Statistical Analysis of Transaction Data

Objective

In predicting the customer purchase behavior, we usually employ classic market basket analysis to discover the regularities between products in large-scale transaction records. Such information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements.

In this study, we are interested to estimate a consumer's total expenditure during Black Friday first and cluster consumers based on their shopping records. The goal is to identify key determinants associated with their purchase capability and then to identify subgroups composed of similar consumers, so that we are able to predict the purchase behavior within each subgroup of consumers. In other words, we can tailor products and branding in a way attractive to the target groups of consumers.

Dataset

The dataset we used is a sample of Black Friday transactions made in a retail store. It contains 537,577 purchase records and 12 variables, including user ID, product ID, gender, age, occupation, current city living in, years of residence, marital status, categories of purchased products and purchase expenditure (in dollars).

Method

To address the first objective, we took consumers' total purchase on Black Friday as response. There are 5,891 observations after aggregating the purchase records for each consumer. We applied log transformation to handle the skewness in total purchase. Exploratory analysis was then performed to investigate how consumers' demographic factors, including age, gender, occupation, marital status etc., affect their expenditure. Furthermore, we employed association rules to develop an intuitive understanding about how these variables related to the response and to each other. Specifically, we segmented the total purchase into three levels – low, mediate and high, using *discrete* function under association rules. Then we restricted the right hand side of the rules to be these three levels in order to get the desired plot for rules.

Before building prediction models to estimate the total purchase per person, we randomly assigned the observations into training set or testing set with ratio 7:3. We fitted multiple linear regression using least squares at first. As there are over 30 dummy variables in the data and we also would like to consider possible interactions between variables, the least squares estimates then are more likely to suffer from large variability, which results in poor predictions for observations in the testing set. Therefore, we also applied other fitting approaches, including Lasso Regression and Principle Components Regression(PCR) in order to select most relevant variables from the data as well as yield better prediction accuracy for observations not used in model training. We used *glmnet( )* in R to perform lasso regression. It automatically gives two kinds of tuning parameter (λ), λ_{min} and λ_{1se} respectively, based on the results of cross validation. We fitted both of them to the lasso, which generated different models. Next, we used *pcr( )* to build a PCR model on the training set and compared its performance to the models we obtained from previous procedures. The best model, with relatively small prediction error, was selected based on ten-fold cross validation methods.

Next, we conducted hierarchical clustering analysis to find subgroups of consumers. Hierarchical clustering is preferred as we do not have enough information to determine the exact number of clusters. we transformed our data to the form of a matrix where the rows represent each consumer, while the columns are listed by product ID and each cell element is the expenditure a given shopper has purchased a given product in order. We just selected 20 most popular products in this store for computational efficiency. Correlation-based distance was used to cluster the consumers with similar preferences for different products. As products with much higher prices than others tend to have a greater effect in measuring the purchase dissimilarities between consumers, we scaled each variable to have standard deviation one before computing the correlation-based distance, so that each product will be given equal importance in the hierarchical clustering performed. For linkage methods, we applied average and complete linkage to yield more balanced, attractive clusters. Furthermore, we tried to perform hierarchical clustering on the first few principal component score vectors rather than the entire data matrix and compared its result to the ones we obtained though clustering on the full data set.

Results

Through the plots from exploratory analysis, we noticed an obvious difference in purchase curves for consumers with different gender and marital status. For married female consumers, their purchase capacity is overall negatively associated with their ages. Those who age between 26-35 have the largest purchase while those who age over 55 have the least. For unmarried female consumers, however, it is hard to predict their purchase capacity simply from the plot. Those who age over 50 have a sharp increase in purchase compared to those who age between 46-50, but are still at a lower level in purchase compared to those who age between 18-45. Similarly, the average purchase for married male consumers who have stayed over than one years in a city is much less than unmarried male consumers. Therefore, we assumed there might exist interaction between gender and marital status. It is worth noting that the results we obtained from association rules are basically consistent with our assumption. After setting support rate to 0.1 and confidence level to 0.8, ten rules were generated (see **Figure.1**). The middle sized points between *genderM* and two kinds of marital status also indicate possible interaction between gender and marital status. However, it seems that there is not too much interaction between other variables. Therefore, we only included the interaction term for gender and marital status in fitting multiple linear regression.

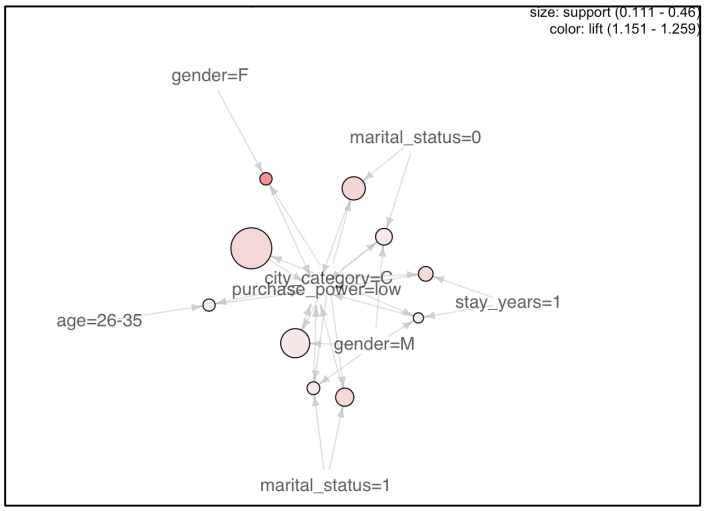


Figure.1 Plot of Association Rules

We fitted least squares regression, lasso regression and PCR on the training set and then calculated the test cross validation error respectively. The coefficient estimates and test errors for each model are shown (see **Table.1**). It shows that the least squares regression gives the smallest test error (0.8471, [0.8346,0.8596]) compared to other methods. In lasso regression, fitting with λ_{min} forces 17 coefficient estimates to be zero due to its penalty term and we obtained a model with relatively small test error (0.8673, [0.8447, 0.8899]). We notice that the coefficient of interaction term is also set to be zero, indicating the interaction between gender and marital status might not have a strong impact on consumers' purchase capacity. When λ_{1se} is applied, lasso forces more coefficients to be zero and only keeps gender, age between 26-35, age over 55, currently living in city B or C as predictors. This also results in a slight increase in test error (0.8797, [0.8573,0.9021]) compared to the model using λ_{min}. We displayed the choice of λ that results from performing ten-fold cross-validation in the lasso regression (see **Figure.2**). The dashed vertical lines indicate the selected value of λ. In this case, the value is relatively small (λ_{1se} =0.0573), indicating that only a small amount of shrinkage relative to the least squares solution have been exerted to yield an optimal fit in lasso regression. In addition, the decrease of the test error is not very evident from the plot. In a case like this we might simply go with the least squares solution, which also accounts for the lower test error obtained from the least squares regression. In performing PCR, we also computed the ten-fold cross-validation error for each possible number of principal components used. We find that the lowest error occurs when 31 components are used. This basically amounts to performing least squares regression as PCR would not achieve dimension reduction when all of the components are used. Although its test error (0.8707, [0.8476,0.8938]) is competitive with the results we obtained using lasso regression, the final model is rather difficult to interpret and it does not perform any kind of variable selection.

