Exploring solutions space

Fabio Morea

introduction

exploring solutions space - trial #2

what is a "good" partition

We want to identify a partition of G.

Specifically we want to identify a **good** partition of G, which means

Single:

- the number of communities is meaningful for interpretation
- communities are not internally disconnected,
- valid communities: mixing parameter 0.5 (weak: for the whole network or strong for each community)
- stable: on repeated trials, we get always the same result (number of communities, and their composition)
- unaffected by ordering: results do not change upon sorting (or shuffling) the order of nodes and edges
- if there is any uncertainty, it is measured at node-level
- outliers can be identified (and interpreted, pruned or highlighted)

This is not trivial, indeed community detection algorithms may fail on one or more of the above points

why we need to explore solution space

Unfortunately, just a single trial of a specific algorithm is not enough to understand if any (or many) of the issues mentioned above are present. Not to mention mitigating or removing them.

We propose two preliminary steps for the choice of algorithm:

1. the algorithm is immune from the problem of internally disconnected communities (i.e. avoid Louvain)
2. the algorithm, with appropriate parameter values, produces a meaningful number of communities $1 < k < n$

then explore stability: repeat the algorithm t times and check how many solutions you get:

- always the same solution. That's easy, you are done.
- a prevalent solution (more than 50% of occurrencies). You can chose the prevalent one.
- few solutions (all below 50%, none is prevaliling). You can chose just one (eg based on modularity score or on the frequency) as above. Or the most common ones (e.g. the top ones that represent at least 50% of the solution space), and perform consensus.
- many different solutions (of the same order of t). Consensus is required. You can either use the whole solution space (which delivers a thorough estimate of uncertainty and outliers) or prune a quantile and proceed with that.

Load network and explore main features

Loading the data to a network g.

```
file_name = "mobility_fvg_sample_01.graphml"
g = igraph::read_graph(file_name, format="graphml")
V(g)$str<-strength(g)
print(g)
```

```
IGRAPH a7b9aec UNW- 203 13487 --
+ attr: name (v/c), id (v/c), str (v/n), weight (e/n)
+ edges from a7b9aec (vertex names):
 [1] Aiello--Amaro      Aiello--Ampezzo    Aiello--Andreis    Aiello--Aquileia
 [5] Aiello--Arba       Aiello--ArtaTerme  Aiello--Attimis    Aiello--Aviano
 [9] Aiello--AzzanoX    Aiello--Bagnaria   Aiello--Basiliano  Aiello--Bertiolo
[13] Aiello--Bicinicco  Aiello--Brugnera   Aiello--Budoia     Aiello--Buja
```

```
[17] Aiello--Buttrio    Aiello--Camino     Aiello--CampTap    Aiello--Campoform
[21] Aiello--Caneva     Aiello--Capriva    Aiello--Carlino    Aiello--Casarsa
[25] Aiello--Cassacco   Aiello--Castions   Aiello--Cavasso    Aiello--Cervignano
[29] Aiello--Chions     Aiello--ChiopVisc  Aiello--Cividale   Aiello--Codroipo
+ ... omitted several edges
```

Communities - a simple approach

as first test we identify communities using infomap. The result is a *igraph community object*

```
#comms <- infomap.community(g)
#comms <- walktrap.community(g)
#comms <-  cluster_louvain(g,resolution = 1.0)
#comms <-  cluster_leiden(g,resolution = 30)
comms <-  label.propagation.community(g)

table(comms$membership)
```

```
  1   2   3   4
128  23  46   6
```

```
commdf <- data.frame(loc = comms$name, comm = comms$membership)
#commdf %>% head(10)
```

```
explore_solutions_space <- function(g, tmax=10, met="IM" ){
    M <- matrix(NA, nrow = vcount(g), ncol = tmax)
    S <- data.frame()
    for (i in 1:tmax) {
        gs <-igraph::permute(g, sample(vcount(g)))
        method <- switch(met,
                 "IM" = igraph::infomap.community,
                 "LV" = igraph::louvain_clusters,
                 "LP" = label.propagation.community)

        memberships <- method(gs)$membership

        M[, i] <- memberships[ match(V(g)$name, V(gs)$name) ]
        for (j in 1:i){
```

```
            if (i != j) {
                sscore<-igraph::compare(M[,i],M[, j], method = "nmi")
                S <- rbind(S, data.frame(i,j, sscore))
             }
        }
    }


    return(list(M = M, S = S))

}
```

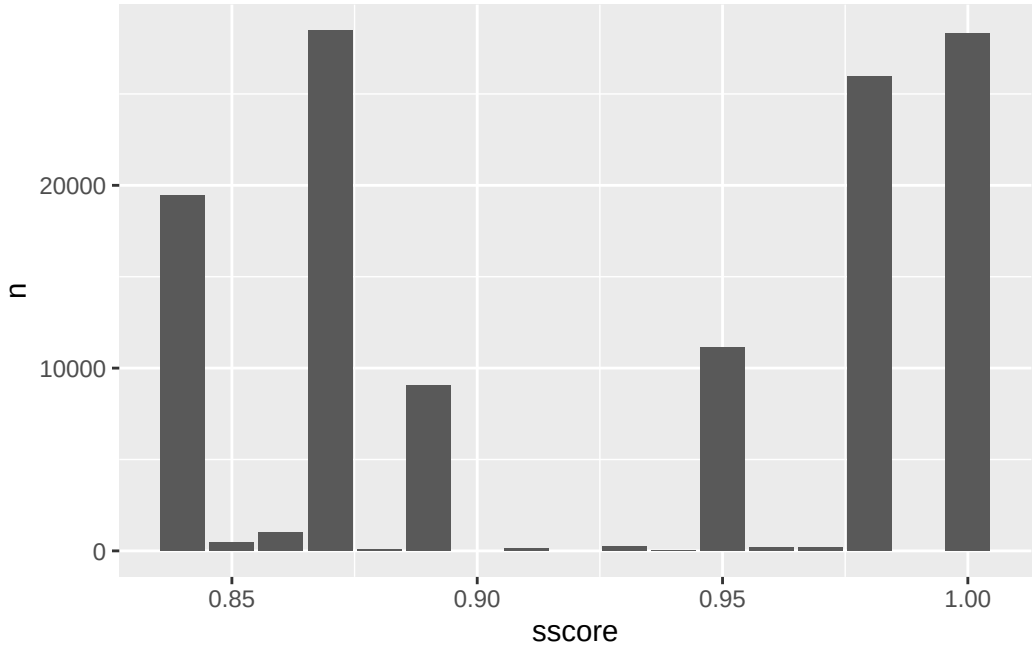
```
tmax = 500
tmp <- explore_solutions_space (g, tmax = tmax, met = "IM")
M <- tmp$M; S <- tmp$S

print(paste("calculated", tmax, "independent solutions"))
```

```
[1] "calculated 500 independent solutions"
```

```
cfs <- S %>%
    mutate(sscore = round(sscore,2))%>%
    group_by(sscore) %>%
    summarize(n=n()) %>%
    mutate(relative_frequency = n / sum(n))

print(paste("there are " ,nrow(cfs), "different configurations"))
```

```
[1] "there are  14 different configurations"
```

```
cfs %>% ggplot(aes(x = sscore, y = n))+geom_col()
```

