```
BeginPackage["SingleNetworks`"];

snetworkdatasingleintimewindows::usage = "description.";
snetworkgraphsinglenodes::usage = "description.";
snetworkdatabinned::usage = "description.";
snetworkdatafxdbucket::usage = "description.";
snetworkdatabinnedintimewindows::usage = "description.";
snetworkdatafxdbucketintimewindows::usage = "description.";
snetworkgraph::usage = "description.";
correlationfunction::usage = "description.";
randomnessfunction::usage = "description.";
GirvanNewmanmodularity::usage = "description.";
randomizedgraphamongcommunities::usage = "description.";
randomnessfunctionformodularity::usage = "description.";
randomnessfunctionformodularityonenullmodel::usage = "description.";
randomnessfunctionformodularitytwonullmodel::usage = "description.";
randomnessvaluesformodularitytwonullmodel::usage = "description.";
```

```
Begin["`Private`"];
```

```
Clear[snetworkdatasingleintimewindows]
snetworkdatasingleintimewindows[feature_,datadimension_]:=Module[{rawaim,pos,aim,campaig
rawaim=Table[Symbol["data"][[i]][[All,feature]],{i,Range@datadimension}];
pos=Table[Partition[Flatten@Table[Position[k,i],{i,{"NA",0}}],1],{k,rawaim}];
aim=Table[Delete[i[[1]],i[[2]]],{i,MapThread[{#1,#2}&,{rawaim,pos}]}];
campaign=Table[Delete[Symbol["data"][[i[[1]]]][[All,2]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
seri=Table[Delete[Symbol["data"][[i[[1]]]][[All,1]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
{aim,campaign,seri}]
```

```
Clear[snetworkgraphsinglenodes]
snetworkgraphsinglenodes[aim_,campaign_,vertexsize_,vertexlabelsize_,imagesize_,vertexco
Module[{binningmembers,aimbaskets,singlesupportvalues,pairs,pairsupportvalues,liftvalues,
allmatrixelements,likelypairs,binarymatrix,graph},
binningmembers=Sort[DeleteDuplicates[aim]];
aimbaskets=Table[DeleteDuplicates[i],{i,Values@GroupBy[Thread[{aim,campaign}],Last->Firs
singlesupportvalues=Table[N[Count[Table[MemberQ[i,j],{i,aimbaskets}],
True]/Length[aimbaskets]],{j,binningmembers}];
pairs=Subsets[binningmembers,{2}];
pairsupportvalues=Table[N[Count[Table[SubsetQ[i,j],{i,aimbaskets}],True]/
Length[aimbaskets]],{j,pairs}];
liftvalues=pairsupportvalues/(DeleteCases[Flatten@UpperTriangularize[Table[singlesupport
[[j]]*singlesupportvalues[[k]],{j,Length[binningmembers]},{k,Length[binningmembers]}],1
allmatrixelements=Sort[Join[pairs,Reverse[pairs,2],Table[{i,i},{i,binningmembers}]]];
likelypairs=Extract[pairs,Position[liftvalues,x_/;x>1]];
binarymatrix=ArrayReshape[Table[If[j==True,1,0],{j,Table[MemberQ[likelypairs,i],
{i,allmatrixelements}]}],{Length@binningmembers,Length@binningmembers}];
graph=AdjacencyGraph[binarymatrix,{GraphLayout->Automatic,DirectedEdges->False,
EdgeShapeFunction->"Line",VertexSize->vertexsize,VertexStyle->vertexcolor,
VertexLabelStyle->Directive[Black,Italic,vertexlabelsize],VertexLabels->Flatten[MapThrea
{#1->Placed[#2,Center]}&,{Range[1,Dimensions[binarymatrix][[1]]],binningmembers}]],
ImageSize->imagesize];{graph,Length@binningmembers}]
```

```
Clear[snetworkdatabinned]
