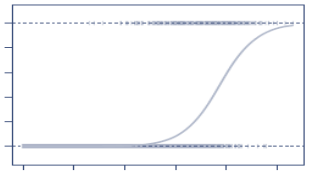
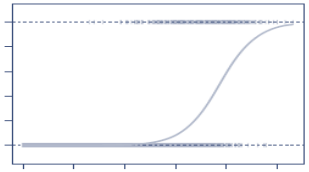


Analysis of Customer Profiles, Product Information and Transaction Histories

Nathan Grossman

Overview



Customer Profiles

What are our customers' buying habits? Which customers are profitable?

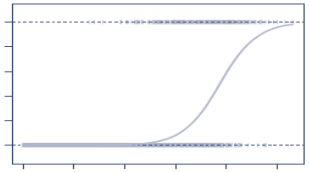
Marketing Effectiveness

What is our return on investment for different sources of customers?

Product Recommendations

Which products tend to be bought by the same customers?

Customer Profiles: Clusters



Behavior-based clusters were found based on

- Six variables reflecting how customers behave while purchasing
- Two variables reflecting how customers behave while purchasing

Key Insights:

- Most customers only made one or two purchases over the last two years. In other words, there were relatively few repeat customers.
- Customers can be grouped into three clusters according their total revenue and mean order interval (or equivalently, total orders).

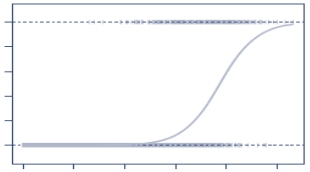
Cluster Number	Total Revenue	Total Orders	Mean Revenue	Mean Order Interval	Mean Price	Mean Discount	Cluster Size
1	$1,011.72	1.0	$1,011.72	100000	$338.74	2.2%	4,215
2	$743.85	2.1	$361.49	222	$245.21	5.1%	6,004
3	$272.11	1.0	$272.11	100000	$231.86	3.8%	26,769

Clusters based on six variables

Cluster Number	Total Revenue	Mean Order Interval	Cluster Size
1	$272.49	100000	26,797
2	$1,014.23	100000	4,187
3	$743.85	222	6,004

Clusters based on two variables

Customer Profiles: Profitability

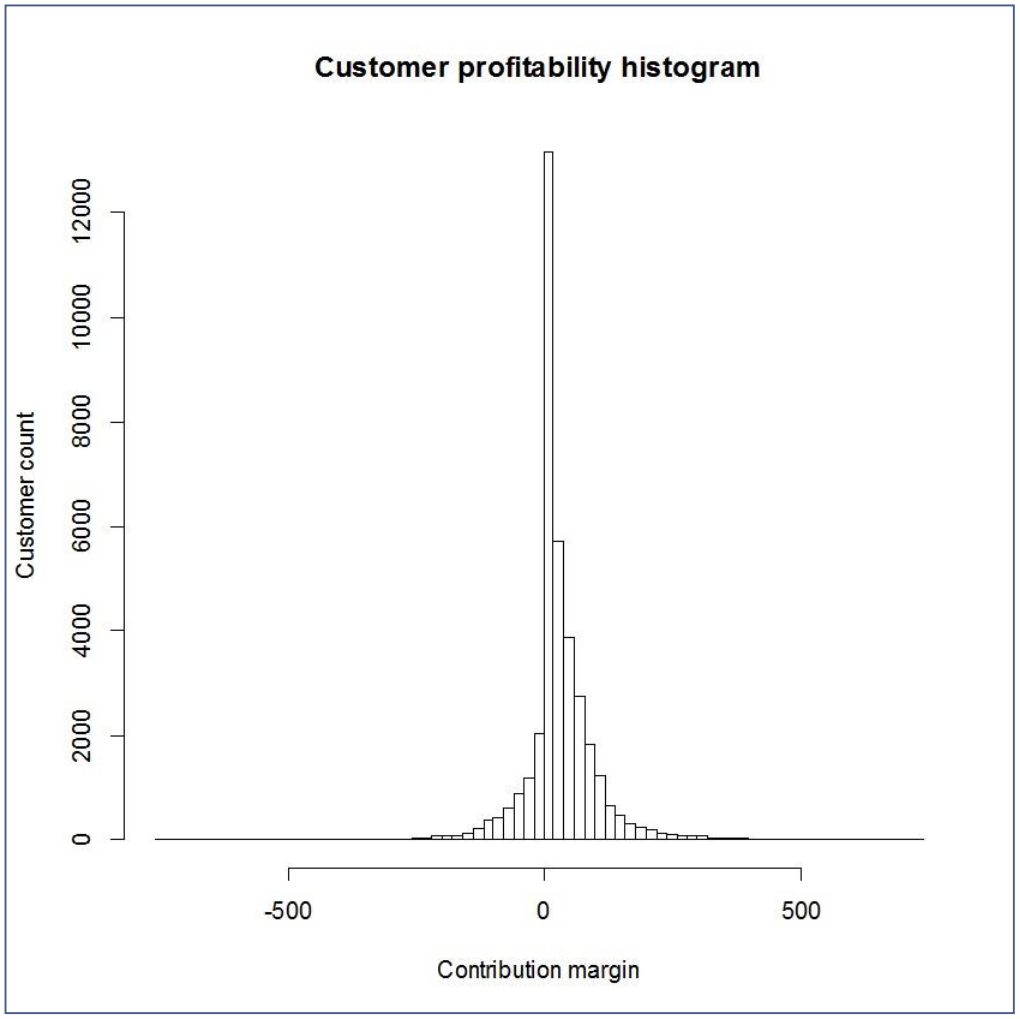


Customers' profitability was measured by the contribution margin (CM) they generated, which was defined as

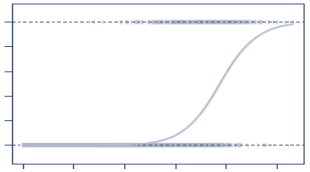
$$CM = Revenue - Quatity \times Cost$$

While the fact that customer profitability follows a distribution like the one found may not be surprising, the value of this exercise lies in the fact that we now know the profitability of each individual customer.

