```r
setwd("/Users/tong/Desktop/R")
test=read.table("spam-test.txt",sep=",")
train=read.table("spam-train.txt",sep=",")
require(ggplot2)
```

```
## Loading required package:  ggplot2
```

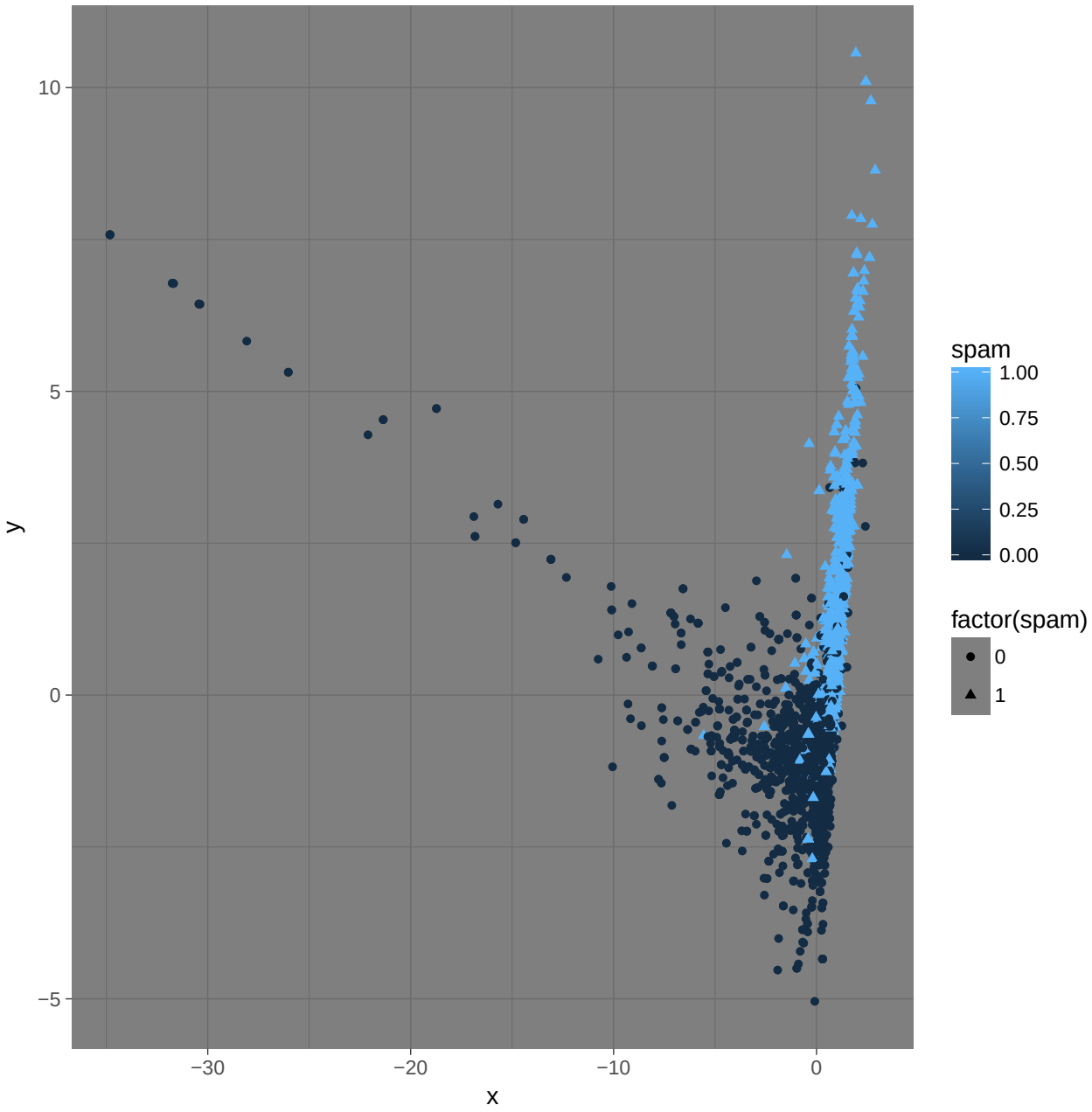
```r
require(GGally)
```

```
## Loading required package:  GGally
```

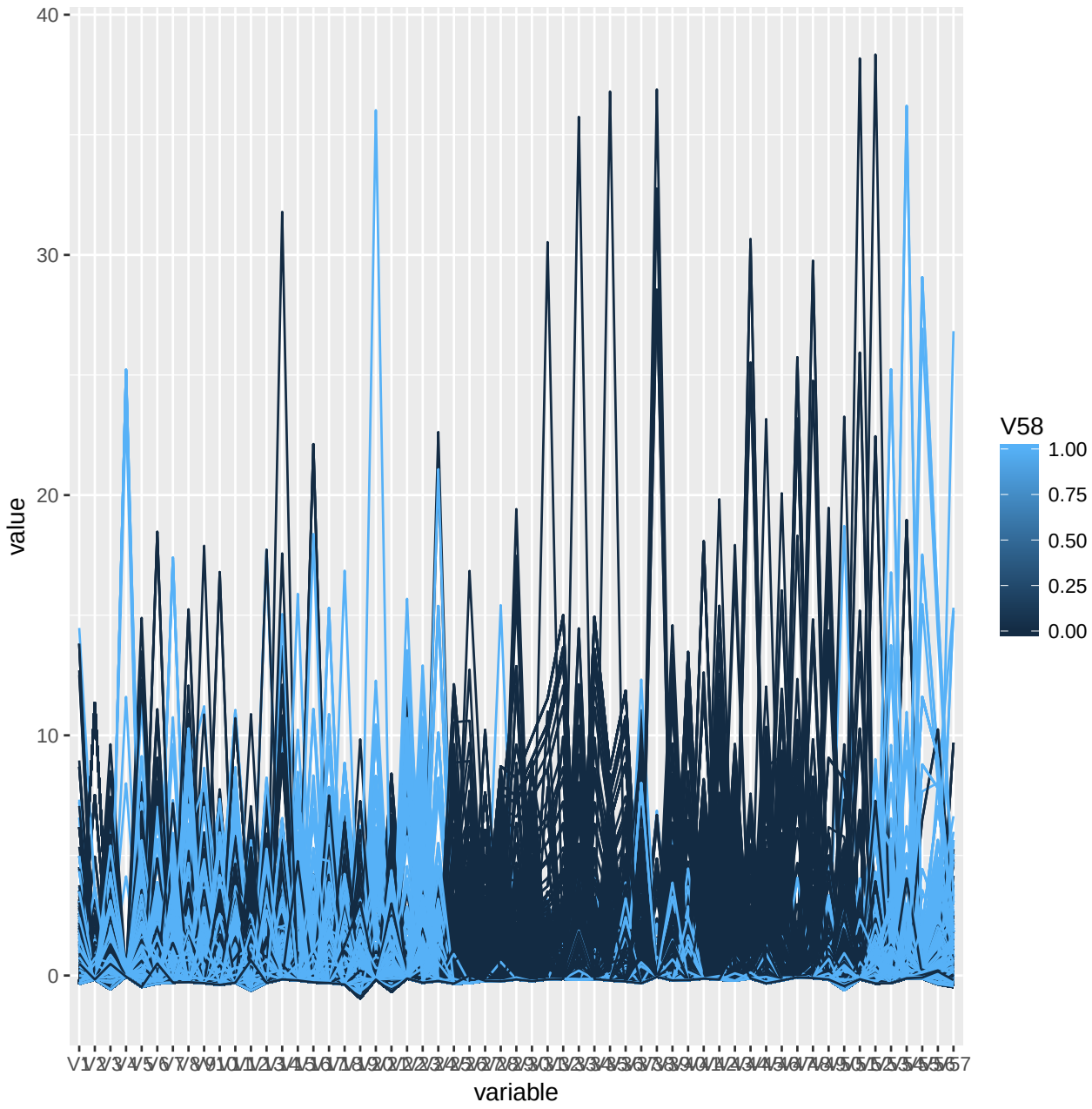
```r
require(gridExtra)
```

```
## Loading required package:  gridExtra
```

```r
#######1
test_sd=data.frame(scale(test[,1:57]),test[,58])
train_sd=data.frame(scale(train[,1:57]),train[,58])
test_log=data.frame(log(1+test[,1:57]),test[,58])
train_log=data.frame(log(1+train[,-58]),train[,58])
train_disc=data.frame(1*(train[,-58]>0),train[,58])
test_disc=data.frame(1*(test[,-58]>0),test[,58])
dat=rbind(train,test)
mds=data.frame(cmdscale(dist(scale(dat[,-58])),k=2),dat[,58])
colnames(mds)=c("x","y","spam")
ggplot(data=mds,aes(x=x,y=y,color=spam,shape=factor(spam)))+geom_point()+theme_dark()
```




```
ggparcoord(dat,columns=1:57,groupColumn=58)
