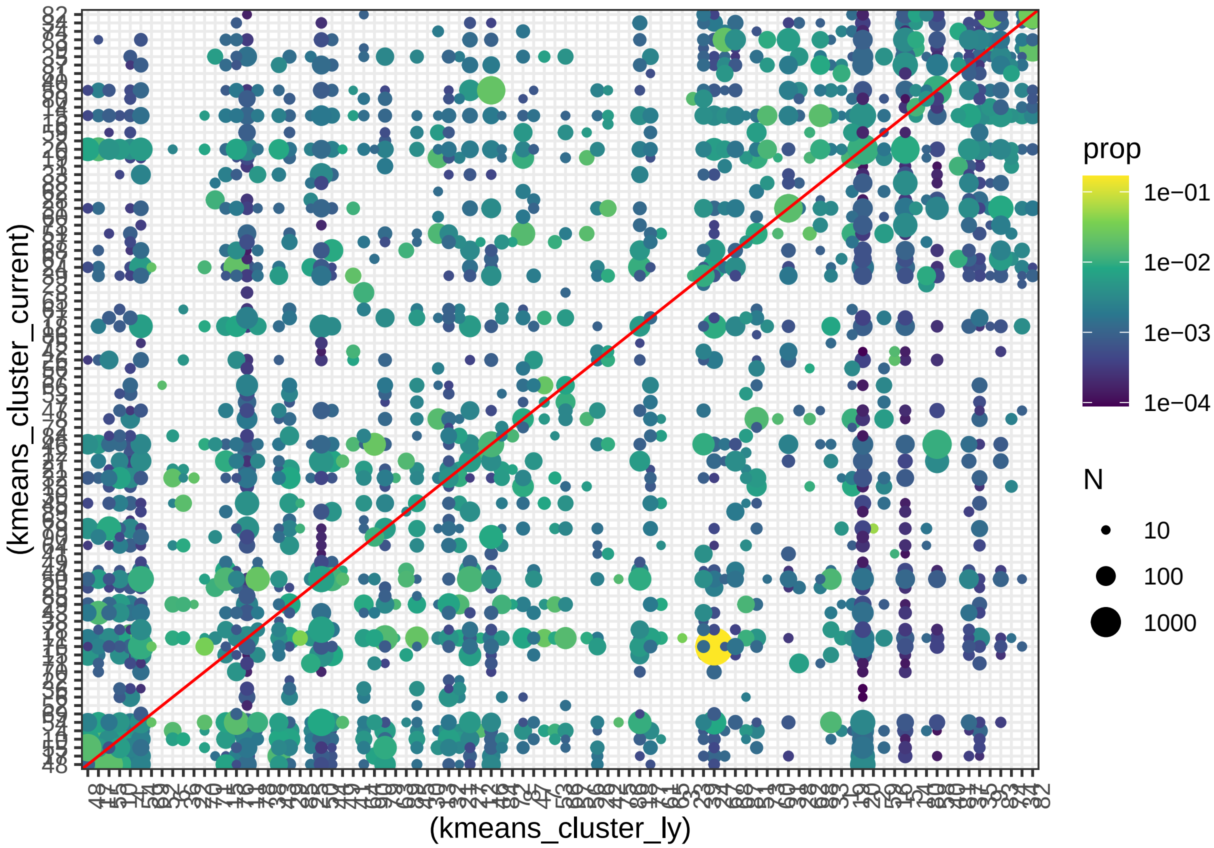


(kmeans_cluster_current)
(kmeans_cluster_ly)
N
10
100
1000
prop
1e−01
1e−02
1e−03
1e−04

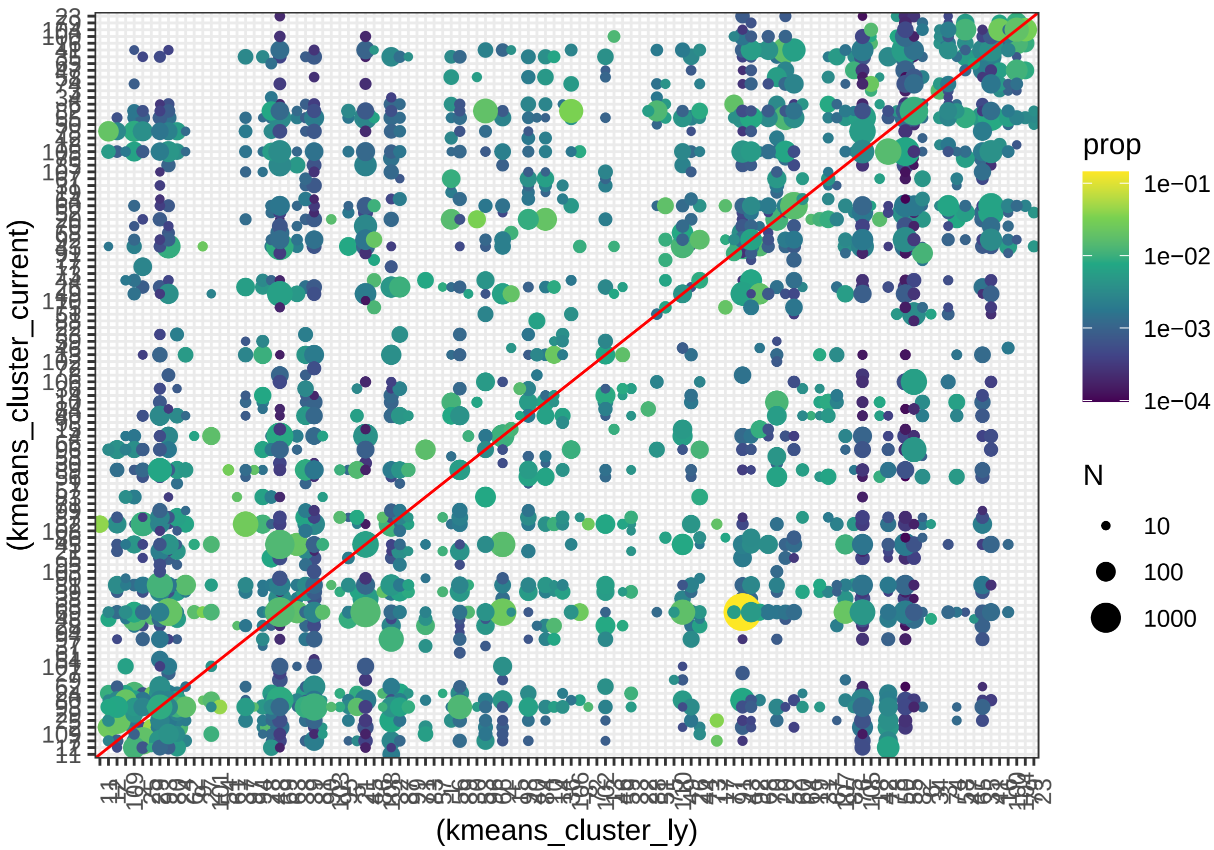

```
## [1] 70
```


(kmeans_cluster_current)
(kmeans_cluster_ly)
prop
1e−01
1e−02
1e−03
1e−04
N
10
100
1000

```
## [1] 90
```


(kmeans_cluster_current)
(kmeans_cluster_ly)
prop
1e−01
1e−02
1e−03
1e−04
N
10
100
1000

```
## [1] 110
```



See how well each scheme predicts log earnings

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

```
## [1] 76871.02
```

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

```
## [1] 76492.18
```