snetworkdatabinned[feature_,step_,data_]:=Module[{rawaim,pos,aim,campaign,seri,min,max,
binningamount,binning},
rawaim=data[[All,feature]];
pos=Partition[Flatten@Table[Position[rawaim,i],{i,{"NA",0}}],1];
aim=Delete[rawaim,pos];
campaign=Delete[data[[All,2]],pos];
seri=Delete[data[[All,1]],pos];
min=Floor[Min[Sort[DeleteDuplicates[aim]]],0.1];
max=Ceiling[Max[Sort[DeleteDuplicates[aim]]]]+step;
binningamount=Length[DeleteCases[BinLists[aim,{min,max,step}],{}]];
binning=Table[Catch[Do[If[IntervalMemberQ[Interval[i],aim[[j]]]==True,Throw[i]],
{i,Partition[Range[min,max,step],2,1]}]],{j,Length[aim]}];
aim=DeleteCases[binning,Null];
{aim,campaign,seri}]
```

```
Clear[snetworkdatafxdbucket]
snetworkdatafxdbucket[feature_,nodenumber_,data_]:=Module[{rawaim,pos,aim,
campaign,seri,bucketsize,aimlabeled,aimpartitioned,bins,repetitivesreport,repetitives,
labelgeneration,binsrearranged,aimbinned},
rawaim=data[[All,feature]];
pos=Partition[Flatten@Table[Position[rawaim,i],{i,{"NA",0}}],1];
aim=Delete[rawaim,pos];
campaign=Delete[data[[All,2]],pos];
seri=Delete[data[[All,1]],pos];
bucketsize=Ceiling@(N@(Dimensions@aim)/nodenumber);
aimlabeled=Thread[Range@Length@aim->aim];
aimpartitioned=Partition[Normal@Sort@Association@aimlabeled,UpTo@bucketsize];
bins=Table[MinMax[i],{i,Values@aimpartitioned}];
repetitivesreport=DeleteCases[Tally@bins,x_/;x[[2]]==1];
repetitives=repetitivesreport[[All,1]];

labelgeneration[x_]:=Table[{repetitives[[x]][[1]]+i*repetitives[[x]][[1]]
/(2*10^(RealDigits@repetitives[[x]][[1]]+1)[[2]]),repetitives[[x]][[2]]+
i*repetitives[[x]][[2]]/(2*10^(RealDigits@repetitives[[x]][[2]]+1)[[2]])},
{i,Range@repetitivesreport[[x]][[2]]}];

binsrearranged=ReplacePart[bins,Flatten[Table[MapThread[#1->#2&,
{Flatten@Position[bins,repetitives[[g]]],labelgeneration[g]}],
{g,Range@Length@repetitives}],1]];

aimbinned=Values@Sort[Flatten[Table[aimpartitioned[[i]]/.
Dispatch@Table[Values@aimpartitioned[[i]][[j]]->binsrearranged[[i]],
{j,Length@aimpartitioned[[i]]}],{i,Length@aimpartitioned}],1],#1[[1]]<#2[[1]]&];
{aimbinned,campaign,seri}]
```

```
Clear[snetworkdatabinnedintimewindows]
snetworkdatabinnedintimewindows[data_,feature_,step_,datadimension_]:=Module[{rawaim,pos
campaign,seri,min,max,binningamount,binning},
rawaim=Table[data[[i]][[All,feature]],{i,Range@datadimension}];
pos=Table[Partition[Flatten@Table[Position[k,i],{i,{"NA",0}}],1],{k,rawaim}];
aim=Table[Delete[i[[1]],i[[2]]],{i,MapThread[{#1,#2}&,{rawaim,pos}]}];
campaign=Table[Delete[data[[i[[1]]]][[All,2]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
seri=Table[Delete[data[[i[[1]]]][[All,1]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
min=Table[Floor[Min[Sort[DeleteDuplicates[i]]],0.1],{i,aim}];
max=Table[Ceiling[Max[Sort[DeleteDuplicates[i]]]+step,{i,aim}];
binningamount=Table[Length[DeleteCases[BinLists[i[[1]],{i[[2]],i[[3]],step}],{}]],
{i,MapThread[{#1,#2,#3}&,{aim,min,max}]}];
binning=Table[Table[Catch[Do[If[IntervalMemberQ[Interval[i],(k[[1]])[[j]]]==True,Throw