Thus we can tailor what we offer to individual customers based on how profitable they are.



Marketing Effectiveness



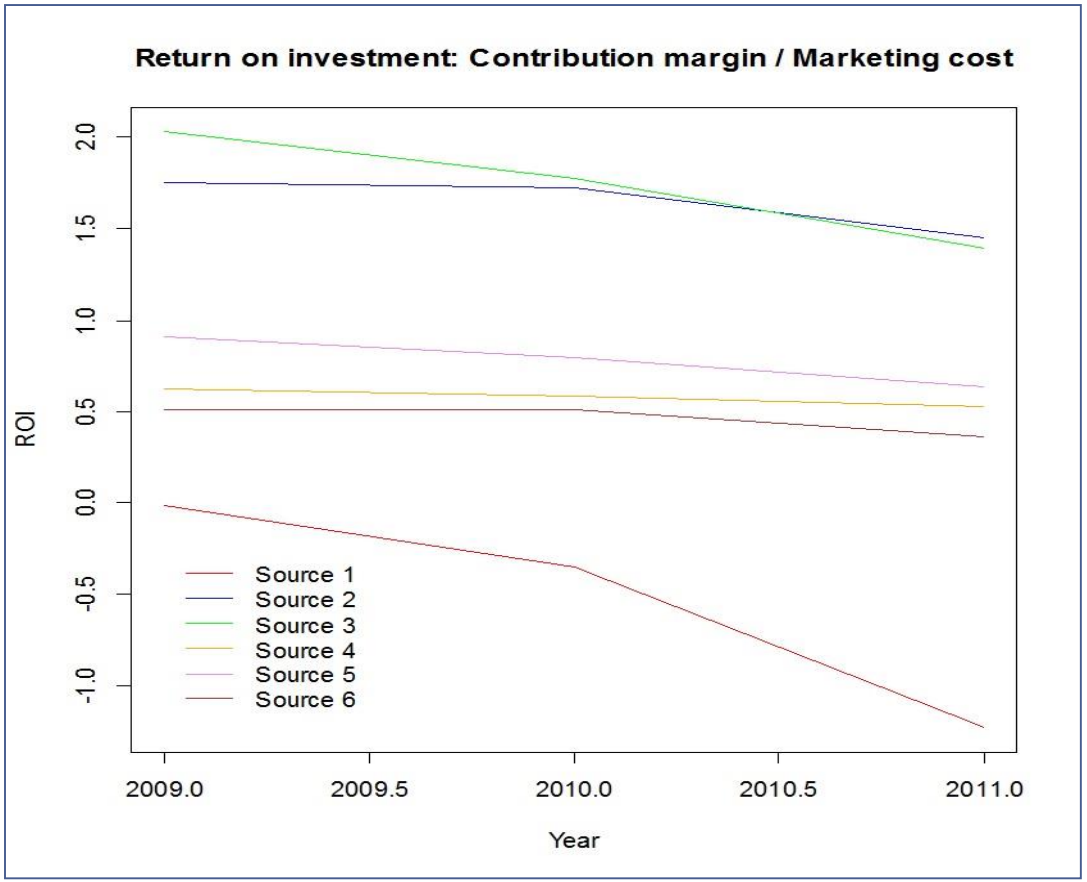
The “same year” return on investment (ROI) in marketing was found for each customer source. “Same year” ROI was defined as

$$ROI = \frac{Contribution\ margin}{Marketing\ cost}$$

where the contribution margin was the difference between total revenues and costs for each customer in their first year with the company

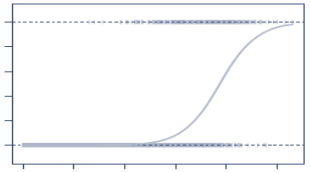
Key Insights:

- Some customer sources have far higher ROI than others.
- The trend has been downward for ROI for all sources.



Year	Source 1	Source 2	Source3	Source 4	Source 5	Source 6
2009	-0.02	1.75	2.03	0.63	0.91	0.51
2010	-0.35	1.72	1.78	0.59	0.80	0.51
2011	-1.23	1.46	1.40	0.53	0.64	0.36

Product Recommendations



Products that tended to be purchased by the same customers were found using Affinity Analysis.

