UNIVERSITÀ DELLA CALABRIA

PROGETTO DI ANALISI DI RETI SOCIALI

HIGH-ENERGY PHYSICS THEORY CITATION NETWORK

DATASET DESCRIPTION

Arxiv HEP-TH (high energy physics theory) citation graph is from the e-print arXiv and covers all the citations within a dataset of 27,770 papers with 352,807 edges. If a paper i cites paper j, the graph contains a directed edge from i to j. If a paper cites, or is cited by, a paper outside the dataset, the graph does not contain any information about this.

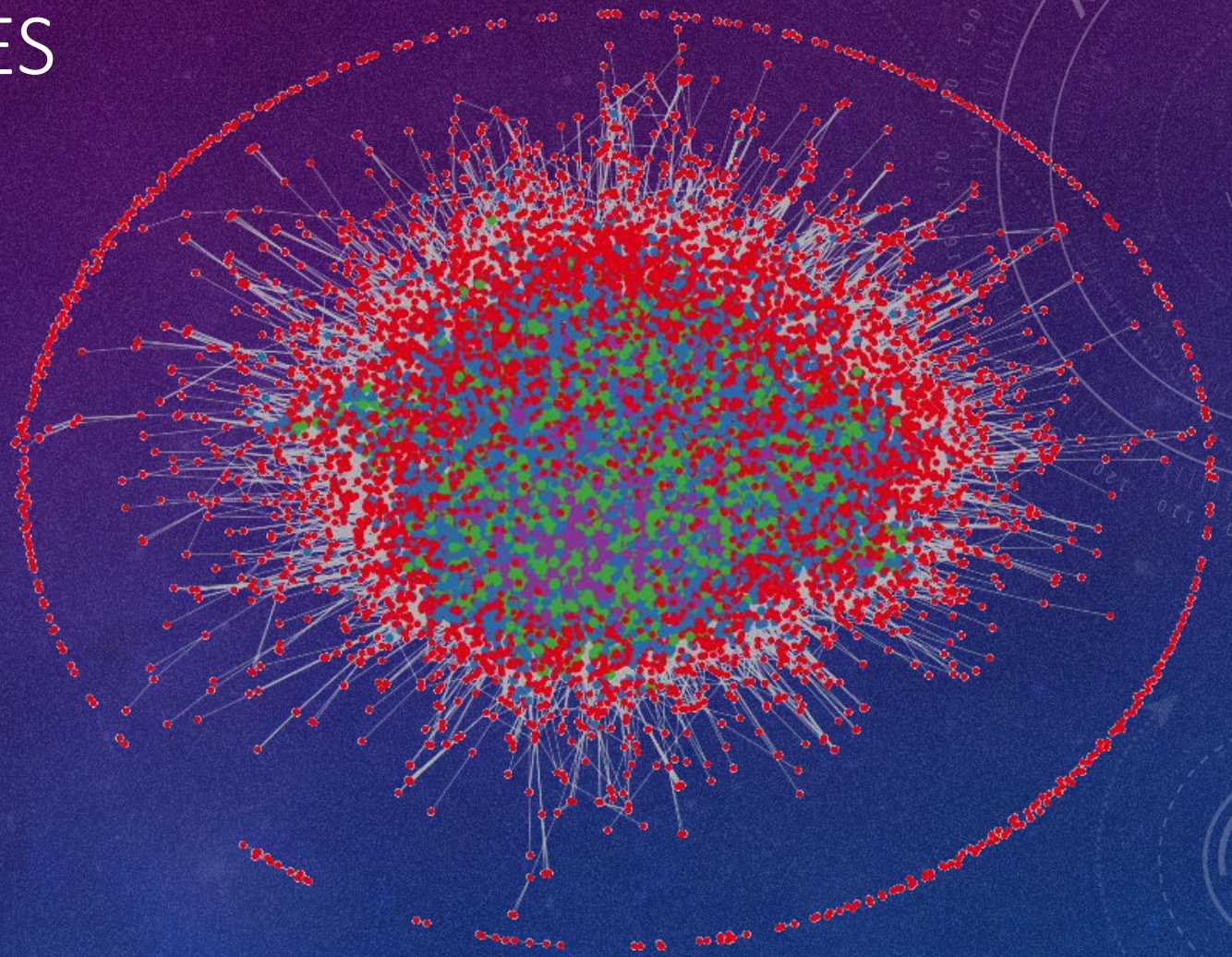
PROJECT OBJECTIVES

- Study the given dataset in Rstudio using the iGraph library.
- Analise network's properties: degree distribution, diameter, reciprocity, mean degree, measures of centrality (degree, closeness, betwenness), mean distance, clustering coefficient.
- Find affinities and differences with other defined network models (Random graph, Small World, Preferential Attachment, etc.).
- Apply a clustering detection algorithm and plot the results.
- Apply a link prediction algorithm.

NET STATIC PROPERTIES

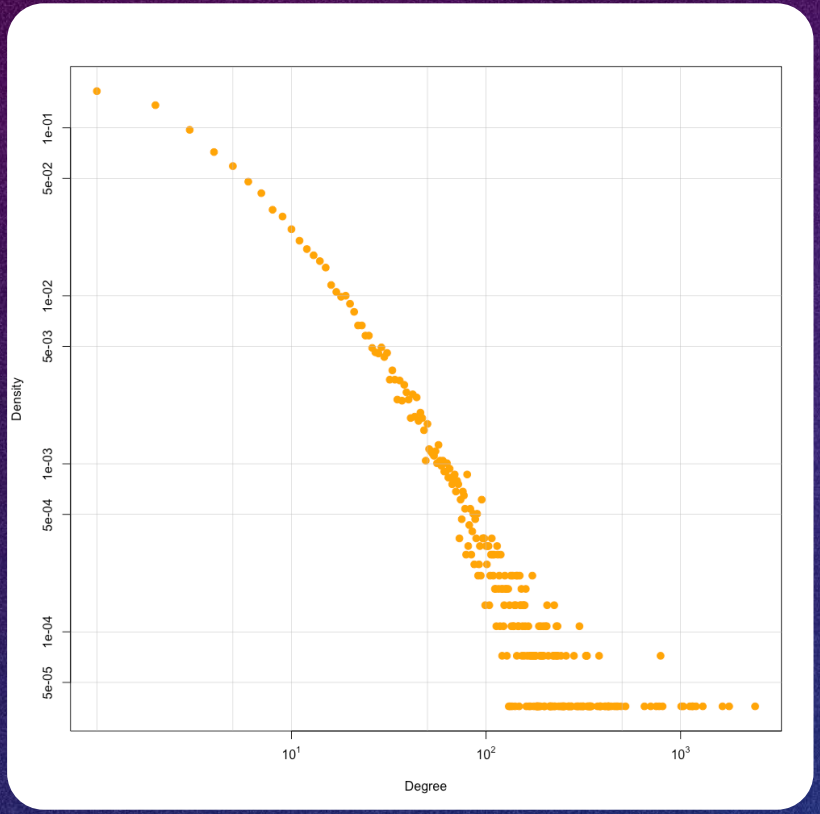
Nodes: 27770
Edges: 352807

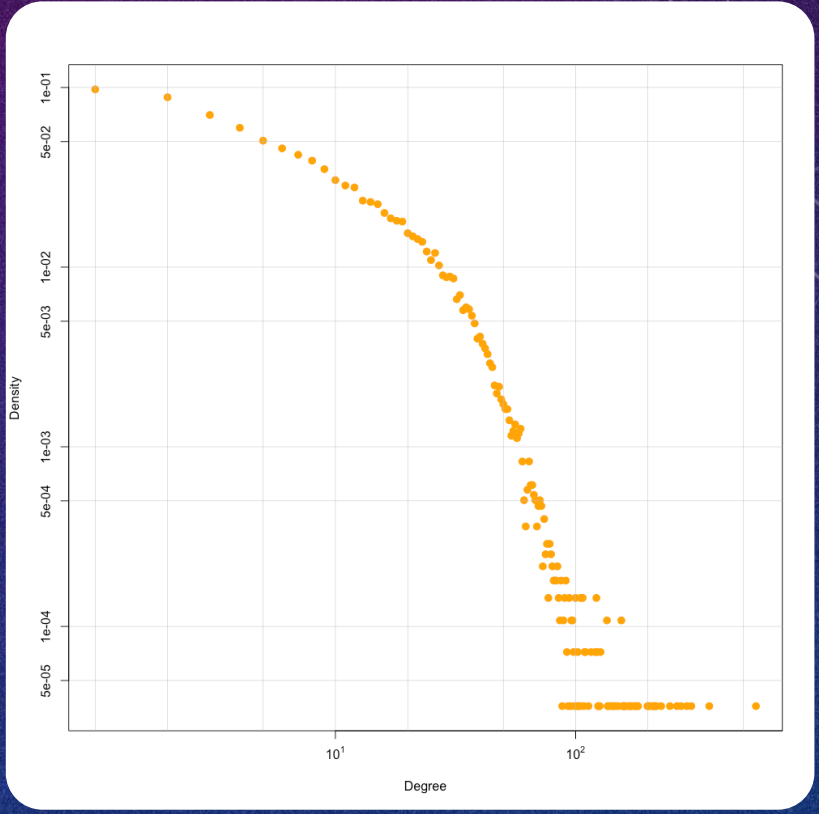
```
Degree:

[1, 6]
(6, 14]
(14, 31]
(31, 2470]
```



Density	Reciprocity	Diameter	Centrality Degree	Centrality Closeness	Centrality Betwenness	Cluster Coefficient	Mean Distance
0.0004575105	0.002738344	37	0.04398212	0.00009274	0.1071624	0.1195691	8.460137

