Vertex Nomination via Seeded Graph Matching

Carey E. Priebe, Youngser Park, Heather Patsolic, Vince Lyzinski Johns Hopkins University

2017-09-17

1 Background

- A short summary of the methodology is here.
- The latest draft of our paper is ~~here~~ (unlinked for the future newer version).
- The poster for SIAMNS16 is here.
- The slide set for JSM2016 is here.

Here we describe our approach to both the simulations and the illustrative experiments, which allows the same code to be used to address a real problem in anger (when we don't know any truth except the VOI x and some seeds $S \leftrightarrow S'$).
If it's a simulation or illustrative experiment: (1) generate G and G' with some shared vertices and some unshared vertices, or start with real data G and G' with a collection of known shared vertices and some unknown or unshared vertices, and (2) randomly pick VOI x and some number of seeds $S \leftrightarrow S'$ from amongst the shared vertices; then embark on our procedure described below. If it's a real problem, with given VOI x and some seeds $S \leftrightarrow S'$, then embark immediately on our procedure described below.

2 Toy Example

input: G, G', seedset $S \leftrightarrow S'$ (pairs of vertices one in G & one in G'), x (vertex of interest in G), $h \leq \ell$.
output: list of (`candidate`,`probability`) where
- `candidates` are non-seed vertices in G' and
- `probability` is for nomination as match for x.

This toy example follows these steps:

1. build a pair of ρ-correlated RDPG random graphs, $G(V, E)$ & $G'(V', E')$, where $|V| = |V'| = 30$
2. remove the last 5 vertices from G' to make $|V(G)| \geq |V(G')|$
3. randomly pick VOI x and s seeds from amongst the shared vertices
4. find S_x, all seeds in $N_h(x)$ in G, and matching S'_x in G', let $s_x = |S_x| = |S'_x|$, if $s_x = 0$, then "`impossible1`"
5. let $C'_x = N_\ell(S'_x)$ be the *candidates* for the match x' to the VOI x for $\ell \geq h$, if $x' \notin C'_x$, then "`impossible2`"
6. find $G_x = \Omega(N_\ell[S_x])$ and $G'_x = \Omega(N_\ell[S'_x])$
7. do `SGM`$(G_x, G'_x, S_x \leftrightarrow S'_x)$ which returns $P = |V_x| \times |V'_x|$ matching probability matrix
8. find $\hat{x}' = \arg\max_{v \in C'_x} P[x, v]$

So now we do `SGM`$(G_x, G'_x, S_x \leftrightarrow S'_x)$ – a smaller SGM problem. (NB: original is this with $h = \infty$.)

NB: Steps 4-8 are the same for simulation, illustrative experiment, and real application.

```
suppressMessages(require(VN))
suppressMessages(require(igraph))

sim <- TRUE # if TRUE, run simulation, otherwise do the HS experiment
HS <- "full" # or "core"
```

```r
# parameters for finding seeds
s <- ifelse(sim,4,12) # number of seeds to be used for SGM
h <- ell <- 1 # max walk for finding neighborhoods

# parameters for SGM
R <- 100     # repeat SGM R times to get averaged P matrix
gamma <- 0.1 # number of iterations for the Frank-Wolfe algorithm

mc <- 2
set.seed(1234+mc)

if (sim) {
    # generate a pair of correlated graphs
    m <- 5  # |J| = junk on G1
    n <- 20 # |W| = shared vertices on G1, not including x and S
    mp <- 0 # |J'| = junk on G2
    d <- 5  # for RDPG, dimension of the random vectors
    corr <- 0.5 # for correlated graphs

    (nV1 <- 1+s+n+m)
    (nV2 <- 1+s+n+mp)
    lpvs <- sample_sphere_surface(dim=d, n=nV1)/1.5 # random vectors for RDPG
    gg <- rdpg.sample.correlated(t(lpvs),corr)
    g1 <- gg[[1]];
    g2 <- gg[[2]];
    g2 <- delete_vertices(g2,v=(nV2+1):nV1); # remove m vertices from G' to make |V(G)| != |V(G')|
    W <- intersect(V(g1),V(g2)) # shared vertices
} else {
    data("HSgraphs")
    if (HS == "full") {
        # rearrange the vertices so that the to-be-selected seeds are valid
        perm.fb <- c(coremap[,1],setdiff(1:vcount(HSfbgraphfull),coremap[,1]))
        perm.fr <- c(coremap[,2],setdiff(1:vcount(HSfriendsgraphfull),coremap[,2]))
        g1 <- permute.vertices(HSfbgraphfull,perm.fb) # (156,1437)
        g2 <- permute.vertices(HSfriendsgraphfull,perm.fr) # (134,668)
        W <- 1:nrow(coremap) # seeds should be selected from the shared (core) vertices
    } else { # core graphs
        g1 <- HSfbgraphcore # (82,513)
        g2 <- HSfrgraphcore # (82,214)
        W <- intersect(V(g1),V(g2)) # shared vertices
    }
}
```

```r
# Randomly select x and S from W, the shared vertices
(x <- sample(W,1))
```

```
# [1] 22
```

```r
W <- setdiff(W,x) # exclude x from W
maxseed <- min(length(W),s)
(S <- sort(sample(W,maxseed)))
```

```
# [1]  1  8 18 25
```

```r
# Determine Sx and C'x, then do SGM
NBDS <- vnsgm(x,S,g1,g2,h,ell,R,gamma,plotF=TRUE)
```

```r
str(NBDS)
```

```
# List of 9
#  $ case     : chr "possible"
#  $ x        : int 22
#  $ S        : int [1:4] 1 8 18 25
#  $ Sx       : int [1:2] 18 25
#  $ Sxp      : int [1:2] 18 25
#  $ Cxp      : int [1:10] 1 4 5 6 9 10 11 14 15 22
#  $ labelsGx : int [1:10] 18 25 22 6 11 13 14 15 23 27
#  $ labelsGxp: int [1:12] 18 25 22 1 4 5 6 9 10 11 ...
#  $ P        : num [1:12, 1:12] 1 0 0 0 0 0 0 0 0 0 ...
```

```r
# Determine x' amongst the candidates based on the matching probability from SGM
Sx <- NBDS$Sx
x <- NBDS$labelsGxp[length(Sx)+1]
x.ind <- which(NBDS$labelsGx==x)
Cxp <- match(NBDS$Cxp,NBDS$labelsGxp)

if (NBDS$case=="possible") {
    prob <- NBDS$P[x.ind,Cxp]
    names(prob) <- NBDS$labelsGxp[Cxp]
    x.prob <- prob[which.max(prob)]
    vhatstar <- as.integer(names(x.prob))
    rank.prob <- rank(-prob,ties.method = "average")
    plot(as.integer(names(prob)), prob, type="h",col=2, lwd=2)
    rank.prob <- matrix(rank.prob,nrow=1); colnames(rank.prob) <- paste0("V",names(prob))
    kable(rank.prob,caption="Rank of matching probability for candidates")
}
```