$$A(i,j) = \frac{sup\{i,j\}}{sup\{i\} + sup\{j\} - sup\{i,j\}}$$

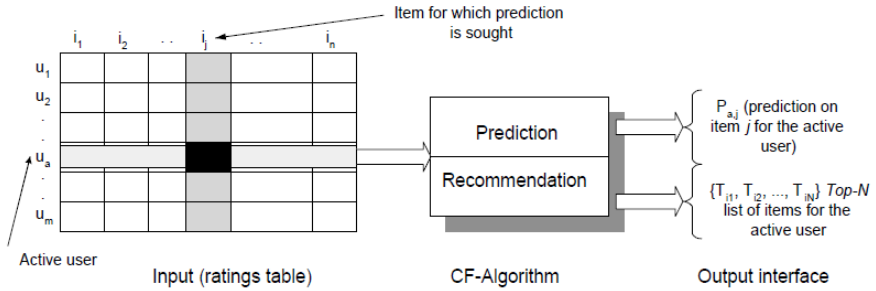
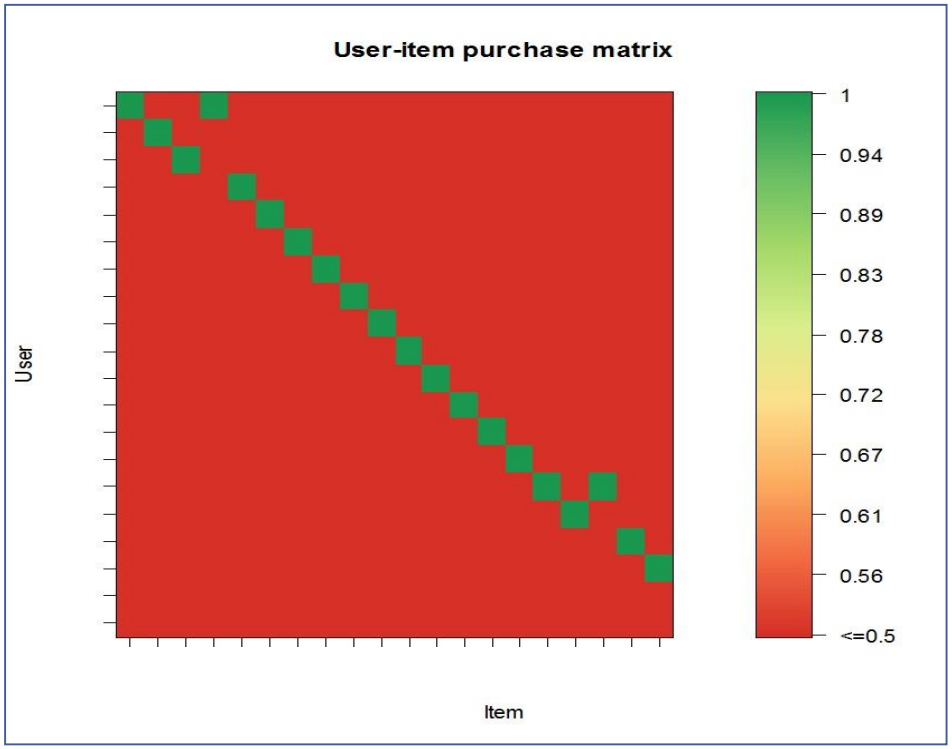
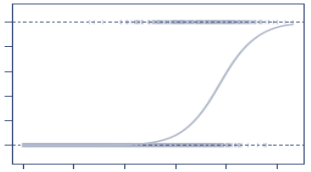


Figure 1: The Collaborative Filtering Process.

The measure of similarity A(i,j) is a Jaccard Index, which basically tells us the percentage of the time that two items are present if either one of them is present.



Product Recommendations



Products that tended to be purchased by the same customers were found using Affinity Analysis.

$$A(i,j) = \frac{sup\{i,j\}}{sup\{i\} + sup\{j\} - sup\{i,j\}}$$

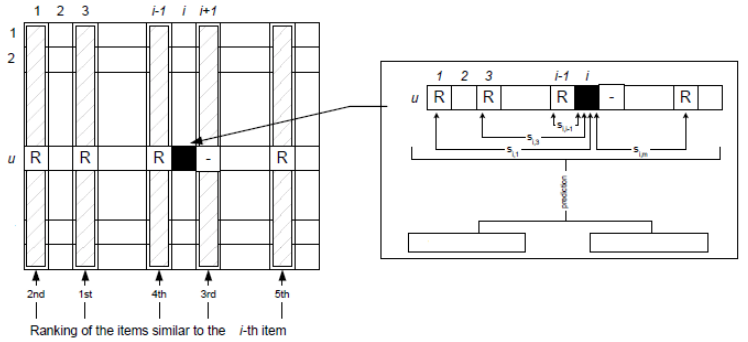
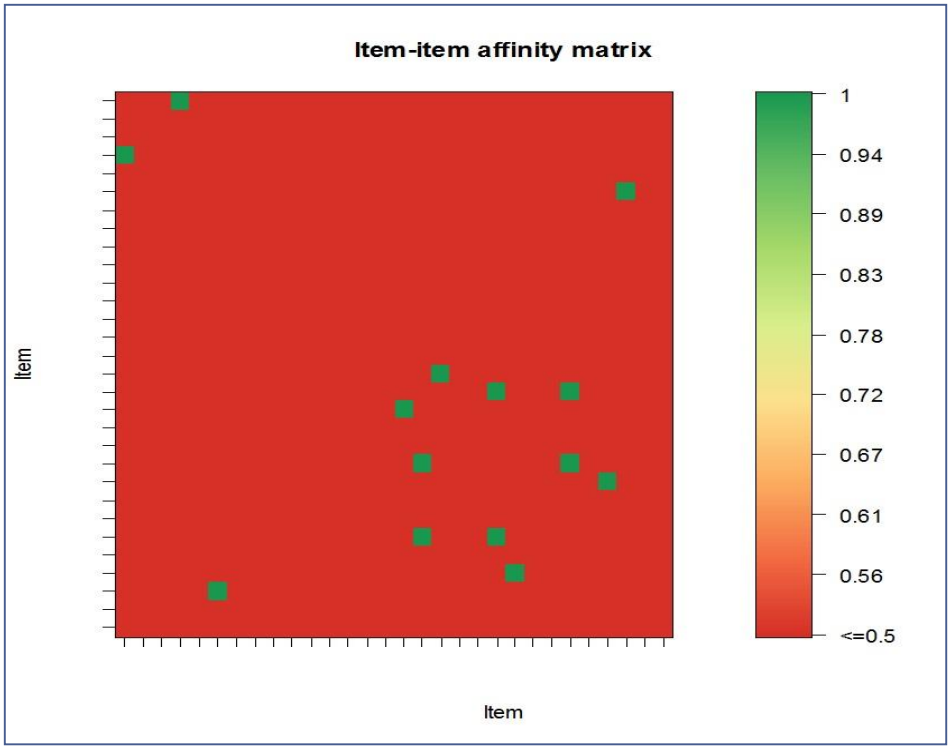
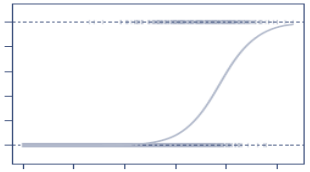


Figure 3: Item-based collaborative filtering algorithm.