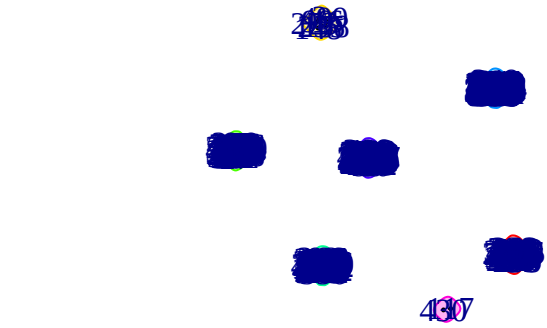


Results vary at each execution, hence we need to explore solutions space

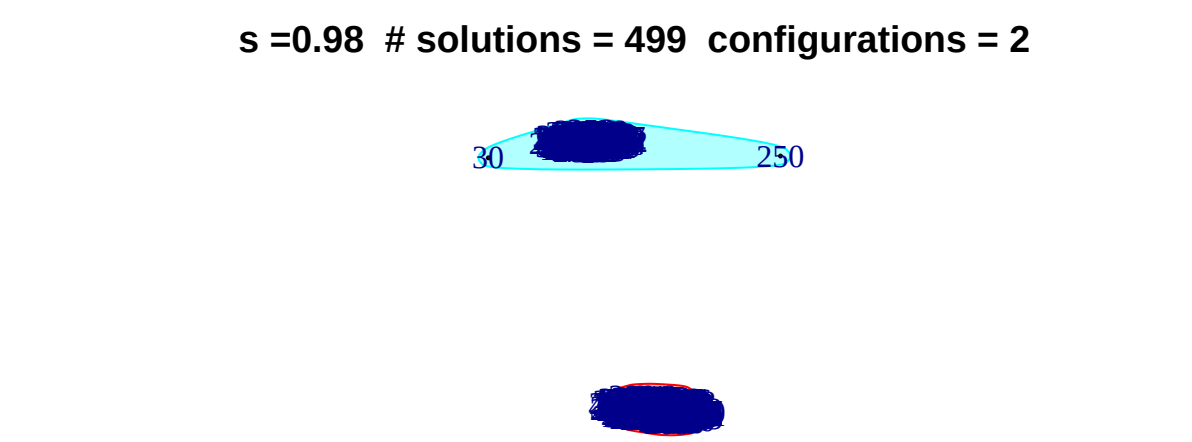
```
sspace <- S %>%
    mutate(weight = round(sscore,2)) %>%
    select(i,j,weight) %>%
    igraph::graph_from_data_frame(directed = FALSE)


cfs <- cfs %>% arrange(-sscore)

for (s in cfs$sscore){
    print(s)
    sspp <- delete_edges(sspace, E(sspace)[edge_attr(sspace)$weight < s])
    sspp <- delete_vertices(sspp, which(degree(sspp) == 0))

    configs <- cluster_label_prop(sspp)
    title <- paste0("s =", s, "  # solutions = ", vcount(sspp), "  configurations = ", max

    plot(configs, sspp, layout = layout.fruchterman.reingold(sspp),
     vertex.size = 2, , edge.width = NA, main = title)
```

```
}
```

