

# Amazon Co-Purchase Network Analysis

---

UCONN MSBAPM FALL 2016

**Chen Chen, Chintha Mithra, Mihir Sanghvi, Samyuktha Kavitha Anand ,Anuj Kumar**

**10/16/2016**

## Contents

Amazon Co-Purchase Network Analysis .....	3
Executive Summary.....	3
Introduction: What is Social Network Analysis (SNA)? .....	3
Goal .....	3
Data Definition.....	3
Data Preparation & Data Cleaning.....	4
Our Approach: .....	6
Step-I: .....	6
Step-II: .....	8
Step-III: .....	9
Step-IV: .....	11
Step-V: .....	12
Conclusion .....	13
References.....	14

# Amazon Co-Purchase Network Analysis

## Executive Summary

The paper contains information about our analysis on amazon co-purchase data. Team used igraph package to explore the dataset and provided insights about the up-selling opportunities mainly based on the community detection and clique detection functions available in igraph package. The purpose of the analysis was to identify the up-selling bundles which could be sold together based on the centrality measures and sales rank of the products.

## Introduction: What is Social Network Analysis (SNA)?

Social Network Analysis (SNA) is the process of investigating social structures through the use of network and graph theories. It is an emerged technique and has more significant applications in biology, communication studies, computer science and more domains, and it is now commonly available as a consumer tool. SNA allows us to answer questions like who are the key actors in a network? How individual member is connected with each other? And which connections in the network are the most important? And etc.

## Goal

In this project our team used SNA to analyze and discover the up-selling opportunities based on the Amazon Co-Purchasing network data, and further, to explore its marketing applications (from the marketing up selling campaign design perspective).

## Data Definition

**Edge list:** Amazon product co-purchasing network, June 01 2003. Network was collected by crawling Amazon website. It is based on *Customers Who Bought This Item Also Bought* feature of the Amazon website

Data Dictionary:

Variable	Description
FromNodeId	Amazon Product Id which was bought
ToNodeId	Amazon Product Id which is also bought with fromnodeId

**Meta data:** Amazon product metadata and reviews from summer 2006. It contains 548,552 different products (Books, music CDs, DVDs and VHS video tapes). Data Dictionary:

Variable	Description
Id	Product id
ASIN	Amazon Standard Identification Number
Title	Name/title of the product
group	Product group (Book, DVD, Video or Music)
salesrank	Amazon Salesrank
similar	ASINs of co-purchased products (people who buy X also buy Y)
categories	Location in product category hierarchy to which the product belongs (separated by  , category id in [])
reviews	Product review information: time, user id, rating, total number of votes on the review, total number of helpfulness votes (how many people found the review to be helpful)

## Data Preparation & Data Cleaning

Data downloaded is a amazon product meta data of flat text with attribute and value pair in assymetric manner. Data prepared should have Id,title,group,salesrank and average rating for the each product.

Below is how the sample data looks like:

```
Id: 0
ASIN: 0771044445
discontinued product

Id: 1
ASIN: 0827229534
title: Patterns of Preaching: A Sermon Sampler
group: Book
salesrank: 396585
similar: 5 0804215715 156101074X 0687023955 0687074231 082721619X
categories: 2
|Books[283155]|Subjects[1000]|Religion & Spirituality[22]|Christianity[12290]|Clergy[12360]|Preaching[12368]
|Books[283155]|Subjects[1000]|Religion & Spirituality[22]|Christianity[12290]|Clergy[12360]|Sermons[12370]
reviews: total: 2 downloaded: 2 avg rating: 5
2000-7-28 cutomer: A2JW67OY8U6HHK rating: 5 votes: 10 helpful: 9
2003-12-14 cutomer: A2VE83MZF98ITY rating: 5 votes: 6 helpful: 5
```

### Packages Used for Data Preparation

foreach: Used to loop faster compared to default for loop

tidyr: Used a method 'separate' to divide the column into two columns using delimiter

data.table: Used read.table function which is faster than write.csv

sqlDf: Used to write sql queries in R using sqlDf function

Step 1: Convert each line of the raw text into row and select the rows which starts with Id, title,group, salesrank and reviews. divide the each row into column with delimiter as 1<sup>st</sup> ":" and eliminate the spaces.

	key	value
1	Id	0
2	Id	1
3	title	Patterns of Preaching: A Sermon Sampler
4	group	Book
5	salesrank	396585
6	reviews	total: 2 downloaded: 2 avg rating: 5
7	Id	2
8	title	Candlemas: Feast of Flames
9	group	Book
10	salesrank	168596
11	reviews	total: 12 downloaded: 12 avg rating: 4.5

Step 2: -Access every row and convert the 'key' as column name and 'value' as row value.

