Problem 1 (25 Credits)

HW4

Michael Brucek (mbrucek), Eduardo Chavez (echavezh), Kevin Grady (grady133), Danny Moncada (monca016), Tian Zhang (zhan7003)

April 11, 2020

```
suppressWarnings(suppressPackageStartupMessages({
  library(igraph)
  library(ggplot2)
  }))
```

Question 0: Data Description (1 credit)

Please load the data from file `EU_email.txt` and `EU_membership.txt`.

The `EU_email.txt` contains email data from a large European research institution. We have anonymized information about all incoming and outgoing email between members of the research institution. There is an edge (i, j) in the network if person i sent person j at least one email. We count the edge between i and j as undirected.

The `EU_membership.txt` file contains "ground-truth" community memberships of the vertices. Each individual belongs to exactly one of 21 departments at the research institute. The 1st column is each individual's ID, and the 2nd column is each department's ID (i.e., membership label).

```
# please write your code below
setwd("~/MSBA 2020 All Files/Spring 2020/MSBA 6430 - Advanced Issues in Business Analytics/HW4")

E = read.table("EU_email.txt", header = F)
# Save as matrix format
E = as.matrix(E)


memberTRUE = read.table("EU_membership.txt", header = F)
# Save as matrix format
memberTRUE = as.matrix(memberTRUE)
```

Question 1 (3 credits)

How many unique vertices? How many edges? Please illustrate the graph.

Hints:

1. To illustrate the graph, you may use `vertex.label=NA, vertex.frame.color=NA, vertex.size=7, edge.arrow.size=0.5, edge.arrow.width=0.5`

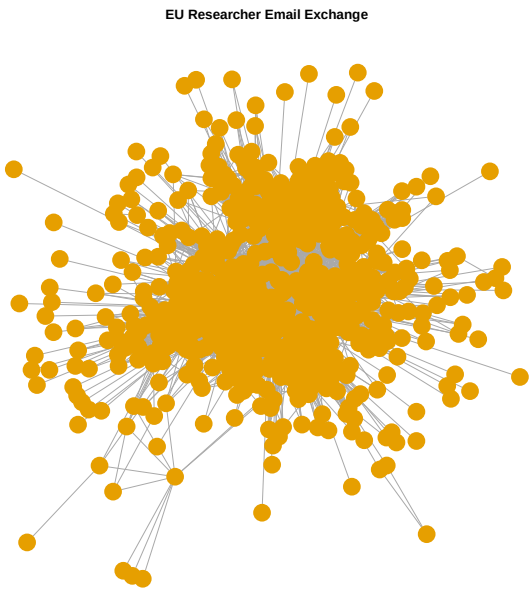
```
g <- graph_from_edgelist(E, directed = FALSE)

# How many unique vertices?
v = gorder(g)
v
```

```
## [1] 714
```

```
# How many edges?
n = gsize(g)
n
```

```
## [1] 7992
```

```
# Graph illustration
set.seed(66)
par(mfrow=c(1,1))
plot(g, vertex.frame.color = NA, vertex.label = NA,
     vertex.size = 7, edge.arrow.size = 0.5, edge.arrow.width = 0.5,
     main = "EU Researcher Email Exchange")
```



Question 2 (4 credits)

Please conduct community detection via `Edge Betweenness`. What's the computational time? What's the modularity? How many communities do we find? Recall that the ground-truth has only 21 communities. Also please take a look at the membership assignment. Does each community have roughly equal members?

Hints:

1. It may run 2-10 minutes depending on your PC.
2. Code is available from the class examples.

```
# Start the timer
t0 = proc.time()
# Generate the communities using edge betweenness
```

```r
EB = edge.betweenness.community(g, directed = FALSE)
# Stop the timer to show computation time and save to a variable
comp_time_eb = proc.time()-t0
# Show the results of our edge betweenness analysis
EB
```

```
## IGRAPH clustering edge betweenness, groups: 205, mod: 0.49
## + groups:
##   $`1`
##    [1]   1   2  11  12  53  54  61  88 105 124 154 156 157 158 159 160 161
##   [18] 162 163 164 165 166 185 190 223 230 232 233 234 235 236 237 239 240
##   [35] 263 359 364 417 519 628 645
##
##   $`2`
##    [1]   3   4   5   7   8  13  36  37 103 114 152 183 184 199 200 201 202
##   [18] 219 247 254 279 280 281 283 290 321 322 330 337 344 354 355 362 380
##   [35] 381 385 387 388 389 390 391 392 393 394 395 398 408 411 412 432 435
##   [52] 439 440 443 444 446 464 467 491 498 502 505 506 521 522 525 534 541
##   + ... omitted several groups/vertices
```

```r
# Since there's 205 groups/communities per the output of the analysis, we won't show the
# output of membership
memberEB = membership(EB)

#############################################################################################
# 1.  What's the computational time?
comp_time_eb[3]
```

```
## elapsed
##  350.54
```

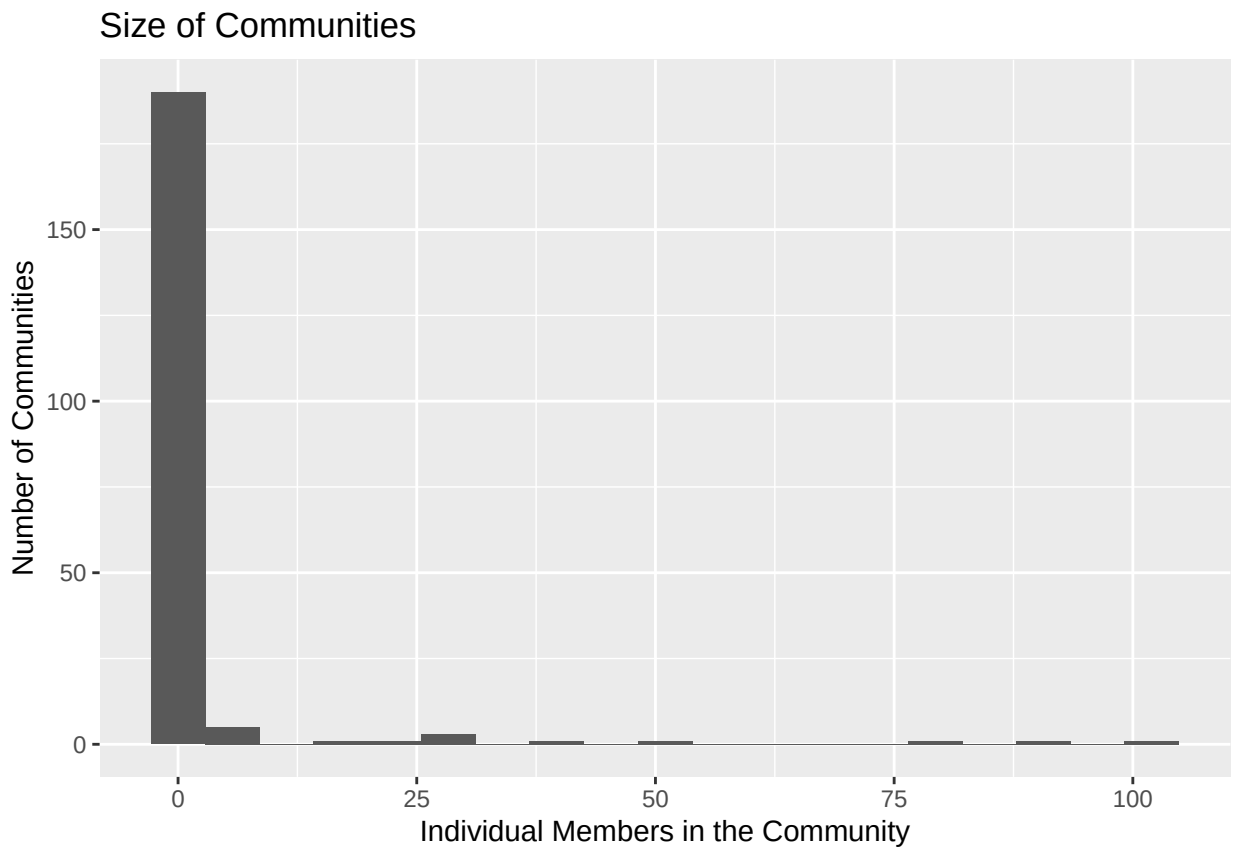
```r
# 2.  What's the modularity?
modularity(EB)
```

```
## [1] 0.488781
```

```r
# 3.  How many communities do we find?
length(communities(EB))
```

```
## [1] 205
```

```r
# 4.  Does each community have roughly equal members?
as.data.frame(table(memberEB)) %>%
    ggplot(aes(x=Freq)) +
    geom_histogram(bins=19) +
    labs(title = "Size of Communities",
          x= "Individual Members in the Community",
          y= "Number of Communities")
```




```
# No, there are many communities that only have one member, while one of the communities
# has 103 members; there is an uneven distribution.
```

