Semiparametric Connectome Inference for Neural Correlates of Cognitive Ability

MT,AA,DLS,VL,YP,JTV,cep,&c
Department of Applied Mathematics and Statistics
Johns Hopkins University

2015-07-01

1 Abstract

We find that more difference twixt two subjects' brain graphs is positively correlated with more difference twixt the two subjects' Composite Creativity Index (CCI) by regressing the semiparametric graph test statistic T_{ij} against $\Delta_{ij} = |CCI_i - CCI_j|$.

- report_brainGraphs_Creativity_Openness.pdf (http://www.cis.jhu.edu/~parky/MRN/report_brainGraphs_Creativity_Openness.pdf)
- Tang, et al., "A semiparametric two-sample hypothesis testing problem for random dot product graphs," subitted for publication. (http://arxiv.org/abs/1403.7249)
- Desikan Region Labels and Descriptions (http://www.cis.jhu.edu/~parky/MRN/Desikan%20Region%20Labels%20and%20Descriptions.pdf) (our regions 36-70 map to 101-135 in this table.)

2 Data

We have 114 weighted symmetric hollow connectomes on $n = 70$ vertices (brain regions). We have the scalar covariate Composite Creativity Index (CCI) for 109 of these graphs. Thus, our data set consists of $m = 109$ pairs (G_i, CCI_i).

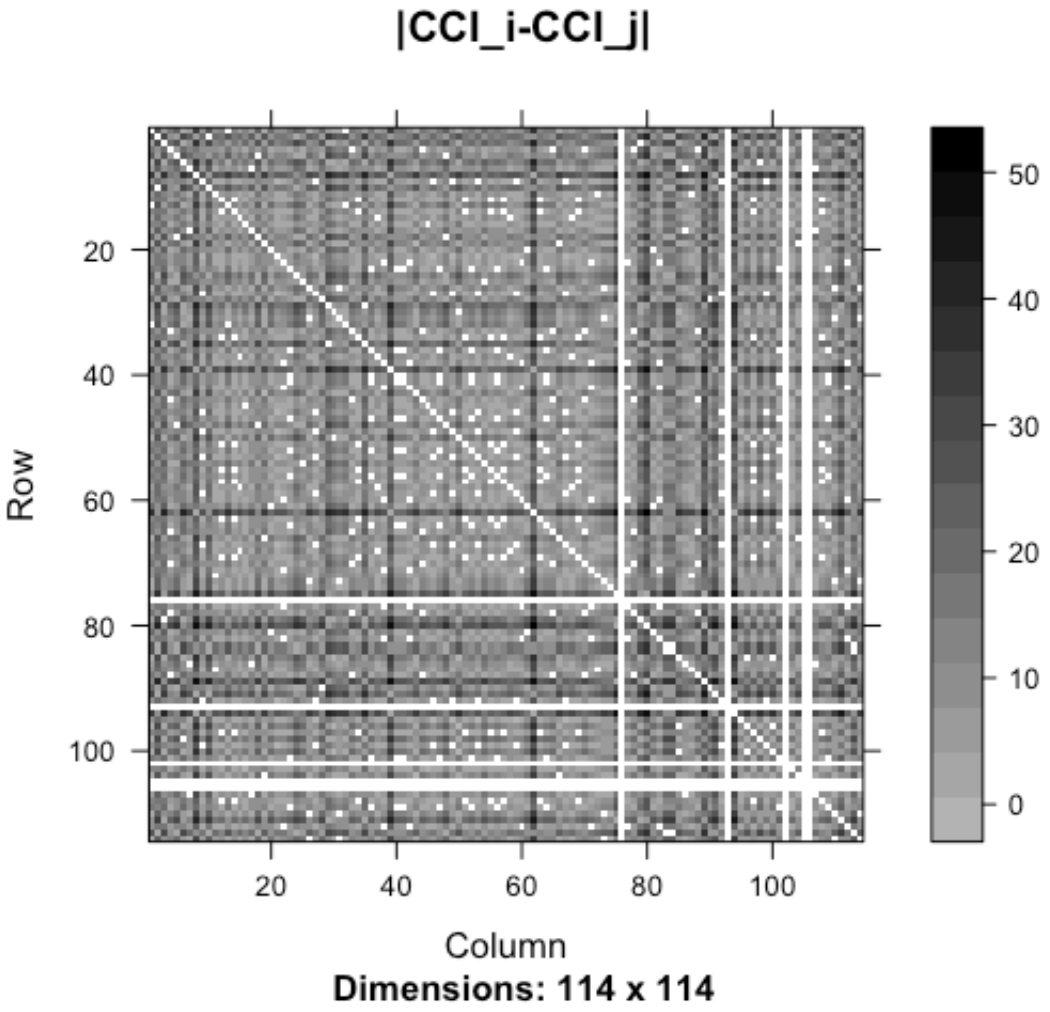
```
con <- url("http://www.cis.jhu.edu/~parky/MRN/cci.txt")
cci <- scan(con)
close(con)
(m <- length(cci))
```

```
# [1] 114
```

```
(cci.na <- which(is.na(cci)))
```

```
# [1]  76  93 102 105 106
```

```
ccidiff <- matrix(0,nrow=m,ncol=m)
for(i in 1:m) for(j in 1:m) ccidiff[i,j] <- abs(cci[i]-cci[j])
diag(ccidiff) <- NA
require(Matrix)
Matrix::image(Matrix(ccidiff),colorkey=TRUE,lwd=0,main="|CCI_i-CCI_j|")
```



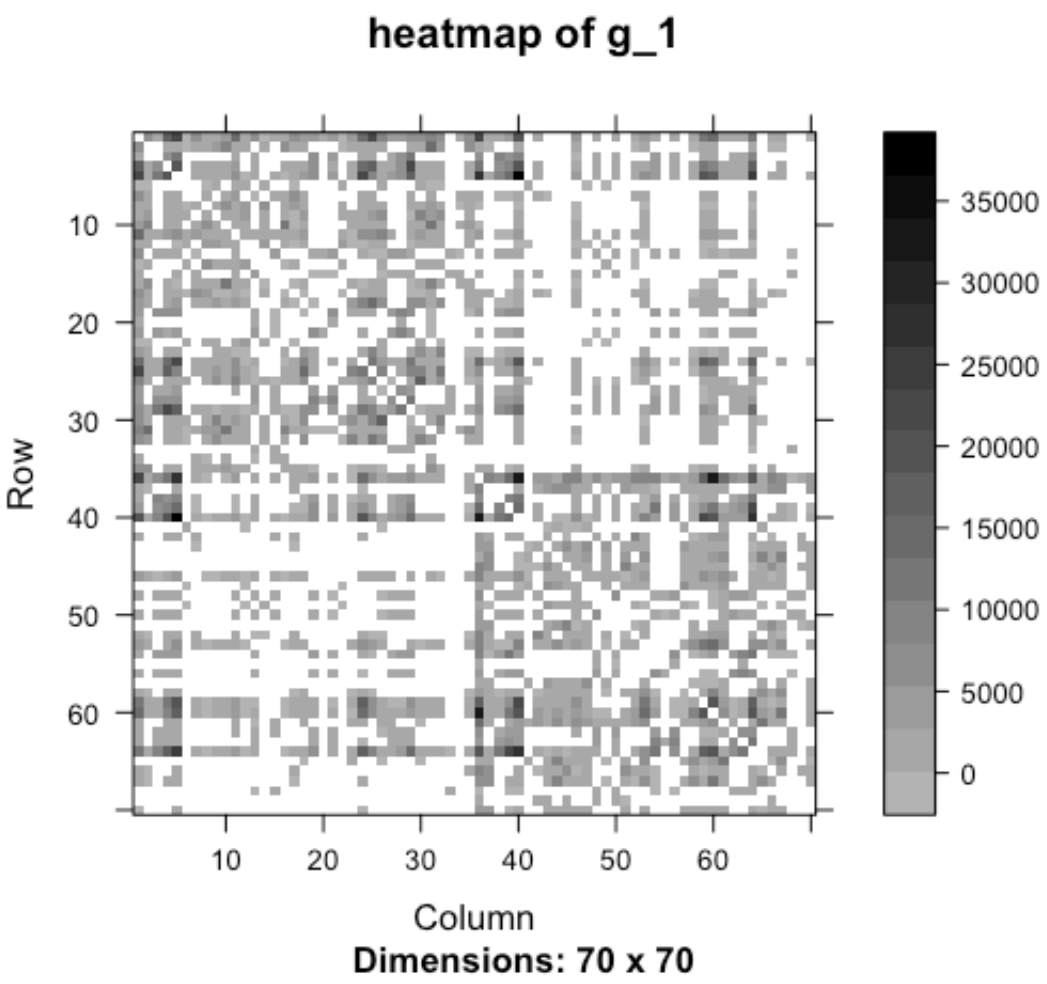
```
ase <- function(A, dim){
    if(nrow(A) >= 400){
        require(irlba)
        A.svd <- irlba(A, nu = dim, nv = dim)
        A.svd.values <- A.svd$d[1:dim]
        A.svd.vectors <- A.svd$v[,1:dim]
        if(dim == 1)
            A.coords <- sqrt(A.svd.values) * A.svd.vectors
        else
            A.coords <- A.svd.vectors %*% diag(sqrt(A.svd.values))
    } else{
        A.svd <- svd(A)
        if(dim == 1)
            A.coords <- A.svd$v[,1] * sqrt(A.svd$d[1])
        else
            A.coords <- A.svd$v[,1:dim] %*% diag(sqrt(A.svd$d[1:dim]))
    }

   return(A.coords)
}

con <- url("http://www.cis.jhu.edu/~parky/MRN/fibergraph.Rbin")
load(con)
close(con)
(m <- length(fibergraph.list))
```