IN-DEGREE & OUT-DEGREE DISTRIBUTION



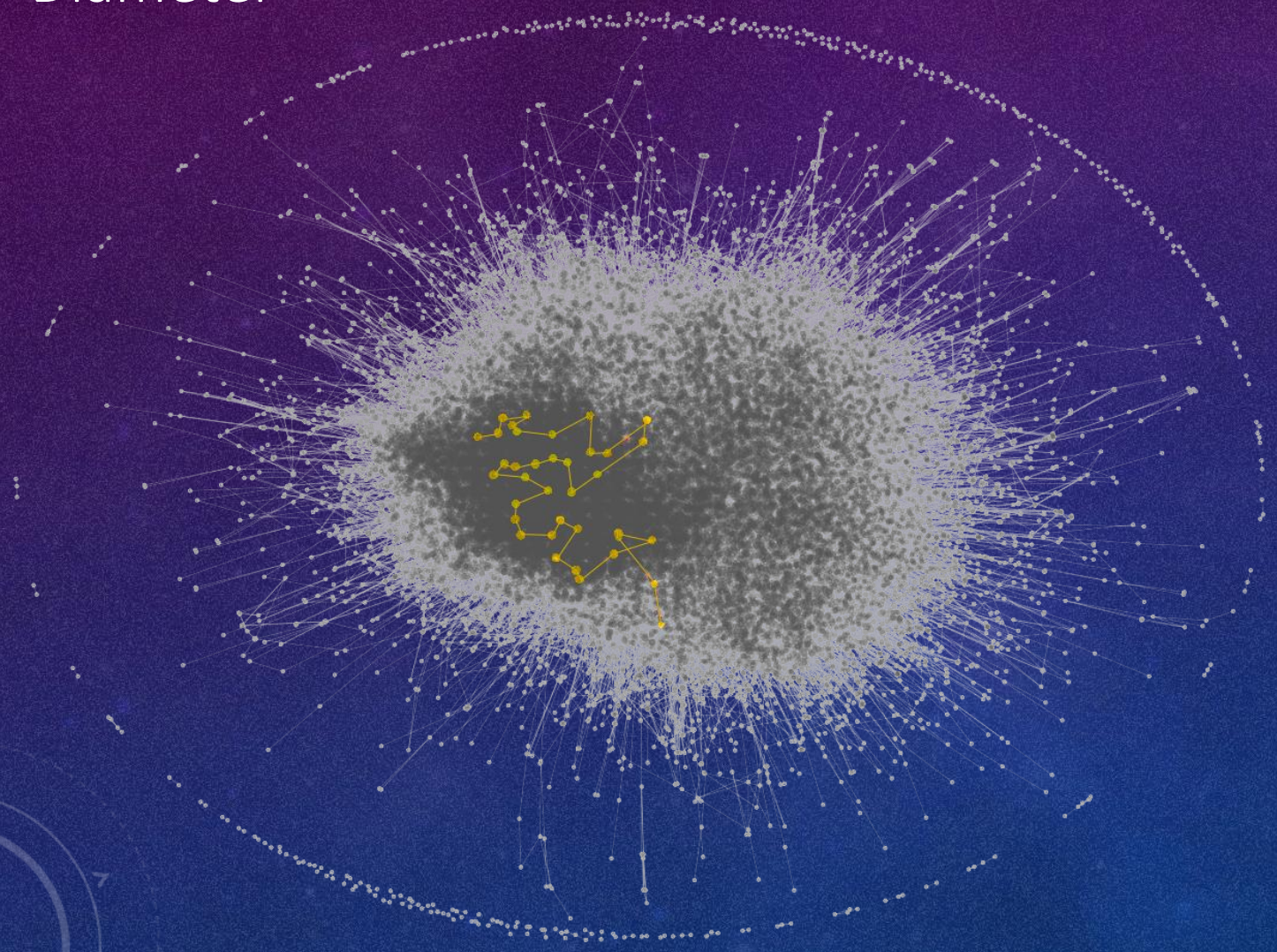


$$P_k \sim k^{-\alpha}$$
$$\alpha \approx 2.057342$$

$$P_k \sim k^{-\alpha}$$
$$\alpha \approx 4.133998$$

NET STATIC PROPERTIES

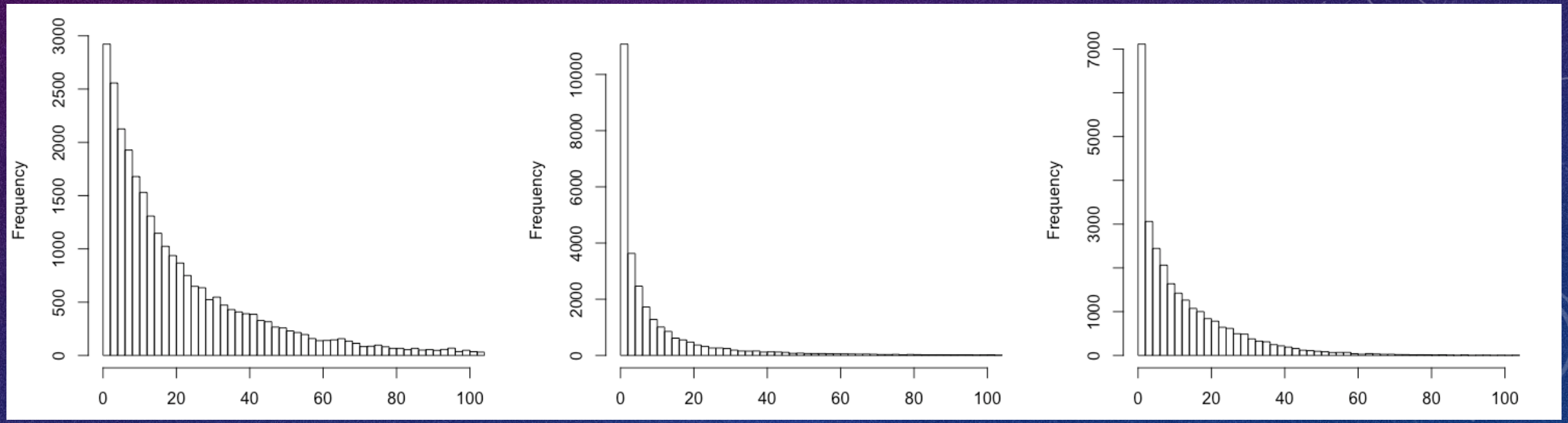
Diameter



Diameter (directed): 37
Diameter (undirected): 15

NET STATIC PROPERTIES

Centrality Degree



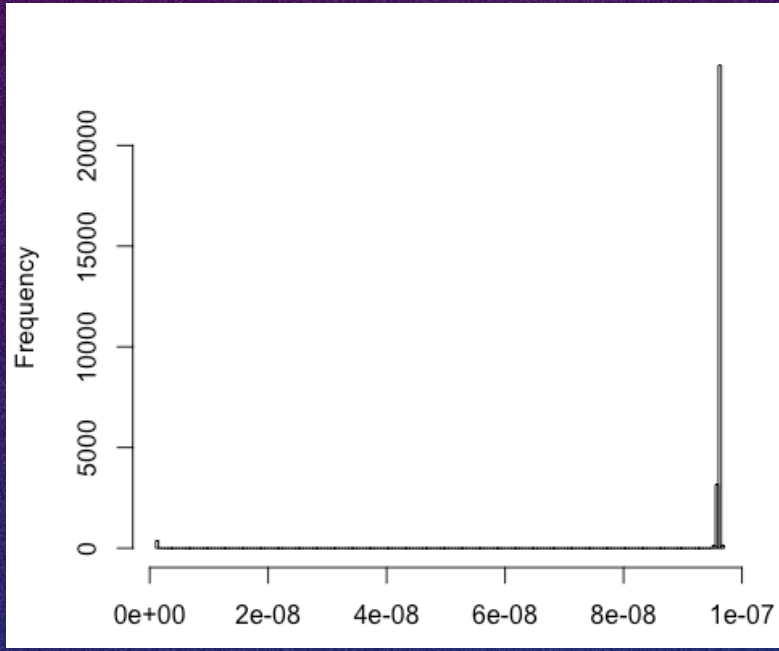
Degree

In-degree

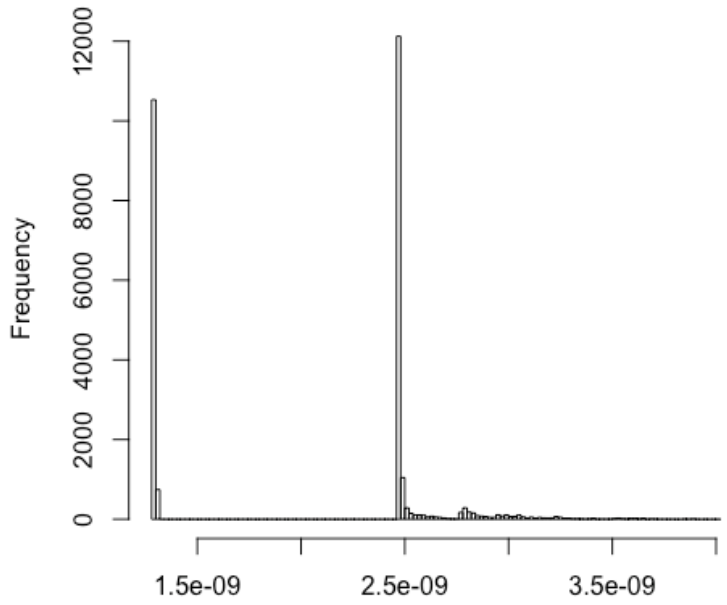
Out-degree

NET STATIC PROPERTIES

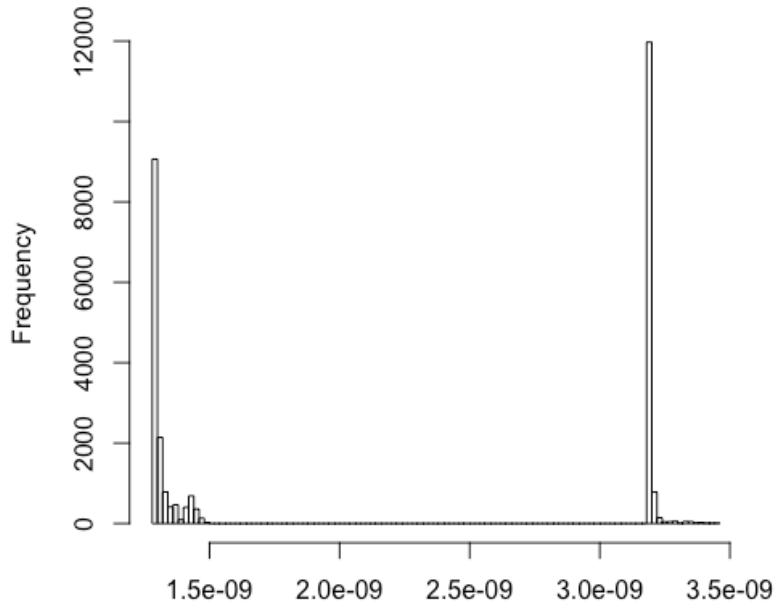