Question 3 (4 credits)

Please conduct community detection via `Walk Trap`. What's the computational time? How many communities do we find? What's the modularity? Please take a look at the membership assignment. Does each community have roughly equal members?

Hints:

1. Code is available from the class examples.

```
# Start the timer
t0=proc.time()
# Generate the communities using walk trap
WT=walktrap.community(g)
# Stop the timer to show computation time and save to a variable
comp_time_wt = proc.time()-t0
# Show the results of our walk trap analysis
WT
```

```
## IGRAPH clustering walktrap, groups: 73, mod: 0.54
## + groups:
```

```
##   $`1`
##    [1]  14  49  50  51  52  66 131 133 134 135 136 194 195 214 224 225 229
##   [18] 248 249 250 316 328 363 374 396 397 416 420 427 428 429 452 479 500
##   [35] 503 508 516 528 530 532 550 568 579 580 591 618 631 636 637 650 677
##   [52] 681 695 696 698
##
##   $`2`
##     [1]   6   9  15  16  35  42  44  47  57  58  59  60  62  63  67  68  69
##    [18]  86  93  94  95  97 117 118 122 123 130 138 139 140 141 142 143 144
##    [35] 145 146 147 148 149 169 186 189 191 192 204 208 209 212 215 216 217
##   + ... omitted several groups/vertices
```

```
# Save the membership to variable
memberWT=membership(WT)

#############################################################################################
# 1.  What's the computational time?
comp_time_wt[3]
```

```
## elapsed
##    0.11
```