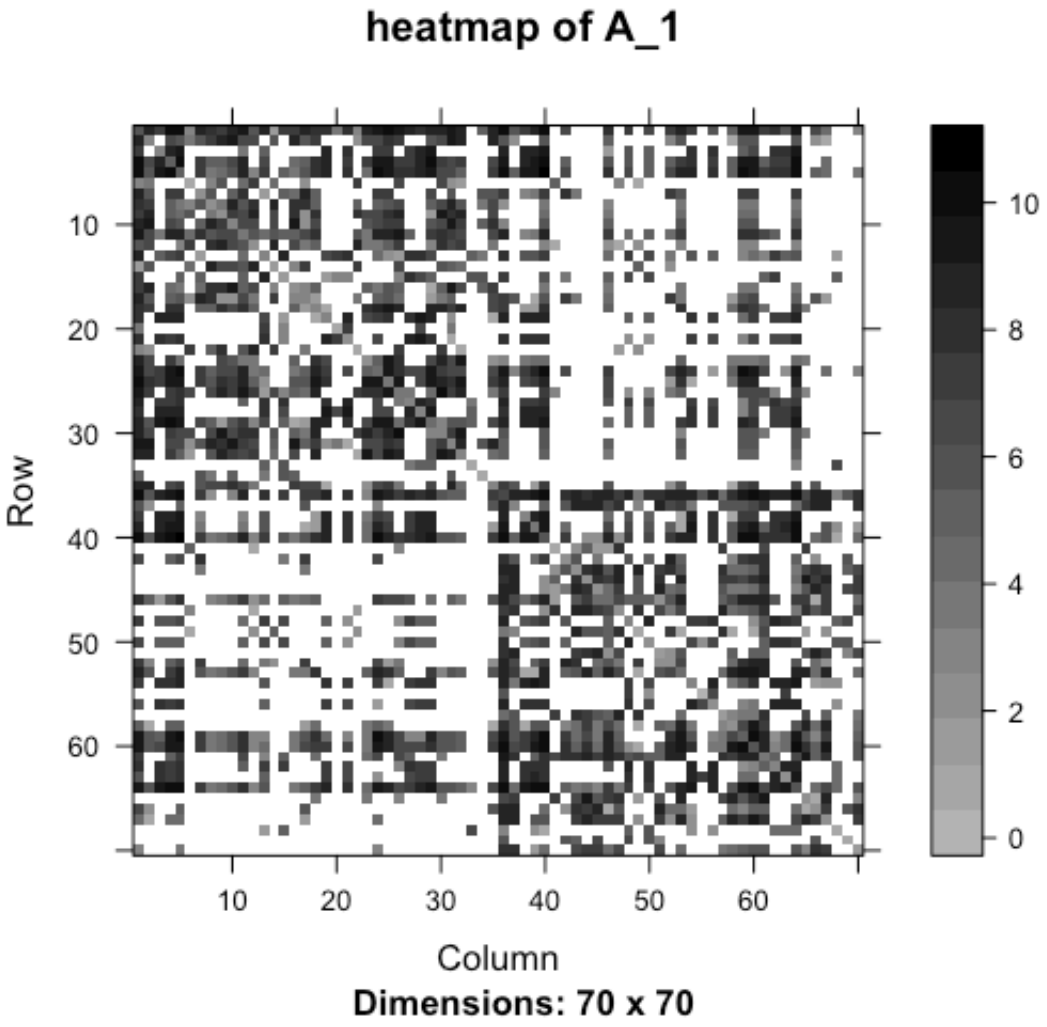
```
# [1] 114
```

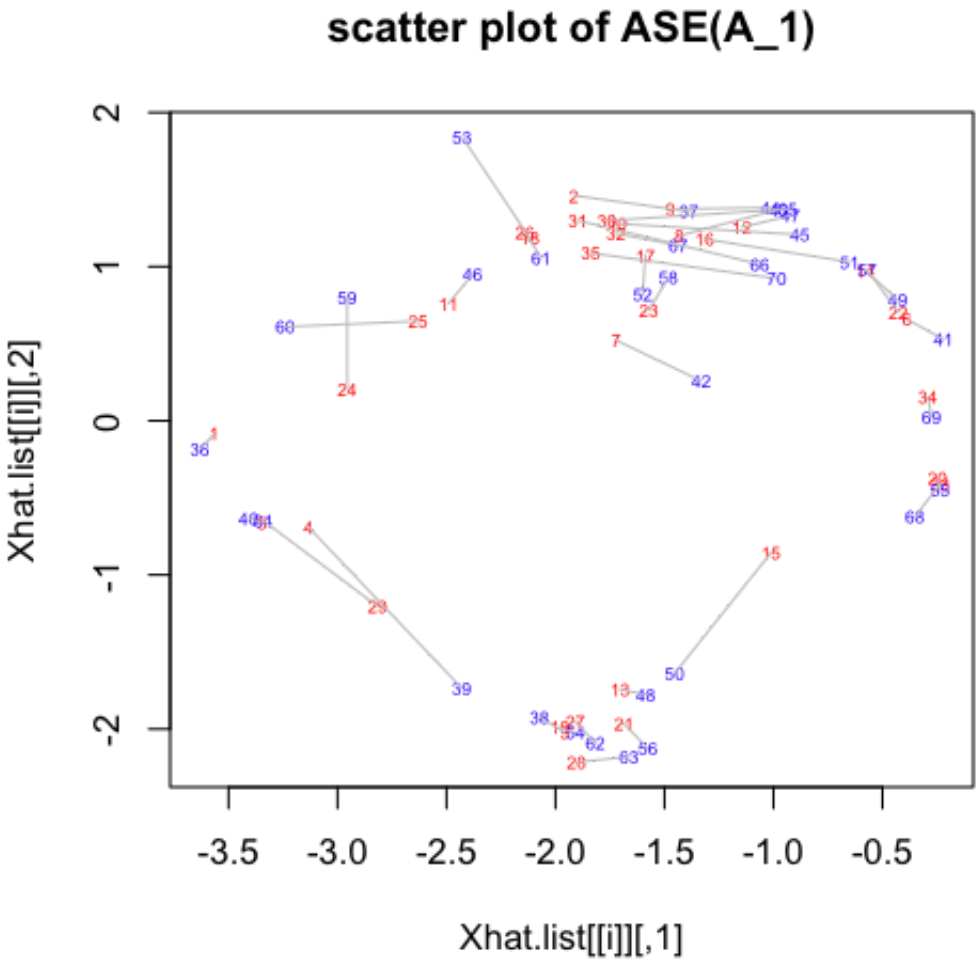
```
(n <- nrow(fibergraph.list[[1]]))
```

```
# [1] 70
```

```
Xhat.list <- list()
dmax <- 20
pcol <- rep(c(2,4),each=n/2)
for(i in 1:m) {
    gi <- fibergraph.list[[i]] ## upper-triangle only
    gi <- (gi + t(gi))
    gi.log <- log(gi + 1)
    A <- gi.log + Diagonal(x=rowSums(gi.log))/(n-1) # diagonal augmentation
    Xhat.list[[i]] <- ase(A, dmax)
    if (i==1) {
        print(Matrix::image(Matrix(gi),lwd=0,colorkey=TRUE,main="heatmap of
g_1"))
        print(Matrix::image(Matrix(A),lwd=0,colorkey=TRUE,main="heatmap of
A_1"))
        plot(Xhat.list[[i]],main="scatter plot of ASE(A_1)",type="n")
        text(Xhat.list[[i]][,1],Xhat.list[[i]][,2],pch=sprintf("%2d",1:n),ce
x=0.5,col=pcol)
        for(v in 1:(n/2))
            segments(Xhat.list[[i]][v,1],Xhat.list[[i]][v,2],Xhat.list[[i]]
[v+n/2,1],Xhat.list[[i]][v+n/2,2],col="grey")
    }
}
```


heatmap of g_1
Row
10
20
30
40
50
60
Column
10
20
30
40
50
60
35000
30000
25000
20000
15000
10000
5000
0
Dimensions: 70 x 70


heatmap of A_1
Row
10
20
30
40
50
60
Column
10
20
30
40
50
60
10
8
6
4
2
0
Dimensions: 70 x 70



3 Embedding Dimension

We embed each graph, via ASE, into R^d .