	least squares	lasso(lambda.min)	lasso(lambda.1se	pcr
(Intercept)	13.073290178	13.19419263	13.30160340	0.0000000000
genderM	0.302952285	0.27869396	0.17787472	0.1565359454
marital_statusmarried	0.017757576	0.00000000	0.00000000	0.0363430583
age18-25	0.133769297	0.00000000	0.00000000	0.0948199041
age26-35	0.268928171	0.11016783	0.02881082	0.1851180167
age36-45	0.226610232	0.04284748	0.00000000	0.1340761047
age46-50	0.179399005	0.00000000	0.00000000	0.0796455362
age51-55	0.135055958	-0.01352031	0.00000000	0.0606608439
age55+	-0.063999070	-0.18128275	-0.05930959	0.0140092323
stay_years1	-0.003186436	0.00000000	0.00000000	-0.0060441903
stay_years2	-0.006577716	0.00000000	0.00000000	0.0014666621
stay_years3	0.028131647	0.00000000	0.00000000	0.0014531023
stay_years4+	0.004569378	0.00000000	0.00000000	-0.0049051561
occupation1	-0.042762693	0.00000000	0.00000000	-0.0115552611
occupation2	-0.023233049	0.00217164	0.00000000	0.0062453437
occupation3	0.134325673	0.09663665	0.00000000	0.0232796689
occupation4	-0.004465121	0.00000000	0.00000000	-0.0055827454
occupation5	0.060978222	0.11186105	0.00000000	0.0215554098
occupation6	-0.126966676	-0.02880059	0.00000000	-0.0218423698
occupation7	-0.106710155	-0.02524419	0.00000000	-0.0289290058
occupation8	-0.153569149	0.00000000	0.00000000	-0.0007978762
occupation9	-0.137543863	0.00000000	0.00000000	-0.0104724796
occupation10	-0.008905319	-0.05237881	0.00000000	0.0166406696
occupation11	-0.049995789	0.00000000	0.00000000	-0.0005800516
occupation12	-0.072633131	0.00000000	0.00000000	-0.0125518958
occupation13	-0.159334446	-0.02585443	0.00000000	-0.0145426672
occupation14	-0.023005583	0.00000000	0.00000000	0.0002817950
occupation15	-0.037251762	0.00000000	0.00000000	-0.0038043893
occupation16	0.115119975	0.10296685	0.00000000	0.0267353130
occupation17	-0.091596605	0.00000000	0.00000000	-0.0155567997
occupation18	0.040274509	0.01344275	0.00000000	0.0102591378
occupation19	0.224926637	0.13447222	0.00000000	0.0288647797
occupation20	0.092211365	0.09741228	0.00000000	0.0256104525
city_categoryB	0.113132704	0.09889866	0.03513020	0.0527993877
city_categoryC	-0.588182989	-0.56908427	-0.53286255	-0.2867924759
genderM:marital_statusmarried	-0.030702040	0.00000000	0.00000000	-0.0454268133
MSE	0.847596476	0.86730023	0.87966461	0.8707155010

Table.1. Coefficient Estimates and Test MES

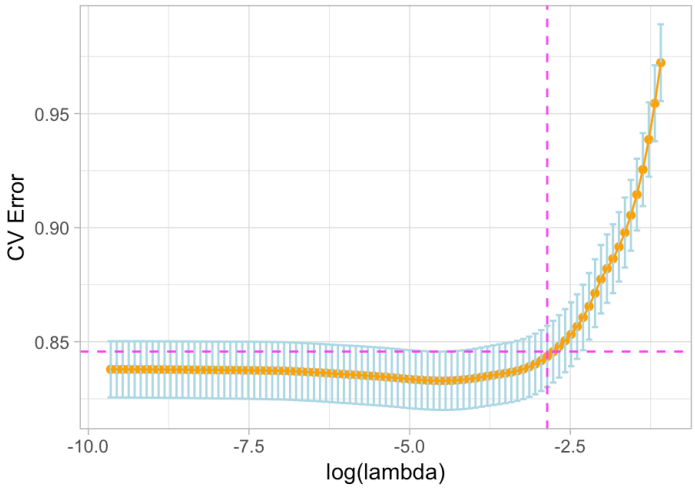


Figure.2 Ten-fold CV error with different choices of λ

The dendrograms we obtained from hierarchical clustering analysis using average or complete linkage are shown (see **Figure.3**). We then cut the two dendrograms at the height that will yield four clusters. It seems that complete linkage leads to more balanced clusters than average linkage. But what they have in common is that both of them separate product P00080342 as a single cluster. That is to say, people who have ever purchased this product in this store could be taken as a subgroup of the consumers. We extracted these consumers from the data and compared their age, gender, occupation and other characteristics. It shows that male consumers with age between 26-45 and more than one year of residence have a higher need for product P00080342. In addition, the purchase capacity for most of them is at a lower level. we also performed hierarchical clustering on the first five principal component score vectors but the clustering results it yields is very different from the ones we obtained using the full data set.

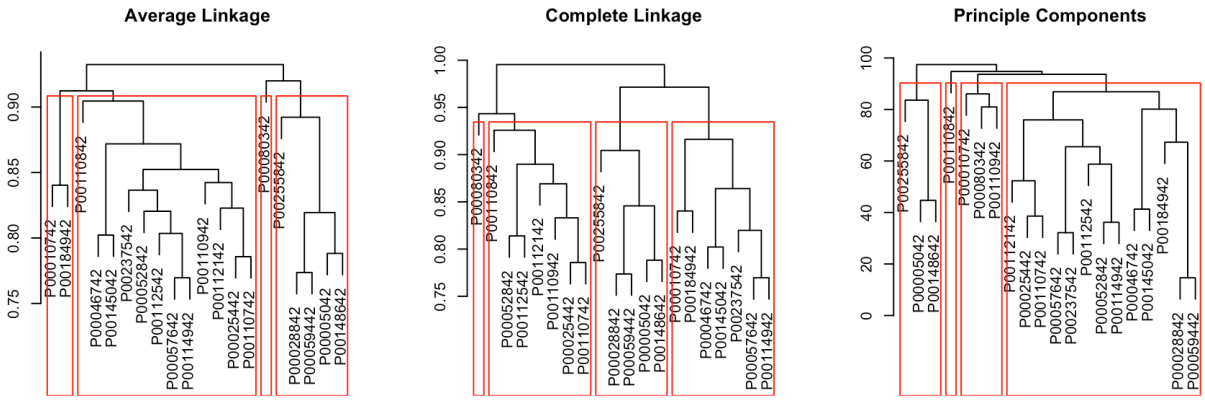


Figure.3 Dendrogram from Hierarchical Clustering

Discussion

Since this data only includes qualitative predictors, we just fitted linear models to explore the relationship between consumers' purchase capacity and their demographic characteristics. If *age* is given at a continuous scale, then we could consider applying Generalized Additive Models(GAMs) to allow the non-linear function for this quantitative variable, which may also help to further improve prediction accuracy for the response. We built four models in this study to predict a consumer's total purchase on Black Friday. Based on the results of cross validation, least squares model yields the lowest test error. In other cases, when the least squares estimates have excessively high variance, the lasso solution can yield a reduction in variance at the expense of a small increase in bias, and consequently generate more accurate predictions. Lasso also performs variable selection, an advantage over ridge regression and principle component regression. Overall, lasso regression usually outperforms other methods and yields a model with less test error and more interpretability.

We employed principle components in estimating the total purchase as well as performing clustering analysis. The relatively worse performance of PCR compared to other methods might result from the fact that the data were generated in such a way that many principal components are required in order to adequately model the response. In contrast, PCR will tend to perform well in cases when the first few principal components are sufficient to capture most of the variation in the predictors as well as the relationship with the response. Actually, performing clustering on the first few principal component score vectors can sometimes give better results than performing clustering on the full data. In this situation, we might think employing principal component score vectors as a way to denoise the data.

There are other problems related to the use of clustering analysis. Clustering methods generally are not very robust to the disturbance of the data. The clustering results would be very different if a subset of observations were removed from the data. In addition, it is difficult to validate the clusters we obtained. For further study, we are interested to investigate whether the clusters that have been found represent the true subgroups of consumers and assign p-values to each cluster to provide more evidence in the real world.

Final_pre

CHUHAN

11/14/2018

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'tibble' was built under R version 3.4.3
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.2
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
require(caTools)
library(ggplot2)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.4.4
```

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 3.4.4
```

```
library(dplyr)

data=read.csv("BlackFriday.csv") %>%
  janitor::clean_names() %>%
  rename(stay_years=stay_in_current_city_years)
data[c(5,8)]=lapply(data[c(5,8)],factor)
length(unique(data$user_id))
```