Centrality Closeness



Degree



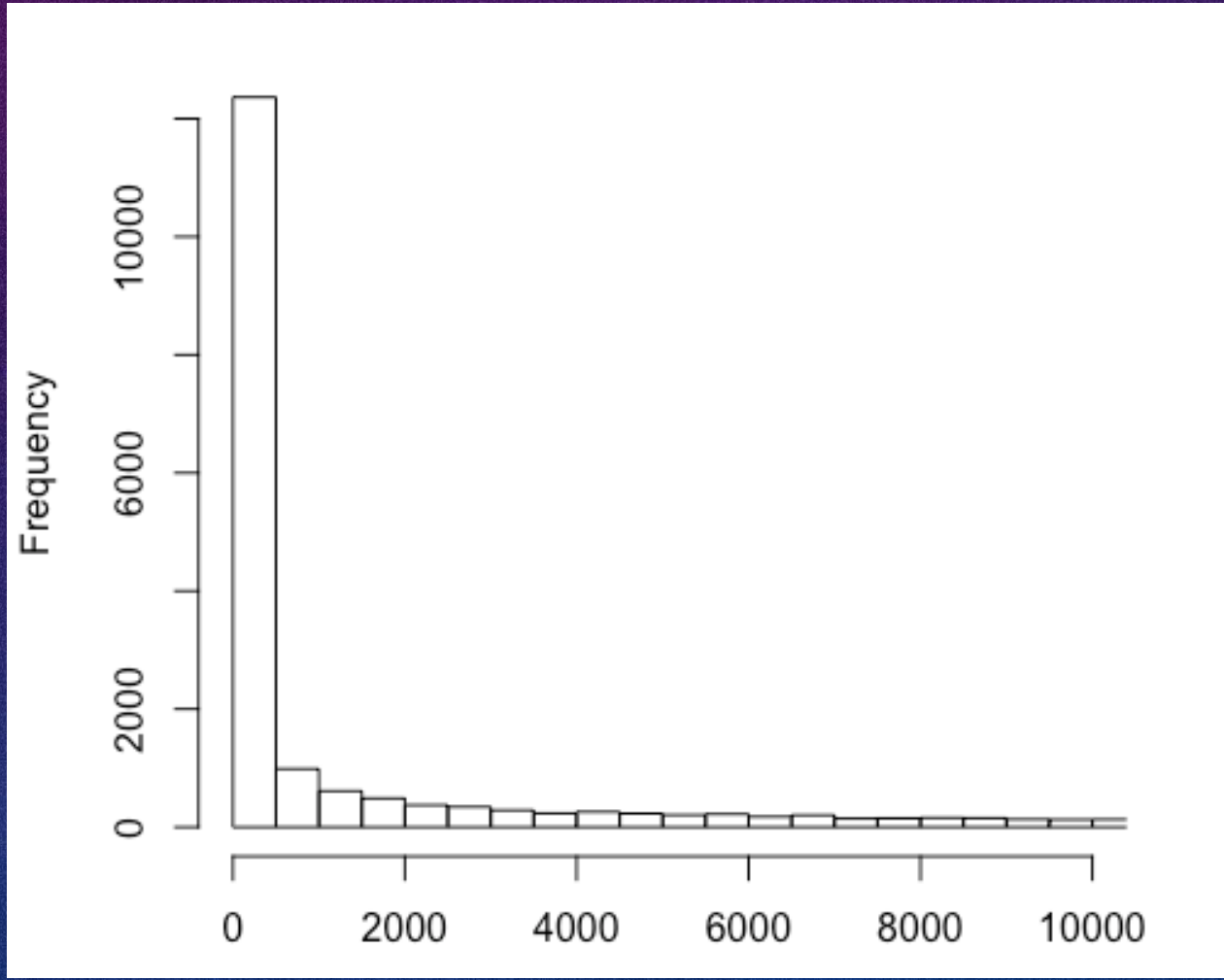
In-degree



Out-degree

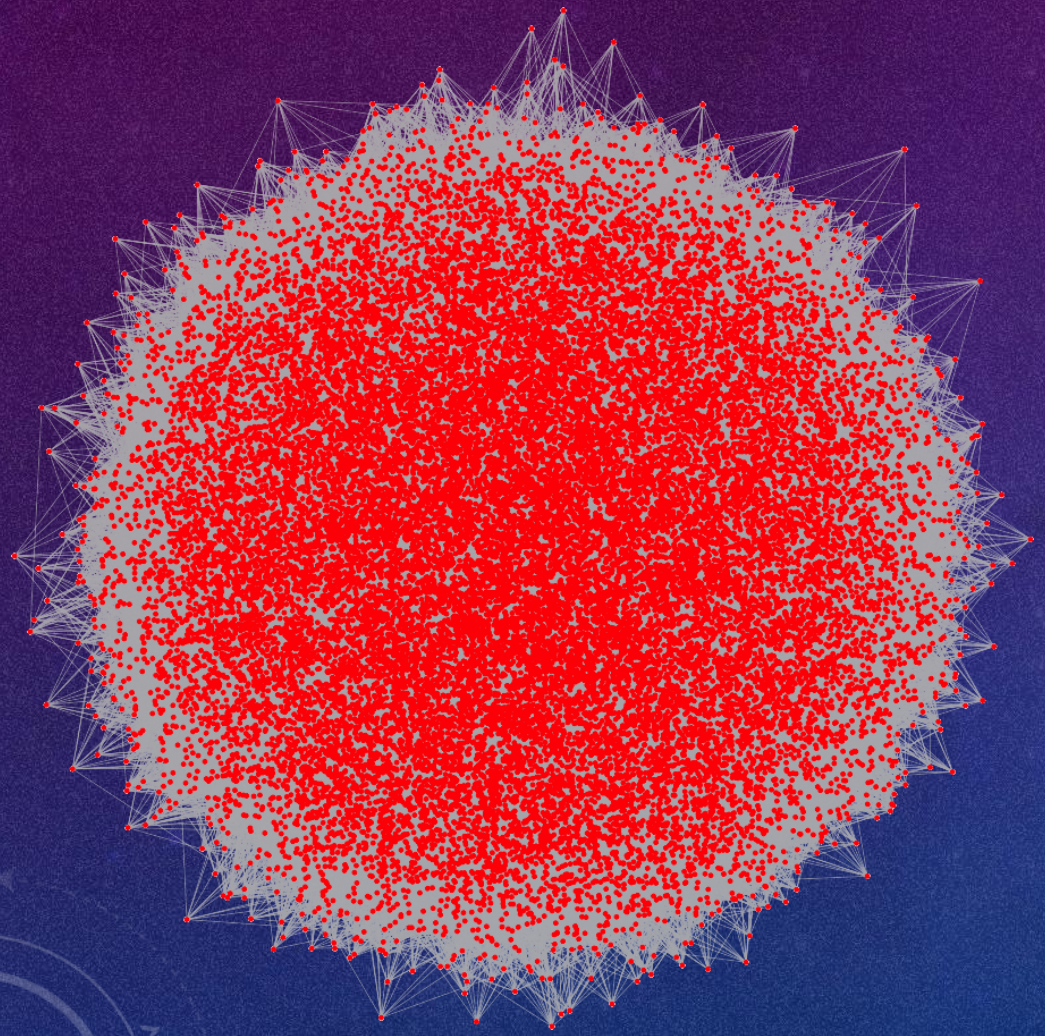
NET STATIC PROPERTIES

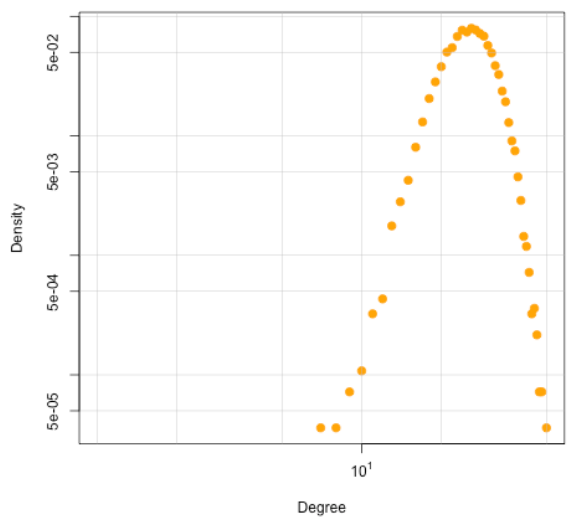
Centrality Betwenness

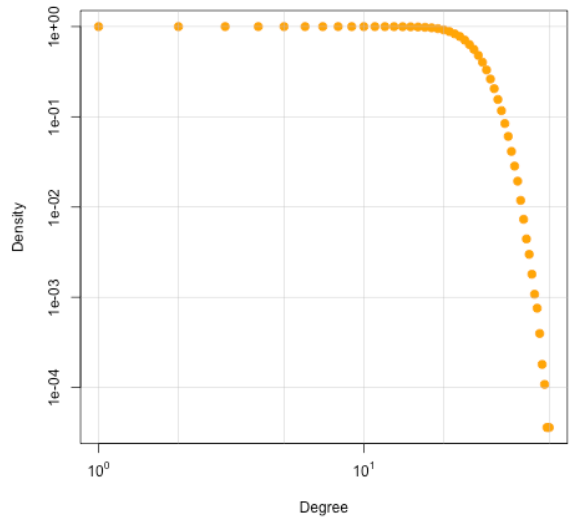


SIMULATED NETS

Random Graph





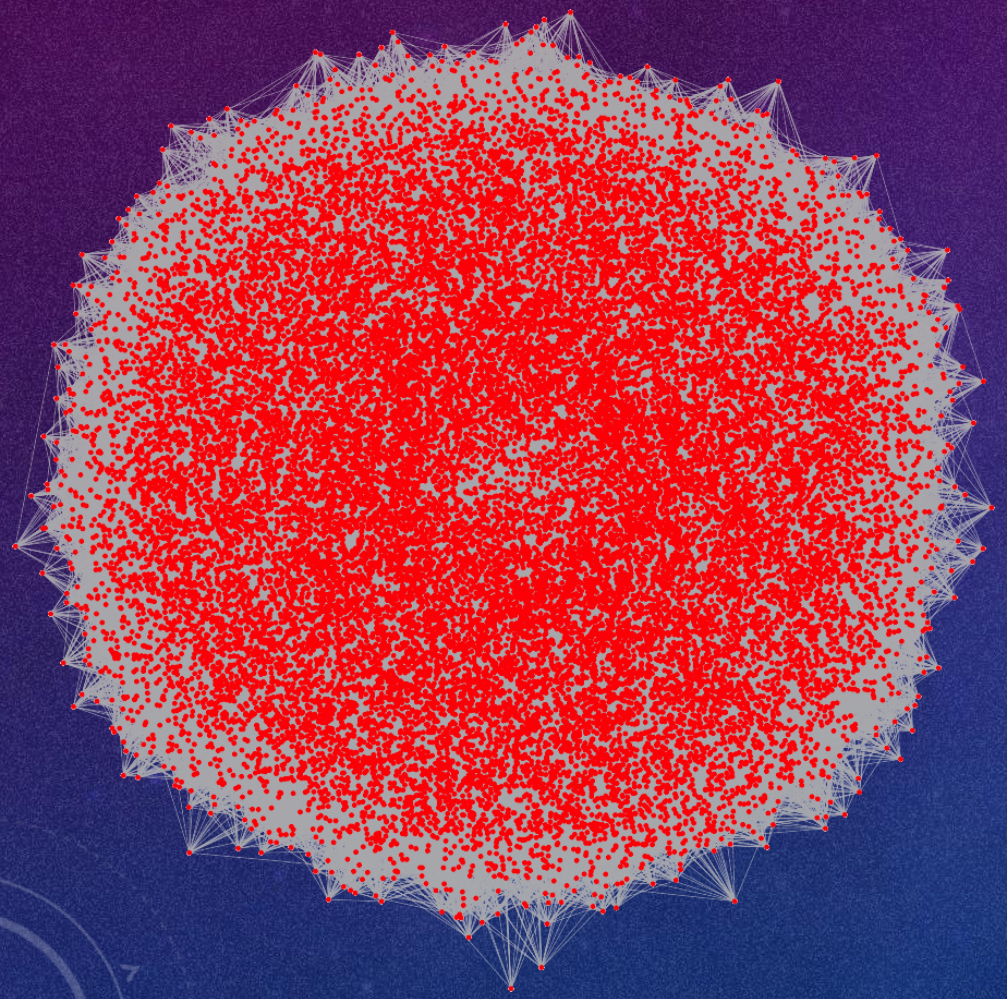


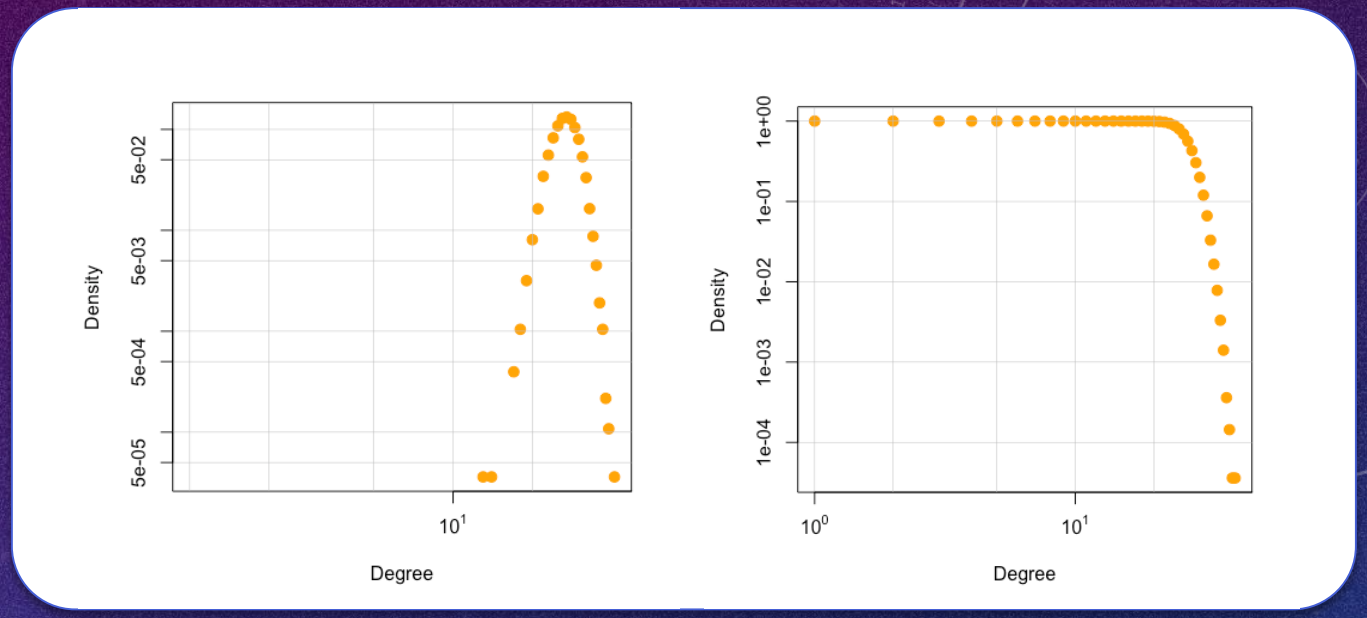
Properties

- Density: 0.0004575105
- Reciprocity: 0.000413824
- Diameter: 5
- Centrality degree: 0.0004247836
- Centrality closeness: 0.04522392