We select the embedding dimension $\hat{d}$ via Zhu & Ghodsi.

See Figures 1 and 2 and Table 1 below.

We shall use embedding dimension $\hat{d} = Q3(ZG2) = 15$ henceforth.

```
## scree plots
source(url("http://www.cis.jhu.edu/~parky/MRN/getElbows.R"))
require(irlba)
elbmat <- matrix(0,3,m)
plot(1:(n-10),type="n",ylim=c(0,250),xlab="embedding dimension",ylab="eigen val
ue")
for (i in (1:m)[-cci.na]) {
    gi <- fibergraph.list[[i]]
    gi <- (gi + t(gi))
    gi.log <- log(gi + 1)
    A <- gi.log + Diagonal(x=rowSums(gi.log))/(n-1)
    vecs <- irlba(A,nrow(gi)-10,ncol(gi)-10)$d
    elb <- getElbows(vecs,plot=F)
    elbmat[,i] <- elb
    points(vecs,type="l",col="grey")
    points(elb,vecs[elb],pch=19,col=2:4,cex=0.5)
}
```

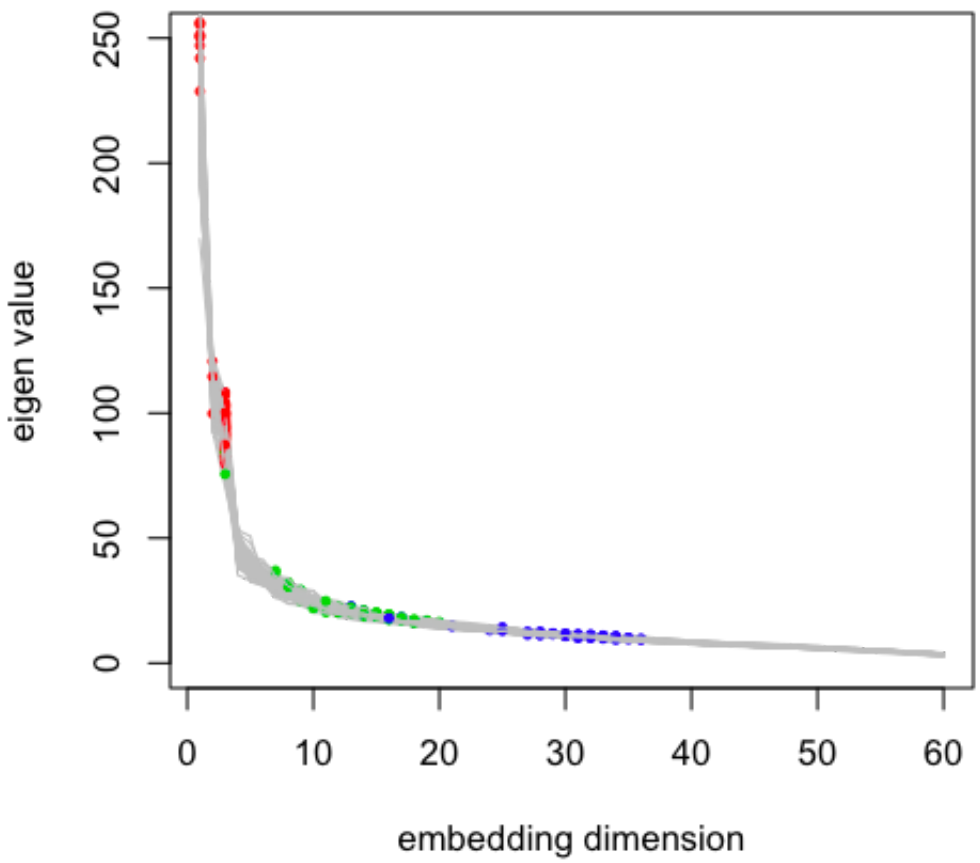


Figure 1: 109 ASE scree plots.

```
boxplot(t(elbmat[,-cci.na]),notch=TRUE,xlab="elbow",ylab="embedding dimension")
```

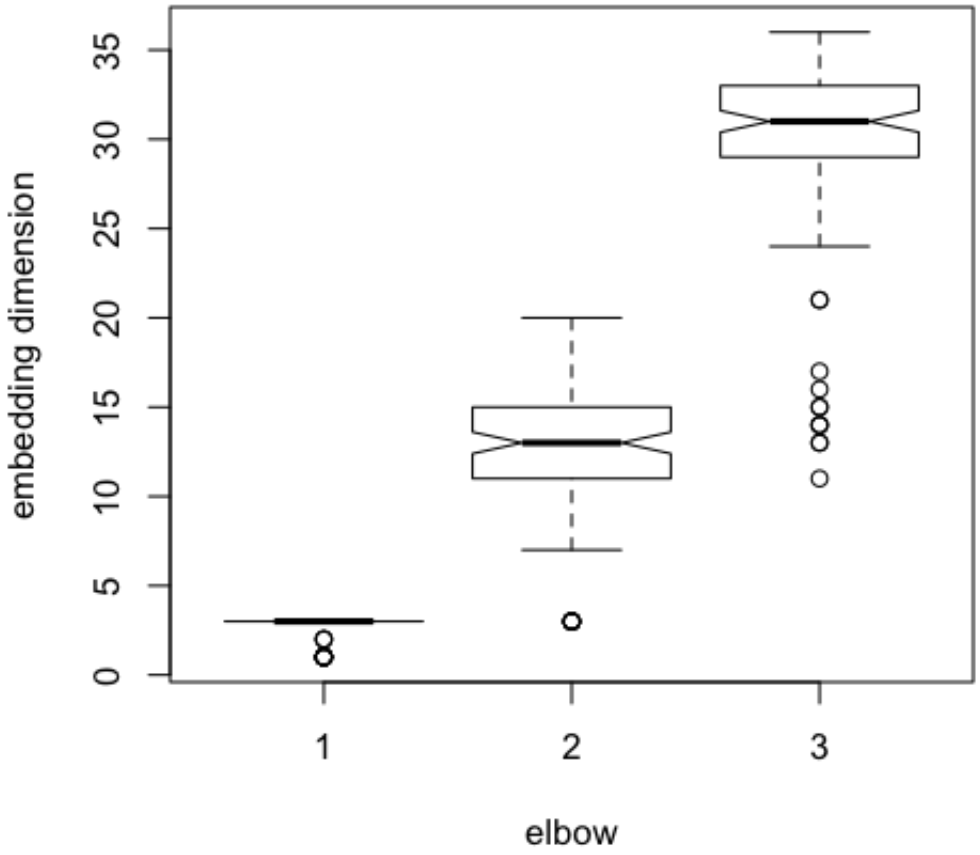


Figure 2: Zhu & Ghodsi boxplots for the 109 graphs: ZG1, ZG2, ZG3.

```
## Table 1
(elbsum <- summary(elbmat[2,-cci.na]))
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#    3.00   11.00   13.00   12.67   15.00   20.00
```

```
(dhat <- elbsum[5])
```

```
# 3rd Qu.
#      15
```