Table 1: Rank of matching probability for candidates

V1	V4	V5	V6	V9	V10	V11	V14	V15	V22
5	7	9	2	9	6	4	9	3	1

3 Simulation

We repeat the above 1000 times by generating new graphs each time, as well as new x and S.

We define the normalized rank of the VOI x in G' with respect to the size of the candidate list C'_x as follows,

$$normalized\ rank = \frac{rank(x') - 1}{|C'_x| - 1},$$

so that values of 0, 0.5, and 1 imply that the VOI is first, half-way down, and last in the candidate list, respectively.

NB: In the tables and plots below,

- `s1` means the number of seeds for `SGM`, $s_x = 1$ and so on.

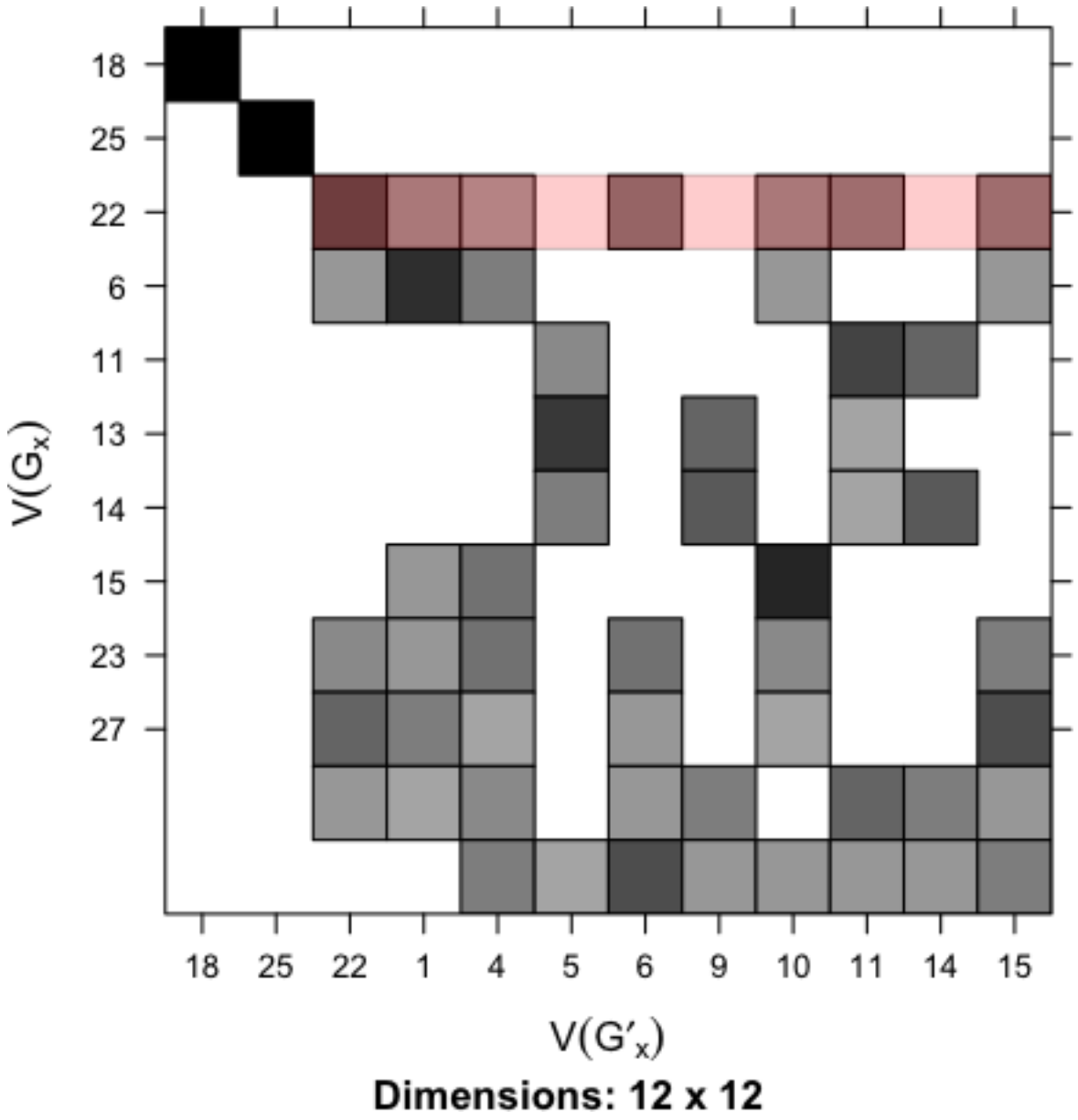


