Data preprocessing

1) Import data and make the conversion from letters to numbers, the codes shown are as follows:

> agaricus.lepiota <- read.csv("E:/GWU/MSCourse/Big Data/Project2/agaricus-lepiota.data", header=FALSE)

>agaricus.data.matrix=matrix(0,nrow(agaricus.lepiota),ncol(agaricus.lepiota))

> for(i in 1:nrow(agaricus.lepiota)){

for(j in 1:ncol(agaricus.lepiota)){

agaricus.data.matrix[i,j] = as.numeric(agaricus.lepiota[i,j]);

}

}

2) Add the column names to data. the codes are shown as follows:

> row=c("type","cap-shape","cap-surface","cap-color","bruises?","odor","gill-attachment","gill-spacing","gill-size","gill-color","stalk-shape","stalk-root","stalk-surface-above-ring","stalk-surface-below-ring","stalk-color-above-ring","stalk-color-below-ring","veil-type","veil-color","ring-number","ring-type","spore-print-color","population","habitat")

> colnames(agaricus.data.matrix)<-row

3) Because there is only one level in the 17th column, delete all the data in 17th column. The codes are shown as follows:

> agaricus.data.matrix=agaricus.data.matrix[,-17]

4) Make the data into two types- label data and feature data. The codes are shown as follows:

> agaricus.label = matrix(agaricus.data.matrix[,1])

> agaricus.feature = agaricus.data.matrix[,2:22]

2.2-D PCA algorithm

1) Make data reduction to eliminate some attributes through Principal Components Analysis(PCA). The codes are shown as follows:

> install.packages("psych")

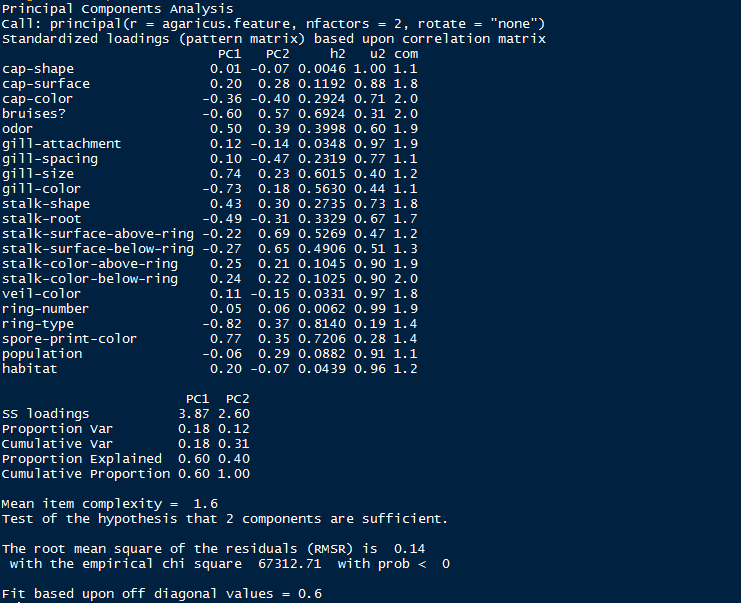
> library(psych)

2) Firstly, do the 2-d PCA. The codes are shown as follows:

> agaricus.2dpca = principal(r = agaricus.feature, nfactors=2, rotate = "none")

> agaricus.2dpca

Result:



3) Get the scores

> agaricus.2dpca.score = agaricus.2dpca$scores

4) Plot the picture of PCA. The codes are shown as follows:

>agaricus.2dpca.score.df=data.frame(agaricus.2dpca.score)

> a=data.frame(agaricus.2dpca.score)

> a$label=agaricus.label

>ggplot(a,aes(x=PC1,y=PC2,colour=factor(label)))+geom\_point()



Figure 1

1. The K-means cluster algorithm

1) The codes for those algorithm

> kmeans2d=character(8124)

> kmeans2d=data.frame(kmeans2d)

> kmeans2d.label=kmeans(agaricus.label,2)

> kmeans2d.label.ma=matrix(kmeans2d.label$cluster)

> kmeans2d$label=kmeans2d.label.ma

> kmeans2d.feature=kmeans(agaricus.2dpca.score,2)

> kmeans2d.feature.ma=matrix(kmeans2d.feature$cluster)

> kmeans2d$feature=kmeans2d.feature.ma

> print(kmeans2d.feature$size)

> kmeans2d$pc1 = agaricus.2dpca.score.df$PC1

> kmeans2d$pc2 = agaricus.2dpca.score.df$PC2

> ggplot(kmeans2d,aes(x=pc1,y=pc2,colour=factor(feature)))+geom\_point()