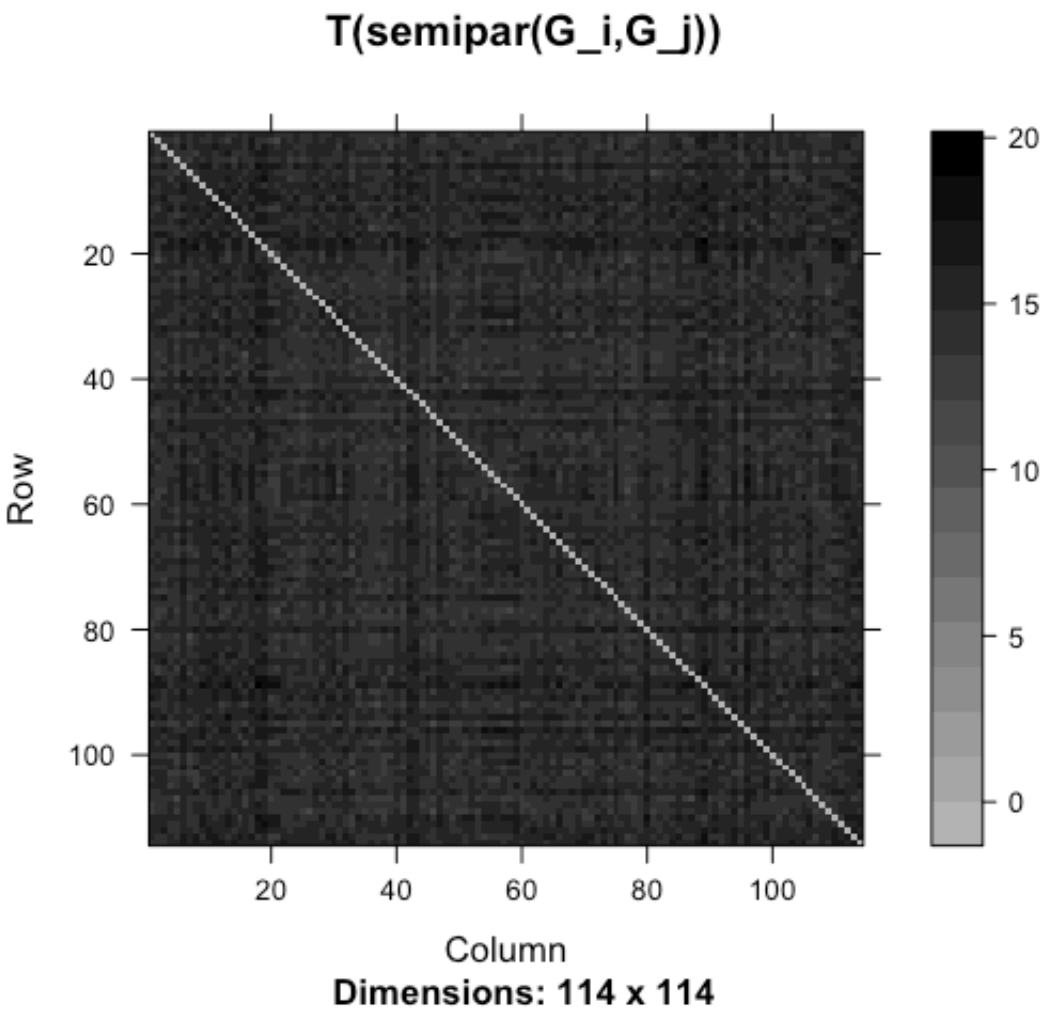
4 ∃ signal!

Large values of the semiparametric graph test statistic T_{ij} are appropriate for rejecting $H_0 : G_i \sim G_j$ in favor of the alternative that the two graphs have different distributions.

[NB: There is much more to say about this – most of it from the Tang et al. semipar graph testing paper and the covariate extension –
but for now let us note that the covariate-aware generative model described below motives our apporach.
(Note also that direct estimation of model parameters will be problematic for this data set, with $\hat{d} = 15$.]

As such, we regress T_{ij}, obtained using embedding dimension $\hat{d} = 15$, against $\Delta_{ij} = |CCI_i - CCI_j|$.
We find positive slope,
indicating that more difference twixt two subjects' brain graphs is positively correlated with more difference twixt the two subjects' CCI.
See Figure 3 and Table 2 below.

```
computeCX <- function(X) {
         P <- X%*%t(X)
         D <- rowSums(P)
         D[D < 0] <- 0
         tmp1 <- eigen(t(X)%*%X)
         tmp2 <- tmp1$vectors %*% diag(1/(tmp1$values^2)) %*% t(tmp1$vectors)
         CX <- sum(diag(tmp2 %*% t(X*sqrt(D)) %*% (X*sqrt(D))))
         return(sqrt(CX))
}

procrustes <- function(X,Y, type = "I", normFlag=FALSE) {
    if(type == "C"){
        X <- X/norm(X, type = "F")*sqrt(nrow(X))
        Y <- Y/norm(Y, type = "F")*sqrt(nrow(Y))
    }
    if(type == "D"){
        tX <- rowSums(X^2)
        tX[tX <= 1e-15] <- 1
        tY <- rowSums(Y^2)
        tY[tY <= 1e-15] <- 1
        X <- X/sqrt(tX)
        Y <- Y/sqrt(tY)
    }

    tmp <- t(X) %*% Y
    tmp.svd <- svd(tmp)
    W <- tmp.svd$u %*% t(tmp.svd$v)
    err <- norm(X%*%W - Y, type = "F")
    if (normFlag) err <- err / (computeCX(X) + computeCX(Y))

    return(list(error = err, W = W))
}

Tmat <- matrix(0,m,m)
for(i in 1:m) {
    for(j in 1:m) {
        Tmat[i,j] <- procrustes(Xhat.list[[i]][,1:dhat], Xhat.list[[j]][,1:dha
t])$error
    }
}
Matrix::image(Matrix(Tmat),colorkey=TRUE,lwd=0,main="T(semipar(G_i,G_j))")
```




```
## slope plot
ylim <- range(Tmat[upper.tri(Tmat)])
plot(ccidiff,Tmat,type="n",ylim=ylim,xlab="Delta_ij = |CCI_i - CCI_j|",yla
b="T_ij = Tsemipar(G_i,G_j)",main="")
ccilm <- lm(c(Tmat[upper.tri(Tmat)])~c(ccidiff[upper.tri(ccidiff)]))
pvalv <- summary(ccilm)$coeff[2,4]
slopev <- ccilm$coeff[2];
points(ccidiff,Tmat,pch=".",col=1)
abline(ccilm$coefficients,col=2,lw=3)
title(paste("dhat =",dhat,", slope =", format(slopev,digits=2)))
```

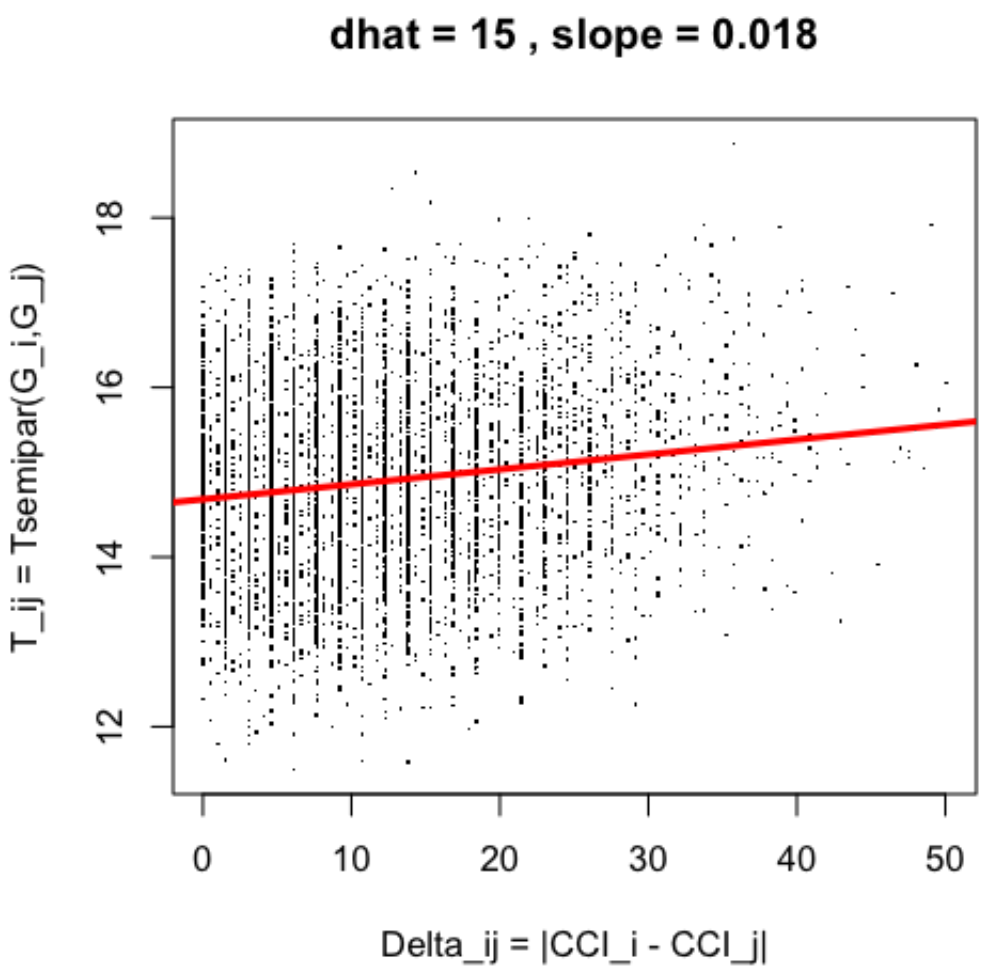


Figure 3: T_ij vs Delta_ij

```
## Table 2
summary(ccilm)
```

```
#
# Call:
# lm(formula = c(Tmat[upper.tri(Tmat)]) ~ c(ccidiff[upper.tri(ccidiff)]))
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -3.3454 -0.7121  0.0129  0.6860  3.6002
#
# Coefficients:
#                                 Estimate Std. Error t value Pr(>|t|)
# (Intercept)                    14.680903   0.022012  666.96   <2e-16 ***
# c(ccidiff[upper.tri(ccidiff)])  0.017617   0.001528   11.53   <2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.031 on 5884 degrees of freedom
#   (555 observations deleted due to missingness)
# Multiple R-squared:  0.0221,  Adjusted R-squared:  0.02193
# F-statistic:   133 on 1 and 5884 DF,  p-value: < 2.2e-16
```