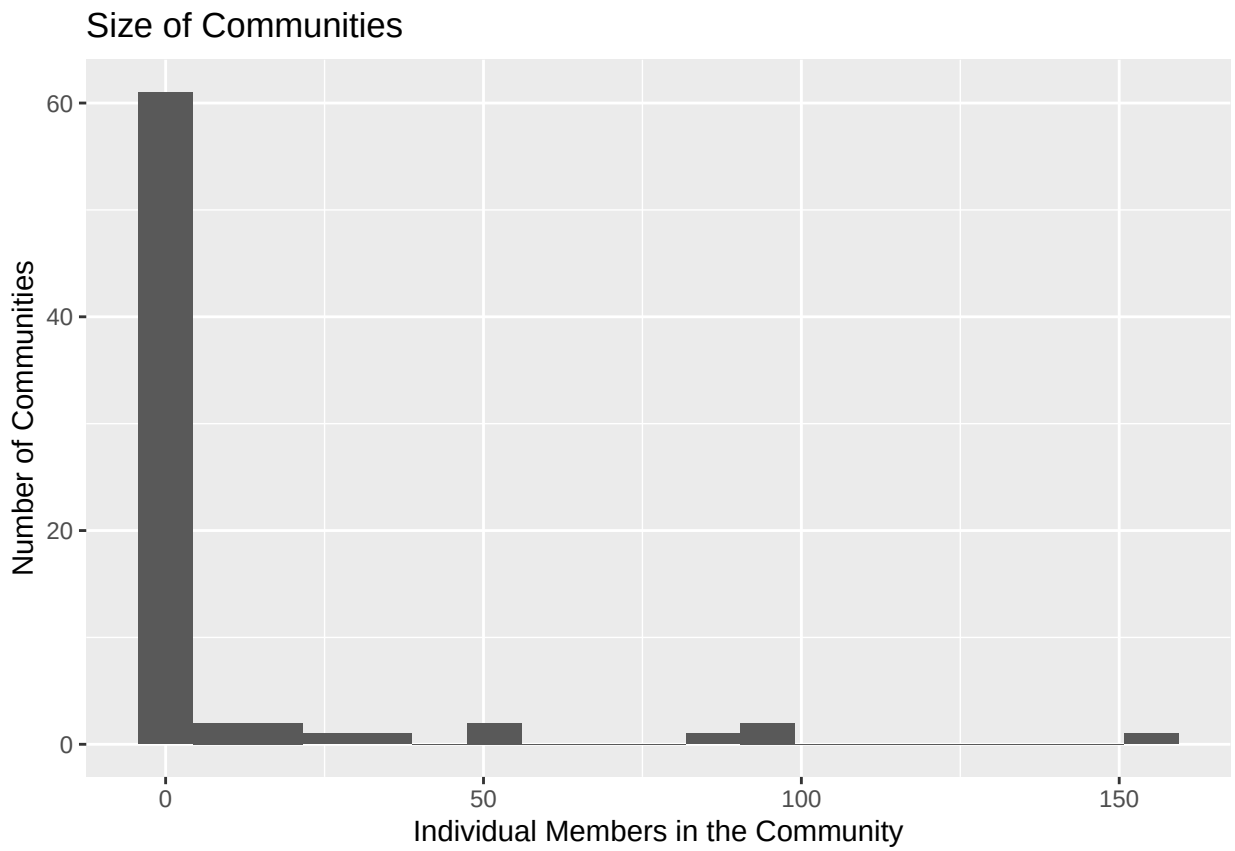
```
# 2.  What's the modularity?
modularity(WT)
```

```
## [1] 0.5396314
```

```
# 3.  How many communities do we find?
length(communities(WT))
```

```
## [1] 73
```

```
# 4.  Does each community have roughly equal members?
as.data.frame(table(memberWT)) %>%
    ggplot(aes(x=Freq)) +
    geom_histogram(bins=19) +
    labs(title = "Size of Communities",
          x= "Individual Members in the Community",
          y= "Number of Communities")
```




```
# The plot demonstrates that each community is much closer in size than the edge
# betweenness, but we still have members who belong to their own island... they must be
# happy to be there right now!  We also see one community with over 150 members.
```

Question 4 (4 credits)

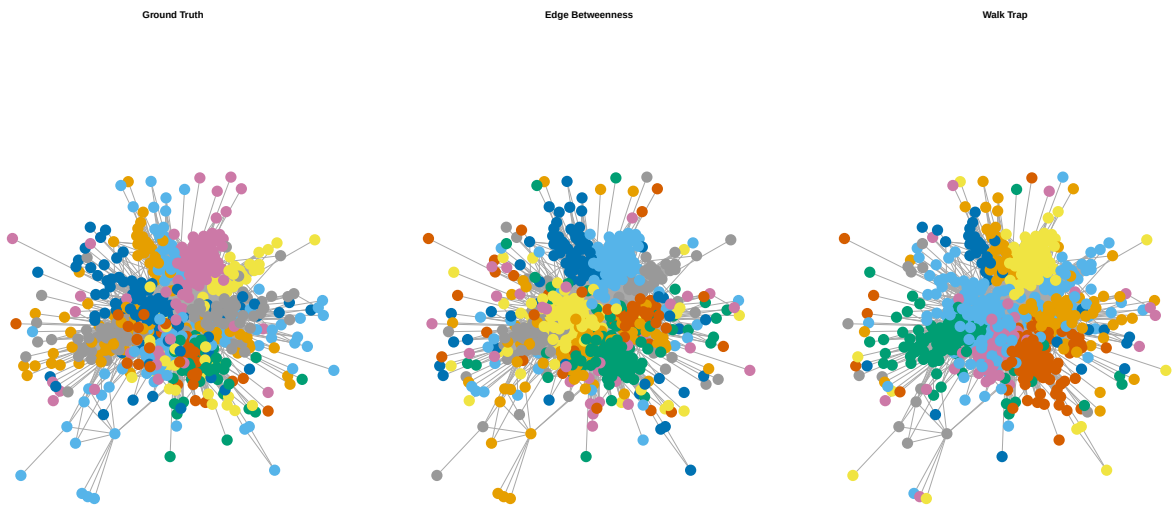
Now illustrate the true community structure, and compare it with those by `edge betweenness` and `walk trap` side-by-side. Whose structure is closer to the truth, `edge betweenness` or `walk trap`?