As data is huge and looping through the entire data would lead to the slower processing which would create memory problems in R, data is divided into several chunks and row values are dumped into comma separated values (CSV) file after reaching 20000 rows of output. The cleaned data would be uploaded in the memory. Every 20000 records that are processed in this manner were merged with the file that is created in the previous steps.

	Id	title	group	salesrank	reviews
1	0	NA	NA	NA	NA
2	1	Patterns of Preaching: A Sermon Sampler	Book	396585	total: 2 downloaded: 2 avg rating: 5
3	2	Candlemas: Feast of Flames	Book	168596	total: 12 downloaded: 12 avg rating: 4.5

**Step 3:** - Access the average rating in the reviews column and create a “average rating” column.

	Id	title	group	salesrank	averagerating
1	0	NA	NA	NA	NA
2	1	Patterns of Preaching: A Sermon Sampler	Book	396585	5
3	2	Candlemas: Feast of Flames	Book	168596	4.5

**Step 4:** - Missing Value Treatment & Data Cleaning

Missing Value Treatment

In the metadata, there are 5868 products which are discontinued products and for those products only Id is available.

There are missing values for four columns: Title, Group, SalesRank, AverageRating

Replace NA's for Title & Group columns with “Discontinued Product”

Replace NA's for SalesRank & AverageRating with 0

## Data Cleaning

There are four main groups of products: Book, DVD, Video or Music

Apart from the four main groups, there are few more groups as follows:

```
> table(coPurchaseMeta$Group)
```

Baby Product	Book	CE	DVD	Music	Software	Sports	Toy	Video
1	393464	4	19761	103139	5	1	8	26127
Video Games								
1								

Collapse Levels other than the four groups

```
> table(coPurchaseMeta$Group)
```

Music	DVD	Video	Book	Others
103139	19761	26127	393464	20

**Step 5:** - Load Edge List and extract Metadata about the nodes present in Edge List

Some products present in edge list are not there in metadata, merged edge list and metadata and created new edge list.

Similarly, there are many products in metadata which are not there in edge list, merged both to create new metadata.

Computed Eigen Value Centrality and degree for the nodes and added it to metadata.

## Final Metadata

	Nodeid	Title	Group	SalesRank	AverageRating	DiscontinuedProduct	Degree	EigenValueCentrality
1	0	Discontinued Product	Others	0	0.0	1	14	1.321183e-03
2	1	Patterns of Preaching: A Sermon Sampler	Book	396585	5.0	0	13	8.601228e-04
3	2	Candlemas: Feast of Flames	Book	168596	4.5	0	13	8.767395e-04
4	3	World War II Allied Fighter Planes Trading Cards	Book	1270652	5.0	0	13	1.856094e-03
5	4	Life Application Bible Commentary: 1 and 2 Timothy an...	Book	631289	4.0	0	289	8.181516e-03
6	5	Prayers That Avail Much for Business: Executive	Book	455160	0.0	0	572	3.755284e-02
7	6	How the Other Half Lives: Studies Among the Teneme...	Book	188784	4.0	0	331	1.098284e-02
8	7	Batik	Music	5392	4.5	0	37	2.145691e-03
9	8	Losing Matt Shepard	Book	277409	4.5	0	69	2.969419e-03

## Final Edge List

	FromNodeid	ToNodeid
1	0	1
2	0	2
3	0	3
4	0	4
5	0	5

## Our Approach:

We have divided our analysis in five steps

### Step-I: Create Network Graph and Diagnose Network

#### Graph Creation

```
coPurchaseNetworkGraph <- graph.data.frame(d = coPurchaseEdgeList, vertices = coPurchaseMeta, directed = T)
```