Dependency invalidates the p-value for slope given in the summary table above.
We assess $H_0 : slope \leq 0$ vs $H_A : slope > 0$ via permutation test and obtain $p \ll 0.05$;
see Figure 4.

```
## H0 : slope <= 0 vs HA : slope > 0 via permutation test
nmc <- 1000
set.seed(12345)
permslope <- NULL
for (mc in 1:nmc) {
    sss <- sample(1:nrow(Tmat))
    Zmat <- Tmat[sss,sss]
    permslope[mc] <- lm(c(Zmat[upper.tri(Zmat)])~c(ccidiff[upper.tri(ccidif
f)]))$coefficients[2]
}
(pval <- sum(permslope>slopev)/nmc)
```

```
# [1] 0.002
```

```
plot(stats::density(permslope,adjust=2),xlab="slope",main="")
title(paste("pval =",pval))
abline(v=slopev,col=2,lwd=2)
```

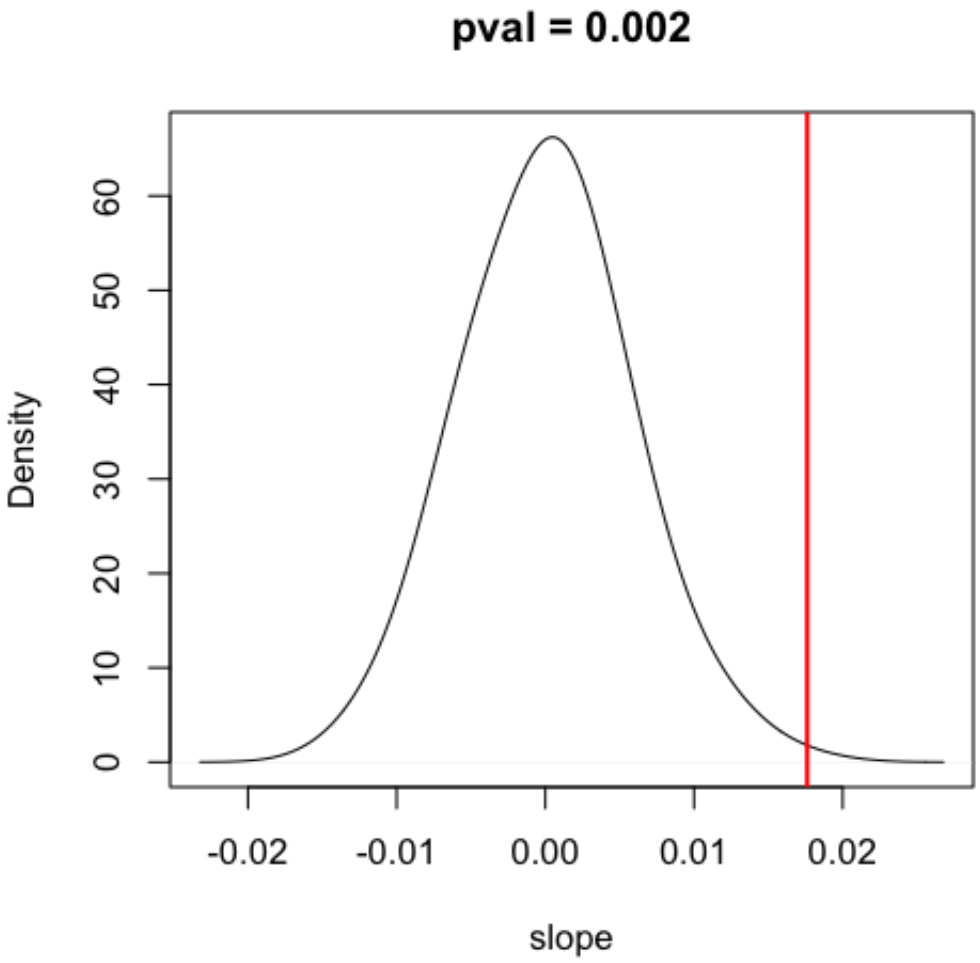
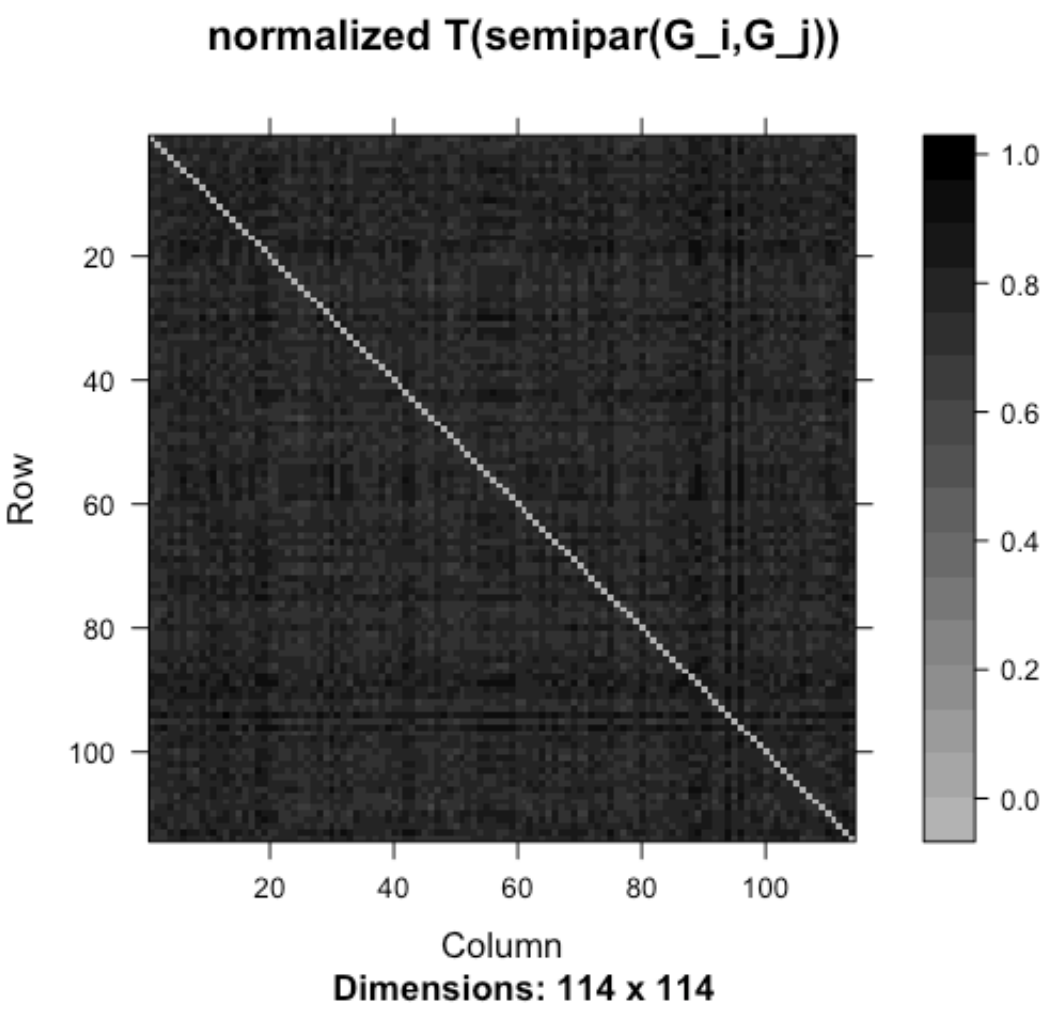


Figure 4: Permutation test null distribution (black) and observed value (red), demonstrating statistical significance.