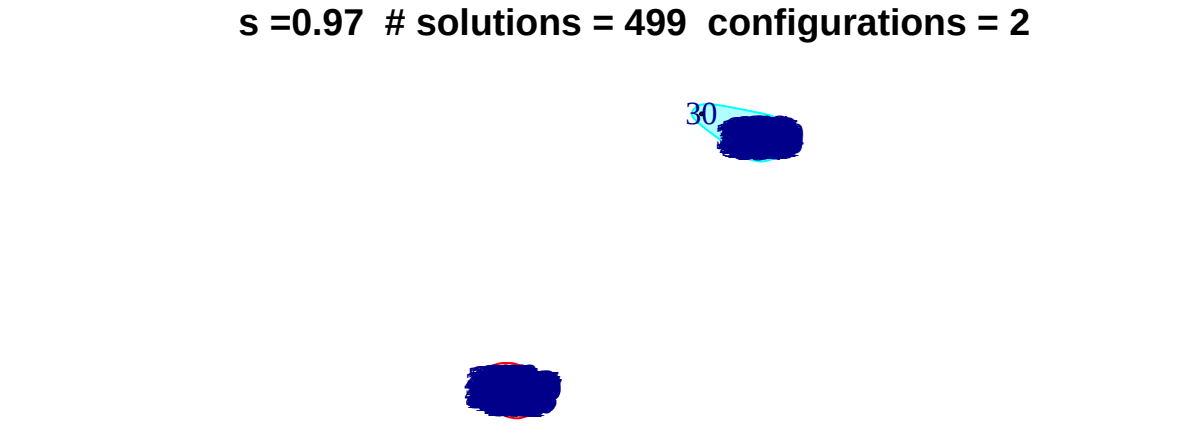
```
[1] 1
```



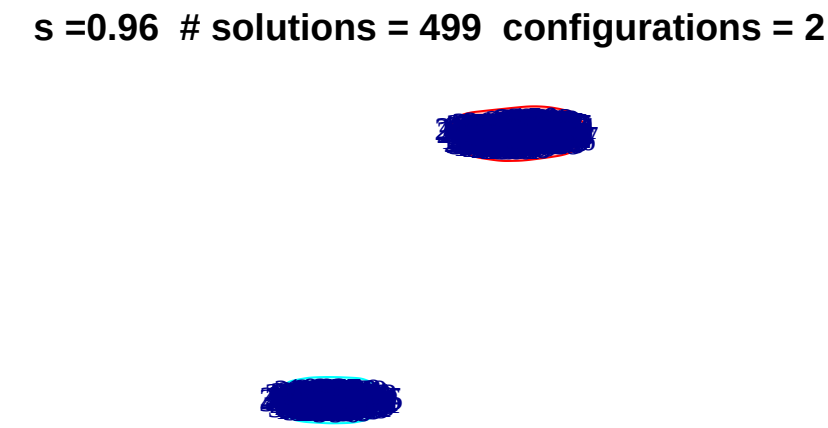

```
[1] 0.98
```


s =0.98 # solutions = 499 configurations = 2
30
250

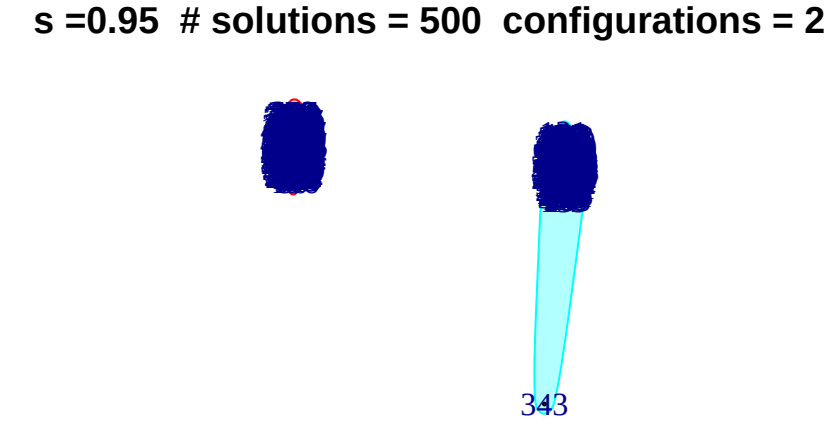
```
[1] 0.97
```


s =0.97 # solutions = 499 configurations = 2
30

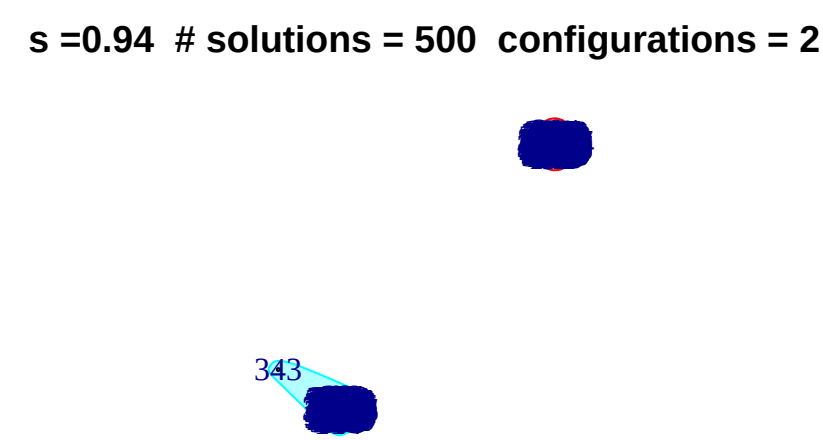
```
[1] 0.96
```


s =0.96 # solutions = 499 configurations = 2

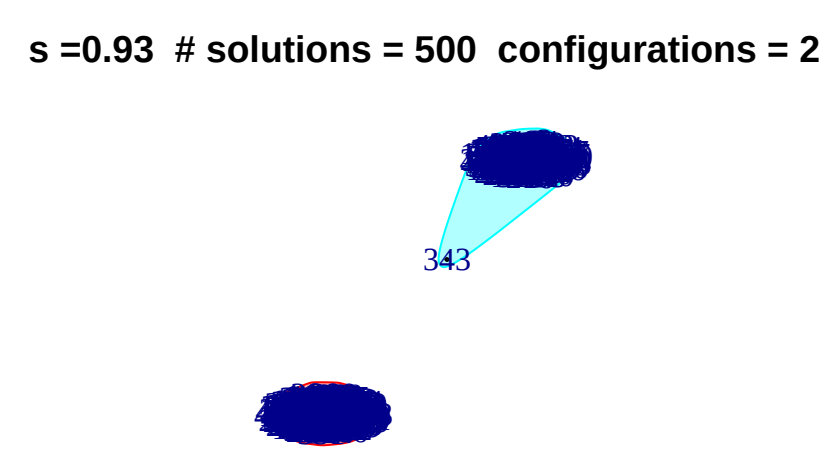
```
[1] 0.95
```


s =0.95 # solutions = 500 configurations = 2
343

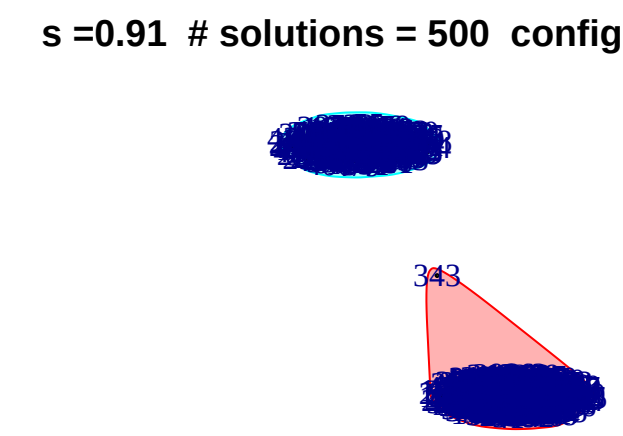
```
[1] 0.94
```


s =0.94 # solutions = 500 configurations = 2
343

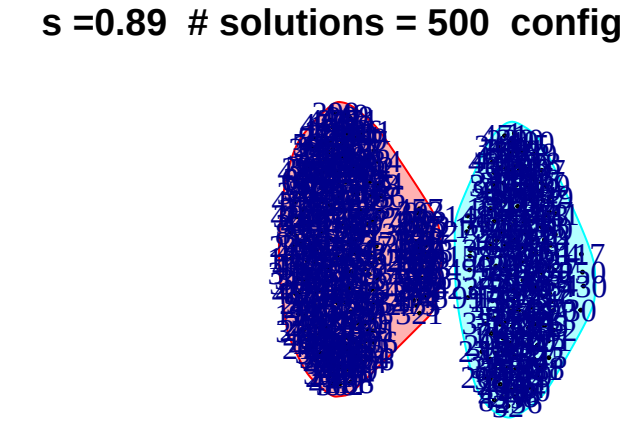
```
[1] 0.93
```


s =0.93 # solutions = 500 configurations = 2
343

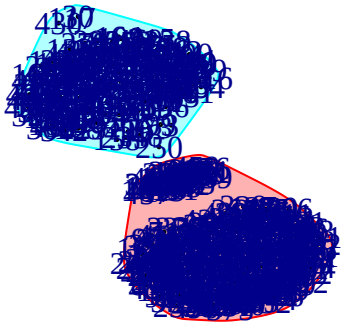
```
[1] 0.91
```


s =0.91 # solutions = 500 configurations = 2

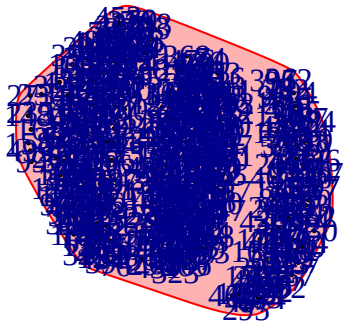
```
[1] 0.89
```


s =0.89 # solutions = 500 configurations = 2

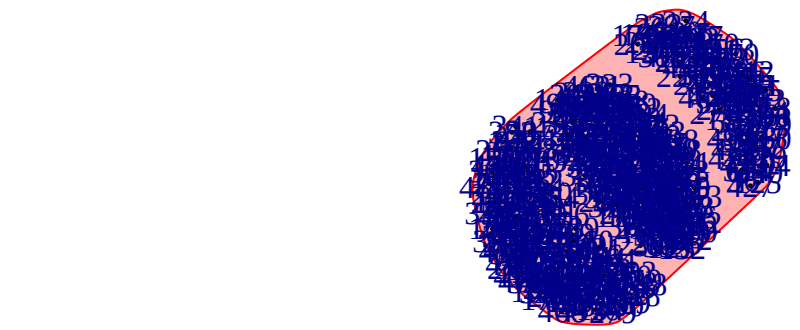
```
[1] 0.88
```


s =0.88 # solutions = 500 configurations = 2

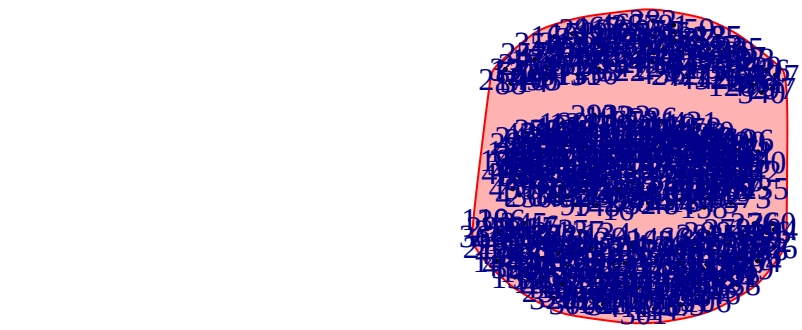
```
[1] 0.87
```


s =0.87 # solutions = 500 configurations = 1

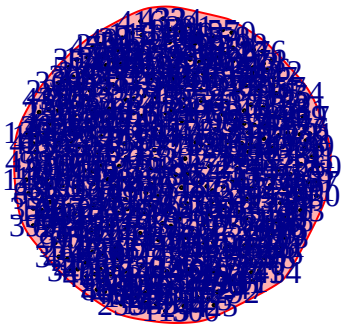
```
[1] 0.86
```


s =0.86 # solutions = 500 configurations = 1

```
[1] 0.85
```


s =0.85 # solutions = 500 configurations = 1

```
[1] 0.84
```