The above function is used to create network graph from data-frames

#### Argument Description:

No.	Argument	Description
1	D	edge list data-frame
2	Vertices	data-frame about vertex metadata
3	Directed	whether or not to create a directed graph

## Network Diagnostics

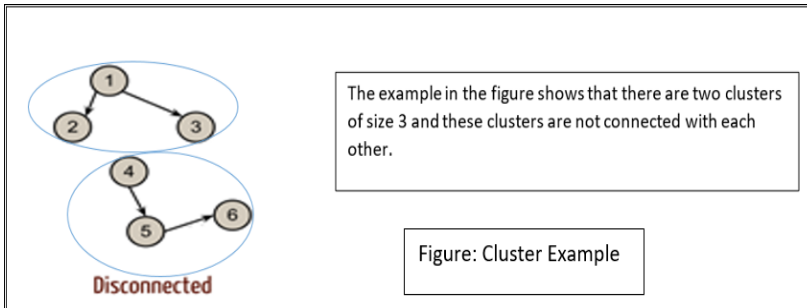
### Clustering

```
> is.connected(coPurchaseNetworkGraph)
[1] FALSE
```

The above function is used to check if network is globally connected, if all nodes are connected to each other in that case it will return TRUE

```
> cl <- clusters(coPurchaseNetworkGraph)
> str(cl)
List of 3
 $ membership: Named num [1:403394] 1 1 1 1 1 1 1 1 1 1 ...
  .. attr(*, "names")= chr [1:403394] "0" "1" "2" "3" ...
 $ csize      : num [1:7] 403364 2 2 3 3 ...
 $ no         : int 7
> cl$csize
[1] 403364      2      2      3      3     15      5
```

The clusters function checks for the connected components in the graph. From the results it is evident that first cluster has 402264 connected vertices and all of these clusters are not connected with each other.

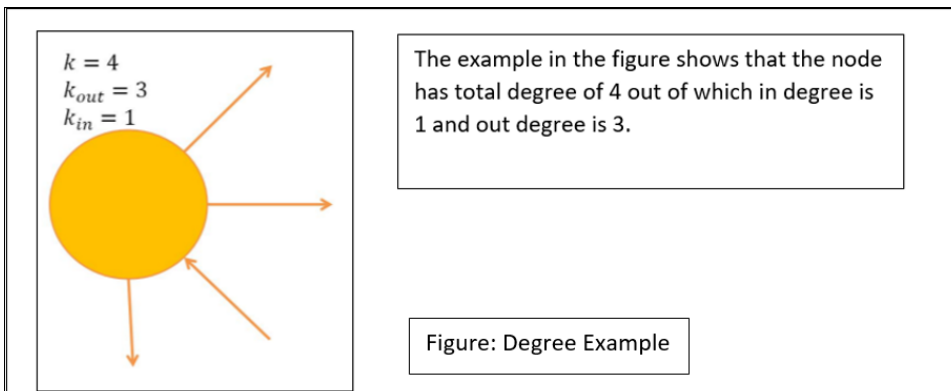


## Degree

```
> Degree<- degree(coPurchaseNetworkGraph, mode = "total")
> Degree<- degree(coPurchaseNetworkGraph, mode = "total")
> Degree<- sort(Degree, decreasing = T)
> head(as.data.frame(Degree))
```

	Degree
1041	2761
45	2497
50	2291
529	1522
783	1184
10030	866

The degree function calculates total number of degree for each vertex.