(Now we repeat the slope significance analysis, using the normalized version of the semipar test statistic … just for completeness.)

```
normT <- Tmat
for(i in 1:m) {
    X <- Xhat.list[[i]][,1:dhat]
    for(j in 1:m) {
        Y <- Xhat.list[[j]][,1:dhat]
        normT[i,j] <- Tmat[i,j] / (computeCX(X) + computeCX(Y))
    }
}
Matrix::image(Matrix(normT),colorkey=TRUE,lwd=0,main="normalized T(semipa
r(G_i,G_j))")
```




```
ccilm2 <- lm(c(normT[upper.tri(normT)])~c(ccidiff[upper.tri(ccidiff)]))
pvalv2 <- summary(ccilm2)$coeff[2,4]
(slopev2 <- ccilm2$coeff[2])
```

```
# c(ccidiff[upper.tri(ccidiff)])
#                   0.0008818183
```

```
nmc <- 1000
set.seed(12345)
permslope2 <- NULL
for (mc in 1:nmc) {
    sss <- sample(1:nrow(normT))
    Zmat <- normT[sss,sss]
    permslope2[mc] <- lm(c(Zmat[upper.tri(Zmat)])~c(ccidiff[upper.tri(ccidif
f)]))$coefficients[2]
}
(pval2 <- sum(permslope2>slopev2)/nmc)
```

```
# [1] 0.001
```

```
plot(stats::density(permslope2,adjust=2),xlab="slope",main="")
title(paste("normalized, pval =",pval2))
abline(v=slopev2,col=2,lwd=2)
```

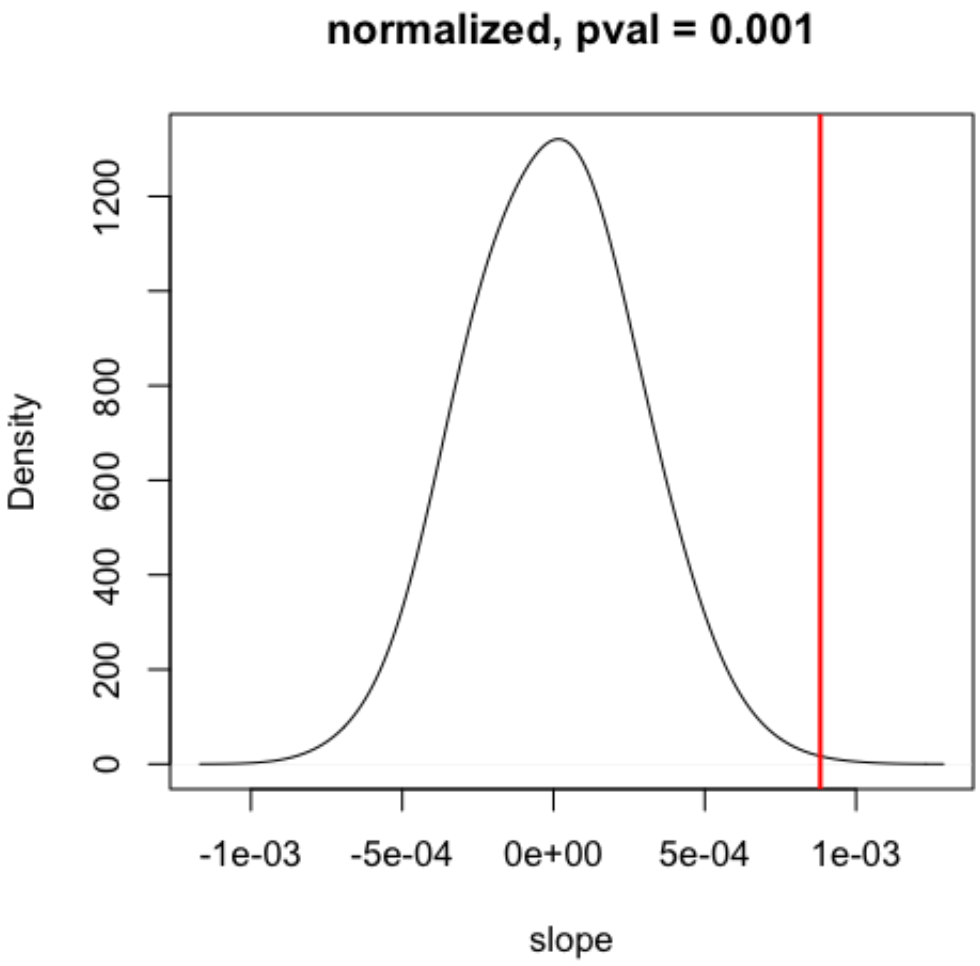


Figure 5 below depicts slopes and permutation test p-values against $\hat{d}$.
(Don't try to run this; this takes a long time, so this plot contains pre-computed values.)

```
## Slope and permutation test p-values against dhat
require(foreach)
nmc <- 10000
dmax <- 60
(dvec <- c(1:20,seq(30,dmax,by=10)))
myslopes <-pval <- numeric(length(dvec))
ind <- 1
set.seed(12345)
for (d in dvec) {
    load(paste0("Smat-Pmat-diagaug-typeI-dhat",d,".Rbin"))
    myslopes[ind] <- lm(c(Smat[upper.tri(Smat)])~c(ccidiff[upper.tri(ccidif
f)]))$coefficients[2]
    permslope <- foreach (mc=1:nmc,.combine='c') %dopar% {
        sss <- sample(1:nrow(Smat))
        Zmat <- Smat[sss,sss]
        out <- lm(c(Zmat[upper.tri(Zmat)])~c(ccidiff[upper.tri(ccidiff)]))$coef
ficients[2]
        out
    }
    pval[ind] <- sum(permslope>myslopes[ind])/nmc
    ind <- ind+1
}
```

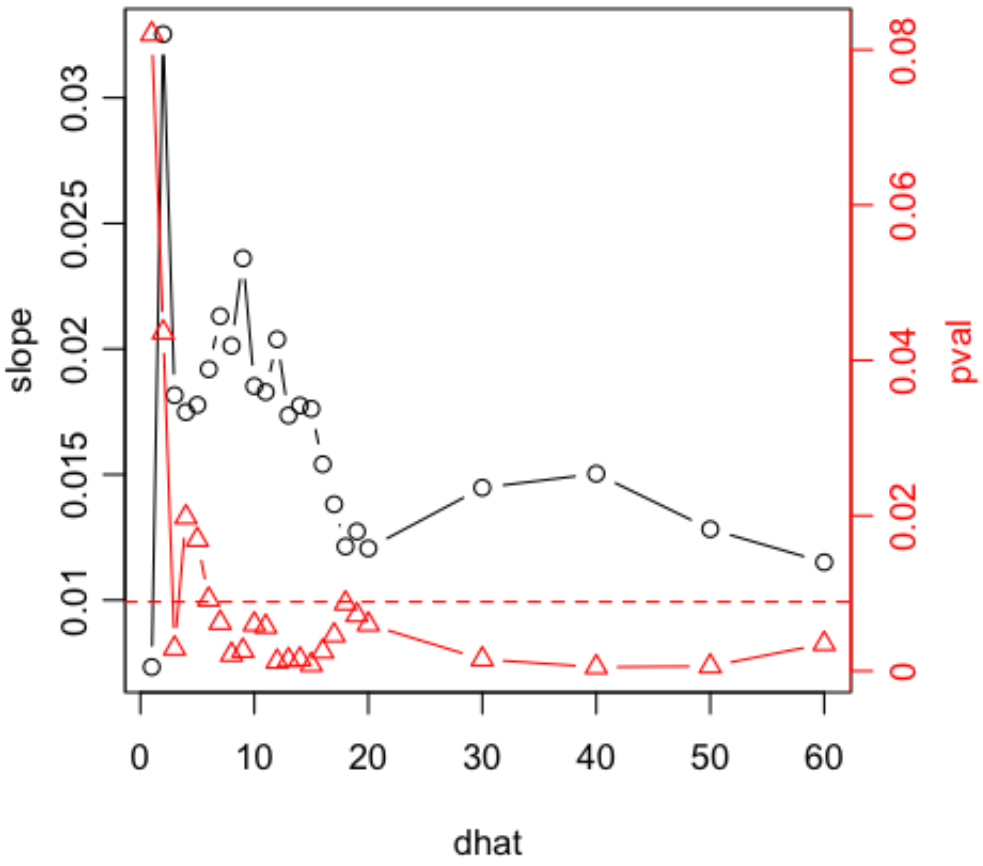


Figure 5: Slopes & permutation test p-values as a function of embedding dimension. (Red dashed horizontal indicates Frobenius p-value.)