2) Plot the pictures when K=2,3,5,7

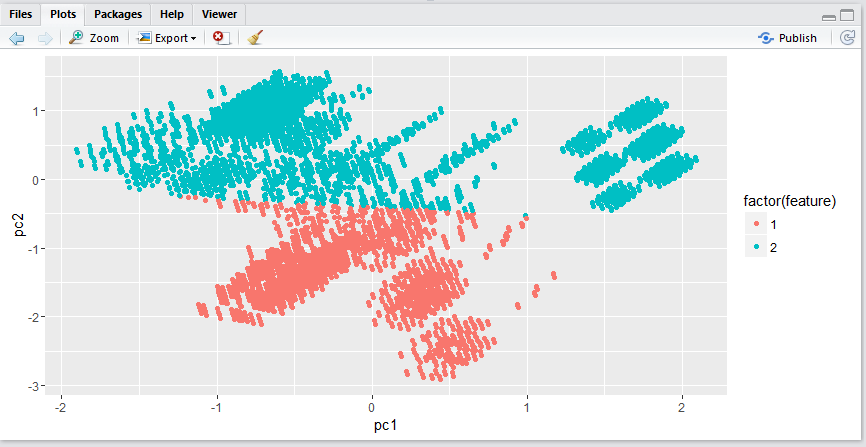


Figure 2. K=2

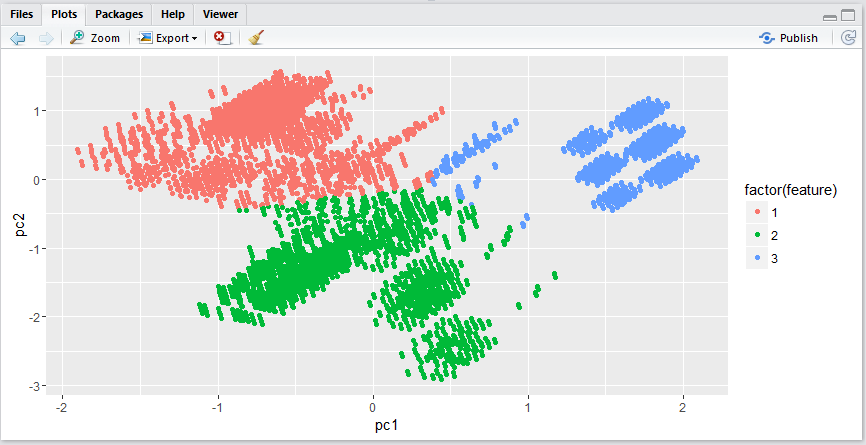


Figure 3. K=3

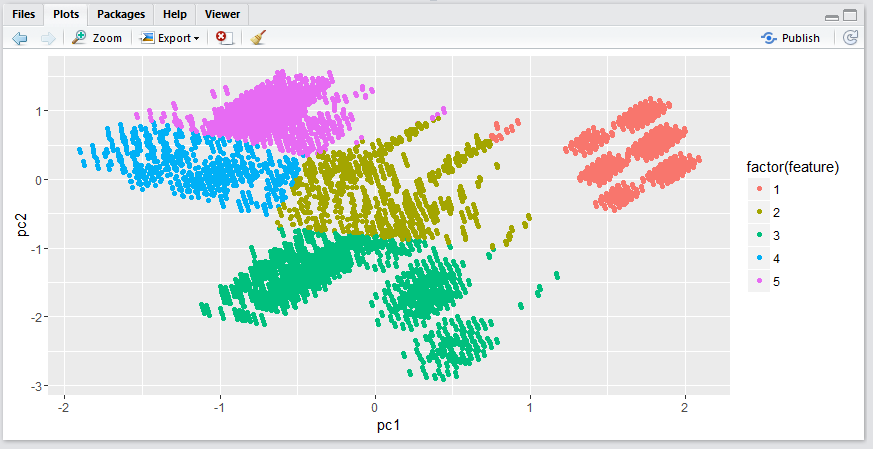


Figure 4. K=5

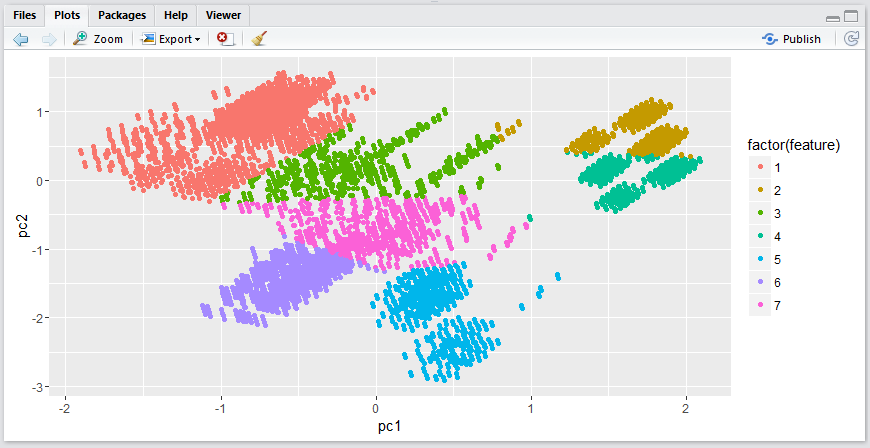


Figure 5. K=7

3) The table of results:

|  |  |
| --- | --- |
| K=2 | 1862, 6262 |
| K=3 | 3714,1895,2515 |
| K=5 | 1753,1072,2035,897,2367 |
| K=7 | 3082,998,773,749,574,1177,771 |

1. The Knn cluster algorithm

1) The codes for Knn algorithm:

> knn.train=agaricus.2dpca.score[1:4062,]

> knn.test=agaricus.2dpca.score[4063:8124,]

> knn.train.label=agaricus.label[1:4062]

> knn.test.label=agaricus.label[4063:8124]

> knn2d=character(8124)