Figure 1: A submatrix of a probability matrix output from SGM. Rows correspond to vertices in $V(G_x)$ and columns correspond to vertices in $V(G'_x)$. The first 2 rows and columns correspond to the seeds (S_x, S'_x), The next row is x. The shaded area (in pink) depicts matching probabilities of vertices in C'_x against x in $V(G_x)$. The vertices with the highest probability among these *candidates* is nominated as our best guess for x'.

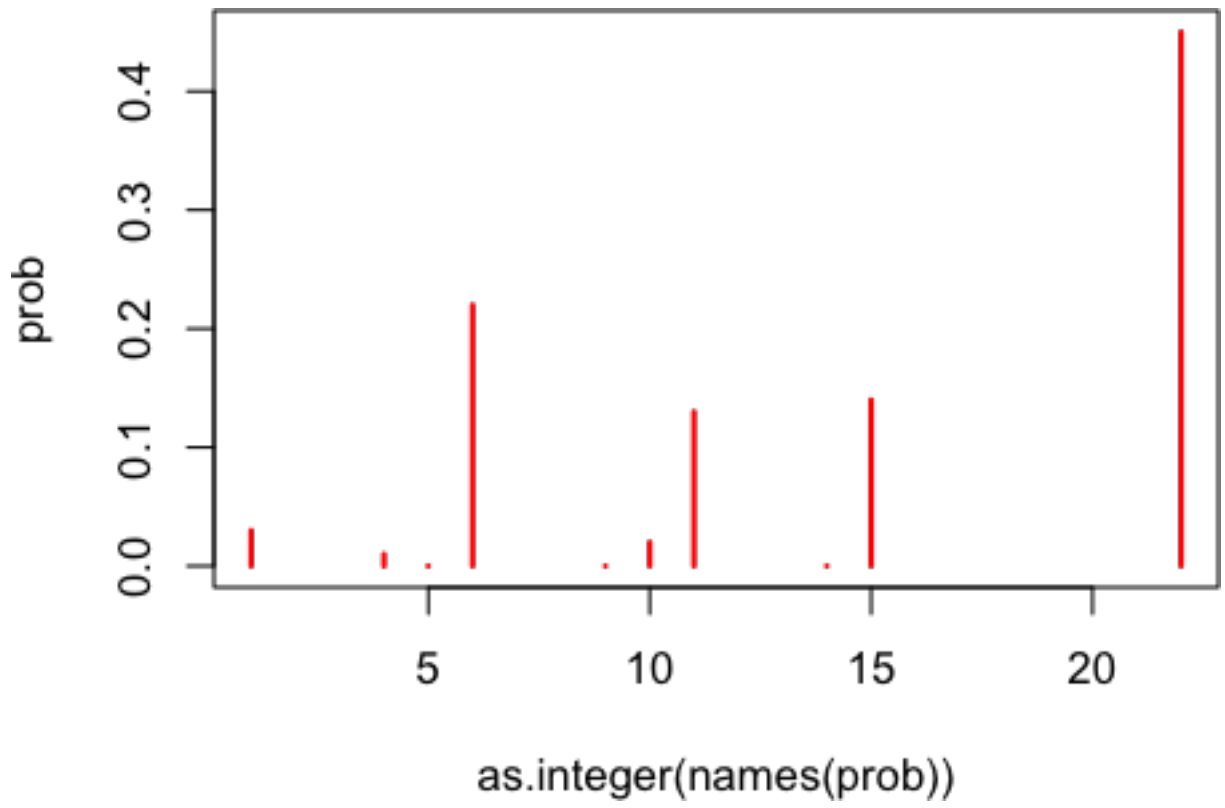


Figure 2: A bar plot of the matching probability of the candidates. The vertex 22 in G'_x has the highest probability so is nominated as x'.