5 Most Important Vertices

To investigate the most important vertices (brain regions),
we decompose our test statistic to assess the effect of leaving *in* and *out* each vertex in turn.

The vertices whose *addition* (*leave-one-in*) yields the largest slope – the most important vertices for our positive slope signal – are regions
38, 44, 53, 65, 37, 19, 48, 54, …

The vertices whose *deletion* (*leave-one-out*) decreases the slope the most – the most important vertices for our positive slope signal – are regions
38, 53, 65, 44, 48, 37, 19, 54, …

These two approaches agree on these top few MIV's (with a few order swaps).
Furthermore, the correlation twixt the two slope vectors is 0.986.
Henceforth we consider only the leave-one-in analysis.

```
calcRanks <- function(dvec,Xhat,ydiff,unif=FALSE,verbose=FALSE)
{
    m <- length(Xhat)
    n <- nrow(Xhat[[1]])

    slopemat.loout <- slopemat.loin <- matrix(0,n,length(dvec));
    colnames(slopemat.loout) <- colnames(slopemat.loin) <- dvec

    ind <- 1
    for (d in dvec) {
        T.loout <- T.loin <- rep(list(matrix(0,m,m)),n)
        if (unif) {
            T.loin <- lapply(1:n, function(x) matrix(runif(m*2),m,m))
            T.loout <- lapply(1:n, function(x) matrix(runif(m*2),m,m))
#            T.loin <- lapply(1:n, function(x) matrix(rnorm(m*2),m,m))
#            T.loout <- lapply(1:n, function(x) matrix(rnorm(m*2),m,m))
        } else {
            for(i in 1:m) {
                Xhat.i <- Xhat[[i]][,1:d]
                if (verbose)  cat("d = ", d, ", i = ", i, "\n")
                for (j in 1:m) {
                    Xhat.j <- Xhat[[j]][,1:d]
                    tmp <- t(Xhat.i) %*% Xhat.j
                    tmp.svd <- svd(tmp)
                    W <- tmp.svd$u %*% t(tmp.svd$v)
                    for (v in 1:n) {
                        T.loin[[v]][i,j]  <- norm(Xhat.i[v,]%*%W-Xhat.j[v,],
"F") # leave-one-in
                        T.loout[[v]][i,j] <- norm(Xhat.i[-v,]%*%W-Xhat.j[-
v,],"F") # leave-one-out
                    }
                }
            }
        }

        slopemat.loin[,ind] <- sapply(T.loin, function(x) lm(c(x[upper.tr
i(x)])~c(ydiff[upper.tri(ydiff)]))$coeff[2])
        slopemat.loout[,ind] <- sapply(T.loout, function(x) lm(c(x[upper.tr
i(x)])~c(ydiff[upper.tri(ydiff)]))$coeff[2])
        ind <- ind+1
    }

    return(list(slope.in=slopemat.loin, slope.out=slopemat.loout))
}
```

```
## observed
dhat <- 15
rankout <- calcRanks(dvec=dhat,Xhat.list,ccidiff,verbose=FALSE)
```

```
(slope.obs <- max(rankout$slope.in))
```

```
# [1] 0.007721802
```

```
which.max(rankout$slope.in)
```

```
# [1] 38
```

```
(ord <- order(rankout$slope.in))
```

```
#  [1] 58 17 61 32 43 33 35 23  7 45 57 49  9 51  4 41 42  2 26 22 10 40 66
# [24] 67 16 12 30 14 31  6  8 27 59 63 13 62 46 36 15 64 11  5 34 39 29 47
# [47] 60  3 69 50  1 55 21 52 28 70 24 20 25 18 68 56 54 48 19 37 65 53 44
# [70] 38
```

```
cor( rowMeans(rankout$slope.out) , -rowMeans(rankout$slope.in) )
```

```
# [1] 0.9863619
```

Figure 6 shows that the three most important vertices (38,44,53), as ranked via our leave-one-in analysis, stand out particularly dramatically.

```
zzz <- rankout$slope.in
plot(sort(zzz),xlab="order",ylab="rank.loin",col="black",cex=0.5)
title("largest are most important")
for (i in 1:3) {
    points(n-i+1, zzz[rev(ord)[i]],col=i+1,bg=i+1,pch=20+i,cex=1)
}
abline(h=0,lty=3)
legend("bottomright",legend=rev(ord)[1:3],pch=21:23,pt.bg=2:4)
```

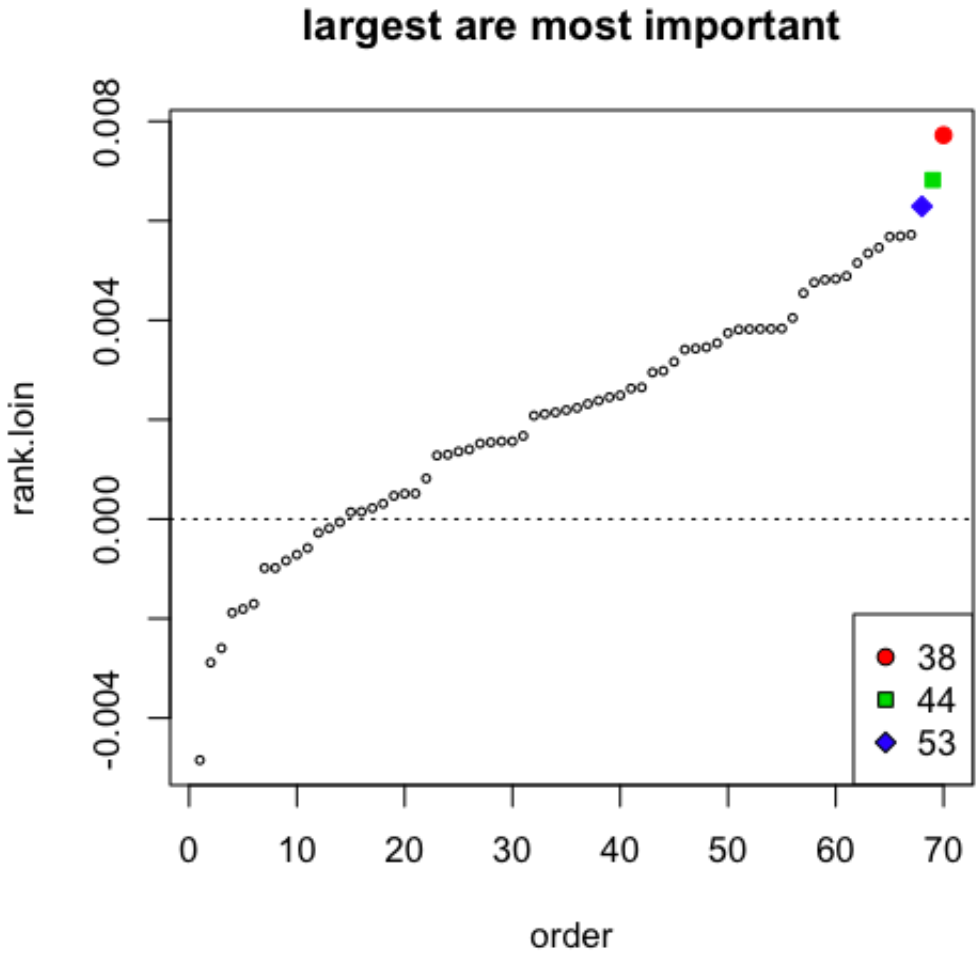


Figure 6: Rank depiction for most important vertices combining leave-one-in

```
# let's use just these three, henceforth
(miv.int <- rev(ord)[1:3])# c( 38, 44, 53)
```

```
# [1] 38 44 53
```

```
nmiv <- length(miv.int)
(slope.int <- rankout$slope.in[miv.int])
```

```
# [1] 0.007721802 0.006820264 0.006289236
```

5.1 P-value

1. By regressing the semiparametric graph test statistic T_{ij} against Δ_{ij},
   we find a positive slope (Figure 3)
   with a permutation test p-value $\ll 0.05$ (Figure 4).

NB: i can find no reason for a crisis of confidence regarding this analysis;
i am confident that this p-value is entirely valid
and that this analysis does indeed demonstrate that
more difference twixt two subjects' brain graphs
is positively correlated with more
difference twixt the two subjects' CCI.
—cep 06/27/2015

2. Now, as for MIVs,
   the leave-one-in analysis tells us that region 38 is the most important vertex.
   Both this MIV ranking and the T vs Δ regression are principled based on our covariate-aware generative model.
   But how to get a p-value?
   Answer: Synthetic Data Analysis!