{i,Partition[Range[k[[2]],k[[3]],step],2,1]}]],{j,Length[k[[1]]]}],{k,MapThread[{#1,#2
{aim,min,max}]}];
aim=Table[DeleteCases[i,Null],{i,binning}];
{aim,campaign,seri}]
```

```
Clear[snetworkdatafxdbucketintimewindows]
snetworkdatafxdbucketintimewindows[data_,feature_,nodenumber_,datadimension_]:=Module[{r
pos,aim,campaign,seri,bucketsize,aimlabeled,aimpartitioned,bins,repetitivesreport,repetit
labelgeneration,binsrearranged,aimbinned},
rawaim=Table[data[[i]][[All,feature]],{i,Range@datadimension}];
pos=Table[Partition[Flatten@Table[Position[k,i],{i,{"NA",0}}],1],{k,rawaim}];
aim=Table[Delete[i[[1]],i[[2]]],{i,MapThread[{#1,#2}&,{rawaim,pos}]}];
campaign=Table[Delete[data[[i[[1]]]][[All,2]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
seri=Table[Delete[data[[i[[1]]]][[All,1]],i[[2]]],
{i,MapThread[{#1,#2}&,{Range@datadimension,pos}]}];
bucketsize=Table[Ceiling@(N@(Dimensions@i)/nodenumber),{i,aim}];
aimlabeled=Table[Thread[Range@Length@i->i],{i,aim}];
aimpartitioned=Table[Partition[Normal@Sort@Association@i[[1]],UpTo@i[[2]]],
{i,MapThread[{#1,#2}&,{aimlabeled,bucketsize}]}];
bins=Table[Table[MinMax[i],{i,Values@k}],{k,aimpartitioned}];
repetitivesreport=Table[DeleteCases[Tally@i,x_/;x[[2]]==1],{i,bins}];
repetitives=Table[i[[All,1]],{i,repetitivesreport}];

labelgeneration[x_,dim_]:=Table[{repetitives[[dim]][[x]][[1]]+i*repetitives[[dim]][[x]
/(2*10^(RealDigits@repetitives[[dim]][[x]][[1]]+1)[[2]]),repetitives[[dim]][[x]][[2]]
i*repetitives[[dim]][[x]][[2]]/(2*10^(RealDigits@repetitives[[dim]][[x]][[2]]+1)[[2]]
{i,Range@repetitivesreport[[dim]][[x]][[2]]}];

binsrearranged=Table[ReplacePart[bins[[o]],Flatten[Table[MapThread[#1->#2&,
{Flatten@Position[bins[[o]],repetitives[[o]][[g]]],labelgeneration[g,o]}],
{g,Range@Length@repetitives[[o]]}],1]],{o,Range@datadimension}];

aimbinned=Table[Values@Sort[Flatten[Table[aimpartitioned[[k]][[i]]/.
Dispatch@Table[Values@aimpartitioned[[k]][[i]][[j]]->binsrearranged[[k]][[i]],
{j,Length@aimpartitioned[[k]][[i]]}],{i,Length@aimpartitioned[[k]]}],1],#1[[1]]<#2[[1]
{k,Range@datadimension}];{aimbinned,campaign,seri}]
```

```
Clear[snetworkgraph]
snetworkgraph[aim_,campaign_,vertexsize_,vertexlabelsize_,imagesize_,vertexcolor_]:=Modu
binningmembers,aimbaskets,singlesupportvalues,pairs,pairsupportvalues,liftvalues,
allmatrixelements,likelypairs,binarymatrix,graph},
binningmembers=Sort[DeleteDuplicates[aim]];
aimbaskets=Table[DeleteDuplicates[i],{i,Values@GroupBy[Thread[{aim,campaign}],Last->Firs
singlesupportvalues=Table[N[Count[Table[MemberQ[i,j],{i,aimbaskets}],
True]/Length[aimbaskets]],{j,binningmembers}];
pairs=Subsets[binningmembers,{2}];
pairsupportvalues=Table[N[Count[Table[SubsetQ[i,j],{i,aimbaskets}],True]/
Length[aimbaskets]],{j,pairs}];
liftvalues=pairsupportvalues/(DeleteCases[Flatten@UpperTriangularize[Table[singlesupport
[[j]]*singlesupportvalues[[k]],{j,Length[binningmembers]},{k,Length[binningmembers]}],1
allmatrixelements=Sort[Join[pairs,Reverse[pairs,2],Table[{i,i},{i,binningmembers}]]];
likelypairs=Extract[pairs,Position[liftvalues,x_/;x>1]];