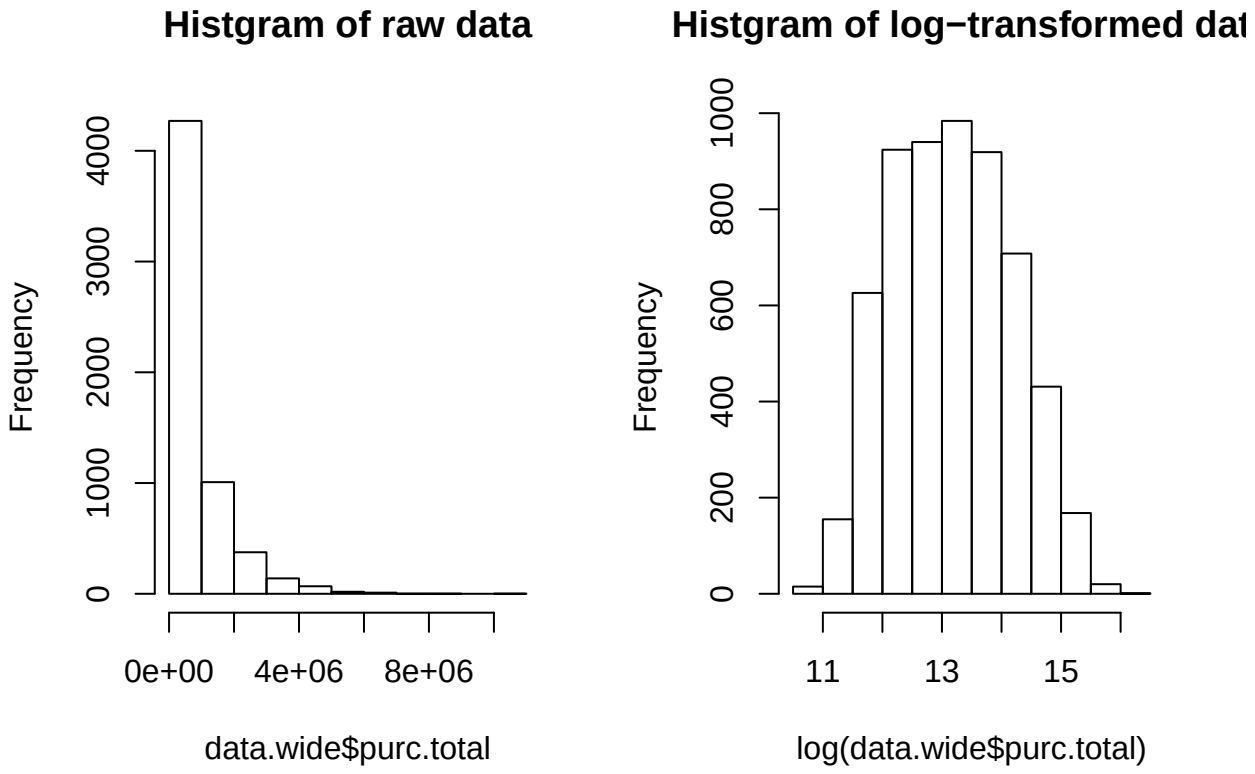
```
## [1] 5891
```

```
summary(data)
```

```
##     user_id            product_id     gender        age
##  Min.   :1000001   P00265242:  1858   F:132197   0-17 : 14707
##  1st Qu.:1001495   P00110742:  1591   M:405380   18-25: 97634
##  Median :1003031   P00025442:  1586              26-35:214690
##  Mean   :1002992   P00112142:  1539              36-45:107499
##  3rd Qu.:1004417   P00057642:  1430              46-50: 44526
##  Max.   :1006040   P00184942:  1424              51-55: 37618
##                    (Other)  :528149              55+  : 20903
##    occupation     city_category stay_years  marital_status
##  4      : 70862   A:144638      0 : 72725   0:317817
##  0      : 68120   B:226493      1 :189192   1:219760
##  7      : 57806   C:166446      2 : 99459
##  1      : 45971                 3 : 93312
##  17     : 39090                 4+: 82889
##  20     : 32910
```

```
##  (Other):222818
##  product_category_1 product_category_2 product_category_3    purchase
##  Min.   : 1.000     Min.   : 2.00      Min.   : 3.0       Min.   :  185
##  1st Qu.: 1.000     1st Qu.: 5.00      1st Qu.: 9.0       1st Qu.: 5866
##  Median : 5.000     Median : 9.00      Median :14.0       Median : 8062
##  Mean   : 5.296     Mean   : 9.84      Mean   :12.7       Mean   : 9334
##  3rd Qu.: 8.000     3rd Qu.:15.00      3rd Qu.:16.0       3rd Qu.:12073
##  Max.   :18.000     Max.   :18.00      Max.   :18.0       Max.   :23961
##                     NA's   :166986     NA's   :373299
```

```
data.wide=data %>%
  select(-product_id,-product_category_1,-product_category_2,product_category_3) %>%
  group_by(user_id,gender,age,occupation,city_category,stay_years,marital_status) %>%
  summarise(purc.total=sum(purchase))


#check the distribution of response
par(mfcol=c(1,2))
hist(data.wide$purc.total,main="Histgram of raw data")
hist(log(data.wide$purc.total),main="Histgram of log-transformed data") #log-transformation
```




```
data.wide$log.purchase=log(data.wide$purc.total)
levels(data.wide$marital_status)=c("unmarried","married")
```

Exploratory Analysis

```
library(gridExtra)
```

```
##
```

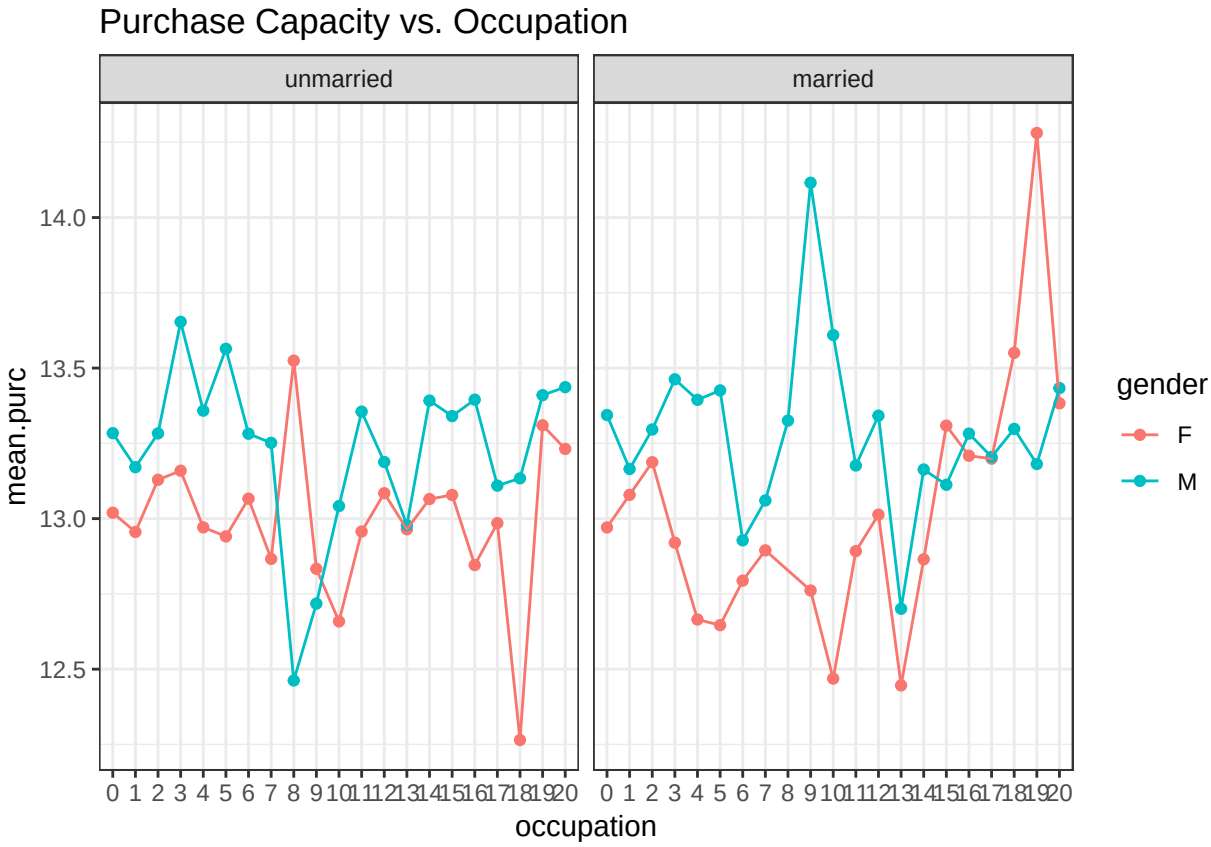
```
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
occupation=data.wide %>%
  group_by (gender,occupation,marital_status) %>%
  summarise(mean.purc=mean(log.purchase))
```