sspace <- S %>%

mutate(weight = round(sscore,2)) %>%

select(i,j,weight) %>%

filter(weight == 1)

```
sspp <- S %>%
    mutate(weight = round(sscore,2)) %>%
    select(i,j,weight) %>%
    filter(weight == 1)%>%
    igraph::graph_from_data_frame(directed = FALSE)


#sspp <- delete_edges(sspace, E(sspace)[edge_attr(sspace)$weight < 1])
    #sspp <- delete_vertices(sspp, which(degree(sspp) == 0))

    configs <- infomap.community(sspp)

    df <- data.frame(nn = configs$names,
                     cc = configs$membership )
```

```
    df <- df %>%
  group_by(cc) %>%
  mutate(csize = n()) %>%
        arrange(-csize)

    for (i in unique(df$cc)){
      recurring_config <- df$nn %>% head(1) %>% as.integer()
      x <- sum(df$cc == i)

      r_memberships <- M[,recurring_config]
      print(paste("configuration ", i, " found ", x, "times and has", max(r_memberships),

    }
```

```
[1] "configuration  3  found  176 times and has 9 communities"
[1] "configuration  6  found  106 times and has 9 communities"
[1] "configuration  4  found  90 times and has 9 communities"
[1] "configuration  2  found  67 times and has 9 communities"
[1] "configuration  1  found  48 times and has 9 communities"
[1] "configuration  5  found  8 times and has 9 communities"
[1] "configuration  7  found  2 times and has 9 communities"
```