> knn2d=data.frame(knn2d)

>knnlabeltest=knn(knn.train,knn.test,knn.train.label,2)

> knn2d=data.frame(knnlabeltest)

> d0=data.frame(knn.test)

> d1=data.frame(knn.train)

> knn2d$PC1=d0$PC1

> knn2d$PC2=d0$PC2

> knn2d1=data.frame(knn.train.label)

> knn2d1$PC1=d1$PC1

> knn2d1$PC2=d1$PC2

> num=0

> for(i in 1:length(knn2d$knnlabeltest)){

+ if(knn2d$knnlabeltest==knn.test.label[i])

+ num=num+1;

+ }

> accuracy=num/4062

> accuracy

2) when K=2,3,5,7 and the training-test are 50%-50%, 60%-40%, 70%-30%, the table of accuracy is shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 50%-50% | 60%-40% | 70%-30% |
| K=2 | 0.683 | 0.896 | 0.913 |
| K=3 | 0.478 | 0.896 | 0.918 |
| K=5 | 0.483 | 0.898 | 0.916 |
| K=7 | 0.481 | 0.903 | 0.920 |

1. LM and GLM methods for 2-d PCA

Use those methods to get liner fits for the data.

1. LM-liner method

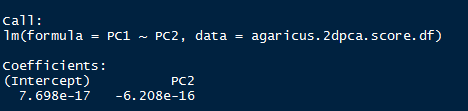
Use the liner method to fit PC1 and PC2. The codes are shown as follows:

> lmdata=agaricus.2dpca.score.df

> lmdata$label=agaricus.label-1

> lmfit=lm(label~PC1+PC2,data=lmdata)

> lmfit



Plot the picture of the LM

> plot(lmfit$residuals,lmfit$fitted)

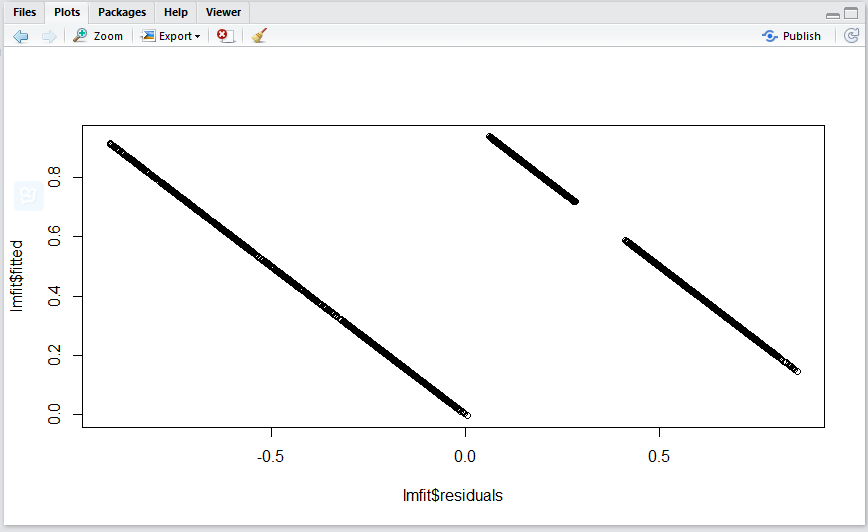


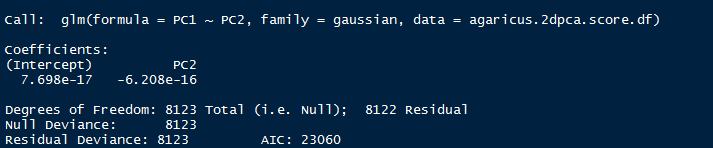
Figure 6

1. Generalized Linear Model

The codes for Gaussian method:

> glmfit.gaussian=glm(label~PC1+PC2,family=gaussian,data=lmdata)

> glmfit.gaussian



> plot(glmfit.gaussian$residuals,glmfit.gaussian$fitted)

Plot the picture of Gaussian method:

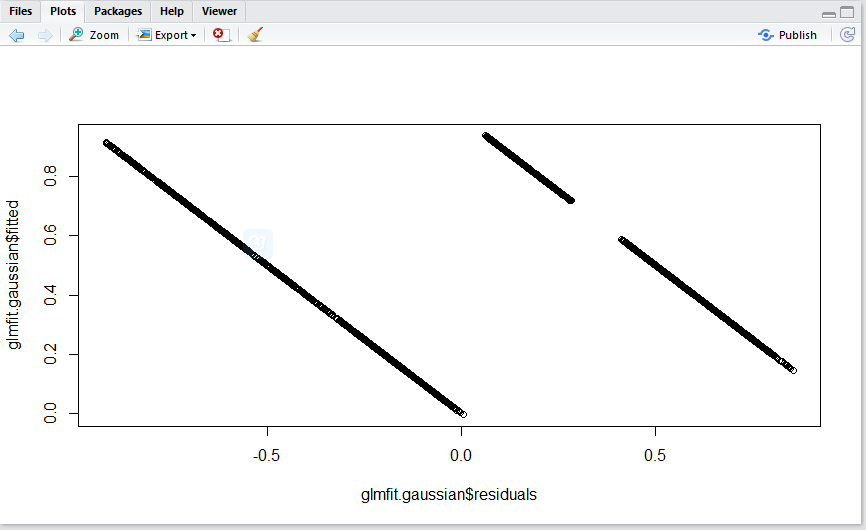
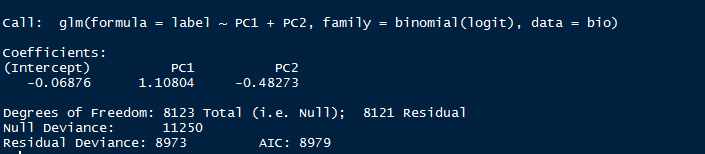