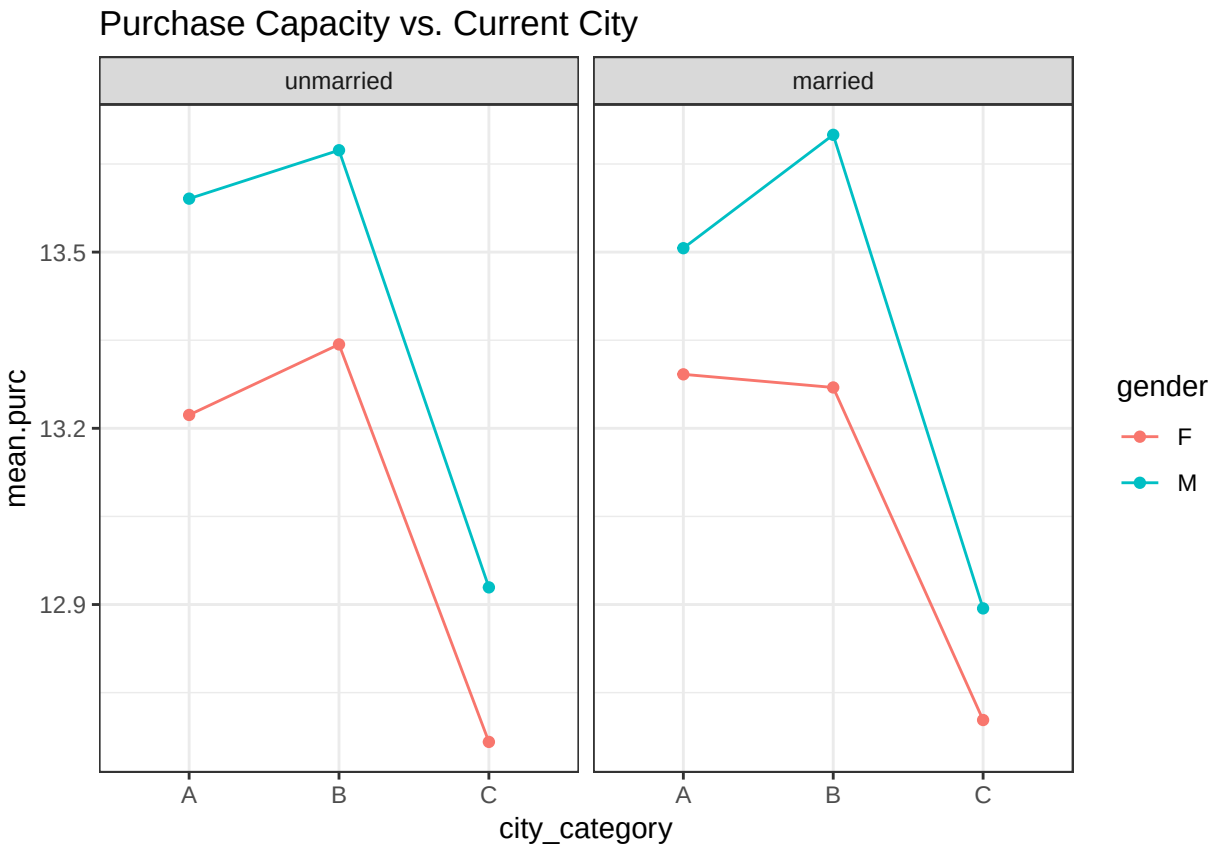
```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
p1=ggplot(data=occupation,aes(x=occupation,y=mean.purc,group=gender,color=gender))+geom_point()+geom_li
```

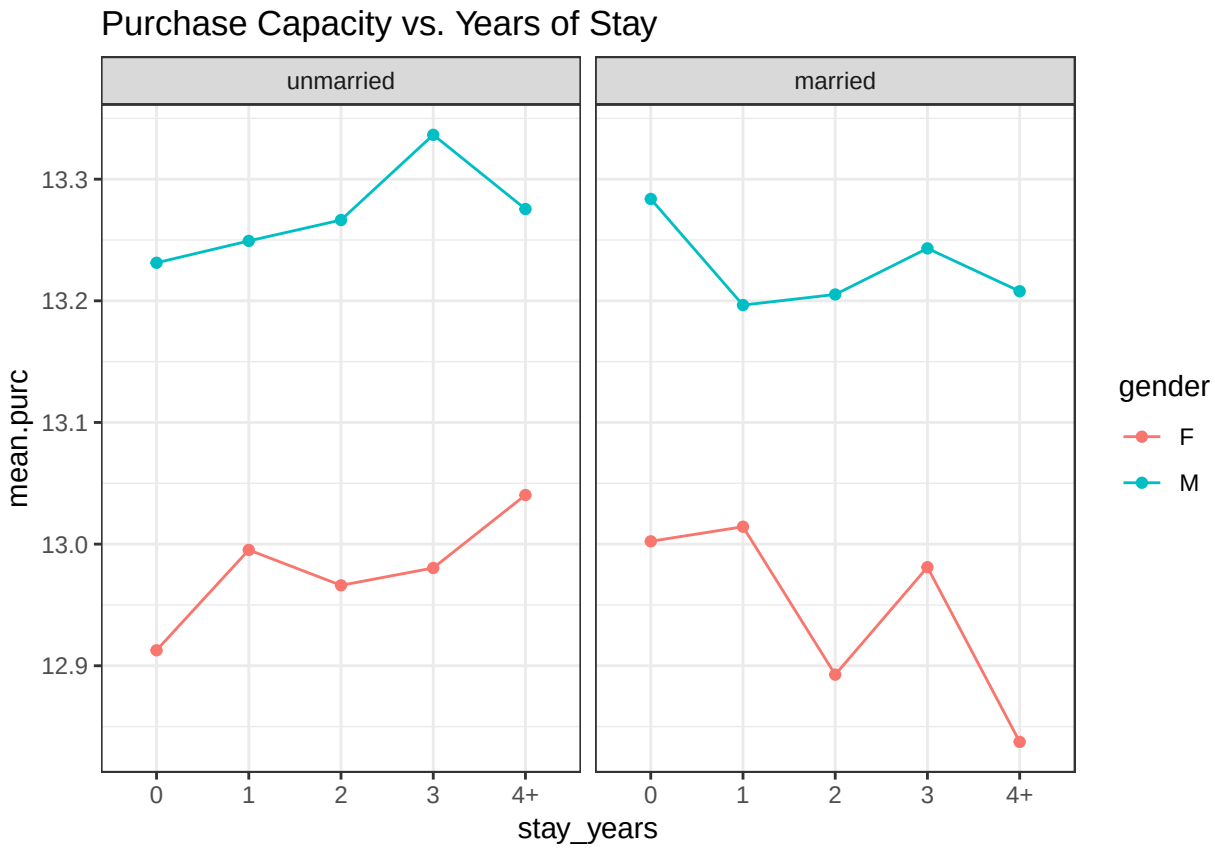



```
age=data.wide %>%
  group_by(gender,age,marital_status) %>%
  summarise(mean.purc=mean(log.purchase))
p2=ggplot(data=age,aes(x=age,y=mean.purc,group=gender,color=gender))+geom_point()+geom_line()+ggtitle("
```


```
city=data.wide %>%
  group_by(gender,city_category,marital_status) %>%
  summarise(mean.purc=mean(log.purchase))
p3=ggplot(data=city,aes(x=city_category,y=mean.purc,group=gender,color=gender))+geom_point()+geom_line()
```



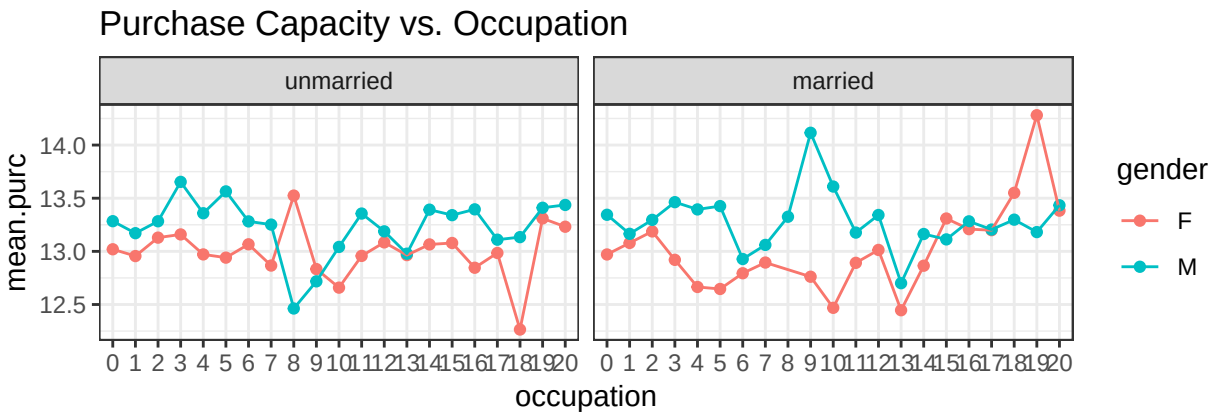

```
years=data.wide %>%
  group_by(gender,stay_years,marital_status) %>%
  summarise(mean.purc=mean(log.purchase))
p4=ggplot(data=years,aes(x=stay_years,y=mean.purc,group=gender,color=gender))+geom_point()+geom_line()+
```