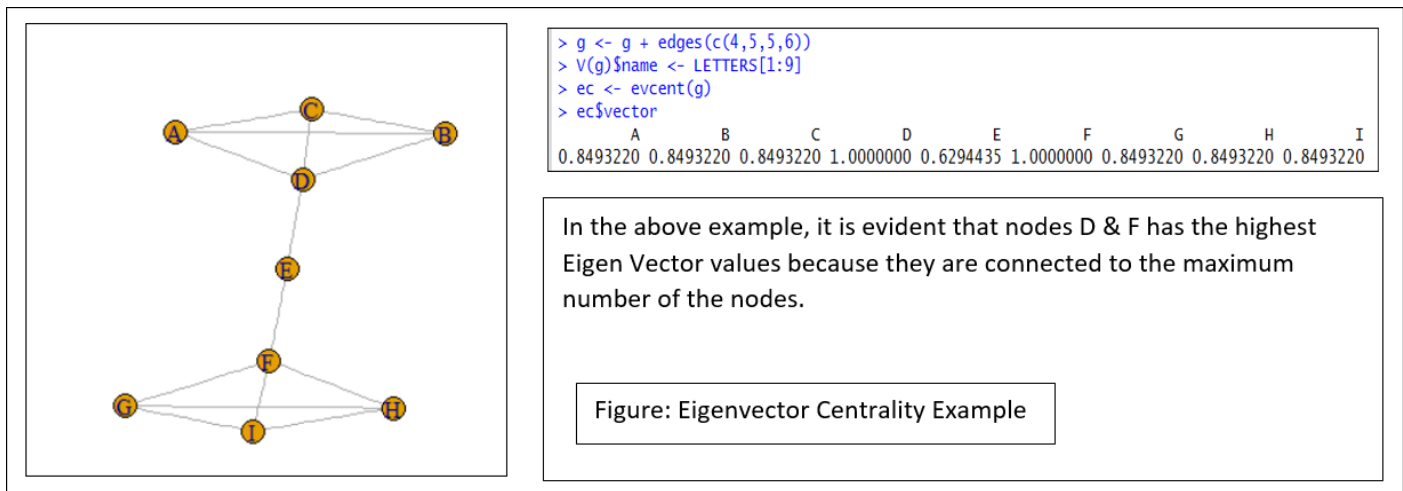


## Eigenvector Centrality

```
> EVcentrality <- evcent(coPurchaseNetworkGraph) # eigenvector centrality
> EVcentrality <- EVcentrality$vector # convert to array
> EVcentrality<- sort(EVcentrality, decreasing = T)
> head(as.data.frame(EVcentrality))
```

	EVcentrality
1041	1.0000000
45	0.9137778
50	0.4430843
1039	0.2470248
1036	0.2304179
529	0.2017369

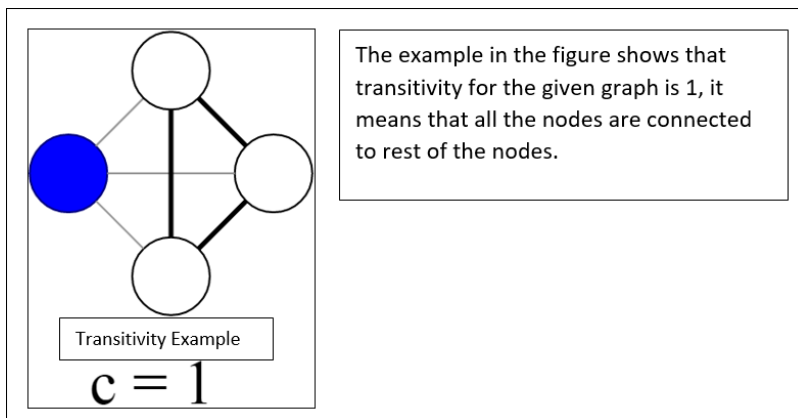
Event function is used to calculate Eigen vector of centrality. The Eigenvector corresponding to the largest Eigenvalue of the adjacency matrix gives a high score to vertices that either have a lot of connections, or are connected to someone with a lot of connections.



### Local Clustering Coefficient: (Transitivity)

Measures the probability that the adjacent vertices of a vertex are connected, quantifies how close its neighbors are to being a clique (complete graph)

The average clustering coefficient of the whole graph is simply the average of the node's local clustering coefficients



```
> avgcc<- transitivity(coPurchaseNetworkGraph, type="average", isolates="zero") #average clustering coeff
> avgcc
[1] 0.4176812
```

The transitivity function is used to calculate clustering coefficient of full graph.