Hints:

1. The true membership is the 2nd column of `EU_membership.txt`.
2. The rest of the code is available from the class examples.

```
# Rebuild the graph using the TRUE membership data
memberTRUE1 = as_membership(memberTRUE[,2])

#Illustrate and compare the community detection results
par(mfrow=c(1,3))
set.seed(66)
plot(g, vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
```

```
     vertex.color = memberTRUE1,
     main = "Ground Truth")

set.seed(66)
plot(g,
     vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     vertex.color = memberEB,
     main = "Edge Betweenness")

set.seed(66)
plot(g,
     vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     vertex.color = memberWT,
     main = "Walk Trap")
```



Based on these three plots, the Edge Betweenness algorithm was able to find the community structure that was closer to the ground truth. The Edge Betweenness plot shows a better representation of the true community structure; the higher modularity of the Walk Trap model (0.53 vs. 0.49) does not signify that it is outperforming the Edge Betweenness model.

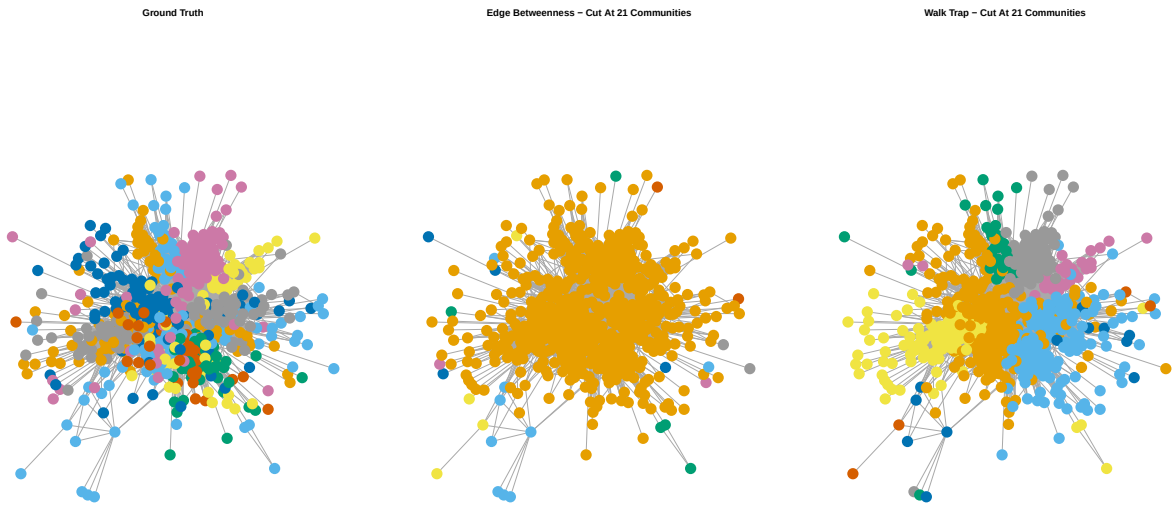
Question 5 (5 credits)

Now we introduce something new. We force each method to allow only 21 communities. And we illustrate the community structure as in the question above. Now which method is closer to the ground truth? Combined with what we learned about the idea of each method (slides regarding EB traversing from 1 community to n, and WT being exactly the opposite), can you provide some interpretation why the other method deteriorate badly?

Hints:

1. We don't need to re-run each method. Simply use the function `cut_at(, no=21)` when specifying each method's `vertex.color`, and we get the new membership assignment.

```
#Illustrate and compare the community detection results
par(mfrow=c(1,3))
set.seed(66)
plot(g,
     vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     vertex.color = memberTRUE1,
     main = "Ground Truth")

set.seed(66)
plot(g,
     vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     vertex.color = cut_at(EB, no=21),
     main = "Edge Betweenness - Cut At 21 Communities")

set.seed(66)
plot(g,
     vertex.frame.color = NA,
     vertex.label = NA,
     vertex.size = 7,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     vertex.color = cut_at(WT, no=21),
     main = "Walk Trap - Cut At 21 Communities")
```



Now that we are setting the cut off at 21 communities (which is the "ground-truth" community structure of each researcher who belongs to exactly one of 21 departments at the research institute), we see that the Walk Trap algorithm *now* outperforms the Edge Betweenness algorithm when trying to determine the true community network structure.