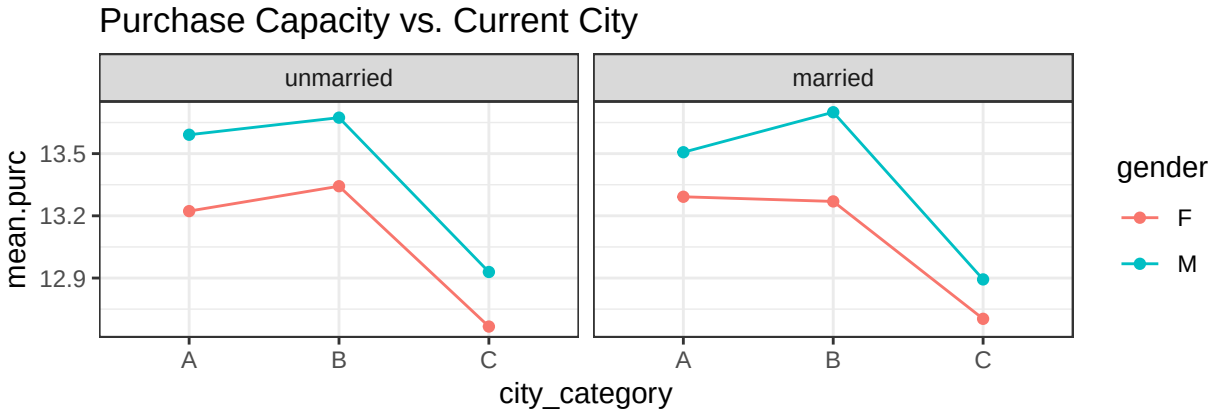



```
#product
product=data %>%
  group_by(product_id) %>%
  summarise(pu.sum=sum(purchase))
popular.prod=product %>% top_n(20)
```

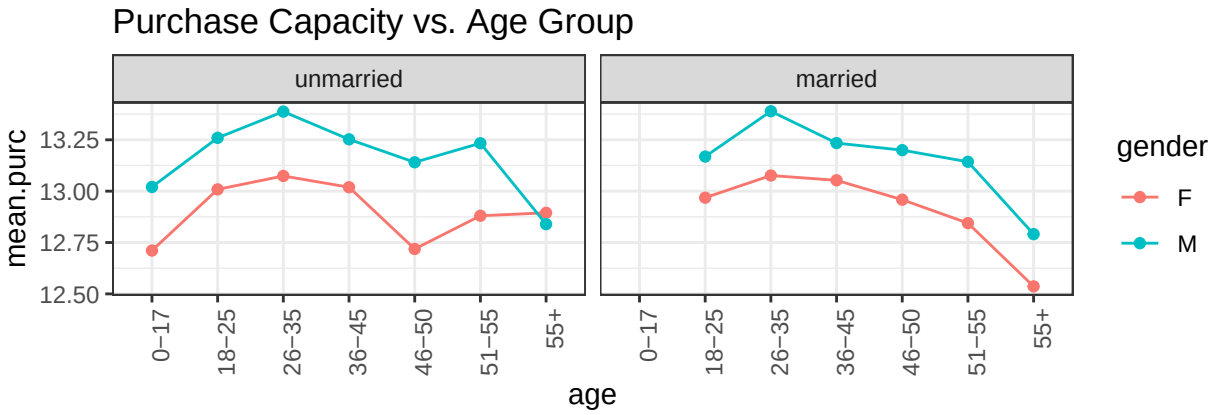
```
## Selecting by pu.sum
```

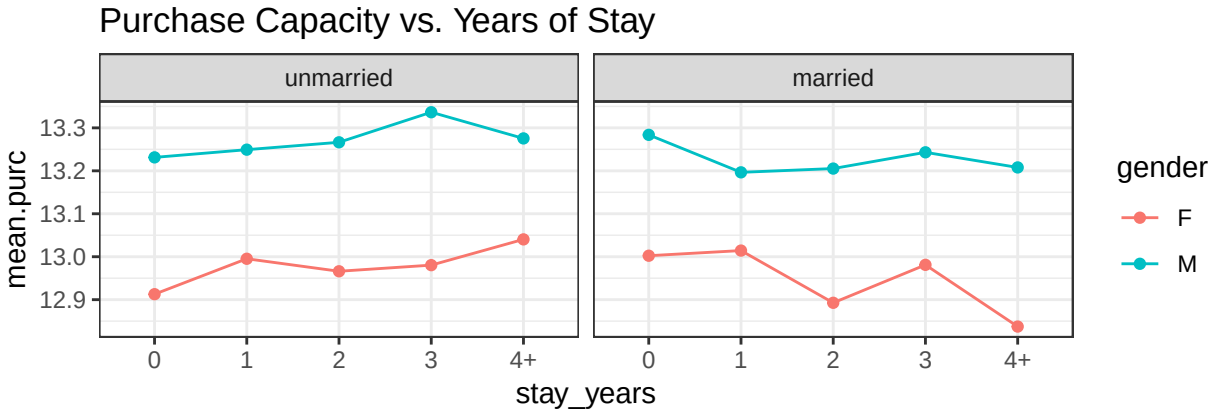
```
prod.names=as.character(popular.prod$product_id)
product.filter=data %>%
  filter(product_id %in% prod.names) %>%
  mutate(log.purchase=log(purchase)) %>%
  group_by(product_id,gender) %>%
  summarise(mean.purc=mean(log.purchase))
p6=ggplot(data=product.filter,aes(x=product_id,y=mean.purc,group=gender,color=gender))+geom_point()+geo

grid.arrange(grobs=list(p1,p3),width=c(3:2))
```






```
grid.arrange(grobs=list(p2,p4))
```





Association rules

```
library(PRIMsrc)

# implements unsupervised discretization
data.wide$purchase_df =as.factor(discretize(data.wide[[8]],method = "cluster",breaks = 3))
data.wide$purchase_capacity=as.factor(
  ifelse(data.wide$purc.total<1.01e+06,"low",
         ifelse((data.wide$purc.total<2.67e+06),"mediate","high")))


transdata= as(data.wide[,c(2:7,11)], "transactions")
inspect(transdata)
transdata=as(transdata, "data.frame")

rules <- apriori(transdata, parameter = list(minlen=2,supp = 0.1, conf = 0.8),
                 appearance =list(rhs=c("purchase_capacity=low","purchase_capacity=mediate","purchase_ca
summary(rules)
rules.sorted <- sort(rules, by="lift")
inspect(rules.sorted)

library(arulesViz)
plot(rules, method="graph", control=list(type="items"))
```