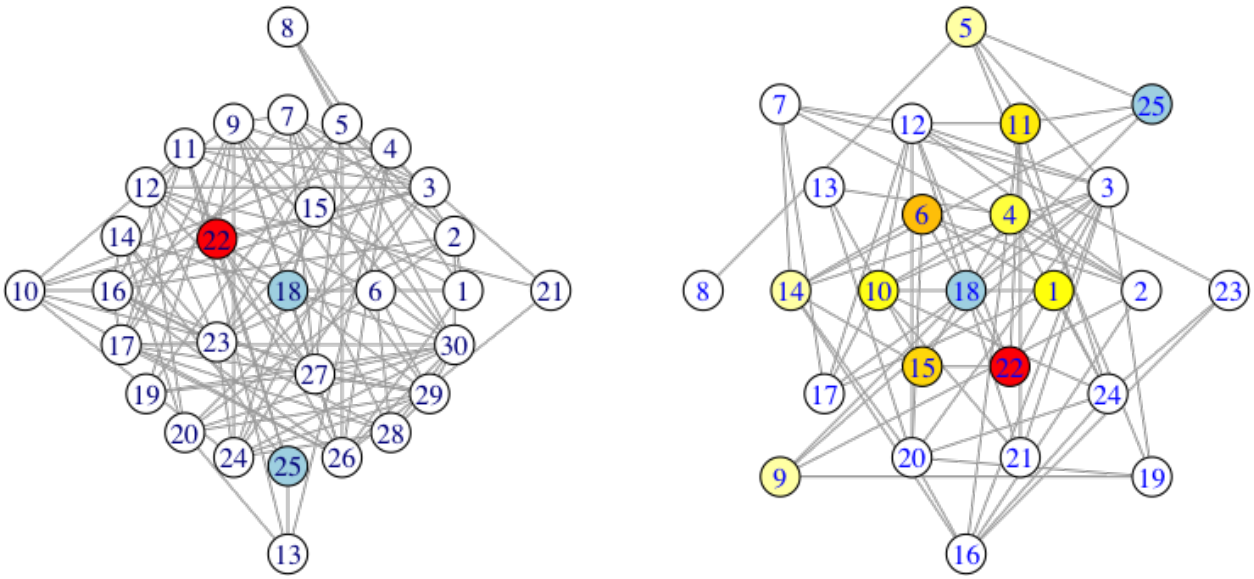


Figure 3: Plot of G (left) and G' (right), where the vertices are colored by red in G: x, cyan on both graphs: S_x, `heat.colors` in G': candidates (the more red the color is, the higher the rank of the candidate is). In both graphs, the center vertex is one of the seeds in S_x, its first neighbors are in the first ring, their first neighbors are in the next ring, and the rest are placed in the outmost ring.

- `mean.ncand` means averaged number of candidates for each case.
- `mean.nrank` means averaged normalized rank for each case.

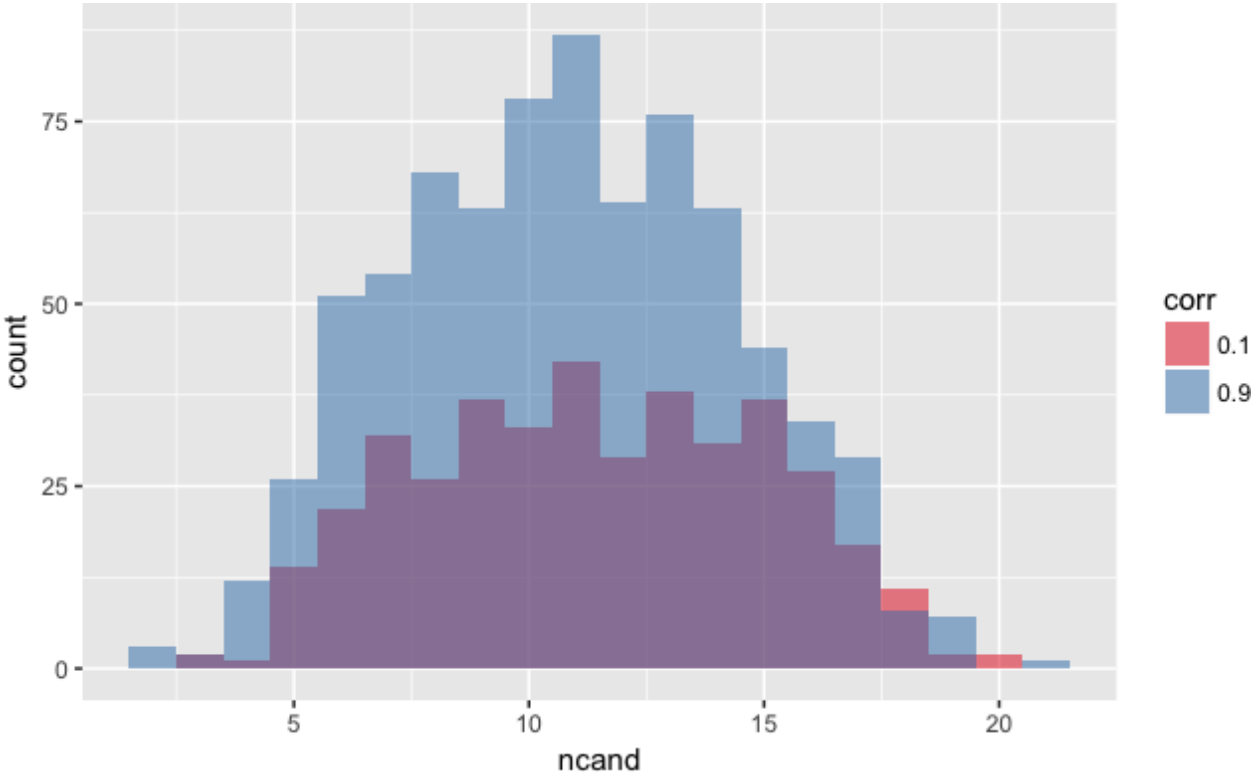
Table 2: A summary of the simulation for `corr=0.1`