- Centrality betwenness: 0.0002294174
- Cluster coefficient: 0.0009278212
- Mean distance: 3.52385

SIMULATED NETS

Small World (Watts, Strogafz)



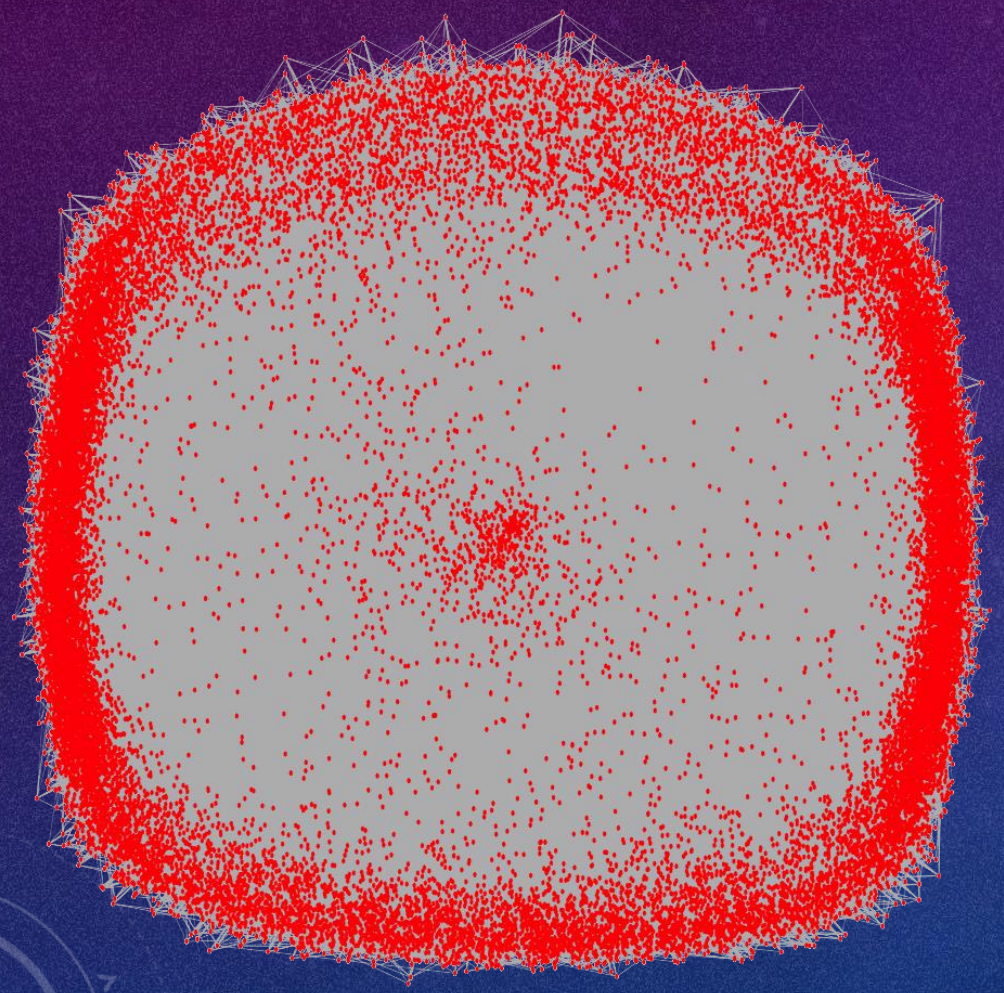


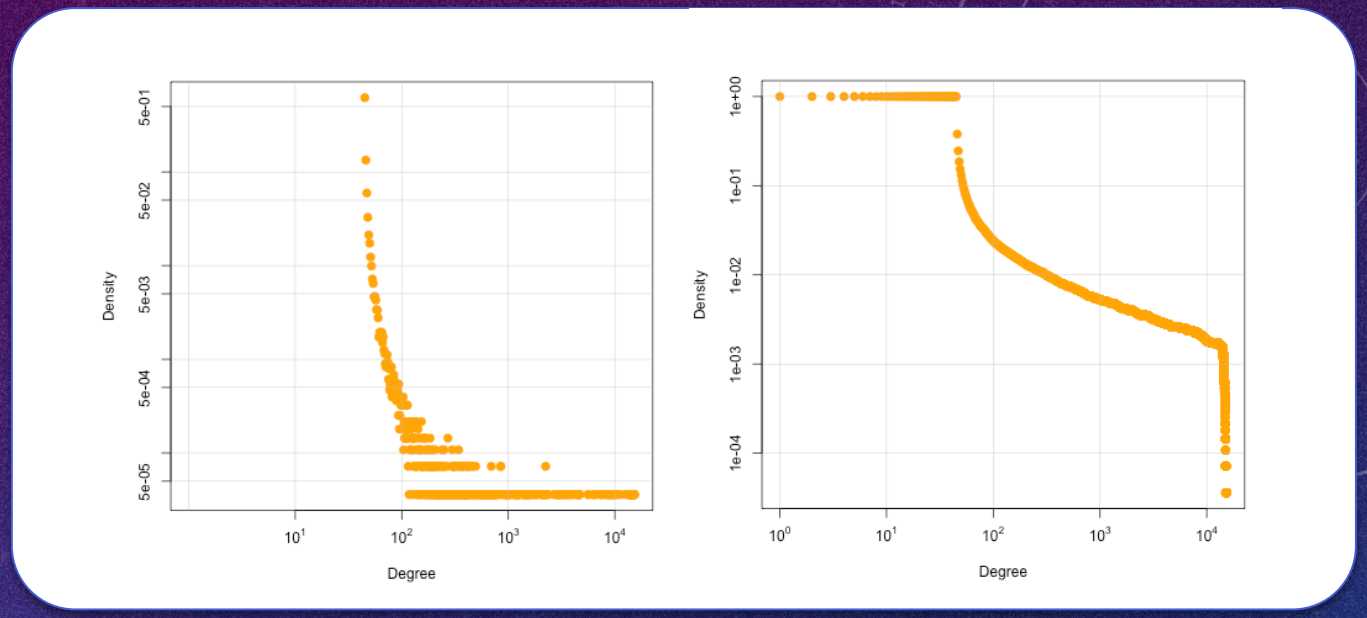
Properties

- Density: 0.0009362959
- Reciprocity: 1
- Diameter: 5
- Centrality degree: 0.0005041593
- Centrality closeness: 0.02202681
- Centrality betwenness: 0.0002070274
- Cluster coefficient: 0.2019233
- Mean distance: 3.738172

SIMULATED NETS

Preferential Attachment (Barabasi, Albert)