Figure 7

The codes for binomial method:

> glmfit.binomial=glm(label~PC1+PC2,family=binomial(logit),data=lmdata)

> glmfit.binomial



> plot(glmfit.binomial$residuals,glmfit.binomial$fitted)

Plot the picture of binomial method

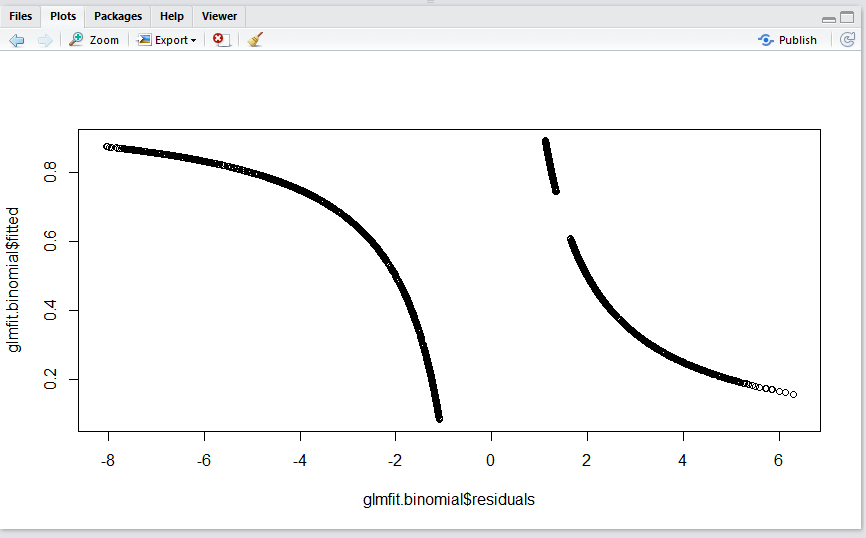


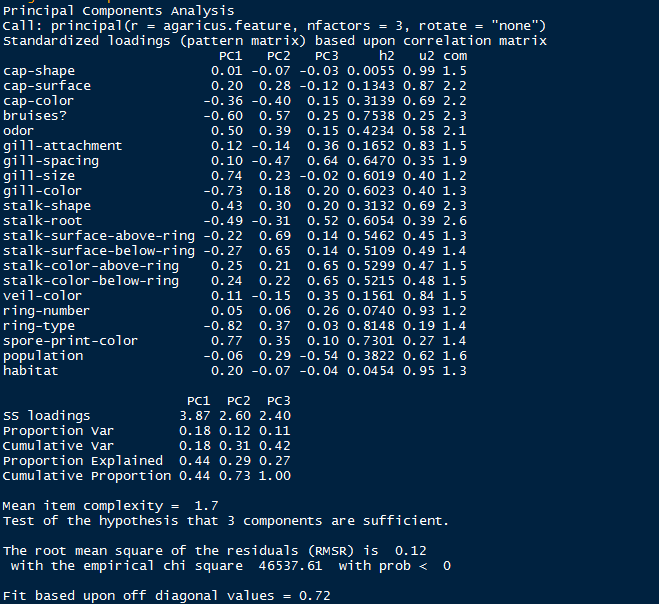
Figure 8

1. Because not all the methods work on all attributes, so just can use those three method to fits PCA 1 and PCA 2. As the figures show, the Binomial method has better fitting result.
2. 3-D PCA algorithm
3. Firstly, do the 2-d PCA. The codes are shown as follows:

> agaricus.3dpca = principal(r = agaricus.feature, nfactors=3, rotate = "none")

> agaricus.3dpca

Result：



2) Get the scores

> agaricus.3dpca.score = agaricus.3dpca$scores

3) Plot the picture for the 3D PCA. The codes are shown as follows:

>agaricus.3dpca.score.df=data.frame(agaricus.3dpca.score)

> install.packages("rgl")

> library(rgl)

> a=data.frame(agaricus.3dpca.score)

> a$label=agaricus.label

>plot3d(a$PC1,a$PC2,a$PC3,xlab="PC1",ylab="PC2",zlab="PC3",col=as.integer(a$label))

