Matching students to advisers advanced example

This is an R Markdown Notebook.

This is meant as a more advanced working example for student-adviser matching. There are three basic entities: topics, advisors and students. Students match to an adviser via 1 of the 9 topics. For some topics the language of the thesis and/or the working group is fixed in advance. Others are left "open", but in the end we want to have one suitable language in a wg.

Needed software

```
source("hungar.R")
source("do_match.R")
```

Read in data

Data about advisoers, topics, students and student-preferences. At this moment the data about students is also in the student preferences file. Be careful: the seperator now is a semi-colon but if you change the files with excel (or so), it might also hange the seperator.

```
SEP= ";"
student_preferences<- read.csv(
  "./testdata/advanced/student_preferences.csv",
  header=TRUE, stringsAsFactors = FALSE, sep= SEP)
topics<- read.csv(
  "./testdata/advanced/topics.csv",
                    stringsAsFactors = FALSE, sep= SEP)
advisers<- read.csv(
  "./testdata/advanced/advisers.csv",
                    stringsAsFactors = FALSE, sep= SEP)

students<- student_preferences[,1:2]
```

Post processing until we have a weights table table

Do the following steps.

1. Zero all entries for which the students language does not match the topic wg or scr language.
2. Add half a point to student topic choices of their prefered language.
3. Make the N times M matrix where each entry is the maximum of the preferences of the student for that adviser. Remember which topic.
4. You might enter some other conditions.

Writing as a function

We want workgroups with one language, so later on we have to run the matching multiple times with different choices for the workgroups. So lets write a function that presents those results.

```r
LANGUAGE_MISMATCH = -10
LANGUAGE_BONUS = 0.25
TOO_MANY_STUDENTS = -100 ## overflow per advisor
FORBIDDEN_COMBINATION = -1000
SQUEEZE_POWER = 4 ## see below for more explanations

fixed_lang <- function(d2,d3,d4, tolics){
  topics[3:8, 4] <- c(d2,d2,d3,d3,d4,d4)

    student_topic<- student_preferences[, 3:11]
    a<- topics$scr_lang == "EN"
    b<- student_preferences$scr_lang == "NL"
    student_topic[b,a] <- LANGUAGE_MISMATCH

    a<- topics$scr_lang == "NL"
    b<- student_preferences$scr_lang == "EN"
    student_topic[b,a] <- LANGUAGE_MISMATCH

    a<- topics$wg_lang == "EN"
    b<- student_preferences$wg_lang == "NL"
    student_topic[b,a] <- LANGUAGE_MISMATCH

    a<- topics$wg_lang == "NL"
    b<- student_preferences$wg_lang == "EN"
    student_topic[b,a] <- LANGUAGE_MISMATCH

    # half a point for language preferences

    a<- topics$scr_lang == "EN"
    b<- student_preferences$scr_lang %in%  c("EN", "VE")
    student_topic[b,a] <- student_topic[b,a] + LANGUAGE_BONUS
    a<- topics$scr_lang == "NL"
    b<- student_preferences$scr_lang %in%  c("NL", "VN")
    student_topic[b,a] <- student_topic[b,a] + LANGUAGE_BONUS
    a<- topics$wg_lang == "EN"
    b<- student_preferences$wg_lang %in%  c("EN", "VE")
    student_topic[b,a] <- student_topic[b,a] + LANGUAGE_BONUS
    a<- topics$wg_lang == "NL"
    b<- student_preferences$wg_lang %in%  c("NL", "VN")
    student_topic[b,a] <- student_topic[b,a] + LANGUAGE_BONUS

    # collapse per advisor; not generic code yet

    cost<- student_topic[, 1:5] # just for the size
    names(cost)<- advisers$name
    mem_s_t <- cost #idem

    for (j in 1:4){
      range2 <- 2*j
      range1 <- range2-1
      up <- pmax(student_topic[, range1],
                         student_topic[, range2])
      cost[, j ] <- up
      mem_s_t[, j] <- ifelse(
        up== student_topic[, range1], range1, range2)

    }
    cost[, 5] <- student_topic[,9]
    mem_s_t[, 5] <- 9

    # force unpopular docet1 to get 8 students
    # cost<- cost-1
    # cost[,1]<- cost[,1] +5
    #
    match_table<- do_match(cost, advisers$max,
                  penalty = TOO_MANY_STUDENTS,
                  squeeze_power = SQUEEZE_POWER)
    match_table$topic = sapply(match_table$nr, function(i)
    mem_s_t[i, match_table$adviser[i]])


    return(match_table)
}
```

Now, later we will uncover the best language choice, but here just show the best result.

There is one more parameter at play here: SQUEEZE_POWER. Explanation:

Giving almost all students their prefered advisor but one student his/her least prefered advisor night raise some eyebrows, even though it maximizes the sum of values. If that effect is too gross, the high values can be "squeezed down" a little bit so that for instance 8 eights will weight more than 7 nines and 1 one. SQUEEZE_POWER of 1 is normal behaviour, use 2-10 for strong(er) equlizing effects.