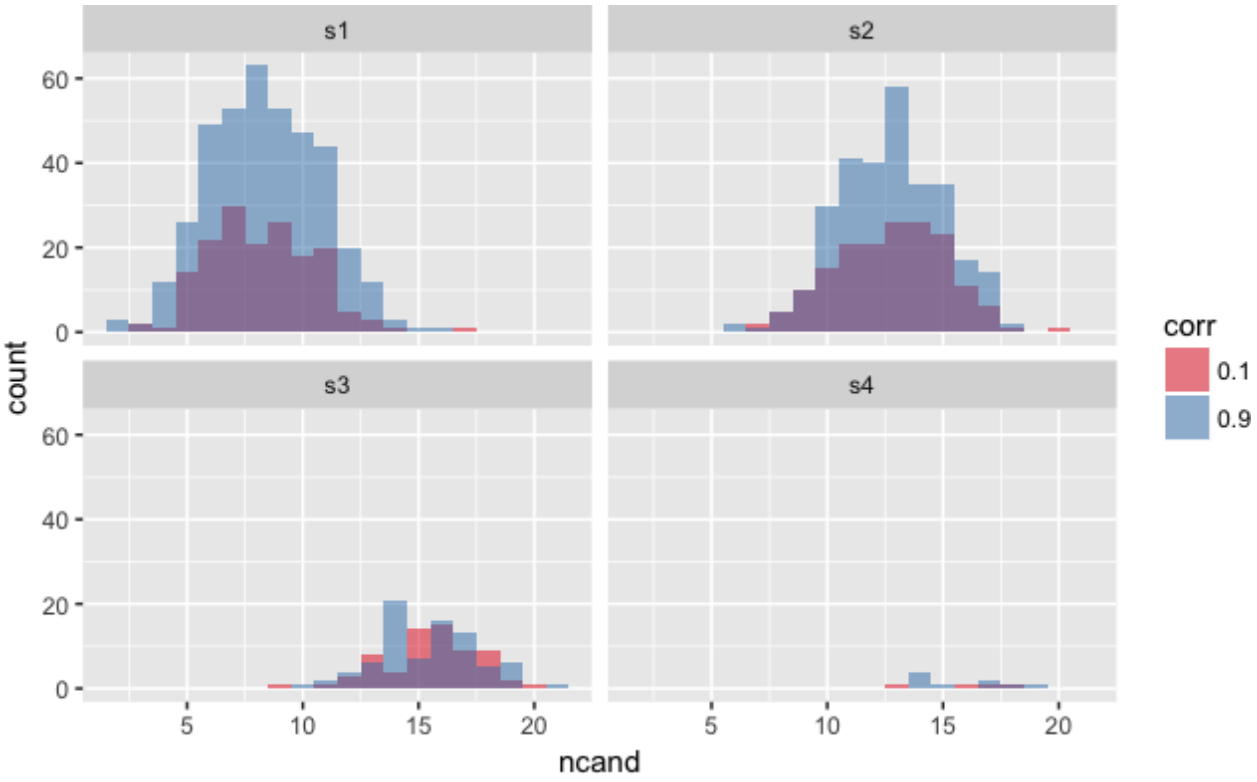
case	seed	count	mean.ncand	mean.nrank
impossible1	s0	232	0.0	NA
impossible2	s0	365	8.8	NA
possible	s1	164	8.3	0.46
possible	s2	168	12.7	0.47
possible	s3	67	15.5	0.42
possible	s4	4	16.0	0.36

Table 3: A summary of the simulation for `corr=0.9`

case	seed	count	mean.ncand	mean.nrank
impossible1	s0	198	0.0	NaN
impossible2	s0	32	7.6	NaN
possible	s1	389	8.4	0.17
possible	s2	290	12.7	0.03
possible	s3	82	15.3	0.01
possible	s4	9	15.8	0.00

3.1 Number of candidates?





3.2 Rank of x'?

4 Real Data

R. Mastrandrea, J. Fournet, and A. Barrat, *Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys*, PLoS ONE, 2015.

We look at two High School friendship networks on over-lapping vertex sets found in our draft. The first network consists of 134 vertices, each representing a particular student, in which two vertices are adjacent if one of the students reported or a survey that the two are friends. The second network, with 156 vertices, consists of a Facebook network of profiles in which two vertices are adjacent if they were friends on Facebook. There are 82 *core* vertices across the two networks for which we know the bijection between the two vertex sets, and it is known that no such bijection exists among the remaining vertices.

4.1 Full Graphs

First, we do the same experiment as above using the full graphs. We randomly select the VOI x and seeds S from the shared vertices (82 core vertices) and repeat for $MC = 1000$ times. Since we are choosing the VOI and seeds to be in the core vertex sets, the VOI will exist in the second network; it just may not exist in C'_x (the candidate set generated by the seeds) – i.e. while the VOI x is guaranteed to have a match x', this vertex will not be found if it is not also in C'_x (`impossible2`).

Table 4: A summary statistics using the full graphs

case	seed	count	mean.ncand	mean.nrank
impossible1	s0	230	0.0	NaN

case	seed	count	mean.ncand	mean.nrank
impossible2	s0	688	12.8	NaN
possible	s01	11	7.9	0.69
possible	s02	27	13.1	0.44
possible	s03	15	20.1	0.51
possible	s04	15	23.5	0.51
possible	s05	9	28.7	0.51
possible	s06	3	36.0	0.51
possible	s07	1	35.0	0.50
possible	s09	1	33.0	0.50