The measure of similarity A(i,j) is a Jaccard Index, which basically tells us the percentage of the time that two items are present if either one of them is present.



Takeaways



Customer Profiles

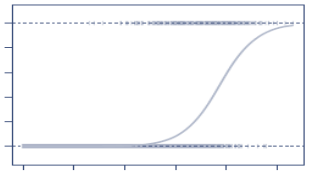
Most of our customers make one-time purchases. If we could increase the percentage of customers that make repeat purchases, we could potentially improve our profitability. We could use knowledge about which customers are the most profitable to tailor our offerings to individual customers.

Marketing Effectiveness

There is a large variance in the return on investment in different marketing sources. If we could reallocate our investments from less effective to more effective sources, we could potentially improve our profitability.

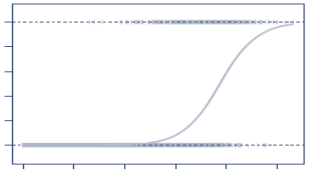
Product Recommendations

We could make cross-sell recommendations to customers based on knowledge of which products tend to be purchased in common by customers. This could potentially increase the percentage of customers that make repeat purchases.



Thank you for your time

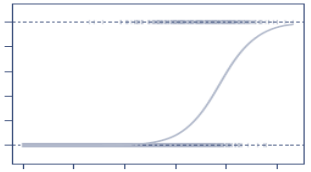
Appendix



customer_transaction_analysis.R Page 1

```
print("Begin Customer Transaction Analysis")

# Clean up environment
rm(list=ls())
graphics.off()
setwd('C:/Users/Nathan/RStudioProjects/customer_transaction_analysis')

# Set up environment
# install.packages('sqldf')
# install.packages('recommenderlab')
# install.packages('arules')
# install.packages('MKmisc')
# install.packages('cluster')
# install.packages('fpc')
library('sqldf')
library('recommenderlab')
library('arules')
library('MKmisc')
library('cluster')
library('fpc')

# Load data
orders2011 = read.csv("Orders2011.csv", stringsAsFactors = FALSE)
orders2011$OrderLineID = as.factor(orders2011$OrderLineID)
orders2011$OrderID = as.factor(orders2011$OrderID)
orders2011$CustomerID = as.factor(orders2011$CustomerID)
orders2011$ProductID = as.factor(orders2011$ProductID)
orders2011$OrderDate = as.Date(orders2011$OrderDate)
print("----------------------------------------")
print("    Orders 2011                         ")
print("----------------------------------------")
str(orders2011)
head(orders2011)

orders2010 = read.csv("Orders2010.csv", stringsAsFactors = FALSE)
orders2010$OrderLineID = as.factor(orders2010$OrderLineID)
orders2010$OrderID = as.factor(orders2010$OrderID)
orders2010$CustomerID = as.factor(orders2010$CustomerID)
orders2010$ProductID = as.factor(orders2010$ProductID)
orders2010$OrderDate = as.Date(orders2010$OrderDate)
print("----------------------------------------")
print("    Orders 2010                         ")
print("----------------------------------------")
str(orders2010)
head(orders2010)

orders2009 = read.csv("Orders2009.csv", stringsAsFactors = FALSE)
orders2009$OrderLineID = as.factor(orders2009$OrderLineID)
orders2009$OrderID = as.factor(orders2009$OrderID)
orders2009$CustomerID = as.factor(orders2009$CustomerID)
orders2009$ProductID = as.factor(orders2009$ProductID)
orders2009$OrderDate = as.Date(orders2009$OrderDate)
print("----------------------------------------")
print("    Orders 2009                         ")
print("----------------------------------------")
str(orders2009)
head(orders2009)
```

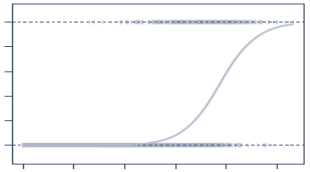
Appendix



customer_transaction_analysis.R **Page 2**

```
historicalOrders = read.csv("HistoricalOrders.csv", stringsAsFactors = FALSE)
historicalOrders$OrderLineID = as.factor(historicalOrders$OrderLineID)
historicalOrders$OrderID = as.factor(historicalOrders$OrderID)
historicalOrders$CustomerID = as.factor(historicalOrders$CustomerID)
historicalOrders$ProductID = as.factor(historicalOrders$ProductID)
historicalOrders$OrderDate = as.Date(historicalOrders$OrderDate)
print("----------------------------------------")
print("    Historical Orders                   ")
print("----------------------------------------")
str(historicalOrders)
head(historicalOrders)

customers = read.csv("Customers.csv", stringsAsFactors = FALSE)
customers$CustomerID = as.factor(customers$CustomerID)
customers$JoinDate = as.Date(customers$JoinDate)
customers$JoinDate = as.numeric(format(customers$JoinDate, "%Y"))
customers$SourceID= as.factor(customers$SourceID)
customers$City = as.factor(customers$City)
customers$State = as.factor(customers$State)
customers$Country = as.factor(customers$Country)
customers$Zipcode = as.factor(customers$Zipcode)
print("----------------------------------------")
print("    Customers                           ")
print("----------------------------------------")
str(customers)
head(customers)

products = read.csv("Products.csv", stringsAsFactors = FALSE)
products$ProductID= as.factor(products$ProductID)
print("----------------------------------------")
print("    Products                            ")
print("----------------------------------------")
str(products)
head(products)

sources = read.csv("Sources.csv", stringsAsFactors = FALSE)
sources$SourceID= as.factor(sources$SourceID)
print("----------------------------------------")
print("    Sources                             ")
print("----------------------------------------")
str(sources)
head(sources)

marketingCosts= read.csv("MarketingCosts.csv", stringsAsFactors = FALSE)
marketingCosts$SourceID= as.factor(marketingCosts$SourceID)
print("----------------------------------------")
print("    Marketing Costs                     ")
print("----------------------------------------")
str(marketingCosts)
head(marketingCosts)


######################
# Process 2011 data #
######################
print(" ")
print("Process 2011 data")
```