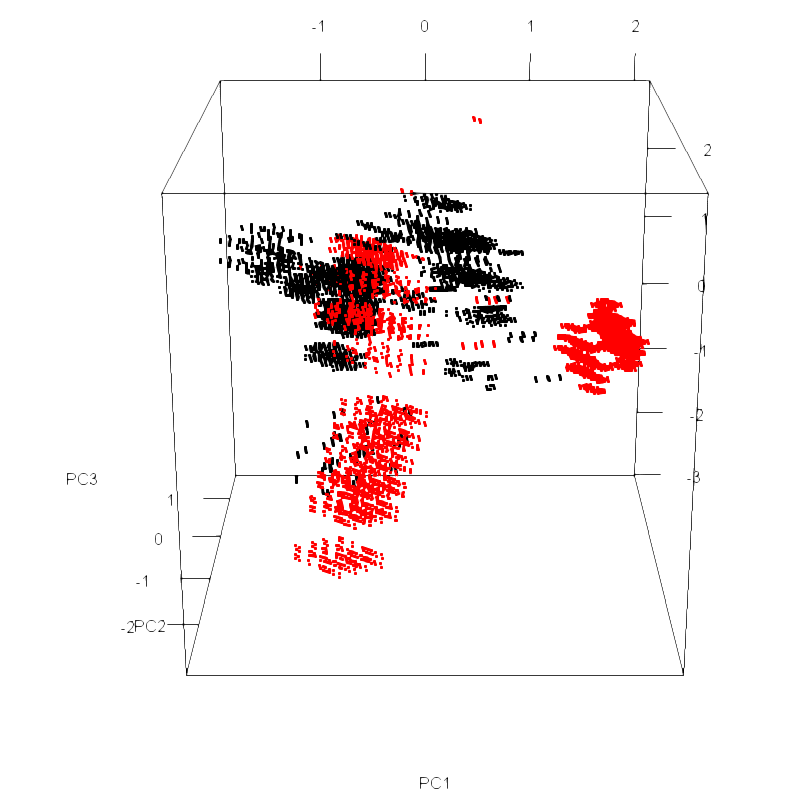


Figure 9

1. The K-means cluster algorithm for 3D PCA

1)The codes for this algorithm:

> kmeans3d=character(8124)

> kmeans3d=data.frame(kmeans3d)

> kmeans3d.label=kmeans(agaricus.label,2)

> kmeans3d.label.ma=matrix(kmeans3d.label$cluster)

> kmeans3d$label=kmeans3d.label.ma

> kmeans3d.feature=kmeans(agaricus.3dpca.score,2)

> kmeans3d.feature.ma=matrix(kmeans3d.feature$cluster)

> kmeans3d$feature=kmeans3d.feature.ma

> print(kmeans3d.feature$size)

> kmeans3d$pc1 = agaricus.3dpca.score.df$PC1

> kmeans3d$pc2 = agaricus.3dpca.score.df$PC2

> kmeans3d$pc3 = agaricus.3dpca.score.df$PC3

> plot3d(kmeans3d$pc1,kmeans3d$pc2,kmeans3d$pc3,xlab="pc1",ylab="pc2",zlab="pc3",col=as.integer(kmeans3d$feature))