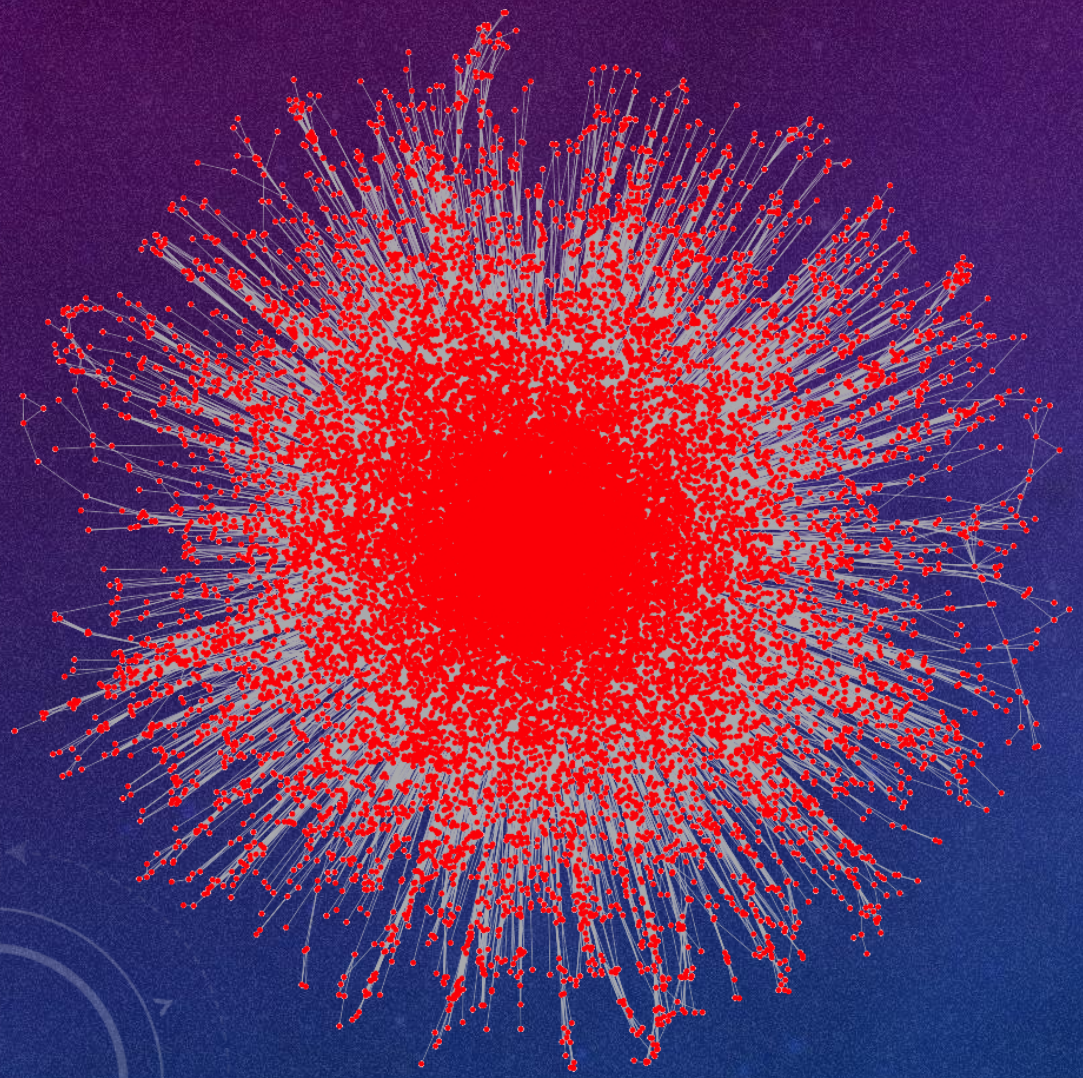


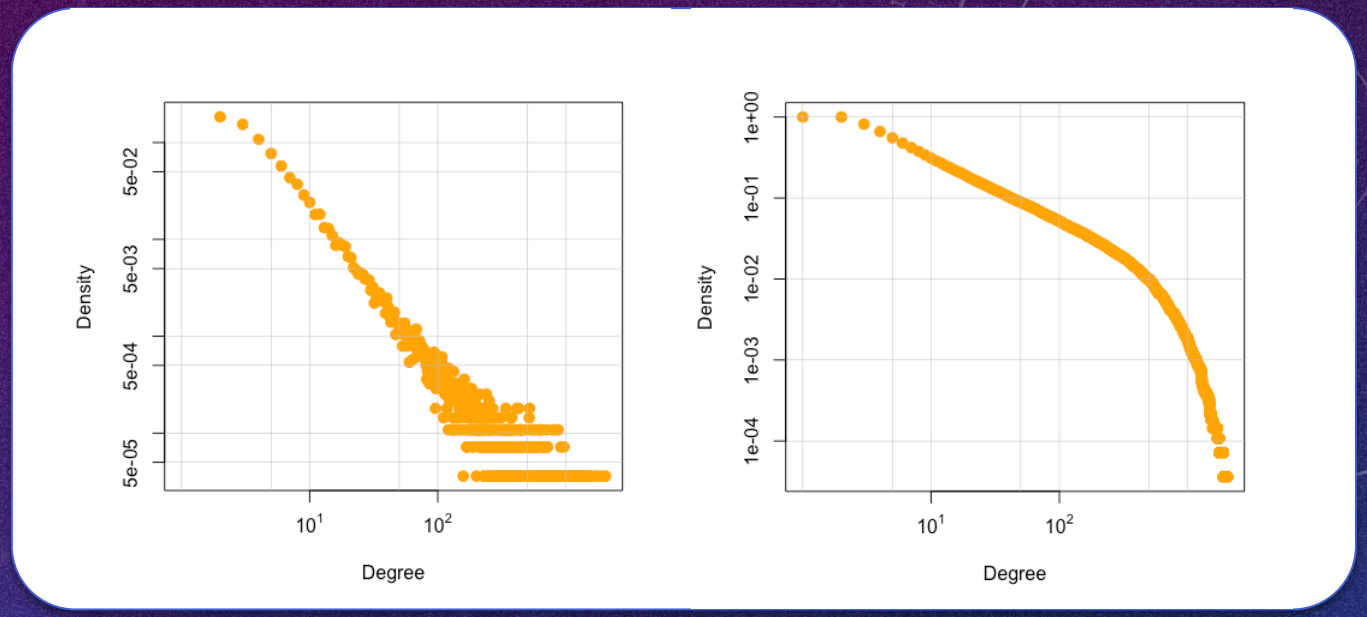
Properties

- Density: 0.0004320354
- Reciprocity: 0
- Diameter: 12
- Centrality degree: 0.1977817
- Centrality closeness: 0.2831583
- Centrality betwenness: 0.0000651589
- Cluster coefficient: 0.003306988
- Mean distance: 2.677891

SIMULATED NETS

Forest Fire (Henley, Drossel, Schwabl)



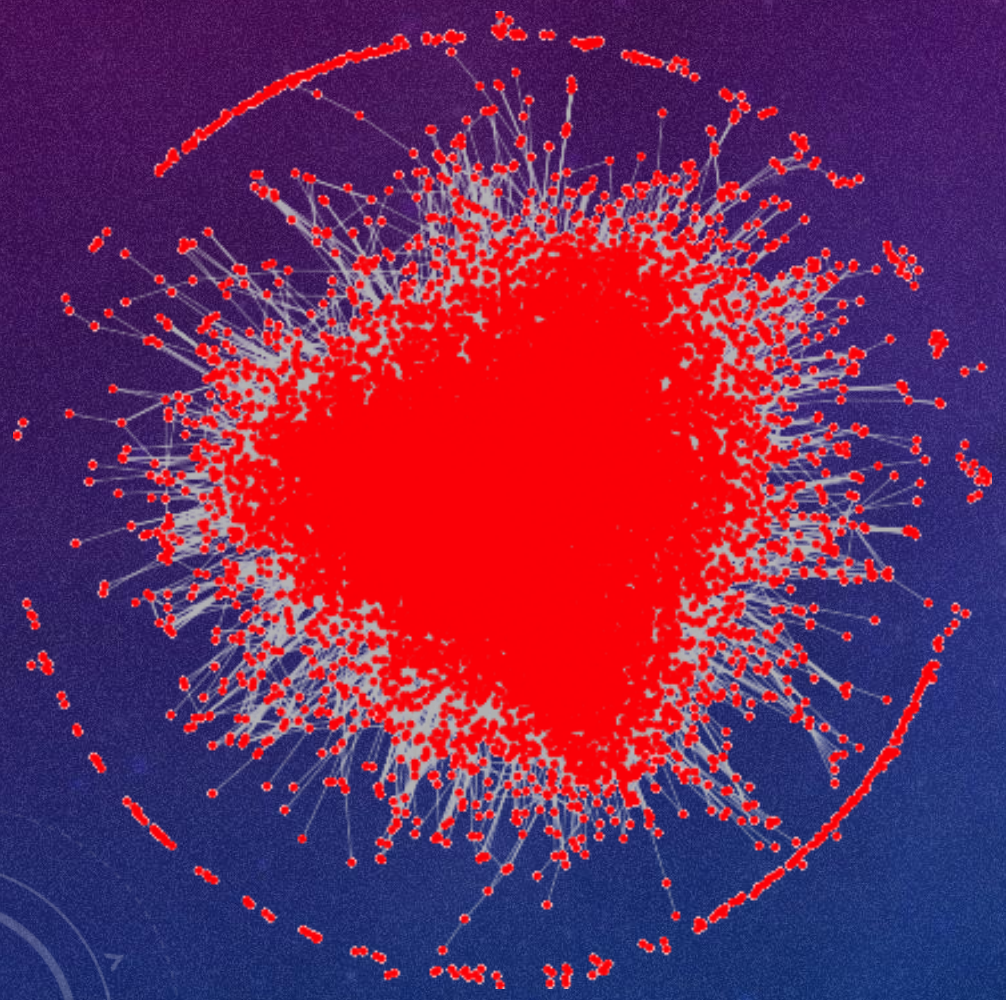


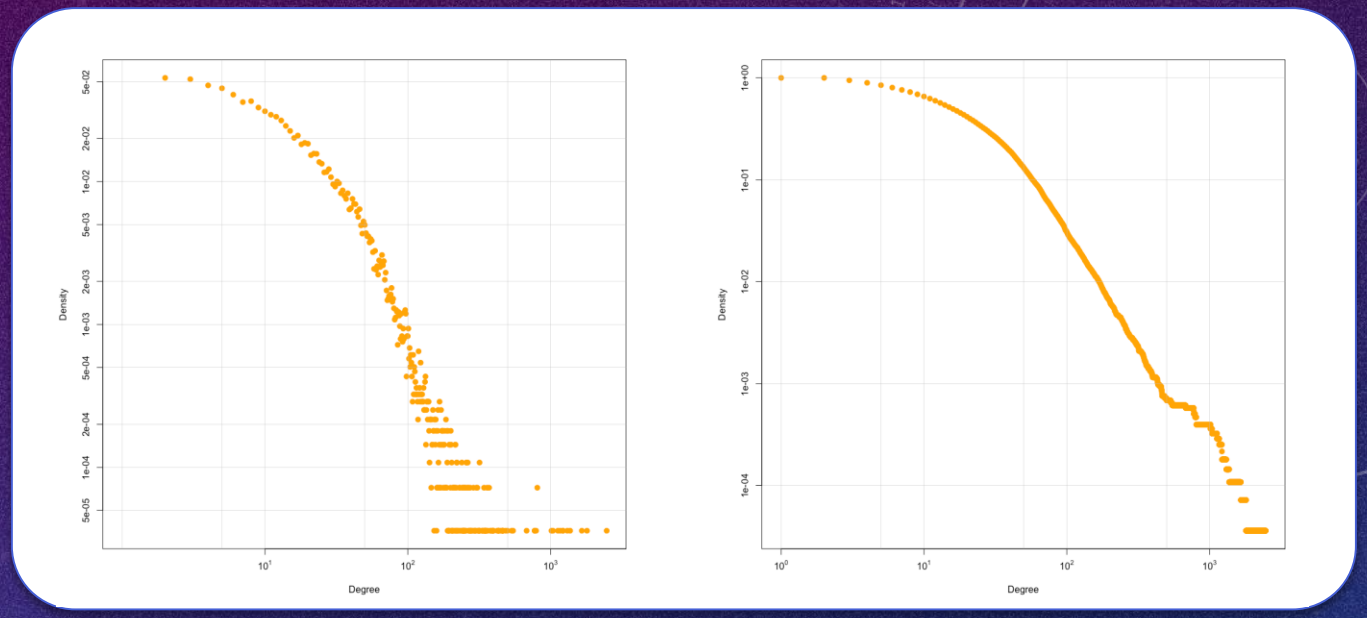
Properties

- Density: 0.0004641655
- Reciprocity: 0
- Diameter: 15
- Centrality degree: 0.03626875
- Centrality closeness: 0.268447
- Centrality betwenness: 0.001163897
- Cluster coefficient: 0.1713556
- Mean distance: 3.483749