least squares regression

```
##
## Attaching package: 'boot'

## The following object is masked _by_ '.GlobalEnv':
##
##     city

## [1] 0.8475965

## [1] 0.01254525

##
## Call:
## glm(formula = log.purchase ~ gender * marital_status + age +
##     stay_years + occupation + city_category, data = data.wide)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.85082  -0.69788   0.03405   0.72130   2.45241
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  13.073290   0.114895 113.785  < 2e-16 ***
## genderM                       0.302952   0.036011   8.413  < 2e-16 ***
## marital_statusmarried         0.017758   0.046630   0.381  0.70335
## age18-25                      0.133769   0.104792   1.277  0.20182
## age26-35                      0.268928   0.105264   2.555  0.01065 *
## age36-45                      0.226610   0.107063   2.117  0.03433 *
## age46-50                      0.179399   0.112131   1.600  0.10967
## age51-55                      0.135056   0.113168   1.193  0.23275
## age55+                       -0.063999   0.116880  -0.548  0.58401
## stay_years1                  -0.003186   0.038746  -0.082  0.93446
## stay_years2                  -0.006578   0.042815  -0.154  0.87791
## stay_years3                   0.028132   0.044253   0.636  0.52500
## stay_years4+                  0.004569   0.044953   0.102  0.91904
## occupation1                  -0.042763   0.053938  -0.793  0.42792
## occupation2                  -0.023233   0.067283  -0.345  0.72988
## occupation3                   0.134326   0.079059   1.699  0.08936 .
## occupation4                  -0.004465   0.052867  -0.084  0.93269
## occupation5                   0.060978   0.094009   0.649  0.51659
## occupation6                  -0.126967   0.070737  -1.795  0.07272 .
## occupation7                  -0.106710   0.050533  -2.112  0.03475 *
## occupation8                  -0.153569   0.225570  -0.681  0.49602
## occupation9                  -0.137544   0.105782  -1.300  0.19356
## occupation10                 -0.008905   0.111916  -0.080  0.93658
## occupation11                 -0.049996   0.088647  -0.564  0.57278
## occupation12                 -0.072633   0.059275  -1.225  0.22049
## occupation13                 -0.159334   0.093480  -1.704  0.08835 .
## occupation14                 -0.023006   0.064098  -0.359  0.71967
## occupation15                 -0.037252   0.085318  -0.437  0.66240
## occupation16                  0.115120   0.070117   1.642  0.10068
## occupation17                 -0.091597   0.054818  -1.671  0.09479 .
## occupation18                  0.040275   0.117889   0.342  0.73264
## occupation19                  0.224927   0.115689   1.944  0.05191 .
```

```
## occupation20                    0.092211   0.065854   1.400  0.16150
## city_categoryB                  0.113133   0.036212   3.124  0.00179 **
## city_categoryC                 -0.588183   0.033303 -17.661  < 2e-16 ***
## genderM:marital_statusmarried -0.030702   0.053862  -0.570  0.56869
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.8412803)
##
##     Null deviance: 5804.7  on 5890  degrees of freedom
## Residual deviance: 4925.7  on 5855  degrees of freedom
## AIC: 15738
##
## Number of Fisher Scoring iterations: 2
```

lasso regression

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.4.3
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-16
```

```
# data partition
split <- initial_split(data.wide, prop = .7)
train <- training(split)
test  <- testing(split)

x=model.matrix(log.purchase~gender*marital_status+age+stay_years+occupation+city_category, data=train[c
x.test=model.matrix(log.purchase~gender*marital_status+age+stay_years+occupation+city_category, data=tes
y=train$log.purchase
y.test=test$log.purchase

# fit model
set.seed (11)
cv.lasso = cv.glmnet(x,y,alpha=1,nfolds=10)
cv.bestlam =cv.lasso$lambda.min
cv.bestlam #0.01178443
```

```
## [1] 0.01090361
```

```
cv.selam=cv.lasso$lambda.1se
cv.selam #0.05730299
```

```
## [1] 0.07692278
```

```
set.seed (11)
lasso.cv.model= glmnet(x,y,alpha=1, lambda=cv.bestlam)
predict.lasso.cv=predict(lasso.cv.model,newx=x.test)
mse.lasso1=mean((predict.lasso.cv - y.test)^2);mse.lasso1
```

```
## [1] 0.8786388
```

```
sd.lasso1=sd((predict.lasso.cv - y.test)^2)/sqrt(nrow(test));sd.lasso1
```

```
## [1] 0.0231901
```

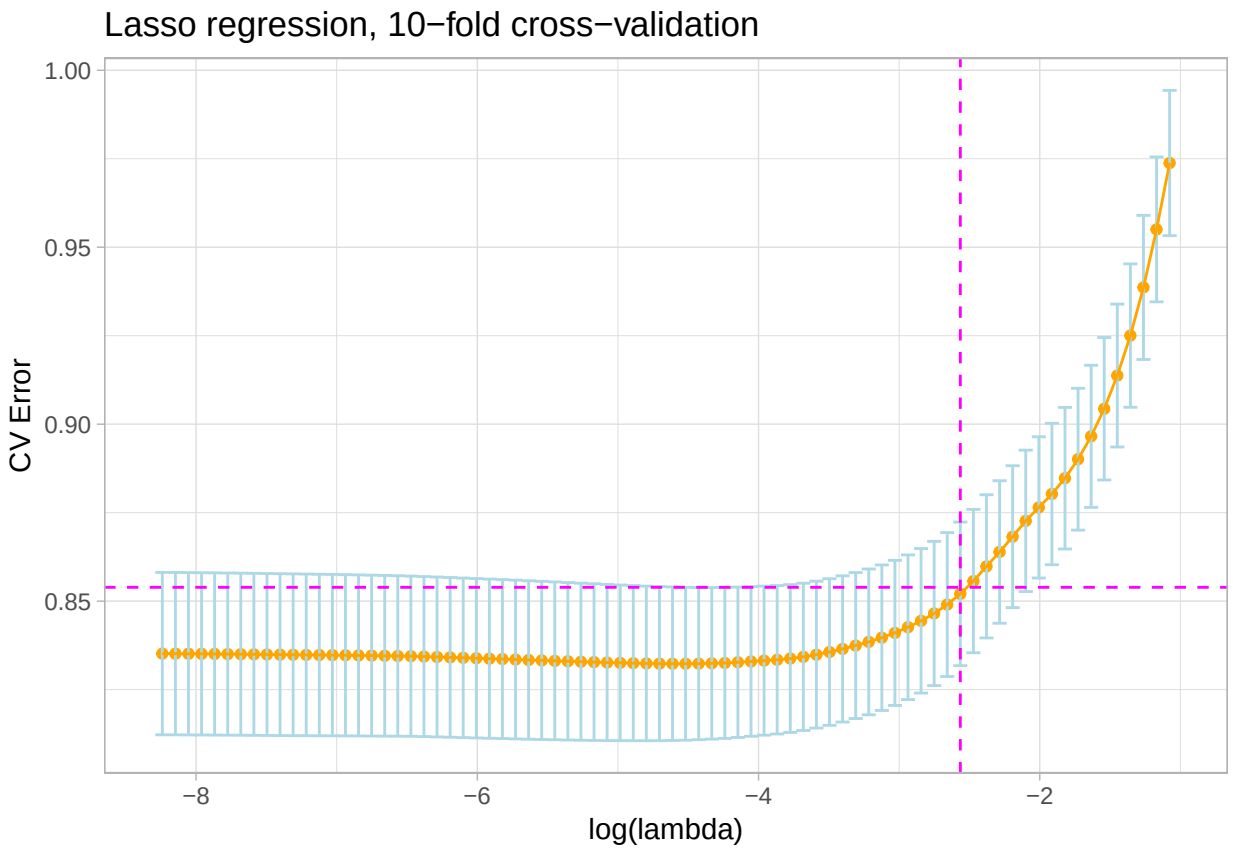
```
lasso.coef1=as.matrix(coef(lasso.cv.model))

lasso.cv.model= glmnet(x,y,alpha=1, lambda=cv.selam)
predict.lasso.cv=predict(lasso.cv.model,newx=x.test)
mse.lasso2=mean((predict.lasso.cv - y.test)^2);mse.lasso2
```