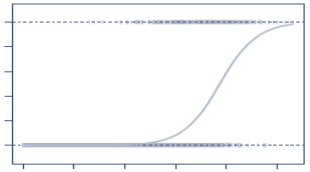
Appendix



customer_transaction_analysis.R **Page 3**

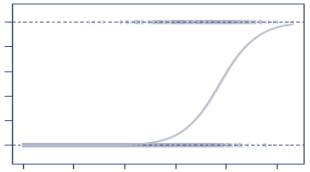
```
print(" ")

# Join order-product-customer data
query = "SELECT orders2011.*, products.ProductPrice, products.ProductCost FROM orders2011 INNER JOIN products ON orders2011.Produc
tID = products.ProductID"
orders2011Products = sqldf(query)

query = "SELECT orders2011Products.*, customers.JoinDate, customers.SourceID, customers.City, customers.State, customers.Country,
customers.Zipcode FROM orders2011Products INNER JOIN customers ON orders2011Products.CustomerID = customers.CustomerID"
orders2011ProductsCustomers = sqldf(query)

# Calculate revenue by source
query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=1"
revenue2011Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=2"
revenue2011Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=3"
revenue2011Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=4"
revenue2011Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=5"
revenue2011Source5 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2011ProductsCustomers WHERE SourceID=6"
revenue2011Source6 = as.numeric(sqldf(query))

revenue2011Source = c(revenue2011Source1, revenue2011Source2, revenue2011Source3, revenue2011Source4, revenue2011Source5, revenue2
011Source6)

print("Revenue in 2011 by source")
print(head(revenue2011Source))
print(" ")
print("Total revenue in 2011 rolled up from revenue by source")
print(sum(revenue2011Source))
print(" ")
print("Total revenue in 2011")
print(sum(orders2011$ExtRevenue))
print(" ")

# Calculate profit (actually contribution margin) by source for new customers
query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=1 AND JoinDate=2011"
profit2011Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=2 AND JoinDate=2011"
profit2011Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=3 AND JoinDate=2011"
profit2011Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=4 AND JoinDate=2011"
profit2011Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=5 AND JoinDate=2011"
profit2011Source5 = as.numeric(sqldf(query))
```

Appendix



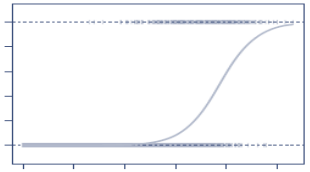

```
query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2011ProductsCustomers WHERE SourceID=6 AND JoinDate=2011"
profit2011Source6 = as.numeric(sqldf(query))

profit2011Source = c(profit2011Source1, profit2011Source2, profit2011Source3, profit2011Source4, profit2011Source5, profit2011Sour
ce6)
print("Profit in 2011 from customers acquired in 2011 by source")
print(head(profit2011Source))
print(" ")

print("End 2011")


######################
# Process 2010 data #
######################
print(" ")
print("Begin 2010")
print(" ")

# Join order-product-customer data
query = "SELECT orders2010.*, products.ProductPrice, products.ProductCost FROM orders2010 INNER JOIN products ON orders2010.Produc
tID = products.ProductID"
orders2010Products = sqldf(query)

query = "SELECT orders2010Products.*, customers.JoinDate, customers.SourceID, customers.City, customers.State, customers.Country,
customers.Zipcode FROM orders2010Products INNER JOIN customers ON orders2010Products.CustomerID = customers.CustomerID"
orders2010ProductsCustomers = sqldf(query)

# Calculate revenue by source
query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=1"
revenue2010Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=2"
revenue2010Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=3"
revenue2010Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=4"
revenue2010Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=5"
revenue2010Source5 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2010ProductsCustomers WHERE SourceID=6"
revenue2010Source6 = as.numeric(sqldf(query))

revenue2010Source = c(revenue2010Source1, revenue2010Source2, revenue2010Source3, revenue2010Source4, revenue2010Source5, revenue2
010Source6)

print("Revenue in 2010 by source")
print(head(revenue2010Source))
print(" ")
print("Total revenue in 2010 rolled up from revenue by source")
print(sum(revenue2010Source))
print(" ")
print("Total revenue in 2010")
```

Appendix



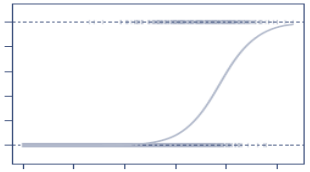

```
print(sum(orders2010$ExtRevenue))
print(" ")

# Calculate profit (actually contribution margin) by source for new customers
query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=1 AND JoinDate=2010"
profit2010Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=2 AND JoinDate=2010"
profit2010Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=3 AND JoinDate=2010"
profit2010Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=4 AND JoinDate=2010"
profit2010Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=5 AND JoinDate=2010"
profit2010Source5 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2010ProductsCustomers WHERE SourceID=6 AND JoinDate=2010"
profit2010Source6 = as.numeric(sqldf(query))

profit2010Source = c(profit2010Source1, profit2010Source2, profit2010Source3, profit2010Source4, profit2010Source5, profit2010Sour
ce6)
print("Profit in 2010 from customers acquired in 2010 by source")
print(head(profit2010Source))
print(" ")

print("End 2010")


######################
# Process 2009 data #
######################
print(" ")
print("Begin 2009")
print(" ")

# Join order-product-customer data
query = "SELECT orders2009.*, products.ProductPrice, products.ProductCost FROM orders2009 INNER JOIN products ON orders2009.Produc
tID = products.ProductID"
orders2009Products = sqldf(query)

query = "SELECT orders2009Products.*, customers.JoinDate, customers.SourceID, customers.City, customers.State, customers.Country,
customers.Zipcode FROM orders2009Products INNER JOIN customers ON orders2009Products.CustomerID = customers.CustomerID"
orders2009ProductsCustomers = sqldf(query)

# Calculate revenue by source
query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=1"
revenue2009Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=2"
revenue2009Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=3"
revenue2009Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=4"
```