SIMULATED NETS

Arxiv HEP-TH



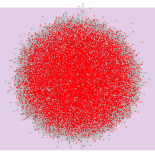
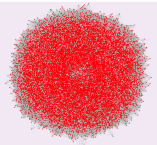
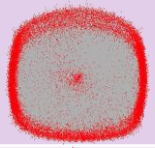
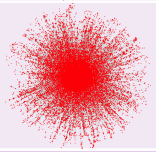
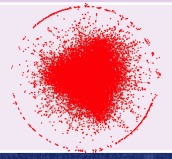


Properties

- Density: 0.0004575105
- Reciprocity: 0.002738344
- Diameter: 37
- Centrality degree: 0.04398212
- Centrality closeness: 0.00009274
- Centrality betwenness: 0.1071624
- Cluster coefficient: 0.1195691
- Mean distance: 8.460137

SIMULATED NETS

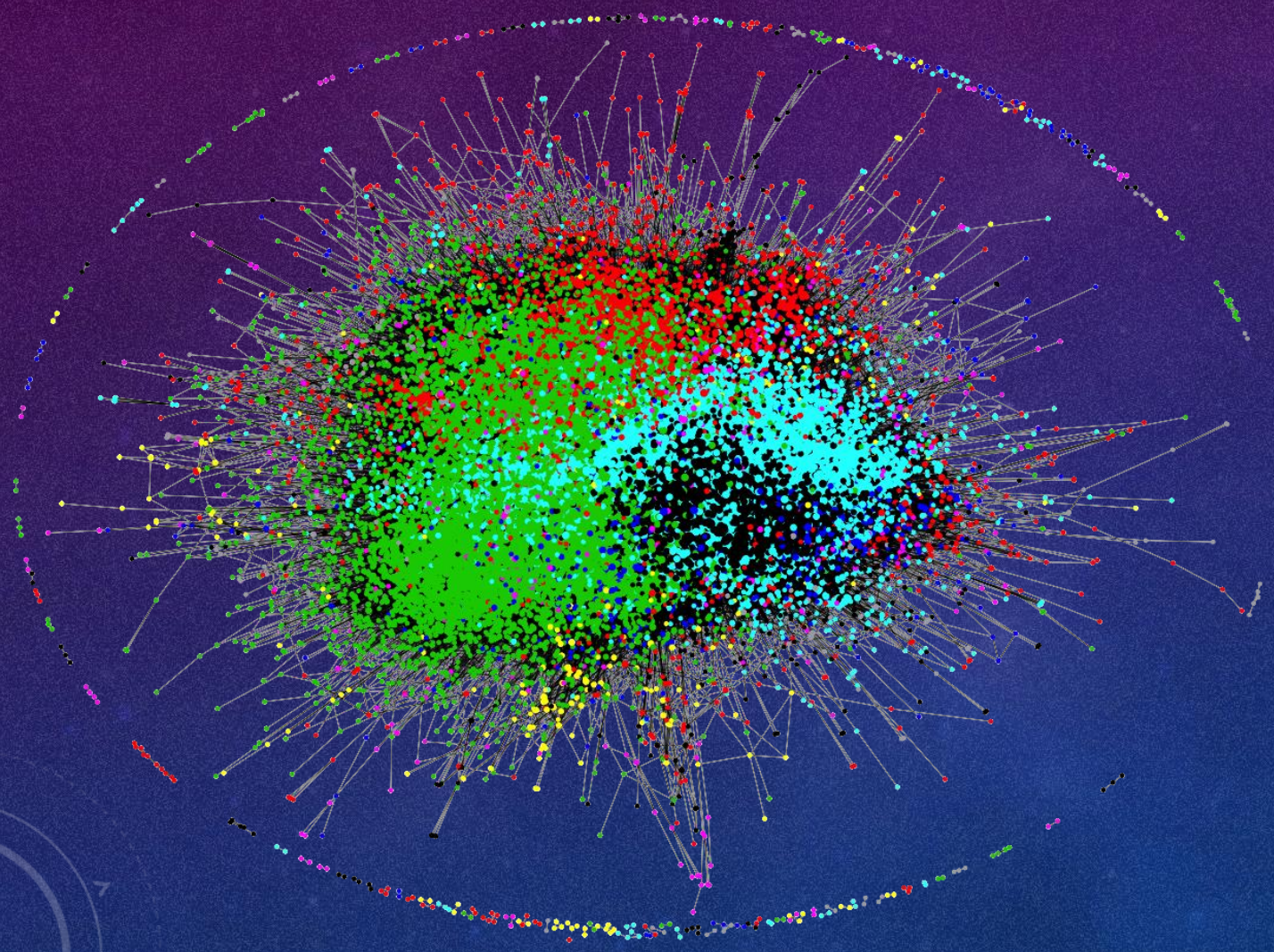
Results comparison

Graph Model	Plot	Density	Reciprocity	Diameter	Centrality Degree	Centrality Closeness	Centrality Betwenness	Cluster Coefficient	Mean Distance
Random Graph		0.0004575105	0.000413824	5	0.0004247836	0.04522392	0.0002294174	0.0009278212	3.52385
Small World		0.0009362959	1	5	0.0005041593	0.02202681	0.0002070274	0.2019233	3.738172
Preferential Attachment		0.0004320354	0	12	0.1977817	0.2831583	0.0000651589	0.003306988	2.677891
Forest Fire		0.0004641655	0	15	0.03626875	0.268447	0.001163897	0.1713556	3.483749
Last Cit	*	0.0004681311	0	12	0.006392267	0.169505	0.0001668864	0.004265398	3.472076
ArXiv HEP-TH		0.0004575105	0.002738344	37	0.04398212	0.00009274	0.1071624	0.1195691	8.460137

* Multiplex graphs (plot not available).

COMMUNITY DETECTION

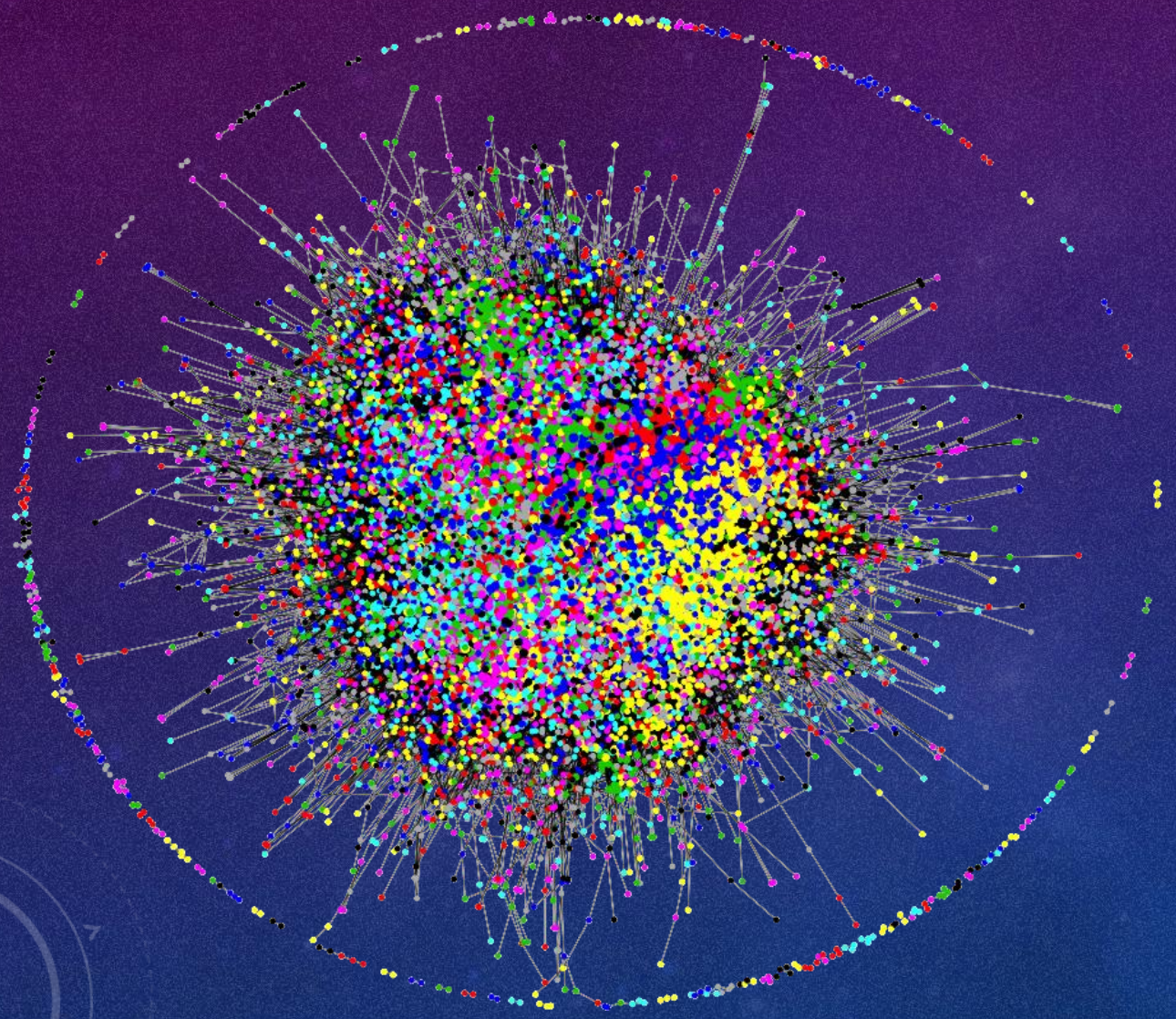
FastGreedy (A Clauset, MEJ Newman, C Moore)