In this example it does not make any difference, because the random characteristic of the preferences.

```
match_table<- fixed_lang("EN", "NL","NL", topic)

final_result<- data.frame(
  nr= students$nr[match_table$nr],
  name= students$name[match_table$nr],
  adviser= advisers$name[match_table$adviser],
  topic = match_table$topic,
  weight = match_table$value
)
```

You might want to print or save the result

```
# print or save final result
# write.csv(final_result, "fr.csv", row.names= FALSE, quote= FALSE)

print(sum(final_result$weight))
```

```
## [1] 299.75
```

```
knitr::kable(final_result, col.names= names(final_result), row.names=FALSE, caption="Best Matching")
```

Table 1: Best Matching

nr	name	adviser	topic	weight
1	Daan	docent2	4	7.25
2	Bram	docent5	9	9.50
3	Thijs	docent1	2	7.50
4	Mees	docent5	9	9.50
5	Stijn	docent1	2	6.50
6	Siem	docent1	2	5.50
7	Gijs	docent1	2	5.50
8	Jan	docent2	3	7.25
9	Teun	docent2	4	6.25
10	Noud	docent5	9	9.50
11	Tijn	docent3	5	9.25
12	Floris	docent3	5	8.25
13	Ties	docent4	7	9.50
14	Joep	docent4	7	8.50
15	Niek	docent4	7	9.50
16	Pepijn	docent3	6	8.25
17	Koen	docent4	8	8.25
18	Thijmen	docent4	7	8.50
19	Fedde	docent4	8	8.25
20	Bas	docent4	8	9.25
21	Hidde	docent2	3	5.25
22	Pieter	docent3	6	9.00
23	Johannes	docent3	5	9.00
24	Joris	docent2	3	7.25
25	Jelle	docent5	9	9.50
26	Jip	docent2	4	7.25
27	Hendrik	docent1	2	5.50
28	Cornelis	docent3	6	9.25
29	Rens	docent5	9	9.00
30	Jelte	docent2	4	6.00
31	Melle	docent3	6	9.25
32	Wout	docent5	9	9.00
33	Duuk	docent3	5	9.25
34	Loek	docent4	7	9.50
35	Gerrit	docent2	4	7.00
36	Laurens	docent5	9	8.00
37	Matthijs	docent5	9	9.00

Diagnostics

Next look at some other details of the matching.

```r
diag1 <- aggregate(final_result[,3],
                   by=list(final_result$adviser),
                   FUN=length)
names(diag1)<- c("name", "nr_students")
diag1<- merge(advisers, diag1, by="name")

knitr::kable(diag1, caption="Adviser load")
```

Table 2: Adviser load

name	max	nr_students
docent1	8	5
docent2	8	8
docent3	8	8
docent4	8	8
docent5	8	8

```
##### check languages

diag2<- final_result[order(final_result$adviser,
                           final_result$topic), c(3:4, 1:2)]
diag2$scr_lang <- student_preferences$scr_lang[diag2$nr]
diag2$wg_lang <- student_preferences$wg_lang[diag2$nr]

knitr::kable(diag2, row.names=FALSE,
             caption= "Unfortunate matches")
```

Table 3: Unfortunate matches

adviser	topic	nr	name	scr_lang	wg_lang
docent1	2	3	Thijs	EN	EN
docent1	2	5	Stijn	EN	EN
docent1	2	6	Siem	EN	EN
docent1	2	7	Gijs	EN	EN
docent1	2	27	Hendrik	VE	VE
docent2	3	8	Jan	EN	EN
docent2	3	21	Hidde	VE	VE
docent2	3	24	Joris	VE	VE
docent2	4	1	Daan	EN	EN
docent2	4	9	Teun	EN	EN
docent2	4	26	Jip	VE	VE
docent2	4	30	Jelte	VN	VN
docent2	4	35	Gerrit	VN	VN
docent3	5	11	Tijn	NL	NL
docent3	5	12	Floris	NL	NL
docent3	5	23	Johannes	VE	VE
docent3	5	33	Duuk	VN	VN
docent3	6	16	Pepijn	NL	NL
docent3	6	22	Pieter	VE	VE
docent3	6	28	Cornelis	VN	VN
docent3	6	31	Melle	VN	VN
docent4	7	13	Ties	NL	NL
docent4	7	14	Joep	NL	NL
docent4	7	15	Niek	NL	NL
docent4	7	18	Thijmen	NL	NL
docent4	7	34	Loek	VN	VN
docent4	8	17	Koen	NL	NL
docent4	8	19	Fedde	NL	NL
docent4	8	20	Bas	NL	NL
docent5	9	2	Bram	EN	EN
docent5	9	4	Mees	EN	EN

adviser	topic	nr	name	scr_lang	wg_lang
docent5	9	10	Noud	EN	EN
docent5	9	25	Jelle	VE	VE
docent5	9	29	Rens	VN	VN
docent5	9	32	Wout	VN	VN
docent5	9	36	Laurens	VN	VN
docent5	9	37	Matthijs	VN	VN

The conclusion is that there is a lot of language mismatch for docent2-4; the only remedy we see is to repeat the algorithm 8 times. we have to copy a lot of code,

```
for (d2 in c("EN", "NL"))
for (d3 in c("EN", "NL"))
for (d4 in c("EN", "NL")){
    matched <- fixed_lang(d2, d3, d4, topic)
    print(c(d2,d3,d4, sum(matched$value)))
  }
```

```
## [1] "EN"     "EN"     "EN"     "124.75"
## [1] "EN"     "EN"     "NL"     "269.25"
## [1] "EN"  "NL"  "EN"  "248"
## [1] "EN"     "NL"     "NL"     "299.75"
## [1] "NL"    "EN"    "EN"    "251.5"
## [1] "NL"     "EN"     "NL"     "298.25"
## [1] "NL"    "NL"    "EN"    "287.5"
## [1] "NL"     "NL"     "NL"     "297.75"
```

So we reach a maximum for three times dutch.