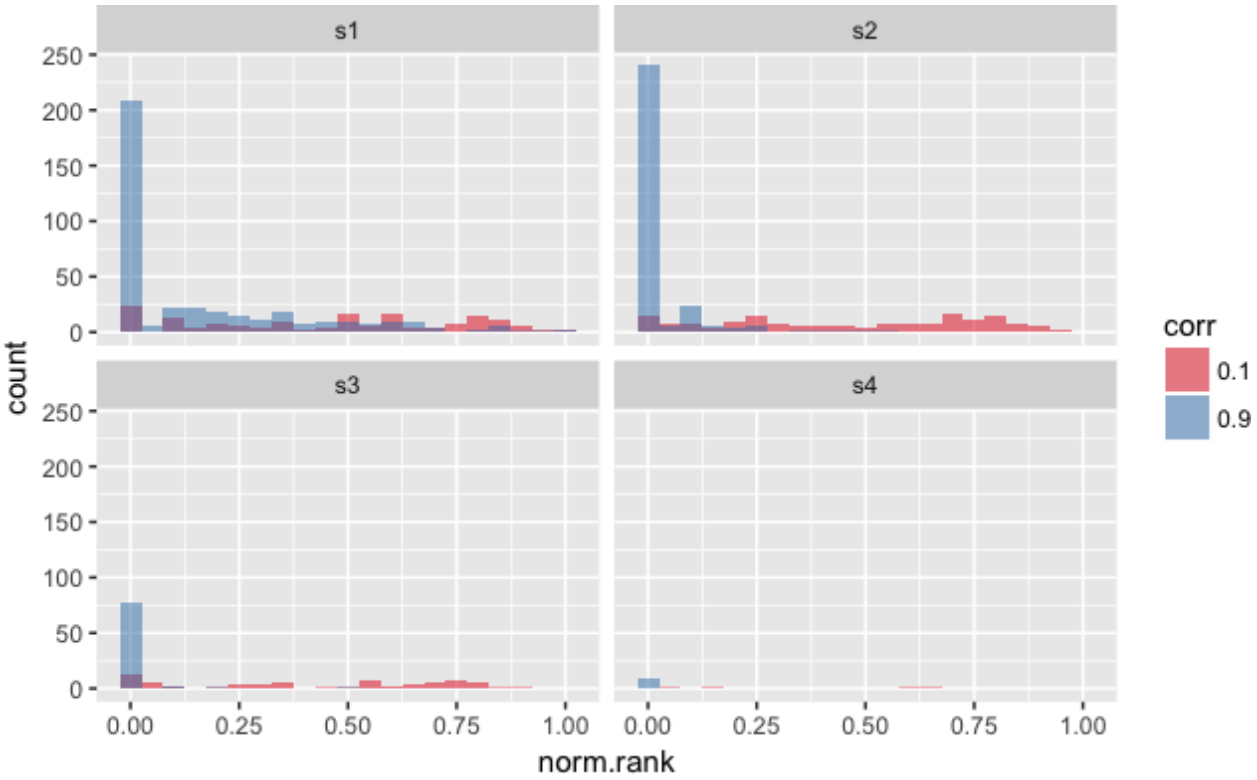


Figure 4: A histogram of the normalized rank of x' as a function of the number of seeds s_x.

Figure 5: A histogram of the normalized rank of x' as a function of the number of candidates and the number of seeds.