5.2 Synthetic Data Analysis

Figure 7 provides an assessment of significance for our most important vertices.
Here we perform a Monte Carlo simulation
to estimate the distribution of max leave-one-in slope
for the synthetic null $\hat{X}(\bar{A})$,
and we see that our identified MIV's
(regions 38,44,53)
are indeed statistically significant.

```
## null
Abar <- Reduce("+", fibergraph.list) / m
Abar.log <-  log((Abar + t(Abar)) + 1)
Xhatbar <- ase(Abar.log + diag(x=rowSums(Abar.log))/(n-1), dhat)

rg.sample.pois <- function(P) {
    n <-  nrow(P)
    U <- matrix(0, nrow = n, ncol = n)
    lambda.vec <- P[col(P) > row(P)]
    U[col(U) > row(U)] <- rpois(n*(n-1)/2, lambda = lambda.vec*(lambda.vec >
0))
    U <-  U + t(U)
    return(U)
}

set.seed(12345)
nullslope <- NULL
nummc <- 100
for (mc in 1:nummc) {
    cat("mc = ", mc, "\n")
    Alist <- lapply(1:m, function(x) rg.sample.pois(Xhatbar %*% t(Xhatbar)))
    Xhatp <- lapply(Alist, function(x) {
                               A.log <- log((x+t(x))/2+1)
                               ase(A.log + diag(rowSums(A.log))/(n-1),dhat)})

    rankout0 <- calcRanks(dvec=dhat,Xhatp,ccidiff,verbose=FALSE)
    nullslope[[mc]] <- max(rankout0$slope.in)
}
```

```
pval3 <- sum(slope.obs < nullslope)/nummc
(cv3 <- sort(nullslope)[0.95*nummc])
```

```
# [1] 0.002383612
```

```
plot(stats::density(nullslope,adjust=2),main=paste("observed = ", format(slop
e.obs,digits=2), ", pval = ", pval3),xlab="slope.in",xlim=c(0,max(c(nullslope,s
lope.int))))
abline(v=cv3,col=1,lty=3,lwd=2)
scol <- rep("grey",n); scol[miv.int] <- c(2:(nmiv+1))
spch <- rep(19,n); spch[miv.int] <- c(21:(21+nmiv-1))
sbg <- rep(0,n); sbg[miv.int] <- c(2:(2+nmiv-1))
scex <- rep(0.5,n); scex[miv.int] <- 1
points(rankout$slope.in,rep(0,n),pch=spch,col=scol,bg=sbg,cex=scex)
points(cv3,0,pch=19)
legend("topright",c("p = 0.05",paste("miv =",miv.int)),col=c(1,scol[miv.int]),p
ch=c(21,spch[miv.int]),pt.bg=1:4,cex=1)
```

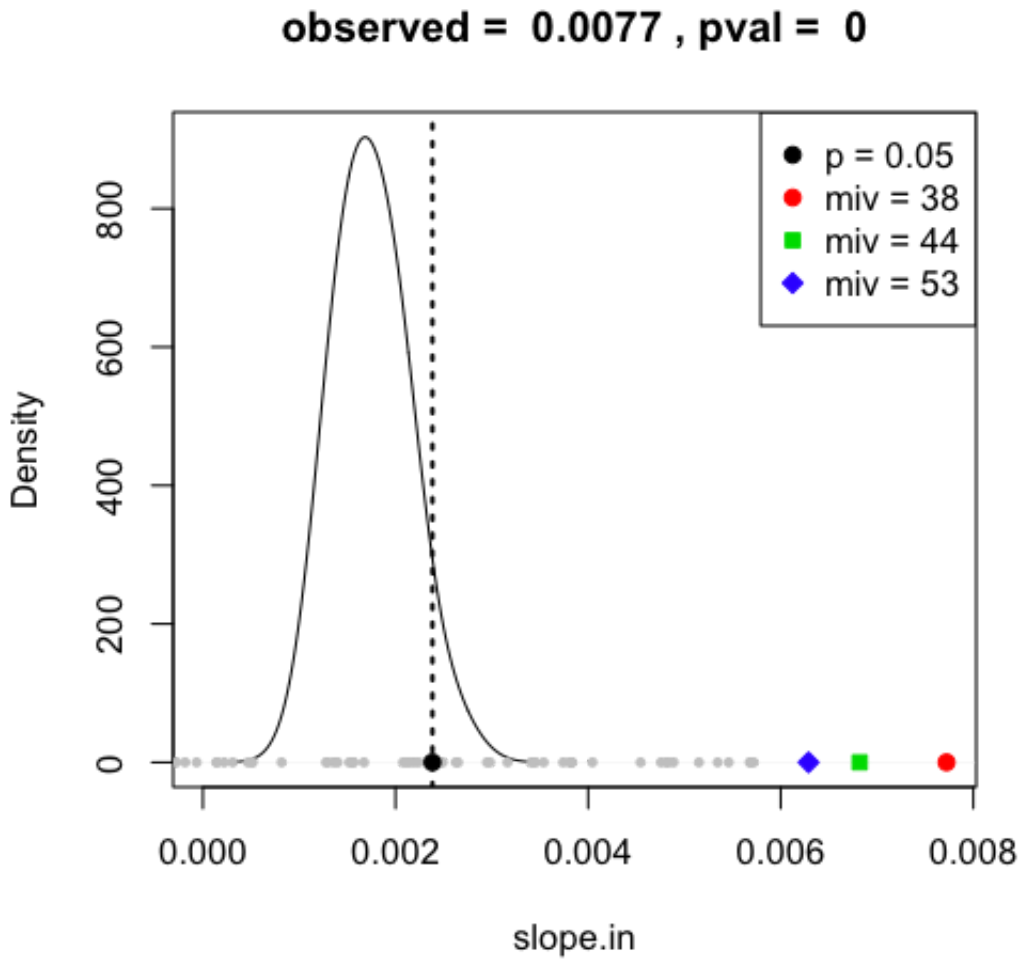


Figure 7: Significance for the most important vertices

6 Covariate Aware Generative Model

i^{th} graph, v^{th} vertex, latent position $X_{iv} \in (R_+)^d$ given by
$X_{iv} = y_i^{t_v} Z_v$
where Z_v is the covariate-unaware latent position for vertex v shared by all graphs,
$y_i \in (1, \infty)$ is the (observed) covariate for graph i,
and $t_v = I\{v \in J\}$ is the indicator for vertex v being covariate-aware (ie $v \in J$ are Important Vertices).

so important vertices $v \in J$
have their latent position magnitude in graph i increased (NB: only increased, in this case) multiplicitively with y_i.

NB: we could generalize, with $t_v \in R_+$ == "how important is v" … but let's not yet?
NB: for rdpPg's (Poisson) we are ok; for rdpg's, we need a further constraint to keep dot product in [0,1].
NB: if $\{Z_v\}$ gives rise to SBM, then $\{X_v\}$ gives rise to DCSBM (Poisson or not)?
NB: $T_{ij}^2 = \sum_{v \in J} d^2(\widehat{y_i Z_v} - \hat{W}\widehat{y_j Z_v}) + \sum_{v \notin J} d^2(\hat{Z}_v - \hat{W}\hat{Z}_v)$

under this model, we have principled approaches to our CCI tasks …
task one: test $H_0 : J = \emptyset$;
approach: test $H_0' : slope(T, \Delta) = 0$ against $H_A' :> 0$.
task two: estimate J.
task twopointone: estimate MIV;
approach: leave-one-in: $\arg\max_v slope(T_v, \Delta)$ where $T^2_{v,ij} = d^2(\hat{X}_{iv} - \hat{W}\hat{X}_{jv})$.

7 Simulation

Consider a 2-block SBM with point masses $(1/2, 1/4)$ and $(1/4, 1/2)$, so $B_{11} = B_{22} = 5/16$ and $B_{12} = B_{21} = 4/16$.
Let $n = 100$, and condition on $n_1 = n_2 = 50$, with the first 50 vertices from block 1 and the second 50 vertices from block 2.
Let the vertex importance indicators be given by $t_1 = t_{51} = 1$ and for all other vertices let $t_v = 0$.
Let $m = 10$.
Let the graph covariates be given by $y_i = 1 + 1/(2i)$ for $i \in [m]$.

Our model gives $X_{iv} = y_i{}^{t_v} Z_v$ where the Z_v's are given by the SBM point masses.

Generate rdpg's (NB: not Poisson!) using the X's, and perform our slope test (Figs 3 & 4) using $d = 2$, and perform our leave-one-in MIV analysis.