binarymatrix=ArrayReshape[Table[If[j==True,1,0],{j,Table[MemberQ[likelypairs,i],
{i,allmatrixelements}]}],{Length@binningmembers,Length@binningmembers}];
graph=AdjacencyGraph[binarymatrix,{GraphLayout->Automatic,DirectedEdges->False,
EdgeShapeFunction->"Line",VertexSize->vertexsize,VertexStyle->vertexcolor,
VertexLabelStyle->Directive[Black,Italic,vertexlabelsize],VertexLabels->Flatten[MapThrea
{#1->Placed[#2,Center]}&,{Range[1,Dimensions[binarymatrix][[1]]],Table[StringRiffle[i,"
{i,binningmembers}]}]]},ImageSize->imagesize];{graph,Length@binningmembers}]
```

```
Clear[correlationfunction]
correlationfunction[network_,choice_]:=Module[{De,CC,BC,DeBC,CCBC,out},
De=VertexDegree[network];
CC=LocalClusteringCoefficient[network];
BC=BetweennessCentrality[network];
DeBC=Which[choice==1,SpearmanRho[De,BC],choice==2,Correlation[De,BC],choice==3,
KendallTau[[De,BC]]];
CCBC=Which[choice==1,SpearmanRho[CC,BC],choice==2,Correlation[CC,BC],choice==3,
KendallTau[[CC,BC]]];
out={DeBC,CCBC}]
```

```
Clear[randomnessfunction]
randomnessfunction[network_,choice_]:=
Module[{randomgraphs,MuDeBC,MuCCBC,XDeBC,XCCBC,SigmaDeBC,SigmaCCBC,ZDeBC,ZCCBC,final},
SeedRandom[17];
randomgraphs=RandomGraph[{VertexCount[network],EdgeCount[network]},1000];
MuDeBC=Mean[Table[correlationfunction[randomgraphs[[i]],choice][[1]],{i,1000}]];
MuCCBC=Mean[Table[correlationfunction[randomgraphs[[i]],choice][[2]],{i,1000}]];
XDeBC=correlationfunction[network,choice][[1]];
XCCBC=correlationfunction[network,choice][[2]];
SigmaDeBC=StandardDeviation[Table[correlationfunction[randomgraphs[[i]],choice][[1]],{i
SigmaCCBC=StandardDeviation[Table[correlationfunction[randomgraphs[[i]],choice][[2]],{i
ZDeBC=(XDeBC-MuDeBC)/SigmaDeBC;
ZCCBC=(XCCBC-MuCCBC)/SigmaCCBC;
final={ZDeBC,ZCCBC}]
```

```
Clear[GirvanNewmanmodularity]
GirvanNewmanmodularity[xx_]:=Module[{network,m,Aij,B,u,β,s,Q,div1,div2,moditer,r1,r11,r1
r21,r22,subfinal1Q,subfinal2Q,finalQ},
network=xx;
m=EdgeCount@network;
Aij=AdjacencyMatrix@network;
B=N@(Aij-Outer[Times,VertexDegree@network,VertexDegree@network]/(2*m));
u=Eigenvectors@B;
β=Eigenvalues@B;
s=(u/.x_/;x<0->-1)/.x_/;x>=0->1;
Q=(Total@(Table[(u[[i]] . Flatten@s[[Flatten@Position[β,Max@β][[1]]]])^2,
{i,VertexList@network}]*β))/(4*m);
div1=Flatten@Position[Flatten@s[[Flatten@Position[β,Max@β][[1]]]],_?Negative];
div2=Flatten@Position[Flatten@s[[Flatten@Position[β,Max@β][[1]]]],_?Positive];
moditer[pos_,B1_]:=Module[{Bg,ug,βg,sg,deltaQ,divv1,divv2,Bg1,Bg2},
Bg=(Table[B1[[i,j]]-KroneckerDelta[i,j]*Total@B1[[i,pos]],{i,Length@B1},{j,Length@B1}]
[[pos,pos]];
ug=Eigenvectors@Bg;
βg=Eigenvalues@Bg;
sg=(ug/.x_/;x<0->-1)/.x_/;x>=0->1;
deltaQ=(Total@(Table[(ug[[i]] . Flatten@sg[[Flatten@Position[βg,Max@βg][[1]]]])^2,
{i,Range@Length@pos}]*βg))/(4*m);
divv1=Flatten@Position[Flatten@sg[[Flatten@Position[βg,Max@βg][[1]]]],_?Negative];
divv2=Flatten@Position[Flatten@sg[[Flatten@Position[βg,Max@βg][[1]]]],_?Positive];
{deltaQ,divv1,divv2,Bg}];
r1=moditer[div1,B];