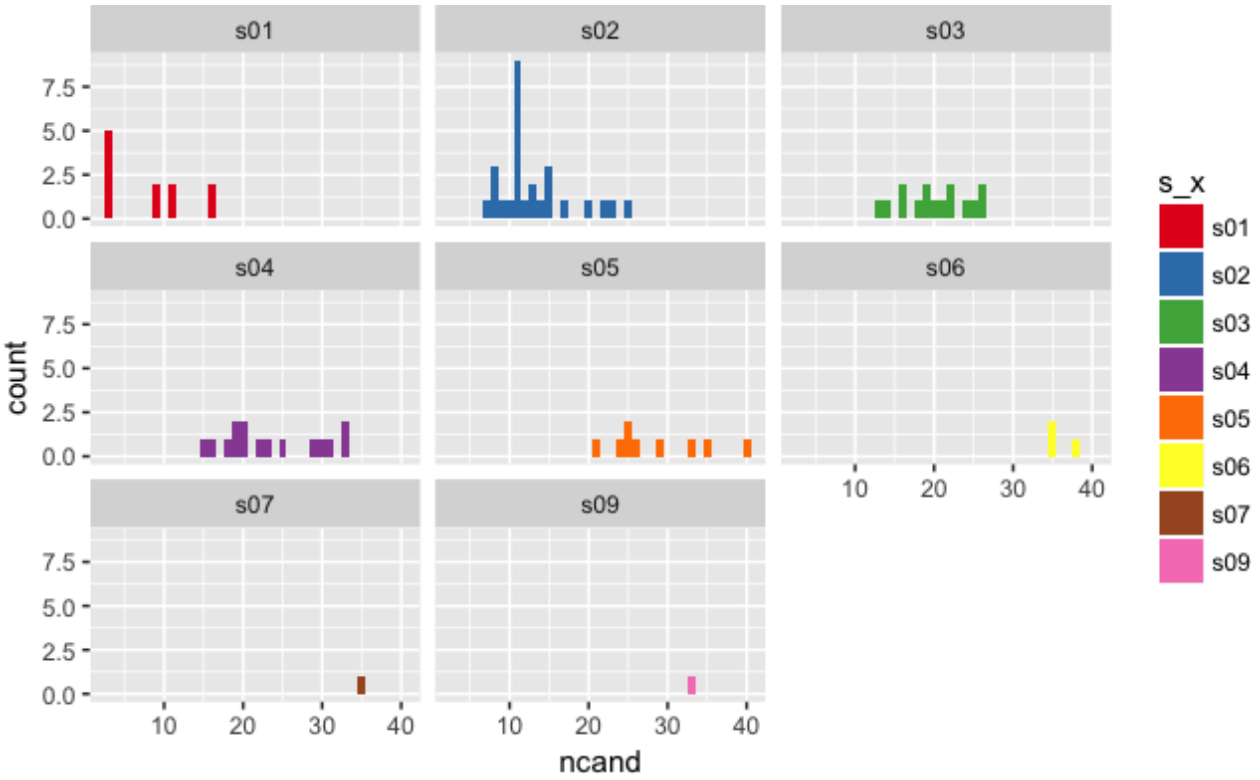
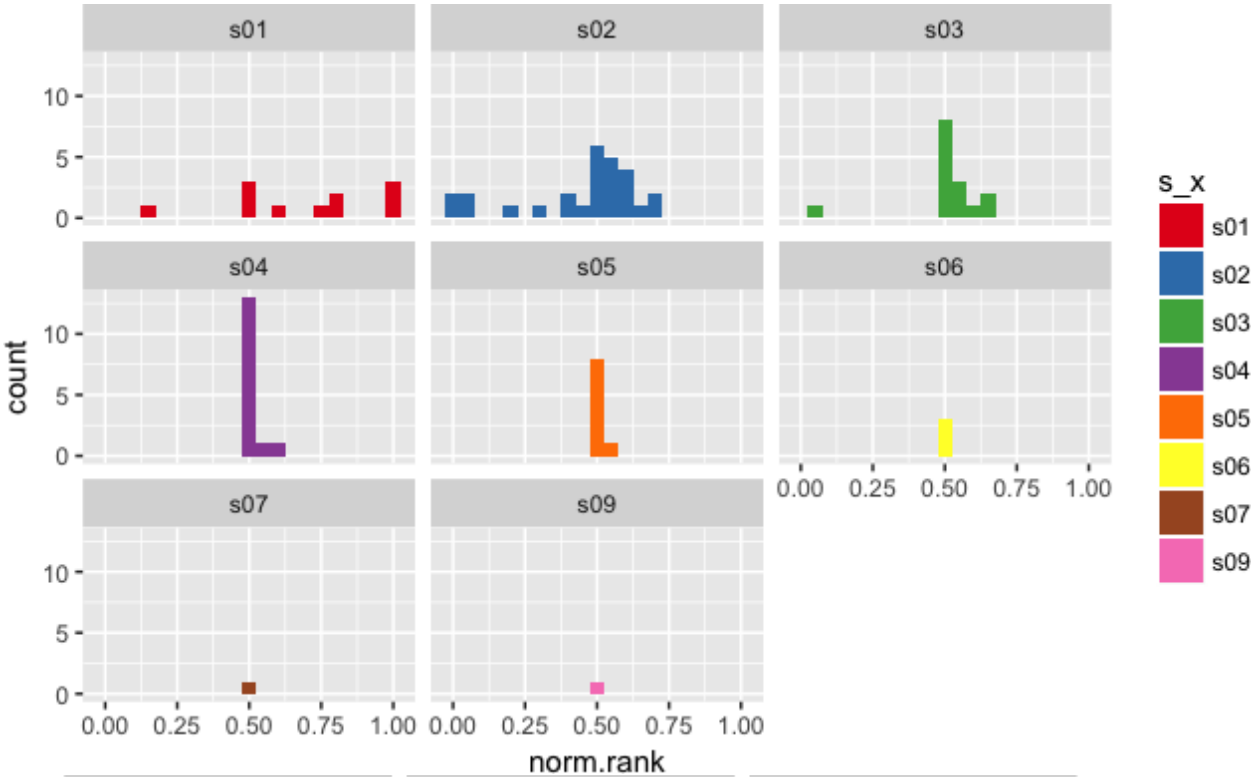


Figure 6: A histogram of the number of candidates as a function of the number of seeds.