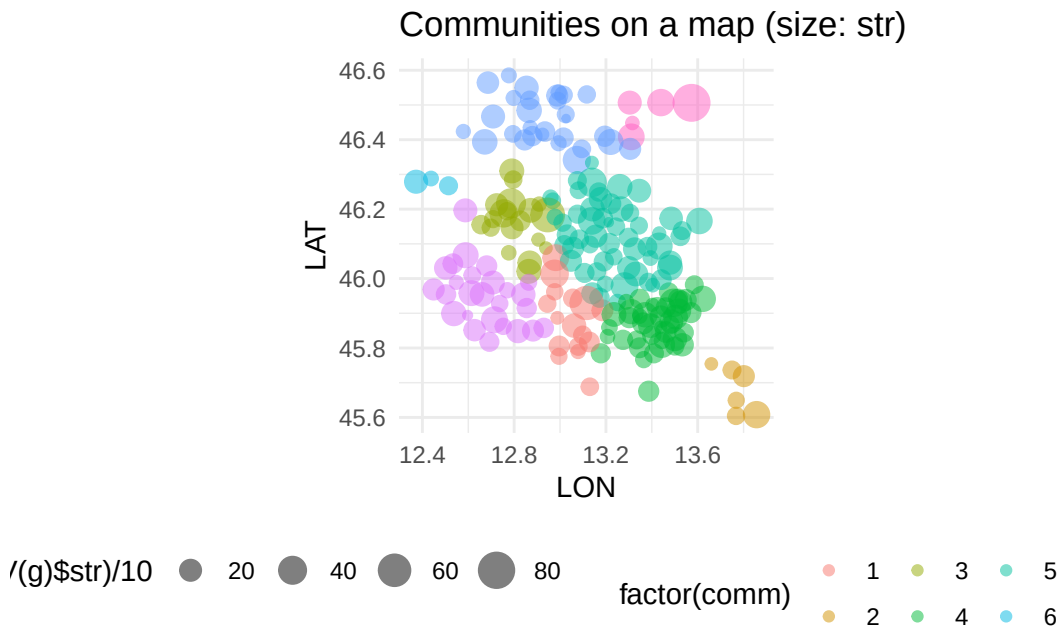
```
    leading_solution = df$nn %>% head(1) %>% as.integer()
    leading_membership <- M[,leading_solution]

    V(g)$stable_membership <- M[,leading_solution]
```

```
commdf <- data.frame(loc = V(g)$name,
                     comm = M[,leading_solution] )

locs <- read_csv('./data/locations.csv') %>%
  left_join(commdf) %>%
    filter(!is.na(comm))
```

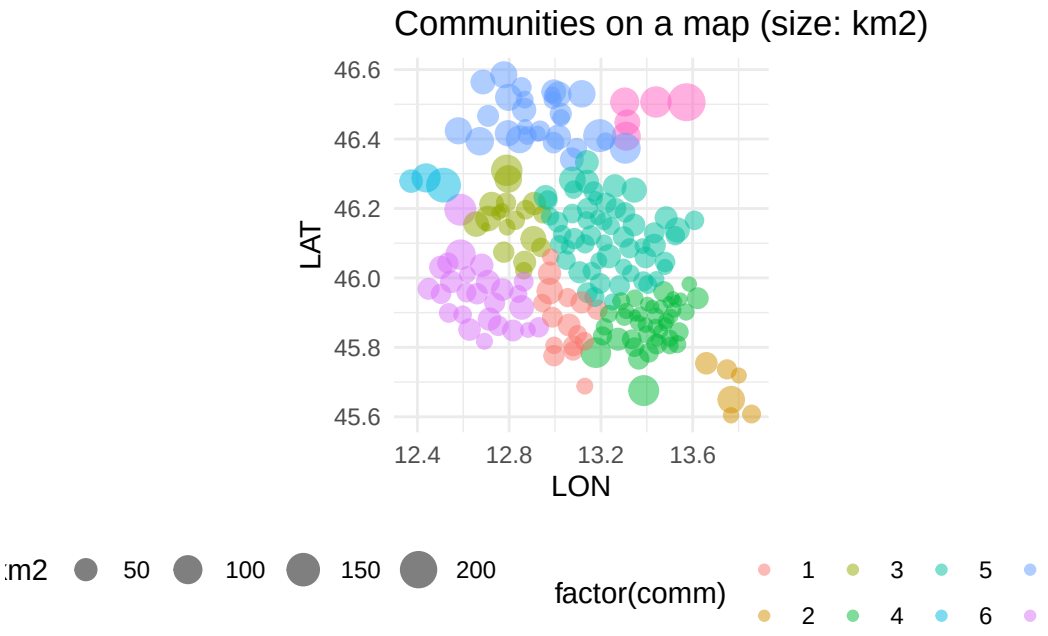
```
Rows: 203 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): location, loc, prov, codISTAT, codCATASTALE
dbl (3): LAT, LON, km2
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Joining with `by = join_by(loc)`
```