**Step-II:** Create communities using InfoMap algorithm

```
# Create Communities
coPurchaseCommunity <- infomap.community(coPurchaseNetworkGraph)
```

The above function is used to create different communities within the network graph, Argument Description:

No.	Argument	Description
1	graph	network graph object

The function returns community object, following descriptive functions could be applied to the community object:

```
> length(coPurchaseCommunity)
[1] 18637
```



The above function gives total number of communities created.

```
> algorithm(coPurchaseCommunity)
[1] "infomap"
```

The above function gives the name of the algorithm which was used to create community.

**Step-III:** Identify top communities based on average degree of the nodes and community visualization

Created following function to identify top communities out of the total communities created by InfoMap function.

```
topCommunities = function(networkGraph, topCommunityCount = 5){
  # Create a new graph where we contract all vertices in a single community to one vertice, and find the average degree
  coPurchaseCommunityGraph <- contract.vertices(networkGraph, V(networkGraph)$Community, vertex.attr.comb = list(Degree ="mean", Community=toString))

  # save average degree associated with communities within a vector
  communityMeanDegreeVector <- V(coPurchaseCommunityGraph)$Degree

  # get community list
  communityList <- V(coPurchaseCommunityGraph)$Community

  # Save community number in a list
  communityNumericVectorList <- lapply(communityList, function(x){as.vector(as.numeric(strsplit(x, ",")[[1]]))})

  # Save community membership into a vector
  communityMembershipVector <- sapply(communityNumericVectorList, function(x){x[1]})

  # Compute number of vertices within a community and save it to a vector
  communityNodeCountVector <- sapply(communityNumericVectorList, function(x){length(x)})

  # Create dataframe
  communityDetailsDataFrame <- data.frame(communityMembershipVector,communityMeanDegreeVector,communityNodeCountVector)

  # Order rows based on communityMeanSalesRankVector
  communityDetailsDataFrame <- arrange(communityDetailsDataFrame,desc(communityMeanDegreeVector))

  # Change column names
  colnames(communityDetailsDataFrame) <- c("communityMembership","communityMeanDegree","communityNodeCount")

  # Return Top Communities
  return(head(communityDetailsDataFrame,topCommunityCount))
}
```

Main igraph function used to develop above function is as follows:

```
# Create a new graph where we contract all vertices in a single community to one vertice, and find the average degree
coPurchaseCommunityGraph <- contract.vertices(networkGraph, V(networkGraph)$Community,
                                              vertex.attr.comb = list(Degree ="mean", Community=toString))
```

The above mentioned function is used to create a new graph, by merging several vertices into one. The vertices in the new graph correspond to sets of vertices in the input graph. In our case we have merged all vertices within one community into a vertex, computed average degree of all the nodes and attached it as a node attribute.

#### Argument Description:

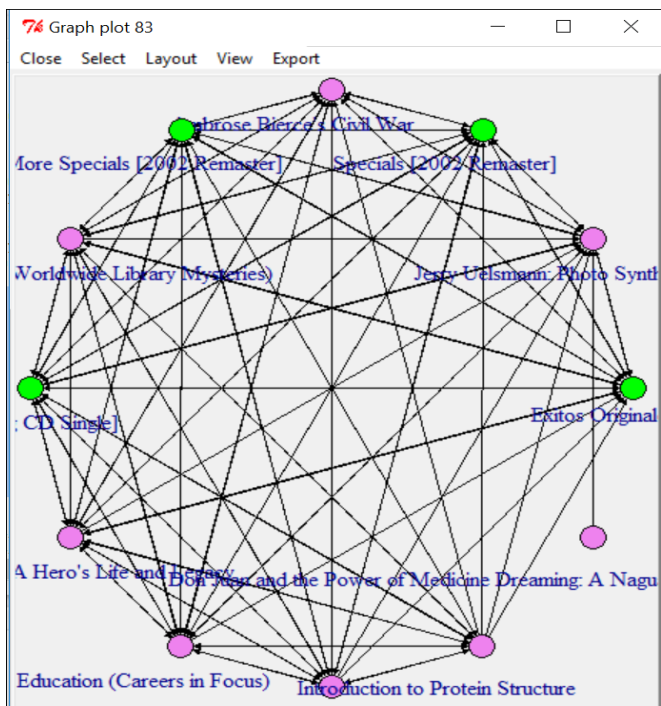
No.	Argument	Description
1	Graph	network graph object
2	Mapping	A numeric vector that specifies the mapping. Its elements correspond to the vertices, and for each element the id in the new graph is given
3	vertex.attr.comb	Specifies how to combine the vertex attributes in the new graph