Appendix




```
revenue2009Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=5"
revenue2009Source5 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue) FROM orders2009ProductsCustomers WHERE SourceID=6"
revenue2009Source6 = as.numeric(sqldf(query))

revenue2009Source = c(revenue2009Source1, revenue2009Source2, revenue2009Source3, revenue2009Source4, revenue2009Source5, revenue2
009Source6)

print("Revenue in 2009 by source")
print(head(revenue2009Source))
print(" ")
print("Total revenue in 2009 rolled up from revenue by source")
print(sum(revenue2009Source))
print(" ")
print("Total revenue in 2009")
print(sum(orders2009$ExtRevenue))
print(" ")

# Calculate profit (actually contribution margin) by source for new customers
query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=1 AND JoinDate=2009"
profit2009Source1 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=2 AND JoinDate=2009"
profit2009Source2 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=3 AND JoinDate=2009"
profit2009Source3 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=4 AND JoinDate=2009"
profit2009Source4 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=5 AND JoinDate=2009"
profit2009Source5 = as.numeric(sqldf(query))

query = "SELECT SUM(ExtRevenue - Qty * ProductCost) FROM orders2009ProductsCustomers WHERE SourceID=6 AND JoinDate=2009"
profit2009Source6 = as.numeric(sqldf(query))

profit2009Source = c(profit2009Source1, profit2009Source2, profit2009Source3, profit2009Source4, profit2009Source5, profit2009Sour
ce6)
print("Profit in 2009 from customers acquired in 2009 by source")
print(head(profit2009Source))
print(" ")

print("End 2009")


########################################
# Aggregate data across multiple years #
########################################

# Aggregate profit (actually contribution margin) for new customers across years by source
profitSource1 = c(profit2009Source1, profit2010Source1, profit2011Source1)
profitSource2 = c(profit2009Source2, profit2010Source2, profit2011Source2)
profitSource3 = c(profit2009Source3, profit2010Source3, profit2011Source3)
profitSource4 = c(profit2009Source4, profit2010Source4, profit2011Source4)
```

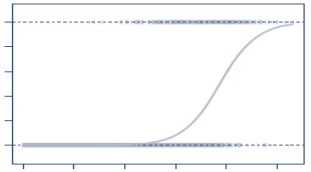
Appendix



customer_transaction_analysis.R Page 7

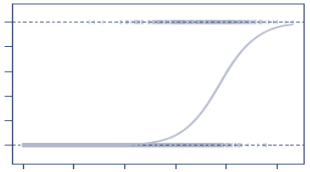
```
profitSource5 = c(profit2009Source5, profit2010Source5, profit2011Source5)
profitSource6 = c(profit2009Source6, profit2010Source6, profit2011Source6)
years = as.numeric(c(2009, 2010, 2011))

# Calculate return on investment (actually contribution margin / marketing cost) for new customers by year and source
return2009Source1 = profit2009Source1 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 1), 3]
return2010Source1 = profit2010Source1 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 1), 3]
return2011Source1 = profit2011Source1 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 1), 3]

return2009Source2 = profit2009Source2 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 2), 3]
return2010Source2 = profit2010Source2 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 2), 3]
return2011Source2 = profit2011Source2 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 2), 3]

return2009Source3 = profit2009Source3 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 3), 3]
return2010Source3 = profit2010Source3 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 3), 3]
return2011Source3 = profit2011Source3 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 3), 3]

return2009Source4 = profit2009Source4 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 4), 3]
return2010Source4 = profit2010Source4 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 4), 3]
return2011Source4 = profit2011Source4 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 4), 3]

return2009Source5 = profit2009Source5 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 5), 3]
return2010Source5 = profit2010Source5 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 5), 3]
return2011Source5 = profit2011Source5 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 5), 3]

return2009Source6 = profit2009Source6 / marketingCosts[(marketingCosts$Year == 2009) & (marketingCosts$SourceID == 6), 3]
return2010Source6 = profit2010Source6 / marketingCosts[(marketingCosts$Year == 2010) & (marketingCosts$SourceID == 6), 3]
return2011Source6 = profit2011Source6 / marketingCosts[(marketingCosts$Year == 2011) & (marketingCosts$SourceID == 6), 3]

# Aggregate return on investment (actually contribution margin / marketing cost) for new customers across years by source
returnSource1 = as.numeric(c(return2009Source1, return2010Source1, return2011Source1))
returnSource2 = as.numeric(c(return2009Source2, return2010Source2, return2011Source2))
returnSource3 = as.numeric(c(return2009Source3, return2010Source3, return2011Source3))
returnSource4 = as.numeric(c(return2009Source4, return2010Source4, return2011Source4))
returnSource5 = as.numeric(c(return2009Source5, return2010Source5, return2011Source5))
returnSource6 = as.numeric(c(return2009Source6, return2010Source6, return2011Source6))

# Plot return on investment (actually contribution margin / marketing cost) for new customers across years by source
returnSource = cbind(returnSource1,returnSource2,returnSource3,returnSource4,returnSource5,returnSource6)
matplot(years, returnSource,type="l",col=c("red","blue","green","orange","violet","brown"),lty=c(1,1),xlab="Year",ylab="ROI")
legend(2009,-0.25,legend=c("Source 1", "Source 2", "Source 3", "Source 4", "Source 5", "Source 6"),col=c("red","blue","green","ora
nge","violet","brown"),lty=c(1,1),box.col="white",bg="white")
title(main = "Return on investment: Contribution margin / Marketing cost")
x11()


#########################################################################
# Aggregate joined order-product-customer data across multiple years #
#########################################################################

# Orders joined with products and customers for the last 3 years
ordProdCust3Yrs = rbind(orders2009ProductsCustomers, orders2010ProductsCustomers, orders2011ProductsCustomers)

# Orders joined with products and customers for the last 2 years
ordProdCust2Yrs = rbind(orders2010ProductsCustomers, orders2011ProductsCustomers)


#################################
```