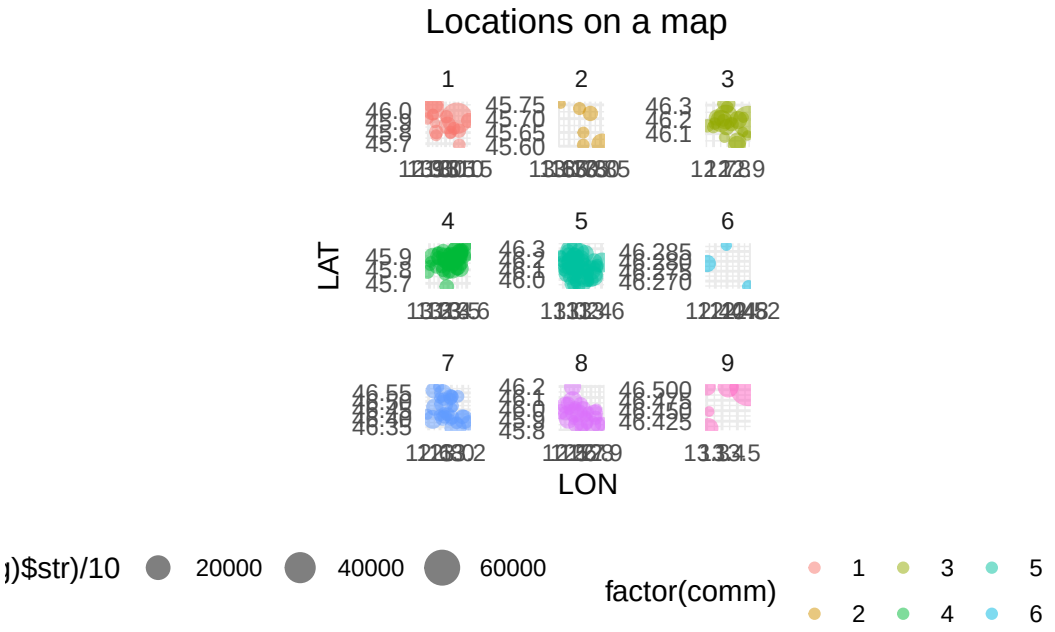
```
locs %>%
  ggplot(aes(x = LON, y = LAT)) +
  geom_point(aes(size = sqrt(V(g)$str)/10, color = factor(comm)), alpha = 0.5) +
  theme_minimal() +
theme(legend.position = 'bottom')+
  theme(aspect.ratio = 1)+
  ggtitle("Communities on a map (size: str)")
```




```
locs %>%
  ggplot(aes(x = LON, y = LAT)) +
  geom_point(aes(size = km2, color = factor(comm)), alpha = 0.5) +
  theme_minimal() +
theme(legend.position = 'bottom')+
  theme(aspect.ratio = 1)+
  ggtitle("Communities on a map (size: km2)")
```




```
locs %>%
  ggplot(aes(x = LON, y = LAT)) +
  geom_point(aes(size = (V(g)$str)/10, color = factor(comm)), alpha = 0.5) +
  #geom_text(aes(label = loc), nudge_y = 0.1) + # Add annotations
  theme_minimal() +
theme(legend.position = 'bottom')+
  theme(aspect.ratio = 1)+
  ggtitle("Locations on a map")+

  facet_wrap(~ comm, scales = 'free')
```



check communities

```
V(g)$community <- M[,leading_solution]

comms$membership <- M[,leading_solution]
V(g)$clabels <- communities::comm_label_as_strongest(g, comms)
```

```
[1] "Codroipo"
[1] "Trieste"
[1] "Spilimber"
[1] "Monfalcone"
[1] "Udine"
[1] "Claut"
[1] "Tolmezzo"
[1] "Pordenone"
[1] "Tarvisio"
```

```
print(unique(V(g)$clabels))
```

```
[1] "C_Monfalcone" "C_Tolmezzo"   "C_Pordenone"  "C_Spilimber"  "C_Udine"
[6] "C_Codroipo"   "C_Tarvisio"   "C_Claut"      "C_Trieste"
```

```
  E(g)$w <- E(g)$weight
  V(g)$community <-V(g)$clabels
  gc <- communities::make_community_network(g)
  gc
```

```
IGRAPH dd70904 UNW- 9 79 --
+ attr: name (v/c), id (v/n), size (v/n), weight (e/n)
+ edges from dd70904 (vertex names):
 [1] C_Claut     --C_Claut      C_Claut     --C_Codroipo
 [3] C_Claut     --C_Monfalcone C_Claut     --C_Pordenone
 [5] C_Claut     --C_Spilimber  C_Claut     --C_Tarvisio
 [7] C_Claut     --C_Tolmezzo   C_Claut     --C_Trieste
 [9] C_Claut     --C_Udine      C_Claut     --C_Codroipo
[11] C_Codroipo  --C_Codroipo   C_Codroipo  --C_Monfalcone
[13] C_Codroipo  --C_Pordenone  C_Codroipo  --C_Spilimber
[15] C_Codroipo  --C_Tarvisio   C_Codroipo  --C_Tolmezzo
+ ... omitted several edges
```

```
  edges_gc<-as_long_data_frame(gc) %>%
      select(from_name, to_name, weight) %>%
      mutate(intra_weight = if_else(from_name == to_name, 0,weight )) %>%
      mutate(self_weigth = if_else(from_name == to_name, weight, 0 )) %>%
      arrange(-weight)

  # mixing parameter of the partition
  mu = sum(edges_gc$intra_weight)/sum(edges_gc$weight)
  print(edges_gc)
```

```
      from_name      to_name    weight intra_weight self_weigth
1       C_Udine      C_Udine 1051876.5          0.0   1051876.5
2   C_Pordenone  C_Pordenone  856470.5          0.0    856470.5
3  C_Monfalcone C_Monfalcone  578713.5          0.0    578713.5
4     C_Trieste    C_Trieste  271212.0          0.0    271212.0
5  C_Monfalcone      C_Udine  154492.0     154492.0         0.0
6    C_Codroipo   C_Codroipo  118502.5          0.0    118502.5
7  C_Monfalcone    C_Trieste  104780.0     104780.0         0.0
8    C_Codroipo      C_Udine   93564.0      93564.0         0.0
9   C_Spilimber  C_Spilimber   81185.0          0.0     81185.0
10 C_Monfalcone      C_Udine   73346.5      73346.5         0.0
11   C_Tolmezzo   C_Tolmezzo   65209.0          0.0     65209.0
```