## Community Visualization:

```
tkplot(  
  communityGraph,  
  vertex.size = 9,  
  vertex.label = V(communityGraph)$Title,  
  vertex.label.dist = -2,  
  edge.arrow.size = 0.5,  
  edge.color = "black",  
  canvas.width = 450,  
  canvas.height = 500,  
  layout= layout.circle,  
  vertex.color=V(communityGraph)$color  
)
```

tkplot function is used to visualize the network graph. Here in the function arguments we have provided vertex and edge attributes including size, label, color, edge arrow size etc.

The output of the above function is as follows:



#### Step-IV: Identify top products based on sales rank

Created following function to identify most popular products based on their sales rank.

```
topCommunityProducts = function(networkGraph,communityMembership,topCommunityProductCount = 5){  
  # Create a subgraph for the given community from the main network graph  
  communityGraph <-induced.subgraph(networkGraph, which(V(networkGraph)$Community==communityMembership))  
  
  # extract nodeId, salesRank, title, group and averageRating from the community graph  
  nodes <- as.vector(as.numeric(V(communityGraph)$name))  
  salesRank <- as.vector(V(communityGraph)$SalesRank)  
  title <- as.vector(V(communityGraph)$Title)  
  group <- as.vector(V(communityGraph)$Group)  
  averageRating <- as.vector(V(communityGraph)$AverageRating)  
  
  # Create a dataframe  
  nodesDataFrame <- data.frame(nodes,title,group,salesRank,averageRating)  
  
  # Filter DataFrame (Remove all rows with salesRank less than 0 )  
  nodesDataFrame <- nodesDataFrame[nodesDataFrame$salesRank>0,]  
  |  
  # Sort DataFrame rows based on salesRank  
  nodesDataFrame <- arrange(nodesDataFrame,salesRank)  
  
  # Rename DataFrame Columns  
  colnames(nodesDataFrame) <- c("NodeId","Title","Group","SalesRank","AverageRating")  
  
  # Return top n products within community based on salesRank (n = topCommunityProductCount)  
  return(head(nodesDataFrame,topCommunityProductCount))  
}
```

Main igraph function used to develop above function is as follows:

```
# Create a subgraph for the given community from the main network graph  
communityGraph <-induced.subgraph(networkGraph,  
                                   which(V(networkGraph)$Community==communityMembership))
```

The subgraph function is used to create a subgraph of a graph, containing only the specified vertices and all the edges among them.

Argument Description:

No.	Argument	Description
1	Graph	network graph object
2	Mapping	A numeric vector that specifies the mapping. Its elements correspond to the vertices, and for each element the id in the new graph is given
3	vertex.attr.comb	Specifies how to combine the vertex attributes in the new graph