r11=If[r1[[1]]>0.001,moditer[r1[[2]],r1[[4]]],{0,0,0,0}];
r12=If[r1[[1]]>0.001,moditer[r1[[3]],r1[[4]]],{0,0,0,0}];
r2=moditer[div2,B];
r21=If[r2[[1]]>0.001,moditer[r2[[2]],r2[[4]]],{0,0,0,0}];
r22=If[r2[[1]]>0.001,moditer[r2[[3]],r2[[4]]],{0,0,0,0}];
subfinal1Q=If[r1[[1]]>0.001,If[r11[[1]]>0.001,If[r12[[1]]>0.001,r12[[1]]+r11[[1]]+r1[[
r11[[1]]+r1[[1]]],If[r12[[1]]>0.001,r12[[1]]+r1[[1]],r1[[1]]]],0];
subfinal2Q=If[r2[[1]]>0.001,If[r21[[1]]>0.001,If[r22[[1]]>0.001,r22[[1]]+r21[[1]]+r2[[
r21[[1]]+r2[[1]]],If[r22[[1]]>0.001,r22[[1]]+r2[[1]],r2[[1]]]],0];
finalQ=Q+subfinal1Q+subfinal2Q]
```

```
Clear[randomizedgraphamongcommunities]
randomizedgraphamongcommunities[network_]:=
Module[{networknodes,communitylist,intralinks,interlinks,intralinkamount,rewiredintra,
interlinknodes,interlinkamount,rewiredinter,rewiredgraph},
networknodes=VertexList@network;
communitylist=FindGraphCommunities[network];
intralinks=EdgeList[Subgraph[network,#]]&/@communitylist;
interlinks=Complement[EdgeList[network],Flatten@intralinks];
intralinkamount=Table[Length@intralinks[[i]],{i,Length@communitylist}];
rewiredintra=Table[#1⧟#2&@@@RandomSample[
Subsets[communitylist[[i]],{2}],intralinkamount[[i]]],{i,Length@communitylist}];
interlinknodes=DeleteDuplicates@Join[interlinks[[All,1]],interlinks[[All,2]]];
interlinkamount=Length@interlinknodes;
rewiredinter=#1⧟#2&@@@RandomSample[Subsets[interlinknodes,{2}],interlinkamount];
rewiredgraph=Graph[networknodes,Flatten[{rewiredintra,rewiredinter}]]]
```

```
Clear[randomnessfunctionformodularity]
randomnessfunctionformodularity[network_,modularitytype_]:=
Module[{X,randomgraphserdösrenyi,modularityerdösrenyi,μerdösrenyi,σerdösrenyi,
randomgraphsdegreesfxd,modularitydegreesfxd,μdegreesfxd,σdegreesfxd,randomgraphscomm,
modularitycomm,μcomm,σcomm,ZScores},

X=Which[modularitytype=="GN",GirvanNewmanmodularity[network],modularitytype=="Wolf",
N@GraphAssortativity[network,FindGraphCommunities[network],"Normalized"->False]];

randomgraphserdösrenyi=RandomGraph[{VertexCount[network],EdgeCount[network]},1000];
modularityerdösrenyi=Which[modularitytype=="GN",Table[GirvanNewmanmodularity[
randomgraphserdösrenyi[[i]]],{i,1000}],modularitytype=="Wolf",Table[N@GraphAssortativity
randomgraphserdösrenyi[[i]],FindGraphCommunities[randomgraphserdösrenyi[[i]]],
"Normalized"->False],{i,1000}]];
μerdösrenyi=Mean[Table[modularityerdösrenyi[[i]],{i,1000}]];
σerdösrenyi=StandardDeviation[Table[modularityerdösrenyi[[i]],{i,1000}]];

randomgraphsdegreesfxd=Table[Symbol["IGDegreeSequenceGame"][Total[AdjacencyMatrix@networ
Method->"VigerLatapy"],{i,1000}];
modularitydegreesfxd=Which[modularitytype=="GN",Table[GirvanNewmanmodularity[
randomgraphsdegreesfxd[[i]]],{i,1000}],modularitytype=="Wolf",Table[N@GraphAssortativity
randomgraphsdegreesfxd[[i]],FindGraphCommunities[randomgraphsdegreesfxd[[i]]],
"Normalized"->False],{i,1000}]];
μdegreesfxd=Mean[Table[modularitydegreesfxd[[i]],{i,1000}]];
σdegreesfxd=StandardDeviation[Table[modularitydegreesfxd[[i]],{i,1000}]];