4.1.1 Number of candidates?

4.1.2 Rank of x'?



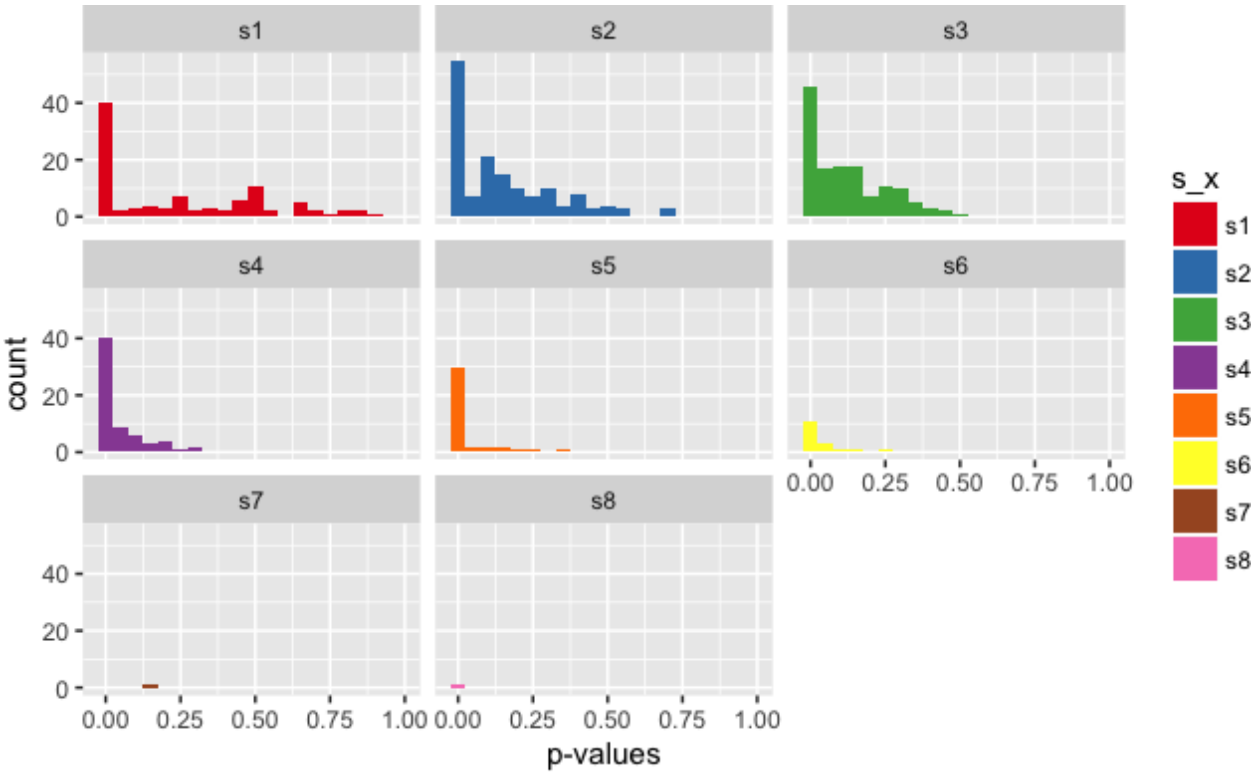


Figure 7: A density of the p-values as a function of the number of seeds.

4.2 Core Graphs

We do the same experiment as above using only the core graphs. We randomly select the VOI x and seeds S from the shared vertices (82 core vertices) and repeat for $MC = 1000$ times.

Table 5: A summary statistics using the core graphs

case	seed	count	mean.ncand	mean.nrank	#pvals<0.05	mean(nrank[pval<0.05])
impossible1	s0	207	0.0	NaN	0	NaN
impossible2	s0	283	8.6	NaN	0	NaN
possible	s1	99	6.4	0.42	40	0.15
possible	s2	150	10.9	0.34	55	0.19
possible	s3	138	14.3	0.35	48	0.21
possible	s4	65	15.4	0.36	42	0.32
possible	s5	39	17.6	0.41	31	0.41
possible	s6	17	20.4	0.36	12	0.38
possible	s7	1	20.0	0.16	0	NaN
possible	s8	1	21.0	0.50	1	0.50

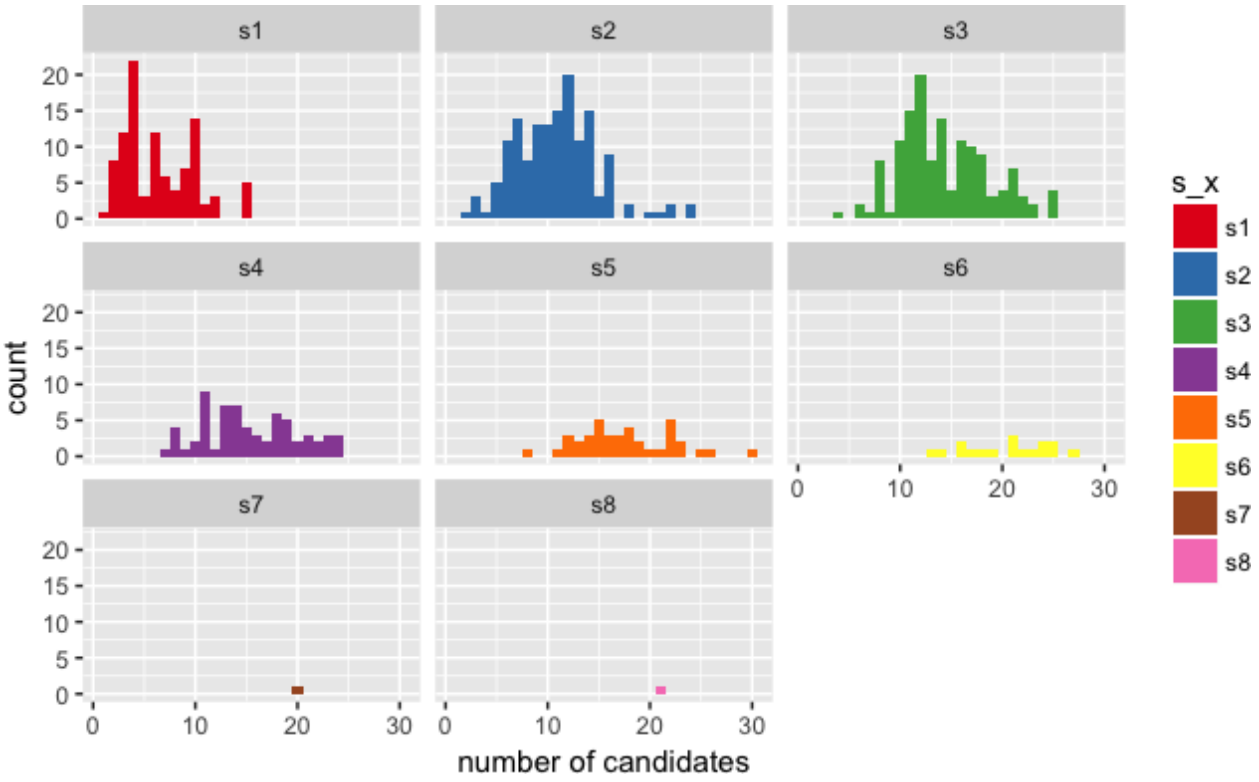


Figure 8: A histogram of the number of candidates as a function of the number of seeds.

4.2.1 Number of candidates?

4.2.2 Rank of x'?