```
## [1] 0.8886866
```

```
sd.lasso2=sd((predict.lasso.cv - y.test)^2)/sqrt(nrow(test));sd.lasso2
```

```
## [1] 0.02291008
```

```
lasso.coef2=as.matrix(coef(lasso.cv.model))

lasso=data.frame(cbind(cv.lasso$nzero,cv.lasso$lambda,cv.lasso$cvm,cv.lasso$cvlo,cv.lasso$cvup))
colnames(lasso) = c("size","lambda","cvm","cvlo","cvup")

ggplot(lasso,aes(x=log(lambda),y=cvm)) + geom_point(color="orange") + geom_errorbar(aes(ymin=cvlo, ymax=
```



Principal Components Regression

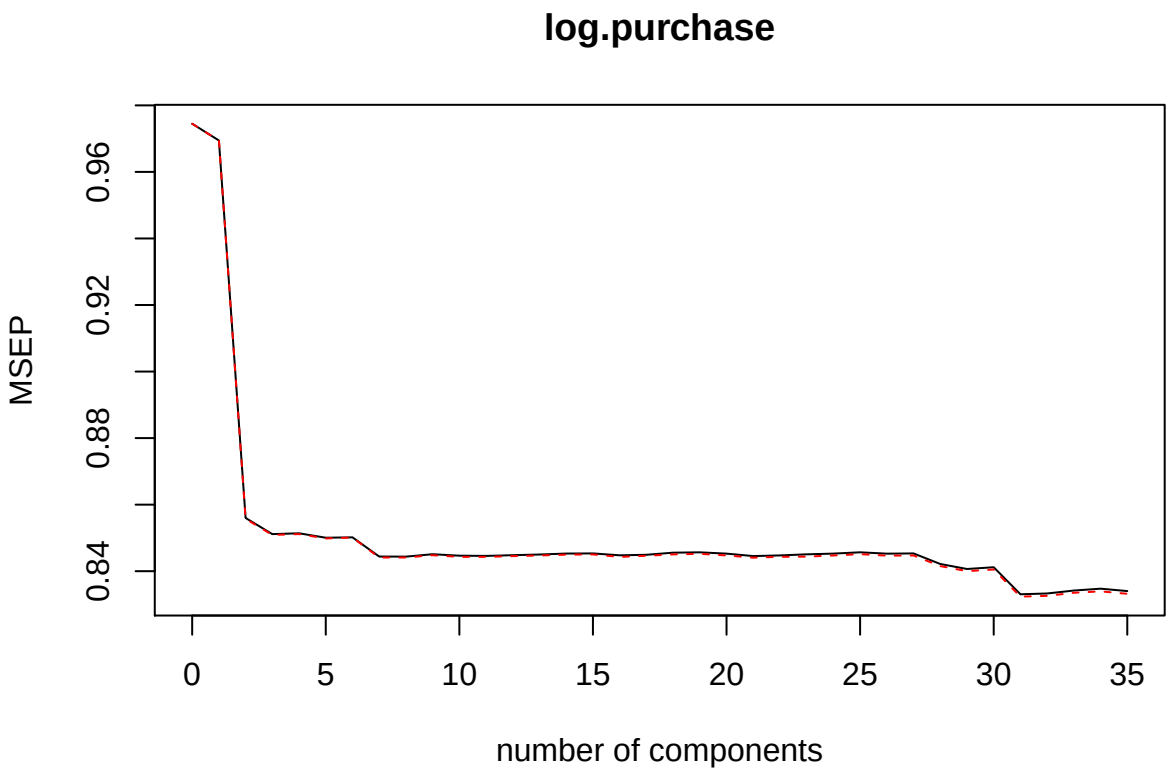
```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.4.4
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(11)
# Perform PCR on the training data and evaluate its test set performance.
pcr.fit=pcr(log.purchase~gender*marital_status+age+stay_years+occupation+city_category,data=train,scale=

# find the best number of components and choose k
validationplot(pcr.fit ,val.type="MSEP" )
```




```
itemp=which.min(pcr.fit$validation$PRESS);itemp
```

```
## [1] 31
```

```
pcr.fit=pcr(log.purchase~gender*marital_status+age+occupation+city_category+stay_years,data=train,scale=
summary(pcr.fit)
```

```
## Data:    X dimension: 4124 35
##  Y dimension: 4124 1
## Fit method: svdpc
## Number of components considered: 31
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.9872   0.9846   0.9256   0.9229   0.9231   0.9225   0.9220
## adjCV       0.9872   0.9846   0.9252   0.9227   0.9230   0.9224   0.9219
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.9191   0.9186   0.9189    0.9186    0.9187    0.9188    0.9189
## adjCV   0.9190   0.9185   0.9188    0.9184    0.9185    0.9187    0.9188
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.9193    0.9194    0.9190    0.9193    0.9187    0.9193
## adjCV    0.9191    0.9191    0.9187    0.9191    0.9185    0.9191
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV       0.9189    0.9192    0.9193    0.9196    0.9195    0.9193
## adjCV    0.9186    0.9189    0.9190    0.9192    0.9193    0.9191
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV       0.9193    0.9194    0.9176    0.9171    0.9174    0.9128
## adjCV    0.9190    0.9191    0.9173    0.9167    0.9170    0.9125
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X              6.5808    11.70    16.49    20.77    24.88    28.83
## log.purchase   0.5193    12.24    12.72    12.73    12.88    12.94
##               7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
## X               32.63    36.15    39.64     42.90     46.09     49.19
## log.purchase    13.55    13.58    13.61     13.72     13.72     13.74
##               13 comps  14 comps  15 comps  16 comps  17 comps  18 comps
## X                52.25     55.30     58.29     61.27     64.24     67.18
## log.purchase     13.74     13.74     13.81     13.94     13.96     14.04
##               19 comps  20 comps  21 comps  22 comps  23 comps  24 comps
## X                70.12     73.04     75.94     78.83     81.69     84.54
## log.purchase     14.05     14.11     14.15     14.15     14.24     14.24
##               25 comps  26 comps  27 comps  28 comps  29 comps  30 comps
## X                87.37     90.17     92.75     94.76     96.52     97.86
## log.purchase     14.26     14.36     14.39     14.75     14.94     14.97
##               31 comps
## X                98.75
## log.purchase     15.83
```

```
predict.pcr=predict(pcr.fit,x.test,ncomp=itemp)
pcr.coef=c(0,pcr.fit$coefficients[,,itemp])
mse.pcr=mean((predict.pcr - y.test)^2);mse.pcr
```

```
## [1] 1.0569
```

```
se.pcr=sd((predict.pcr - y.test)^2)/sqrt(nrow(test));se.pcr
```

```
## [1] 0.028322
```

model selection

```
coef.matrix=cbind(glm.coef,lasso.coef1,lasso.coef2,pcr.coef)
colnames(coef.matrix)=c("least squares","lasso(lambda.min)","lasso(lambda.1se","pcr")
MSE=c(glm.mse,mse.lasso1,mse.lasso2,mse.pcr)
coef.matrix=rbind(coef.matrix,MSE)
```

```
coef.matrix
```

```
##                                least squares lasso(lambda.min)
## (Intercept)                     13.073290178      13.196134817
## genderM                          0.302952285       0.264737653
## marital_statusmarried            0.017757576       0.000000000
## age18-25                         0.133769297       0.000000000
## age26-35                         0.268928171       0.102792065
## age36-45                         0.226610232       0.066050267
## age46-50                         0.179399005       0.045857790
## age51-55                         0.135055958      -0.002538047
## age55+                          -0.063999070      -0.233010178
## stay_years1                     -0.003186436       0.000000000
## stay_years2                     -0.006577716       0.000000000
## stay_years3                      0.028131647       0.054600504
## stay_years4+                     0.004569378      -0.011356452
## occupation1                     -0.042762693       0.000000000
## occupation2                     -0.023233049       0.000000000
## occupation3                      0.134325673       0.074498977
## occupation4                     -0.004465121       0.000000000
## occupation5                      0.060978222       0.000000000
## occupation6                     -0.126966676      -0.045077990
## occupation7                     -0.106710155       0.000000000
## occupation8                     -0.153569149       0.000000000
## occupation9                     -0.137543863      -0.061428315
## occupation10                    -0.008905319      -0.092708546
## occupation11                    -0.049995789       0.000000000
## occupation12                    -0.072633131       0.000000000
## occupation13                    -0.159334446      -0.113602520
## occupation14                    -0.023005583       0.000000000
## occupation15                    -0.037251762       0.000000000
## occupation16                     0.115119975       0.094646610
## occupation17                    -0.091596605      -0.037509429
## occupation18                     0.040274509       0.021330628
## occupation19                     0.224926637       0.084161434
## occupation20                     0.092211365       0.147152578
## city_categoryB                   0.113132704       0.105131660
## city_categoryC                  -0.588182989      -0.570674795
## genderM:marital_statusmarried   -0.030702040       0.003950854
## MSE                              0.847596476       0.878638778
##                                lasso(lambda.1se           pcr
## (Intercept)                        13.35220415  0.0000000000
## genderM                             0.11501533  0.1154632365
## marital_statusmarried               0.00000000 -0.0221475331
## age18-25                            0.00000000 -0.0224802600
## age26-35                            0.00000000  0.0438928488
## age36-45                            0.00000000  0.0275608749
## age46-50                            0.00000000  0.0170734434
## age51-55                            0.00000000 -0.0171868457
## age55+                             -0.07564805 -0.0717372841
## stay_years1                         0.00000000 -0.0025095060
## stay_years2                         0.00000000 -0.0118456116
## stay_years3                         0.00000000  0.0235062883
## stay_years4+                        0.00000000  0.0125017695
```