Edge Betweenness takes the number of shortest paths going through an edge (with edges that have higher betweenness identified as the "bridge"" between two communities) while *cutting* edges as it works through the structure. EB starts with **1** community and grows until it has covered the whole network, so by setting a cutoff of 21 communities the algorithm is forced to cut *more* edges to fit that number and therefore doesn't expand like we saw in plot for Question 4. This is demonstrated by the lack of different colors on the fringes of the plot; it was unable to branch out after reaching the cutoff at 21 and we are left with a big splash of orange in the center of the plot.

Walk Trap works completely opposite to the EB algorithm, starting with n number of communities and *merging* vertices together to form the community structure. More specifically, it defines a distance r between a vertex i and a community C using probabilities of random walks. When r is small the vertex i is in the community C. One of the benefits to Walk Trap is that is can handle an undirected network, like we are faced in this example; we are mapping the emails of researchers coming from and going to different offices - we suspect that the vast majority are **outbound** emails, and the plot shows us that Walk Trap can capture what is going on between any two vertices (researcher1 and researcher 2) even with this cutoff in place.

Question 6 (4 credits)

Now let's check the modularity of those new community structures, and see if it agrees with our illustrations above. Comparing the results with their unrestricted counter parts. What do you see? Also check the modularity of the ground-truth structure.

Hints:

1. Use the function `modularity(g, )` and provide the membership assignment of each method.

```r
# Get modularity results for our original unrestricted method
print("Modularity of EB - Original", quote=FALSE)
```

```
## [1] Modularity of EB - Original
```

```r
modularity(g, memberEB)
```

```
## [1] 0.488781
```

```r
print("Modularity of WT - Original", quote=FALSE)
```

```
## [1] Modularity of WT - Original
```

```r
modularity(g, memberWT)
```

```
## [1] 0.5396314
```

```r
print("Modularity of Ground Truth", quote=FALSE)
```

```
## [1] Modularity of Ground Truth
```

```r
modularity(g, memberTRUE[,2])
```

```
## [1] 0.4909872
```

```r
EB_diff = modularity(g, memberTRUE[,2]) - modularity(g, memberEB)
WT_diff = modularity(g, memberTRUE[,2]) - modularity(g, memberWT)
abs(EB_diff) < abs(WT_diff)
```

```
## [1] TRUE
```

```r
# Based on these results, the Edge Betweenness algorithm was able to capture a better
# picture of the true community structure because it was much closer to the ground truth
# modularity.


# Comparing modularity for the restricted community structures on the original data
print("Modularity of EB - Cut At 21", quote=FALSE)
```

```
## [1] Modularity of EB - Cut At 21
```

```r
modularity(g, cut_at(EB, no=21))
```

```
## [1] 0.003361634
```

```r
print("Modularity of WT - Cut At 21", quote=FALSE)
```

```
## [1] Modularity of WT - Cut At 21
```

```r
modularity(g, cut_at(WT, no=21))
```

```
## [1] 0.5282268
```

```r
print("Modularity of Ground Truth", quote=FALSE)
```

```
## [1] Modularity of Ground Truth
```

```r
modularity(g, memberTRUE[,2])
```

```
## [1] 0.4909872
```

```r
EB21_diff = modularity(g, memberTRUE[,2]) - modularity(g, cut_at(EB, no=21))
WT21_diff = modularity(g, memberTRUE[,2]) - modularity(g, cut_at(WT, no=21))
abs(EB21_diff) > abs(WT21_diff)
```

```
## [1] TRUE
```

```r
# After setting the cutoff to 21 communities, Walk Trap outperforms Edge Betweenneess.
```

- Thanks for a wonderful semester Professor Xuan Bi and Agnes!

Done!

Congratulations!