Appendix



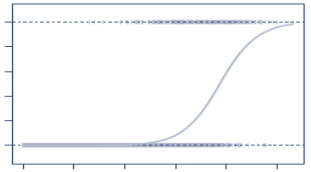

```
# Find behavior-based clusters #
################################

totalRevenuesPerOrder = aggregate(ordProdCust2Yrs$ExtRevenue ~ ordProdCust2Yrs$OrderID, FUN=sum)
colnames(totalRevenuesPerOrder) = c("OrderID", "Revenue")

query = "SELECT DISTINCT OrderID, CustomerID, OrderDate FROM ordProdCust2Yrs"
distinctOrders = sqldf(query)

query = "SELECT distinctOrders.*, totalRevenuesPerOrder.Revenue FROM distinctOrders INNER JOIN totalRevenuesPerOrder ON distinctOr
ders.OrderID = totalRevenuesPerOrder.OrderID"
distinctOrdersTotalRevenues = sqldf(query)


#########################################################################
# Calculate mean interval (in days) between orders (by customer) as  #
#      (last order date - first order date) / (total orders - 1)      #
#########################################################################

lastOrderDate = aggregate(distinctOrdersTotalRevenues$OrderDate ~ distinctOrdersTotalRevenues$CustomerID, FUN=max)
colnames(lastOrderDate) = c("CustomerID", "LastOrderDate")
print(head(lastOrderDate))
print(lastOrderDate[which(lastOrderDate$CustomerID==26943), ])

firstOrderDate = aggregate(distinctOrdersTotalRevenues$OrderDate ~ distinctOrdersTotalRevenues$CustomerID, FUN=min)
colnames(firstOrderDate) = c("CustomerID", "FirstOrderDate")
print(head(firstOrderDate))
print(firstOrderDate[which(firstOrderDate$CustomerID==26943), ])

# Total orders by customer
totalOrders = aggregate(distinctOrdersTotalRevenues$OrderDate ~ distinctOrdersTotalRevenues$CustomerID, FUN=length)
colnames(totalOrders) = c("CustomerID", "TotalOrders")
print(head(totalOrders))
print(totalOrders[which(totalOrders$CustomerID==26943), ])

print(head(totalOrders[which(totalOrders$TotalOrders==3), ]))
print( (lastOrderDate[which(lastOrderDate$CustomerID==26943), 2] - firstOrderDate[which(firstOrderDate$CustomerID==26943), 2]) / (
totalOrders[which(totalOrders$CustomerID==26943), 2] - 1) )

query = "SELECT lastOrderDate.*, firstOrderDate.FirstOrderDate FROM lastOrderDate INNER JOIN firstOrderDate on lastOrderDate.Custo
merID = firstOrderDate.CustomerID"
orderHistory = sqldf(query)
query = "SELECT orderHistory.*, totalOrders.TotalOrders FROM orderHistory INNER JOIN totalOrders on orderHistory.CustomerID = tota
lOrders.CustomerID"
orderHistory = sqldf(query)

query = "SELECT ( (orderHistory.LastOrderDate - orderHistory.FirstOrderDate) / (orderHistory.TotalOrders - 1) ), orderHistory.Cust
omerID FROM orderHistory"
meanOrderInterval = sqldf(query)
colnames(meanOrderInterval) = c("MeanOrderInterval", "CustomerID")

# Sanity check
print("Mean order interval for customer 26943 equals")
print(meanOrderInterval[which(meanOrderInterval$CustomerID==26943), ])

# Total revenue by customer
totalRevenue = aggregate(distinctOrdersTotalRevenues$Revenue ~ distinctOrdersTotalRevenues$CustomerID, FUN=sum)
colnames(totalRevenue) = c("CustomerID", "TotalRevenue")
```