2) Plot the pictures when K=2,3,5,7

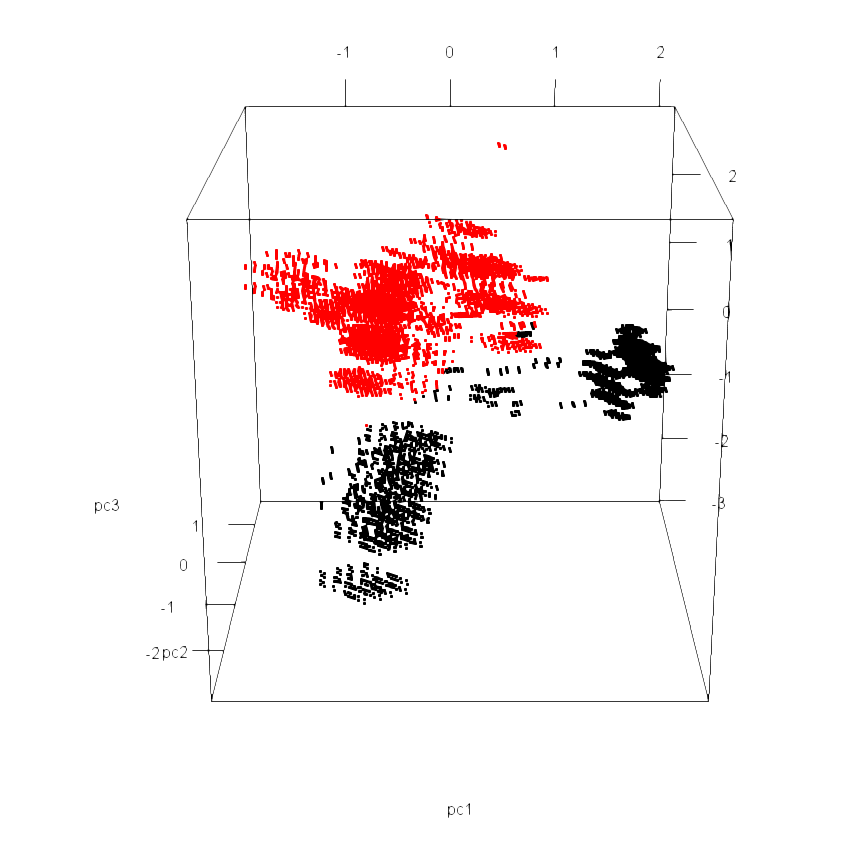


Figure 10. K=2

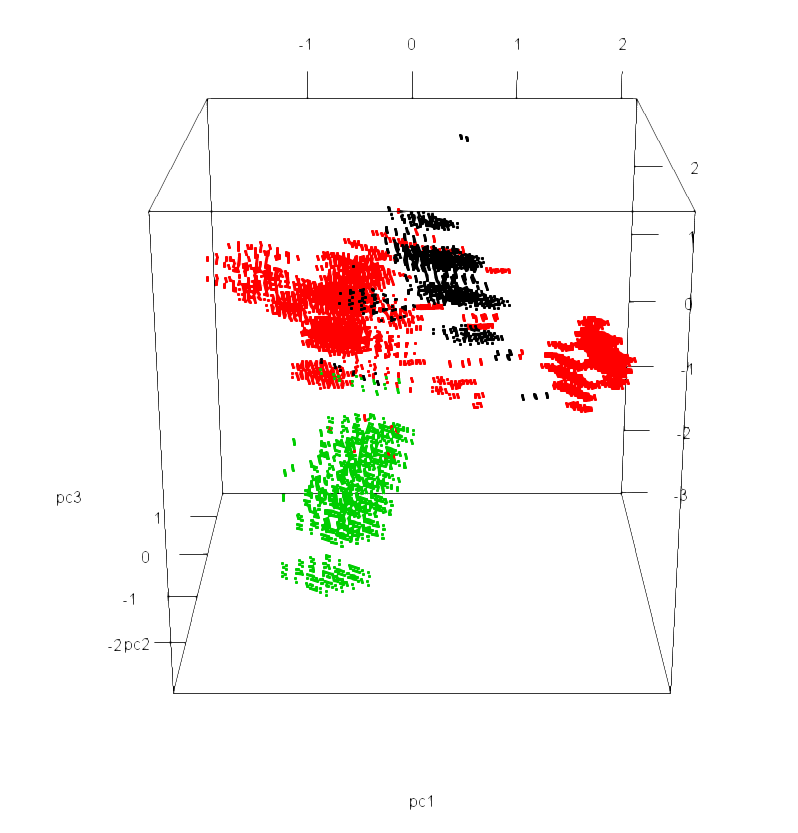


Figure 11. K=3

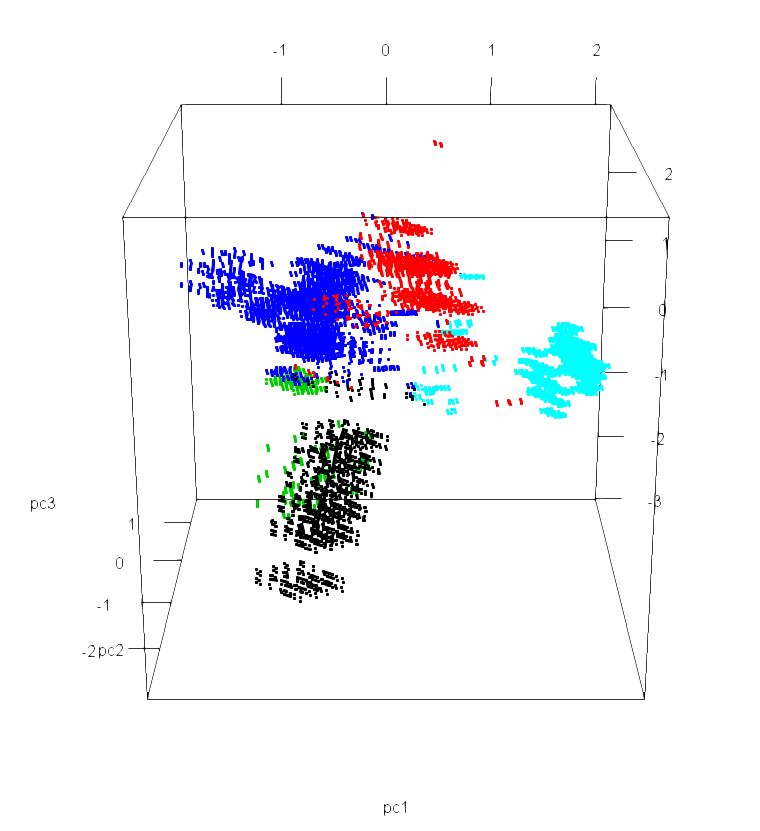


Figure 12. K=5

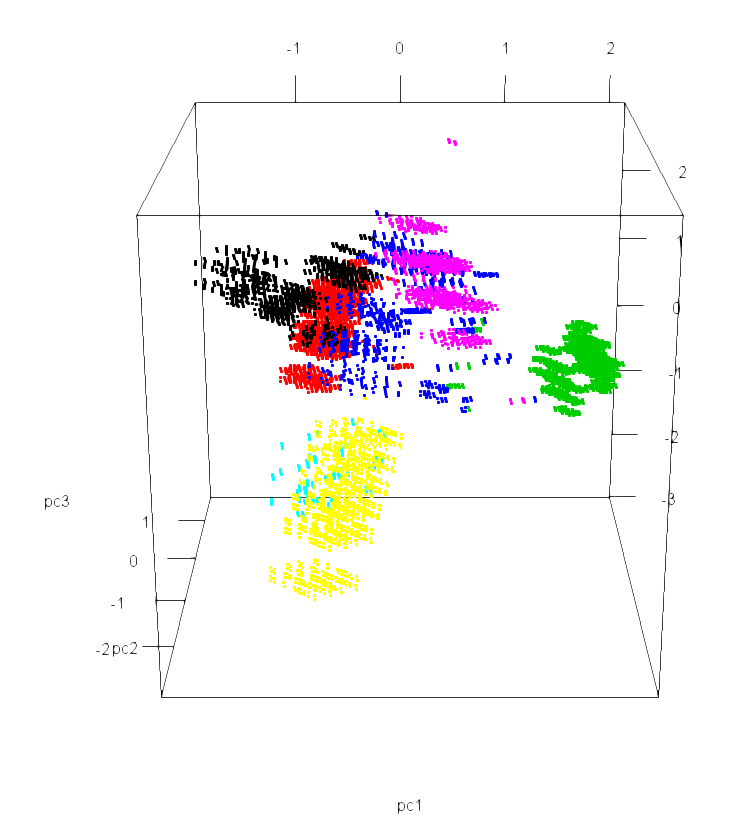


Figure 13. K=7

3) The table of results:

|  |  |
| --- | --- |
| K=2 | 3431,4693 |
| K=3 | 1146,5465,1513 |
| K=5 | 1362,1142,359,3357,1904 |
| K=7 | 1103,1945,1769,982,210,800,1315 |

1. Knn cluster algorithm for 3D PCA

1) The codes for knn algorithm:

> knn.train=agaricus.3dpca.score[1:4062,]

> knn.test=agaricus.3dpca.score[4063:8124,]

> knn.train.label=agaricus.label[1:4062]

> knn.test.label=agaricus.label[4063:8124]

> knn3d=character(8124)

> knn3d=data.frame(knn2d)

> knnlabeltest=knn(knn.train,knn.test,knn.train.label,2)

> knn3d=data.frame(knnlabeltest)

> d0=data.frame(knn.test)

> d1=data.frame(knn.train)

> knn3d$PC1=d0$PC1

> knn3d$PC2=d0$PC2

> knn3d$PC3=d0$PC3

> knn3d1=data.frame(knn.train.label)

> knn3d1$PC1=d1$PC1