```
## occupation1                          0.00000000  0.0076323929
## occupation2                          0.00000000 -0.0189182911
## occupation3                          0.00000000 -0.0129962854
## occupation4                          0.00000000 -0.0039438196
## occupation5                          0.00000000 -0.0180541962
## occupation6                          0.00000000 -0.0300741255
## occupation7                          0.00000000  0.0009746685
## occupation8                          0.00000000 -0.0045614931
## occupation9                          0.00000000 -0.0249681860
## occupation10                         0.00000000  0.0018772019
## occupation11                         0.00000000  0.0015360013
## occupation12                         0.00000000  0.0275883498
## occupation13                         0.00000000 -0.0239265292
## occupation14                         0.00000000  0.0124108194
## occupation15                         0.00000000  0.0204068592
## occupation16                         0.00000000  0.0399248761
## occupation17                         0.00000000  0.0541722666
## occupation18                         0.00000000 -0.2881711552
## occupation19                         0.00000000 -0.0018484860
## occupation20                         0.00000000 -0.0101959965
## city_categoryB                       0.01768155  0.0237128366
## city_categoryC                      -0.51534554 -0.0186372025
## genderM:marital_statusmarried        0.00000000  0.0365467352
## MSE                                  0.88868661  1.0568995239
```

Clustering analysis on products

```
#filter popular products
trans.wide=data %>%
  select(-product_category_1,-product_category_2,-product_category_3)%>%
  filter(product_id %in% prod.names) %>%
  spread(key=product_id,value=purchase,fill=0)


#Hierarchical Clustering
x=trans.wide[8:27]
sd.x=scale(x)
distance <- as.dist(1-cor(sd.x)) #convcerts to correlation-based distance matrix

#average linkage
hc.average =hclust(distance, method ="average")
hc.clusters=cutree(hc.average,4)
table(hc.clusters)
```

```
## hc.clusters
##  1  2  3  4
##  5  2 12  1
```

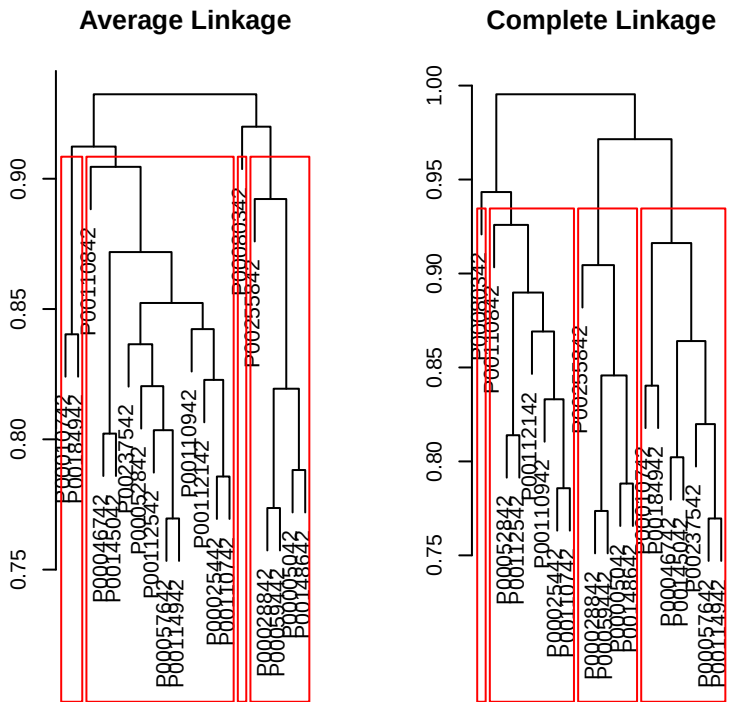
```
par(mfrow =c(1,3))
plot(hc.average, main="Average Linkage", xlab="", sub="",ylab="")
rect.hclust(hc.average,k=4)

#complete linkage
hc.complete =hclust(distance, method ="complete")
```

```r
hc.clusters=cutree(hc.complete,4)
table(hc.clusters)
```

```
## hc.clusters
## 1 2 3 4
## 5 7 7 1
```

```r
plot(hc.complete, main="Complete Linkage", xlab="", sub="",ylab="")
rect.hclust(hc.complete,k=4)

sub.id=trans.wide%>%
  filter(P00080342!=0)%>%
  select(user_id) %>%
  unlist()

consumer=data.wide%>%
  filter(user_id %in% sub.id)
summary(consumer)
```

```
##     user_id        gender     age         occupation  city_category
##  Min.   :1000010   F:264   0-17 : 26   7      :146   A:247
##  1st Qu.:1001598   M:922   18-25:184   4      :144   B:423
##  Median :1003163           26-35:437   0      :129   C:516
##  Mean   :1003100           36-45:251   1      :108
##  3rd Qu.:1004614           46-50:104   17     : 85
##  Max.   :1006039           51-55:105   12     : 67
##                            55+  : 79   (Other):507
##  stay_years   marital_status   purc.total       log.purchase
##  0 :146     unmarried:679    Min.   :  54413   Min.   :10.90
##  1 :426     married  :507    1st Qu.: 441312   1st Qu.:13.00
##  2 :219                      Median : 901219   Median :13.71
##  3 :191                      Mean   :1305911   Mean   :13.68
##  4+:204                      3rd Qu.:1814949   3rd Qu.:14.41
##                              Max.   :6817493   Max.   :15.74
##
```



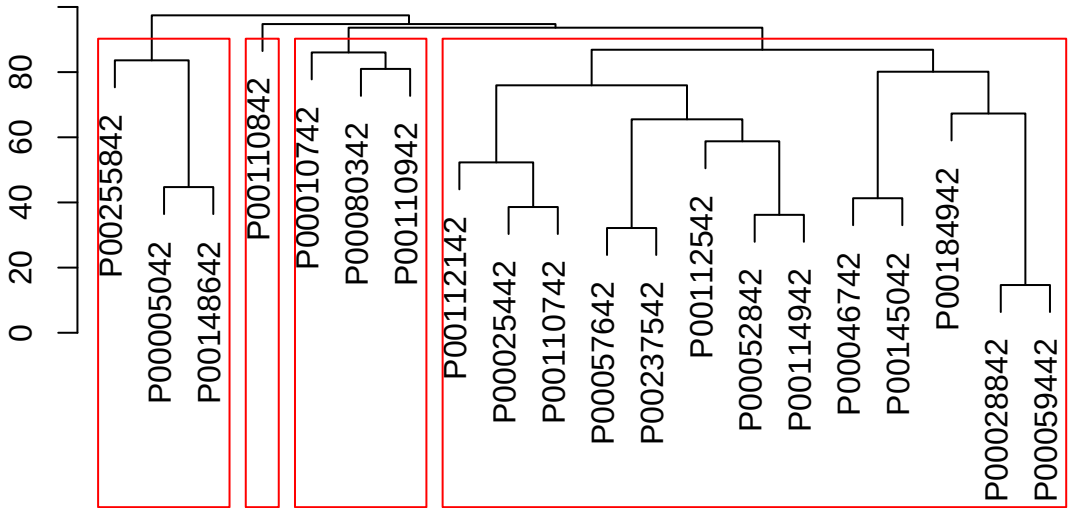
Principal Components Analysis

PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance

```
#Principal Components Analysis
pr.out=prcomp(t(sd.x))
hc.out.pc=hclust(dist(pr.out$x[ ,1:10]),
                 method="complete")
hc.clusters=cutree(hc.out.pc,4)
table(hc.clusters)
```

```
## hc.clusters
##  1  2  3  4
##  3  3 13  1
```

```
plot(hc.out.pc,
     main="Principle Components",
     xlab="", sub="",ylab="")
rect.hclust(hc.out.pc,k=4)
```


Principle Components
0
20
40
60
80
P00255842
P00005042
P00148642
P00110842
P00010742
P00080342
P00110942
P00112142
P00025442
P00110742
P00057642
P00237542
P00112542
P00052842
P00114942
P00046742
P00145042
P00184942
P00028842
P00059442