- Length: 300
- Modularity: 0.5263447

COMMUNITY DETECTION

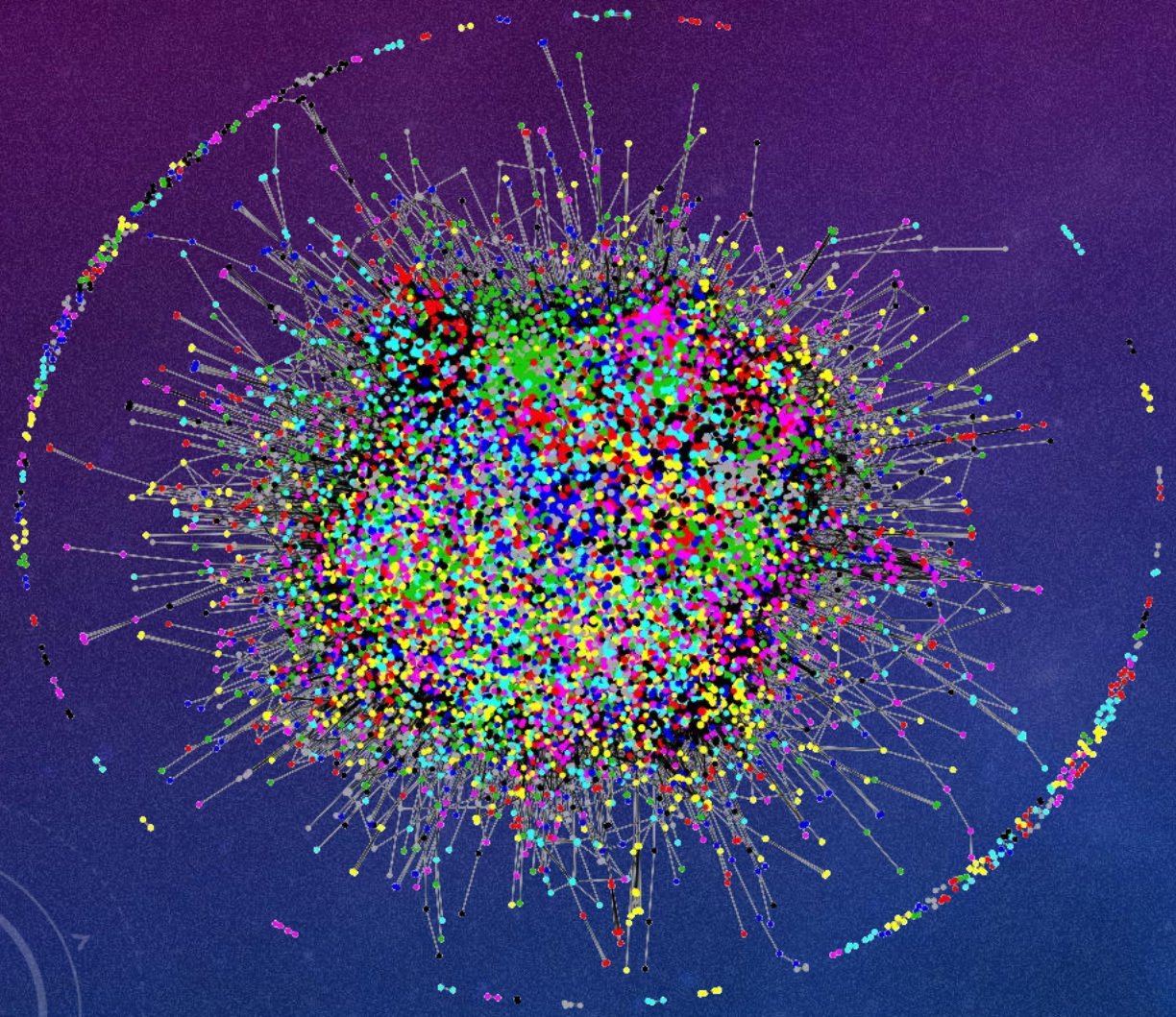
WalkTrap (Pascal Pons, Matthieu Latapy)



- Length: 2913
- Modularity: 0.546389

COMMUNITY DETECTION

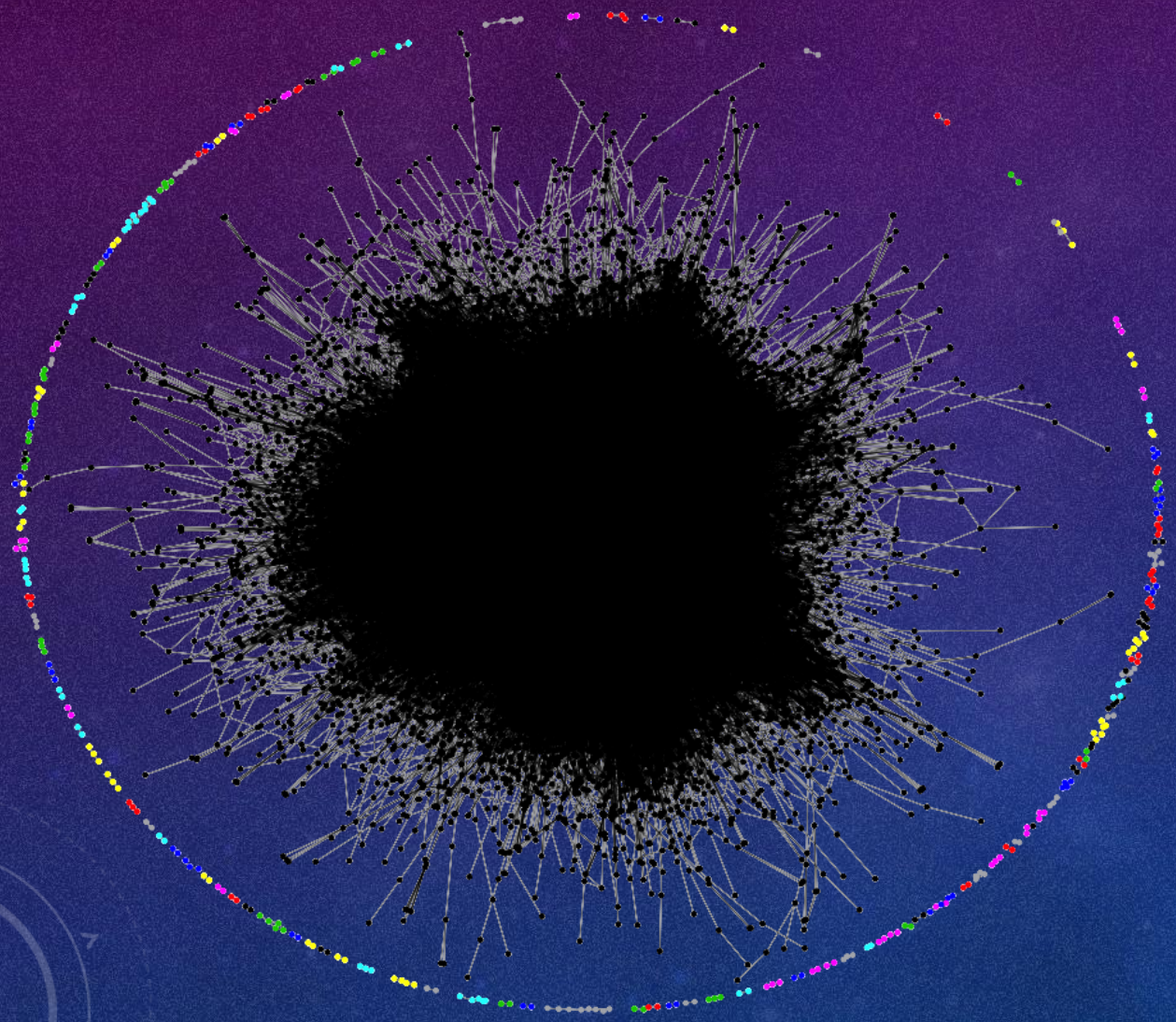
InfoMap (Rosvall, Bergstrom)



- Length: 1272
- Modularity: 0.5363231

COMMUNITY DETECTION

Eigenvector (MEJ Newman)



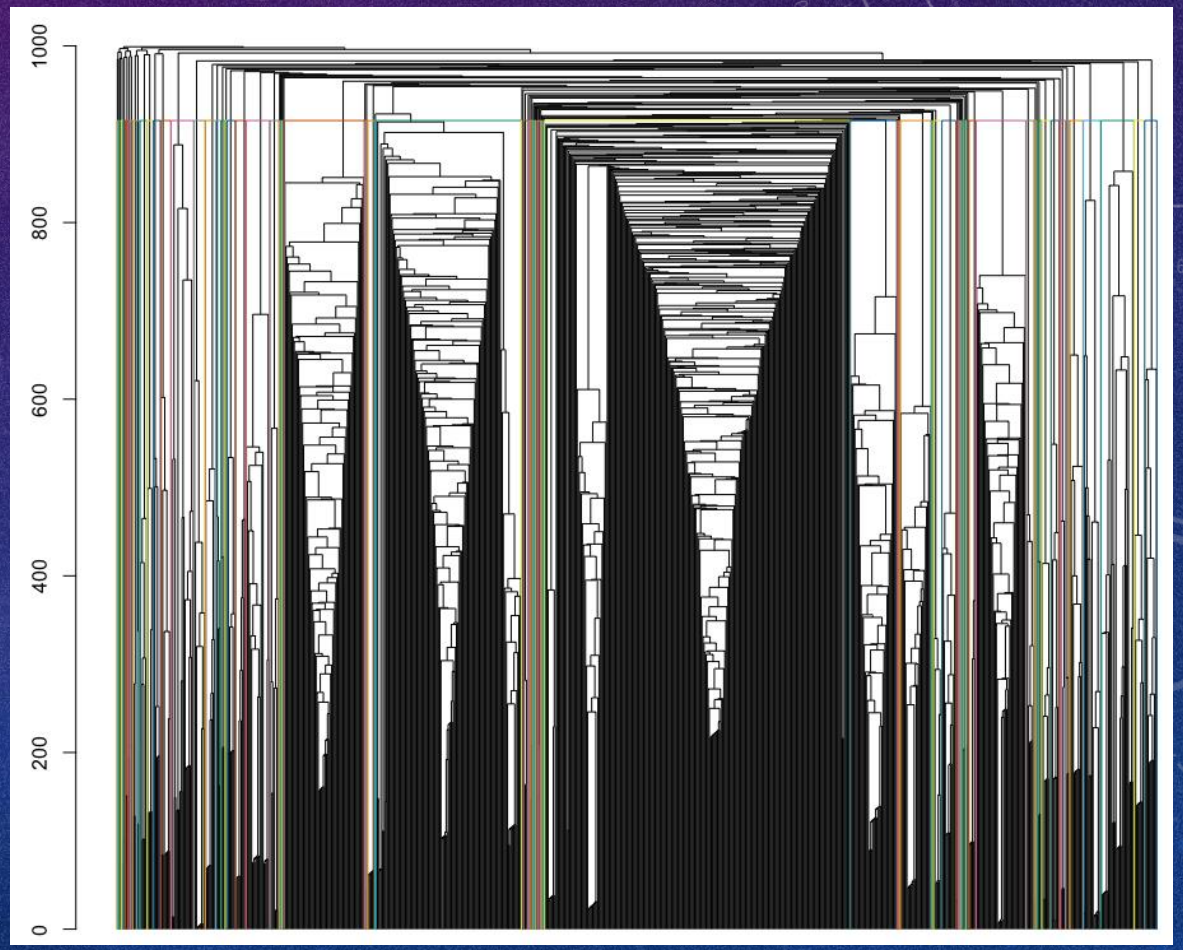
- Length: 143
- Modularity: 0.001503724

COMMUNITY DETECTION

Edge-Betwenness (Girvan, Newman)