> knn3d1$PC2=d1$PC2

> knn3d1$PC3=d1$PC3

> num=0

> for(i in 1:length(knn3d$knnlabeltest)){

+ if(knn3d$knnlabeltest[i]==knn.test.label[i])

+ num=num+1;

+ }

> accuracy=num/4062

> accuracy

2) when K=2,3,5,7 and the training-test are 50%-50%, 60%-40%, 70%-30%, the table of accuracy for is shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 50%-50% | 60%-40% | 70%-30% |
| K=2 | 0.720 | 0.954 | 0.977 |
| K=3 | 0.872 | 0.951 | 0.976 |
| K=5 | 0.798 | 0.946 | 0.974 |
| K=7 | 0.574 | 0.945 | 0.9737 |

1. LM and GLM methods for 3D PCA
2. LM-liner model

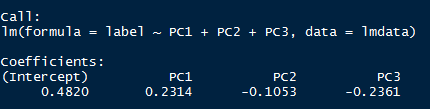
Use the liner method to fit PC1, PC2 and PC3. The codes are shown as follows:

> lmdata=agaricus.3dpca.score.df

> lmdata$label=agaricus.label-1

> lmfit=lm(label~PC1+PC2+PC3,data=lmdata)

> lmfit



> plot(lmfit$residuals,lmfit$fitted)

Plot the picture of the LM

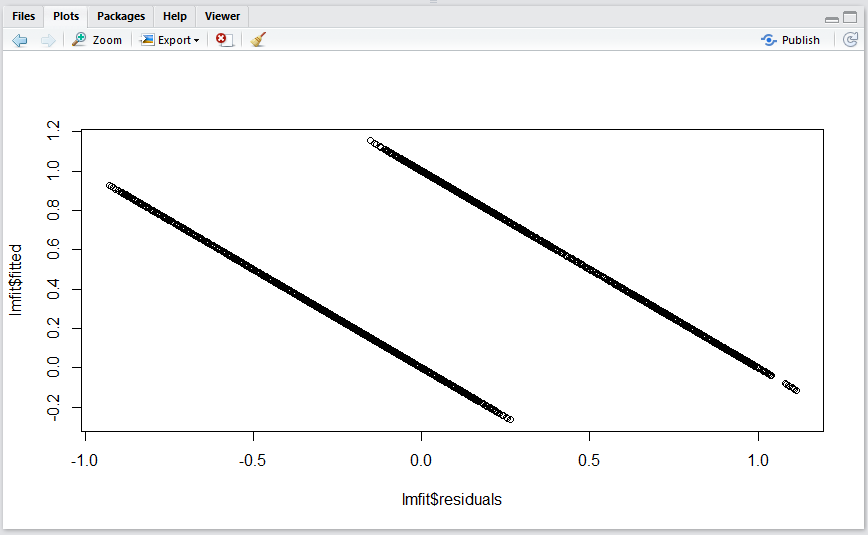


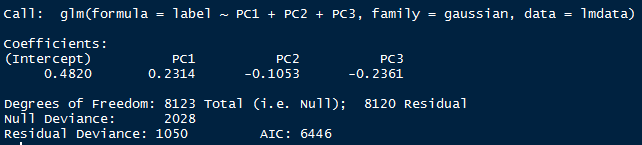
Figure 14

1. Generalized Linear Model

The codes for Gaussian method:

> glmfit.gaussian=glm(label~PC1+PC2+PC3,family=gaussian,data=lmdata)

> glmfit.gaussian



> plot(glmfit.gaussian$residuals,glmfit.gaussian$fitted)

Plot the picture for the Gaussian method:

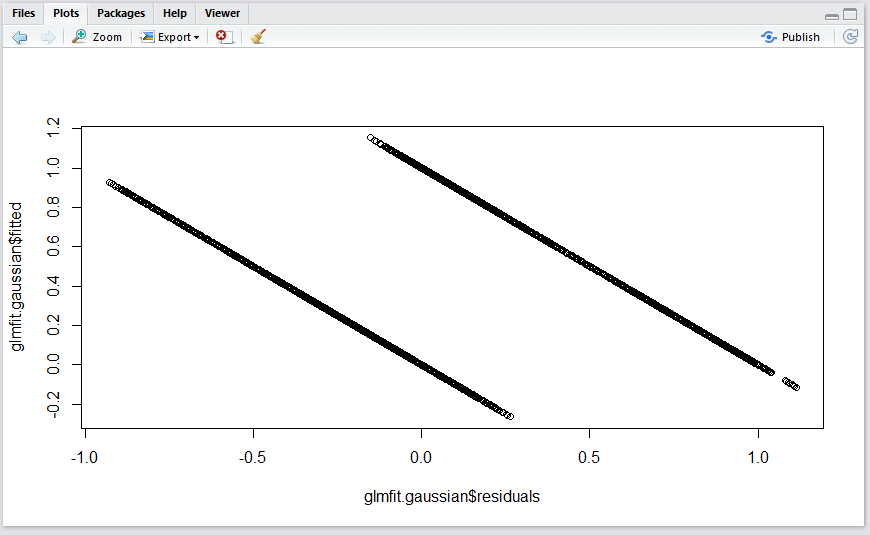
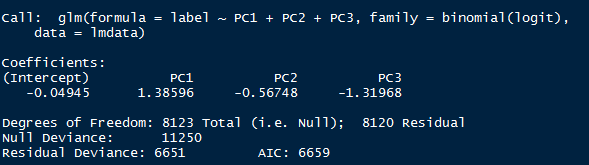


Figure 15

The codes for binomial method:

> glmfit.binomial=glm(label~PC1+PC2+PC3,family=binomial(logit),data=lmdata)

> glmfit.binomial



> plot(glmfit.binomial$residuals,glmfit.binomial$fitted)

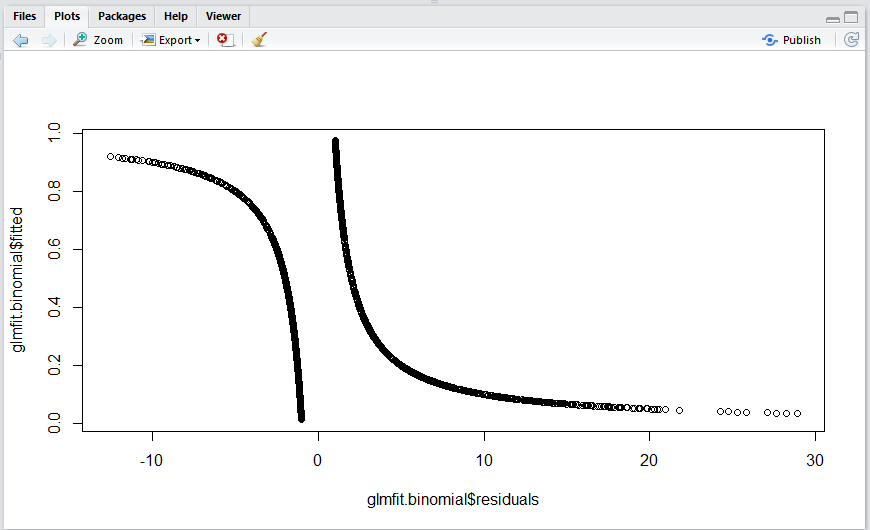


Figure 16

1. As we can see in the result, the AIC of Gaussian method is smaller than Binomial, therefore, Gaussian gets the better fitting result.
2. Discussion of results from the experiments
3. As the table of result of knn shows, the larger the training set divided, the higher accuracy gained in general. What comes opposite to our imagination is, however, bigger value of k does not represent the higher accuracy. Guess it can reasoning to that data with different label was not separate clearly in the 2-dimension or 3-dimension space constructed by PCA.
4. In 3-D clustering, compared figure 9 with figure 10,11,12 and 13, we can say that when the number of cluster equals 5, the black part in figure 9 basically consist of the red and blue part in figure 12, and the red part in figure 9 consist of the rest part in figure12, which means this clustering model is good.
5. In 2-D clustering, compared figure 1 with figure 2,3,4 and 5, we can say that when the number of cluster equals 7, the red part in figure 1 basically consist of the red, light green and blue part in figure 5, and the green part in figure 1 consist of the rest part in figure 5, which means this clustering model is acceptable.
6. General speaking, we deem that plots from 2-d clustering help us understand the data best, since 2 dimensions’ space is easy to imaging.