12	C_Pordenone	C_Spilimber	49518.0	49518.0	0.0
13	C_Codroipo	C_Udine	47625.5	47625.5	0.0
14	C_Pordenone	C_Spilimber	38510.0	38510.0	0.0
15	C_Codroipo	C_Pordenone	34689.0	34689.0	0.0
16	C_Monfalcone	C_Trieste	34153.0	34153.0	0.0
17	C_Pordenone	C_Udine	33216.0	33216.0	0.0
18	C_Codroipo	C_Monfalcone	31161.5	31161.5	0.0
19	C_Spilimber	C_Udine	25421.5	25421.5	0.0
20	C_Tolmezzo	C_Udine	24772.5	24772.5	0.0
21	C_Codroipo	C_Monfalcone	24193.0	24193.0	0.0
22	C_Codroipo	C_Pordenone	20569.0	20569.0	0.0
23	C_Tolmezzo	C_Udine	20267.0	20267.0	0.0
24	C_Spilimber	C_Udine	19893.5	19893.5	0.0
25	C_Trieste	C_Udine	16776.5	16776.5	0.0
26	C_Pordenone	C_Udine	15021.5	15021.5	0.0
27	C_Trieste	C_Udine	14719.0	14719.0	0.0
28	C_Tarvisio	C_Tarvisio	10987.5	0.0	10987.5
29	C_Monfalcone	C_Pordenone	9029.5	9029.5	0.0
30	C_Codroipo	C_Trieste	8548.0	8548.0	0.0
31	C_Monfalcone	C_Pordenone	7825.0	7825.0	0.0
32	C_Pordenone	C_Trieste	7728.5	7728.5	0.0
33	C_Tarvisio	C_Udine	7656.0	7656.0	0.0
34	C_Codroipo	C_Spilimber	7464.0	7464.0	0.0
35	C_Tarvisio	C_Udine	5049.0	5049.0	0.0
36	C_Tarvisio	C_Tolmezzo	3819.0	3819.0	0.0
37	C_Codroipo	C_Spilimber	2729.5	2729.5	0.0
38	C_Monfalcone	C_Tolmezzo	2423.0	2423.0	0.0
39	C_Tarvisio	C_Tolmezzo	2274.5	2274.5	0.0
40	C_Tolmezzo	C_Trieste	2168.0	2168.0	0.0
41	C_Monfalcone	C_Spilimber	2007.0	2007.0	0.0
42	C_Codroipo	C_Tolmezzo	1918.0	1918.0	0.0
43	C_Tarvisio	C_Trieste	1570.5	1570.5	0.0
44	C_Monfalcone	C_Tarvisio	1566.5	1566.5	0.0
45	C_Claut	C_Pordenone	1552.5	1552.5	0.0
46	C_Pordenone	C_Tolmezzo	1425.5	1425.5	0.0
47	C_Pordenone	C_Trieste	1269.5	1269.5	0.0
48	C_Claut	C_Spilimber	1265.0	1265.0	0.0
49	C_Monfalcone	C_Tolmezzo	1223.0	1223.0	0.0
50	C_Pordenone	C_Tarvisio	1035.0	1035.0	0.0
51	C_Monfalcone	C_Spilimber	1025.5	1025.5	0.0
52	C_Spilimber	C_Trieste	988.5	988.5	0.0
53	C_Pordenone	C_Tolmezzo	927.5	927.5	0.0
54	C_Codroipo	C_Tarvisio	787.5	787.5	0.0

```
55   C_Codroipo    C_Trieste     704.5        704.5         0.0
56      C_Claut      C_Claut     653.5          0.0       653.5
57   C_Codroipo   C_Tolmezzo     644.5        644.5         0.0
58  C_Spilimber   C_Tolmezzo     599.0        599.0         0.0
59   C_Tolmezzo    C_Trieste     580.0        580.0         0.0
60      C_Claut  C_Pordenone     501.0        501.0         0.0
61 C_Monfalcone   C_Tarvisio     321.5        321.5         0.0
62  C_Pordenone   C_Tarvisio     312.0        312.0         0.0
63  C_Spilimber   C_Tarvisio     308.5        308.5         0.0
64  C_Spilimber    C_Trieste     275.0        275.0         0.0
65  C_Spilimber   C_Tolmezzo     269.5        269.5         0.0
66      C_Claut      C_Udine     177.5        177.5         0.0
67   C_Tarvisio    C_Trieste     134.0        134.0         0.0
68   C_Codroipo   C_Tarvisio     120.5        120.5         0.0
69  C_Spilimber   C_Tarvisio     100.0        100.0         0.0
70      C_Claut    C_Trieste      57.5         57.5         0.0
71      C_Claut   C_Tolmezzo      45.5         45.5         0.0
72      C_Claut C_Monfalcone      33.5         33.5         0.0
73      C_Claut  C_Spilimber      32.0         32.0         0.0
74      C_Claut      C_Udine      31.0         31.0         0.0
75      C_Claut   C_Codroipo      22.0         22.0         0.0
76      C_Claut   C_Codroipo      12.0         12.0         0.0
77      C_Claut C_Monfalcone      10.5         10.5         0.0
78      C_Claut   C_Tarvisio       6.0          6.0         0.0
79      C_Claut   C_Tolmezzo       5.5          5.5         0.0
```

```
print(mu)
```

```
[1] 0.2416913
```

```
names <- unique(c(edges_gc$from_name, edges_gc$to_name))


 comm_df <- tibble(name = names) %>%
  mutate(total_weight = map_dbl(name,
      ~ sum(edges_gc$weight[edges_gc$from == .x | edges_gc$to == .x])))%>%
  mutate(intra_weight = map_dbl(name,
      ~ sum(edges_gc$intra_weight[edges_gc$from == .x | edges_gc$to == .x])))%>%
     mutate(self_weight = total_weight - intra_weight) %>%
     mutate(x = self_weight / intra_weight) %>%
```