randomgraphscomm=Table[randomizedgraphamongcommunities[network],1000];
modularitycomm=Which[modularitytype=="GN",Table[GirvanNewmanmodularity[
randomgraphscomm[[i]]],{i,1000}],modularitytype=="Wolf",Table[N@GraphAssortativity[
randomgraphscomm[[i]],FindGraphCommunities[randomgraphscomm[[i]]],"Normalized"->False],
{i,1000}]];
μcomm=Mean[Table[modularitycomm[[i]],{i,1000}]];
σcomm=StandardDeviation[Table[modularitycomm[[i]],{i,1000}]];

ZScores={(X-μerdösrenyi)/σerdösrenyi,(X-μdegreesfxd)/σdegreesfxd,(X-μcomm)/σcomm}]
```

```
Clear[randomnessfunctionformodularityonenullmodel]
randomnessfunctionformodularityonenullmodel[network_]:=
Module[{X,randomgraphscomm,modularitycomm,μcomm,σcomm,ZScore},

X=N@GraphAssortativity[network,FindGraphCommunities[network],"Normalized"->False];

randomgraphscomm=Table[randomizedgraphamongcommunities[network],1000];
modularitycomm=Table[N@GraphAssortativity[randomgraphscomm[[i]],
FindGraphCommunities[randomgraphscomm[[i]]],"Normalized"->False],{i,1000}];
μcomm=Mean[Table[modularitycomm[[i]],{i,1000}]];
σcomm=StandardDeviation[Table[modularitycomm[[i]],{i,1000}]];

ZScore=(X-μcomm)/σcomm]
```

```
Clear[randomnessfunctionformodularitytwonullmodel]
randomnessfunctionformodularitytwonullmodel[network_]:=
Module[{X,randomgraphserdösrenyi,modularityerdösrenyi,μerdösrenyi,σerdösrenyi,
randomgraphscomm,modularitycomm,μcomm,σcomm,ZScores},

X=N@GraphAssortativity[network,FindGraphCommunities[network],"Normalized"->False];

randomgraphserdösrenyi=RandomGraph[{VertexCount[network],EdgeCount[network]},1000];
modularityerdösrenyi=Table[N@GraphAssortativity[
randomgraphserdösrenyi[[i]],FindGraphCommunities[randomgraphserdösrenyi[[i]]],
"Normalized"->False],{i,1000}];
μerdösrenyi=Mean[Table[modularityerdösrenyi[[i]],{i,1000}]];
σerdösrenyi=StandardDeviation[Table[modularityerdösrenyi[[i]],{i,1000}]];

randomgraphscomm=Table[randomizedgraphamongcommunities[network],1000];
modularitycomm=Table[N@GraphAssortativity[randomgraphscomm[[i]],
FindGraphCommunities[randomgraphscomm[[i]]],"Normalized"->False],{i,1000}];
μcomm=Mean[Table[modularitycomm[[i]],{i,1000}]];
σcomm=StandardDeviation[Table[modularitycomm[[i]],{i,1000}]];

ZScores={(X-μerdösrenyi)/σerdösrenyi,(X-μcomm)/σcomm}]
```

```
Clear[randomnessvaluesformodularitytwonullmodel]
randomnessvaluesformodularitytwonullmodel[network_]:=
Module[{X,randomgraphserdösrenyi,modularityerdösrenyi,randomgraphscomm,modularitycomm},

X=N@GraphAssortativity[network,FindGraphCommunities[network],"Normalized"->False];

randomgraphserdösrenyi=RandomGraph[{VertexCount[network],EdgeCount[network]},1000];
modularityerdösrenyi=Table[N@GraphAssortativity[
randomgraphserdösrenyi[[i]],FindGraphCommunities[randomgraphserdösrenyi[[i]]],
"Normalized"->False],{i,1000}];

randomgraphscomm=Table[randomizedgraphamongcommunities[network],1000];
modularitycomm=Table[N@GraphAssortativity[randomgraphscomm[[i]],
FindGraphCommunities[randomgraphscomm[[i]]],"Normalized"->False],{i,1000}];

{X,modularityerdösrenyi,modularitycomm}]
```

```
End[];
EndPackage[];
```