Length: 85
Modularity: 0.5078089

**Values from a subnet of 1000 nodes.*



COMMUNITY DETECTION

Results comparison

Algorithm	Length	Modularity
FastGreedy	300	0.5263447
WalkTrap	2913	0.546389
InfoMap	1272	0.5363231
EigenVector	143	0.001503724
EdgeBetwenness	85*	0.5078089*

**Values from a subnet of 1000 nodes.*

LINK PREDICTION

R function *trainAndTest*: create the subgraphs test and training from the complete graph

```
trainAndTest<-function(g){
    test<-g-E(g)
    train<-g-E(g)
    for (v in V(g)){
        allOutVertex<-adjacent_vertices(g, v, mode = c("out"))
        total<-lengths(allOutVertex)
        if(total<2)
            next
        toTrain<-floor(total*0.8)
        for(i in 1:toTrain){
            train <- add.edges(train,c(v,allOutVertex[[1]][i]));
    }
    toTrain<-toTrain+1
    for(i in toTrain:total)
        test <- add.edges(test,c(v,allOutVertex[[1]][i]),attr=list(type=1))
    }
    trainAndTest <- list("train" = train, "test" = test)
    return (trainAndTest)
}
```

LINK PREDICTION

R function *addEdgesTest*: add random edges to test

```
addEdgesTest<-function(test,train,edgesToAdd){
    vertexList<-V(test)
    for (i in 1:edgesToAdd){
        ok<-F
        while(!ok){
                from <- as.numeric(sample(vertexList,1))
                to <- as.numeric(sample(vertexList,1))
                if(!are.connected(train,from, to) && !are.connected(test,from, to))
                    ok<-T
        }
        test <- add.edges(test,c(from,to),attr=list(type=0))
    }
    return (test)
}
```

LINK PREDICTION

R function *linkPrediction (part 1)*: calculate accuracy of link prediction algorithm

```
linkPrediction<-function(test,scoreMatrix,theta){
    tp<-0
    tn<-0
    fp<-0
    fn<-0
    edgeList<-E(test)
    for(i in 1:length(edgeList)){
        e<-edgeList[i]
        from<-match(ends(test,e)[1], V(test)$name)
        to<-match(ends(test,e)[2], V(test)$name)
        l<-e$type
        s<-scoreMatrix[to,from]
        if(l==1 && s>theta){
            tp<-tp+1
            next
        }
        if(l==0 && s<theta){
            tn<-tn+1
            next
        }
```

LINK PREDICTION

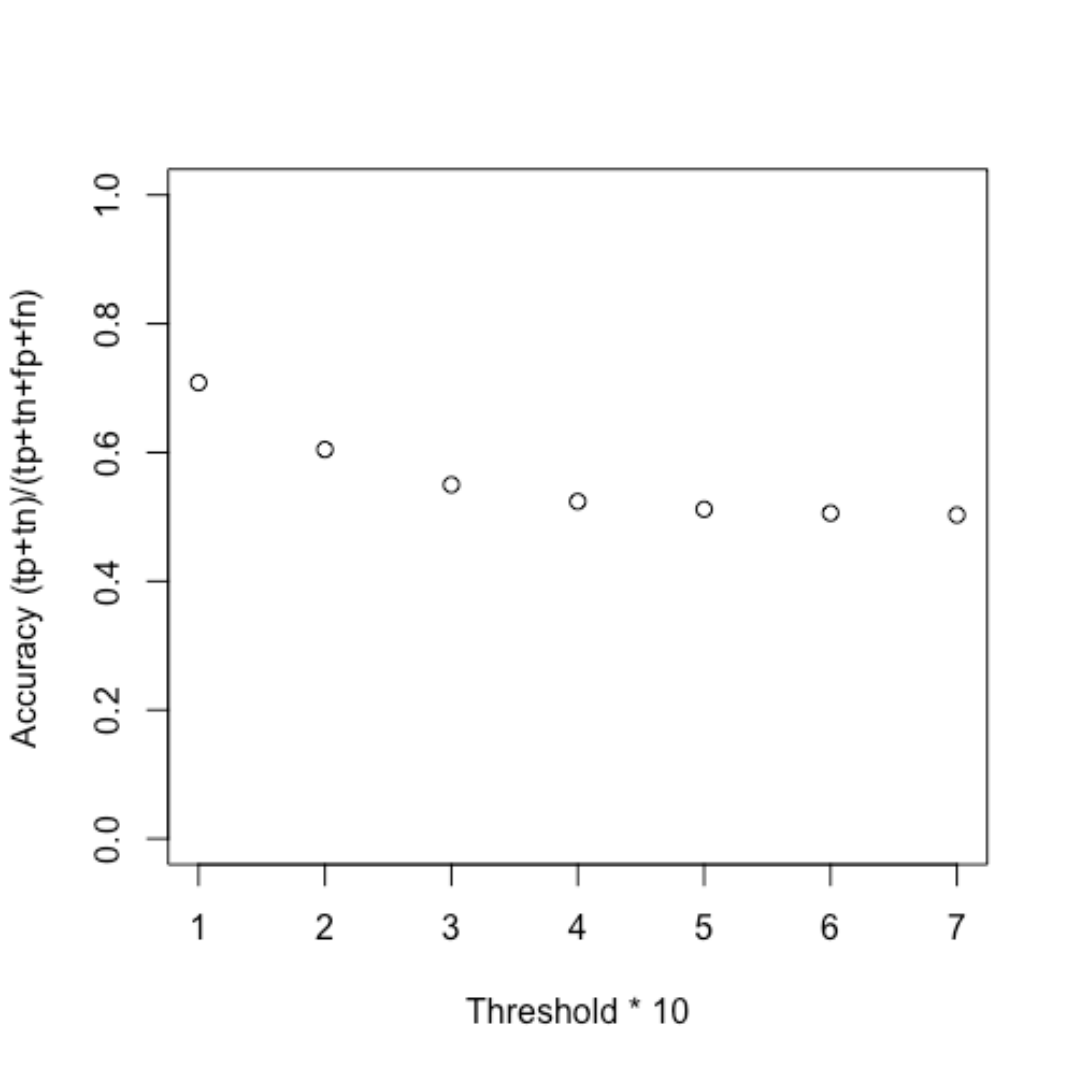
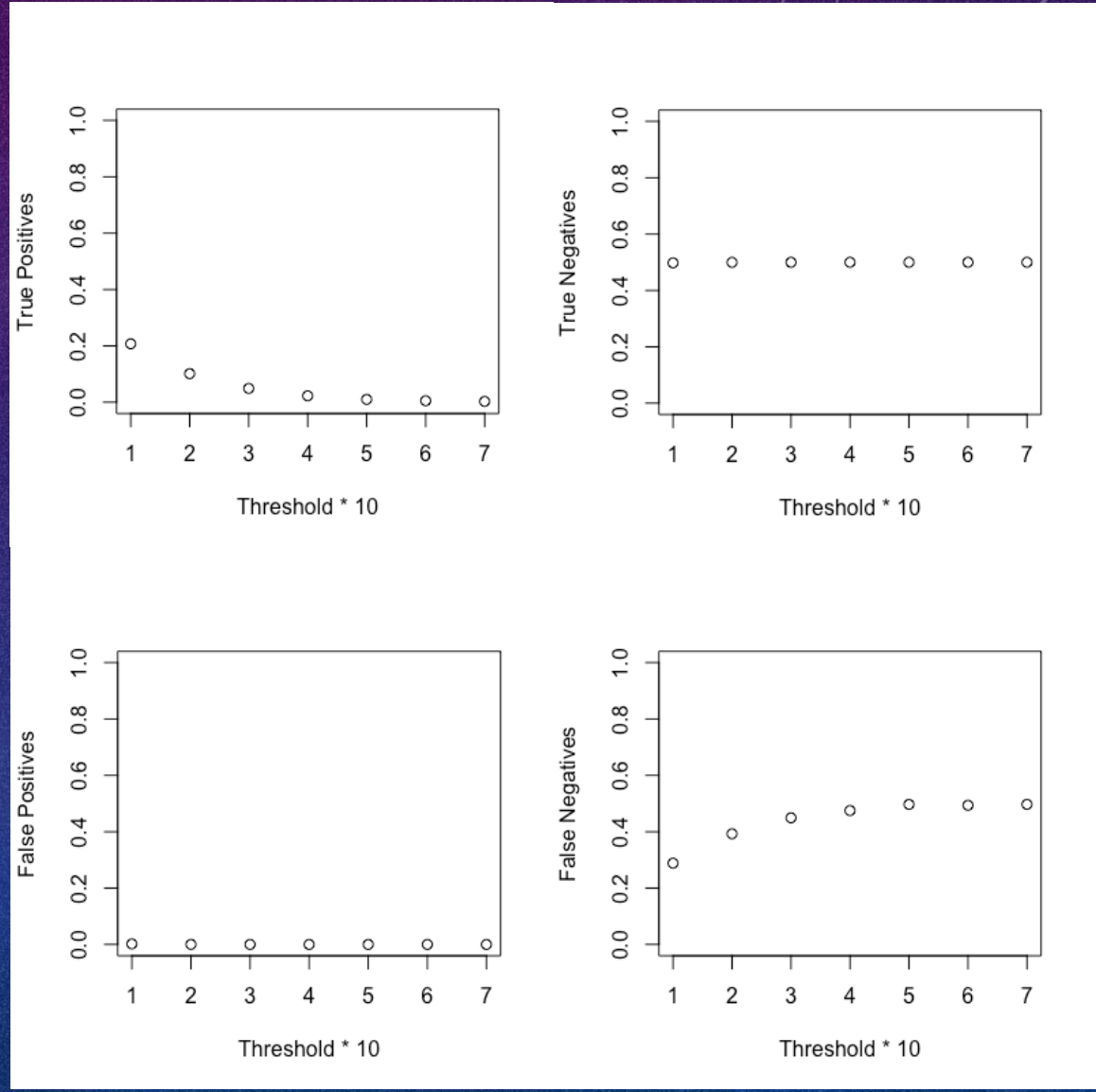
R function *linkPrediction (part 2)*: calculate accuracy of link prediction algorithm

```
        if(l==0 && s>theta){
            fp<-fp+1
            next
        }
        if(l==1 && s<theta){
            fn<-fn+1
            next
        }
    }
    accuracy<-(tp+tn)/(tp+tn+fp+fn)
    lp <- list(“accuracy”=accuracy, "tp”=tp,"tn"=tn,"fp"=fp,"fn"=fn)
    return (lp)
}
```

LINK PREDICTION

R commands

```
trainAndTestNets<-trainAndTest(g)
test<-trainAndTestNets$test
train<-trainAndTest$train
test<-addEdgesTest(test,train,ecount(test))
scoreMatrix<-similarity(train,vids = V(train), mode = c("out"), loops = F, method = c("jaccard"))
lp1<-linkPrediction(test,scoreMatrix,0.1)
lp2<-linkPrediction(test,scoreMatrix,0.2)
lp3<-linkPrediction(test,scoreMatrix,0.3)
…
```

LINK PREDICTION

Accuracy