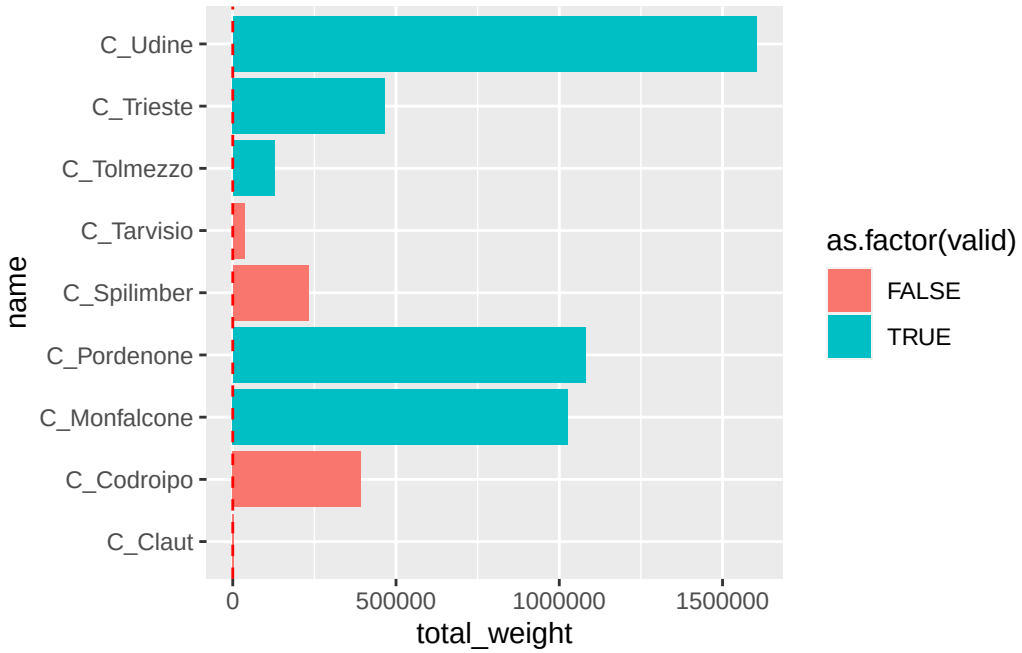
```
    mutate(valid = (self_weight > intra_weight))%>%
  arrange(-total_weight)

 print(comm_df %>% filter(valid == TRUE))
```

```
# A tibble: 5 x 6
  name         total_weight intra_weight self_weight     x valid
  <chr>               <dbl>        <dbl>       <dbl> <dbl> <lgl>
1 C_Udine           1603906.      552029     1051876.  1.91 TRUE
2 C_Pordenone       1079600       223130.     856470.  3.84 TRUE
3 C_Monfalcone      1026304.      447591      578714.  1.29 TRUE
4 C_Trieste          465664.      194452.     271212   1.39 TRUE
5 C_Tolmezzo         128571        63362       65209   1.03 TRUE
```

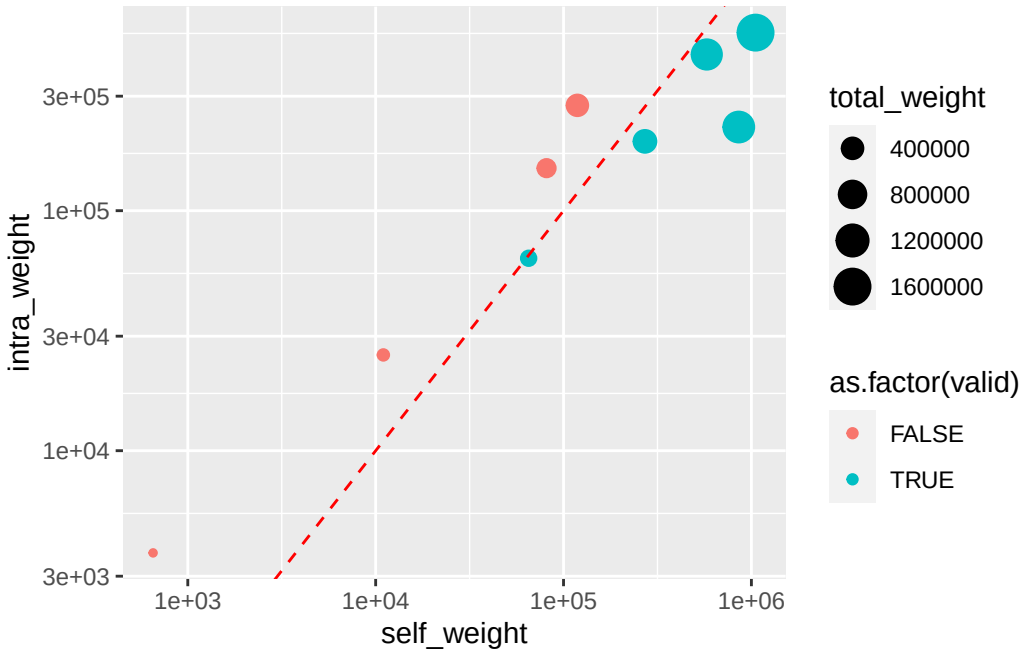
```
 print(comm_df %>% filter(valid == FALSE))
```

```
# A tibble: 4 x 6
  name        total_weight intra_weight self_weight     x valid
  <chr>              <dbl>        <dbl>       <dbl> <dbl> <lgl>
1 C_Codroipo       393255       274752.     118502. 0.431 FALSE
2 C_Spilimber      231592.      150406.      81185  0.540 FALSE
3 C_Tarvisio        36048        25060.      10988. 0.438 FALSE
4 C_Claut            4405         3752.        654. 0.174 FALSE
```

```
 comm_df %>% ggplot(aes(y = name, x = total_weight, fill = as.factor(valid)))+
    geom_col()+
    geom_vline(xintercept = 0.5, linetype = "dashed", color = 'red')
```




```
comm_df %>% ggplot(aes(x = self_weight, y = intra_weight, fill = as.factor(valid)))+
    geom_point(aes(size = total_weight, color = as.factor(valid)))+
    geom_abline(intercept = 0.0, slope = 1, linetype = "dashed", color = 'red')+
  scale_x_log10()+ scale_y_log10()
```


intra_weight
3e+05
1e+05
3e+04
1e+04
3e+03
1e+03
1e+04
1e+05
1e+06
self_weight
total_weight
400000
800000
1200000
1600000
as.factor(valid)
FALSE
TRUE