## Step-V: Identify Cliques within the top community which includes top product of the community

Created following function to export cliques including a specific product within the community

```
exportCliques = function(networkGraph,communityMembership,productNodeId){  
  # Create a subgraph for the given community from the main network graph  
  communityGraph <- induced.subgraph(networkGraph, which(V(networkGraph)$Community==communityMembership))  
  
  # Identify cliques from the community graph  
  # The output of the following function is a list object consisting of vectors of vertices of cliques  
  CliqueVertexList = cliques(communityGraph,min = 4,max = 4)  
  
  # convert the vector of vertices for a given clique to graph object and save it to list object  
  CliqueList <- lapply(CliqueVertexList, function (x) { induced_subgraph(communityGraph, x) })  
  
  # check if the given productNodeId is present within clique graph objects  
  ProductBasedCliqueFlagList = sapply(CliqueList, function(x){as.vector(table(V(x)$name==productNodeId))[2]})  
  ProductBasedCliqueFlagList <- ifelse(is.na(ProductBasedCliqueFlagList),0,1)  
  
  # Filter out all clique graph objects where productNodeId is not present  
  ProductBasedCliqueList <- CliqueList[ProductBasedCliqueFlagList==1]  
  
  cliquesDataFrameList <- lapply(ProductBasedCliqueList,  
                                function(y) {as.data.frame(list(NodeId=as.numeric(V(y)$name),  
                                                                Title = as.character(V(y)$Title),Group=as.vector(V(y)$Group),  
                                                                SalesRank=as.numeric(V(y)$SalesRank),AverageRating=as.numeric(V(y)$AverageRating),  
                                                                CliqueId = 1) , stringsAsFactors=FALSE))})  
  
  cliqueDataFrame <- data.frame() # Prepare |  
  tempCliqueFrame <- data.frame()  
  
  for(i in 1:length(cliquesDataFrameList)){  
    tempCliqueFrame <- cliquesDataFrameList[[i]]  
    tempCliqueFrame$CliqueId = i  
    cliqueDataFrame = rbind(cliqueDataFrame,tempCliqueFrame)  
  }  
  cliqueDataFrame$CommunityMembership = communityMembership  
  cliqueDataFrame$PopularProductId = productNodeId  
  
  write.csv(cliqueDataFrame,"cliquesData.csv",row.names = F)  
  View(cliqueDataFrame)  
}
```

The above function writes csv file in following format:

	NodeId	Title	Group	SalesRank	AverageRating	CliqueId	CommunityMembership	PopularProductId
1	528	Viennese Memories	Music	274843	0.0	1	3	600
2	529	An Outline of Philosophy	Book	581223	4.0	1	3	600
3	597	Sunday Jews	Book	539223	2.0	1	3	600
4	600	American Patriot	Music	3430	4.5	1	3	600
5	528	Viennese Memories	Music	274843	0.0	2	3	600
6	529	An Outline of Philosophy	Book	581223	4.0	2	3	600
7	599	Masada	Music	54181	4.5	2	3	600
8	600	American Patriot	Music	3430	4.5	2	3	600

Main igraph function used to create above function is as follows:

```
CliqueVertexList = cliques(communityGraph,min = 4,max = 4)
```

The cliques function is used to find all cliques within a particular graph.

Argument Description:

No.	Argument	Description
1	Graph	network graph object
2	min	Minimum number of cliques
3	max	Maximum number of cliques


## Conclusion

With the help of this project we are able to identify the list of products which could be sold together as bundles (up-selling). The final output is a csv file which includes all possible cliques of top products within top co-purchase communities which could be used for up-selling campaign and promotions design.

### The Ultimate TOEFL iBT® Test Prep Savings Bundle 1st Edition

by Educational Testing Service (Author)

★★★★★ 2 customer reviews




**Paperback**  
\$64.60

Other Sellers  
from \$58.88

Buy new

Only 14 left in stock (more on the way).

Ships from and sold by Amazon.com. Gift-wrap available.

 Prime


Want it Tuesday, Oct. 18? Order within 10 hrs 6 mins and choose **Two-Day Shipping** at checkout. [Details](#)

 Prime **\$64.60**

List Price: ~~\$95.00~~ Save: \$30.40 (32%)

30 New from **\$58.88**

Qty: 1

 Add to Cart

## References

[https://en.wikipedia.org/wiki/Social\\_network\\_analysis](https://en.wikipedia.org/wiki/Social_network_analysis)

Co-Purchase Network Edge List Data

<https://snap.stanford.edu/data/amazon0601.html>

Amazon Metadata

<https://snap.stanford.edu/data/amazon-meta.html>

Dataset source: J. Leskovec, L. Adamic and B. Adamic. The Dynamics of Viral Marketing. ACM Transactions on the Web (ACM TWEB), 1(1), 2007.

<http://kateto.net/netscix2016>

<https://en.wikipedia.org/wiki/Igraph>

<http://econometricsense.blogspot.com/2012/04/introduction-to-social-network-analysis.html>

[https://en.wikipedia.org/wiki/Network\\_motif](https://en.wikipedia.org/wiki/Network_motif)

<http://igraph.org/r/doc/>