Appendix



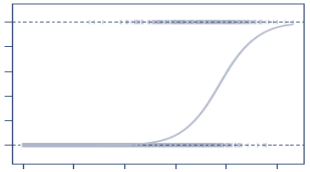

```
# Sanity check
print(head(totalRevenue))
print(totalRevenue[which(totalRevenue$CustomerID==26943), ])

# Mean revenue per order by customer
meanOrderRevenue = aggregate(distinctOrdersTotalRevenues$Revenue ~ distinctOrdersTotalRevenues$CustomerID, FUN=mean)
colnames(meanOrderRevenue) = c("CustomerID", "MeanRevenue")

# Sanity check
print("Sanity check mean revenue per order by customer")
print(meanOrderRevenue[which(meanOrderRevenue$CustomerID==26943), ])

# Mean price per item by customer
meanPriceCustomer = aggregate(ordProdCust2Yrs$ProductPrice ~ ordProdCust2Yrs$CustomerID, FUN=mean)
colnames(meanPriceCustomer) = c("CustomerID", "MeanPrice")

# Sanity check
print("Sanity check mean price per item by customer")
print(meanPriceCustomer[which(meanPriceCustomer$CustomerID==17551), ])

testResult = sqldf("SELECT ProductPrice FROM ordProdCust2Yrs WHERE CustomerID=17551")
print(mean(testResult[,1]))

print(sqldf("SELECT * FROM ordProdCust2Yrs WHERE CustomerID=17551"))

# Mean discount per item by customer
meanDiscountCustomer = aggregate(ordProdCust2Yrs$UnitDiscountPercent ~ ordProdCust2Yrs$CustomerID, FUN=mean)
colnames(meanDiscountCustomer) = c("CustomerID", "MeanDiscount")

# Sanity check
print("Sanity check mean discount per item by customer")
print(meanDiscountCustomer[which(meanDiscountCustomer$CustomerID==17551), ])

testResult = sqldf("SELECT UnitDiscountPercent FROM ordProdCust2Yrs WHERE CustomerID=17551")
print(mean(testResult[,1]))

print(sqldf("SELECT * FROM ordProdCust2Yrs WHERE CustomerID=17551"))

# Pack mean order revenue, order interval, item price and item discount into customer profile
customerProfile = merge(totalRevenue, totalOrders, by="CustomerID")
customerProfile = merge(customerProfile, meanOrderRevenue, by="CustomerID")
customerProfile = merge(customerProfile, meanOrderInterval, by="CustomerID")
customerProfile = merge(customerProfile, meanPriceCustomer, by="CustomerID")
customerProfile = merge(customerProfile, meanDiscountCustomer, by="CustomerID")

# Replace NA's with 1,000,000 for mean order intervals since NA's imply no repeat orders
customerProfile$MeanOrderInterval[is.na(customerProfile$MeanOrderInterval)] = 100000
customerProfile$MeanOrderInterval = as.numeric(customerProfile$MeanOrderInterval)

# Sanity check
print(customerProfile[which(customerProfile$CustomerID==26943), ])
print(customerProfile[which(customerProfile$CustomerID==17551), ])

# Remove customer ID's from profiles before inputting profiles to K-means clustering algorithm
clusterProfile = subset(customerProfile, select = -CustomerID)

# Find clusters (based on 6 variables) using K-means clustering
```

Appendix




```
clusters = kmeans(clusterProfile, 3, nstart=20)
print(clusters$centers)
print(clusters$size)

write.csv(clusters$centers, file="Clusters Centers.csv")
write.csv(clusters$size, file="Clusters Size.csv")

# Find clusters (based on 2 variables) using K-means clustering
clusterProfile2 = clusterProfile
clusterProfile2 = subset(clusterProfile2, select = -MeanRevenue)
clusterProfile2 = subset(clusterProfile2, select = -MeanPrice)
clusterProfile2 = subset(clusterProfile2, select = -MeanDiscount)
clusterProfile2 = subset(clusterProfile2, select = -TotalOrders)

clusters2 = kmeans(clusterProfile2, 3, nstart=20)
print(clusters2$centers)
print(clusters2$size)

write.csv(clusters2$centers, file="Clusters Centers 2.csv")
write.csv(clusters2$size, file="Clusters Size 2.csv")


####################################
# Rank customers on profitability #
####################################

# Profit (actually contribution margin) by customer
Profit = ordProdCust2Yrs$ExtRevenue - ordProdCust2Yrs$Qty * ordProdCust2Yrs$ProductCost
ordProdCustProf2Yrs = cbind(ordProdCust2Yrs, Profit)
profCust2Yrs = tapply(ordProdCustProf2Yrs$Profit, ordProdCustProf2Yrs$CustomerID, sum)
customerContributionMargin = as.numeric(profCust2Yrs)
print(head(customerContributionMargin))
hist(customerContributionMargin, 80, ylab="Customer count",xlab="Contribution margin",main="Customer profitability histogram")
x11()


#############################################
# Find cross-sell product recommendations #
#############################################

# Find product recommendations using using affinity analysis
ordProdCust = ordProdCust2Yrs[1:30, ]

# Define number of rows, columns and elements of user-item matrix
numberRows = length(unique(ordProdCust$CustomerID))
numberColumns = length(unique(ordProdCust$ProductID))
numberElements = length(unique(ordProdCust$CustomerID)) * length(unique(ordProdCust$ProductID))
userItemMatrix = (1 : numberElements)
dim(userItemMatrix) = c(numberRows, numberColumns)

# Define row and column names of user-item matrix
rowNames = unique(ordProdCust$CustomerID)
columnNames = unique(ordProdCust$ProductID)
dimnames(userItemMatrix) = list( c(rowNames), c(columnNames) )

# Populate user-item matrix with
#     1's wherever the row-th user purchased the column-th product
#     0's everywhere else
```

Appendix




```
userItemMatrix[ , ] = 0
numberIterations = length(ordProdCust$CustomerID)
for(i in 1:numberIterations) userItemMatrix[ordProdCust$CustomerID[i], ordProdCust$ProductID[i]] = 1

corPlot(t(userItemMatrix[1:20,1:20]), lab.both.axes=TRUE)
title(main="User-item purchase matrix",xlab="Item",ylab="User")
x11()

# Cast user-item matrix as "realRatingMatrix" data type defined in "recommender" package
realUserItemMatrix = as(userItemMatrix, "realRatingMatrix")
print(getRatingMatrix(realUserItemMatrix))

# Binarize user-item data
binaryUserItemMatrix = binarize(realUserItemMatrix, minRating=1)
# print(as(binaryUserItemMatrix, "matrix"))

# Cast user-item matrix as "itemMatrix" data type defined in "arules" package
itemUserMatrix = as(userItemMatrix, "itemMatrix")

# Perform affinity analysis using user-item data
affinityMatrix = affinity(itemUserMatrix)
print(affinityMatrix)

# Present affinity matrix visually
corPlot(affinityMatrix, lab.both.axes=TRUE)
title(main="Item-item affinity matrix",xlab="Item",ylab="Item")

print("End Customer Transaction Analysis")
```