```




```
#######2
#LDA and QDA
colnames(train_sd)[58]="spam"
colnames(train_log)[58]="spam"
colnames(test_sd)[58]="spam"
```

```r
colnames(test_log)[58]="spam"
colnames(train_disc)[58]="spam"
colnames(test_disc)[58]="spam"
require(MASS)
```

```
## Loading required package:  MASS
```

```r
lda_sd = lda(data=train_sd,spam~.)
lda_log = lda(data=train_log,spam~.)
qda_sd=qda(data=train_sd,spam~.)
qda_log=qda(data=train_log,spam~.)
n_train=dim(train)[1]
n_test=dim(test)[1]
#Test error of scaled dataset.
sum(predict(lda_sd,test_sd)$class!=test_sd$spam)/n_test
```

```
## [1] 0.1029987
```

```r
#Train error of scaled dataset.
sum(predict(lda_sd,train_sd)$class!=train_sd$spam)/n_train
```

```
## [1] 0.1017281
```

```r
#Test error of log-transformed dataset.
sum(predict(lda_log,test_log)$class!=test_log$spam)/n_test
```

```
## [1] 0.06518905
```

```r
#Train error of log-transformed dataset.
sum(predict(lda_log,train_log)$class!=train_log$spam)/n_train
```

```
## [1] 0.06031953
```

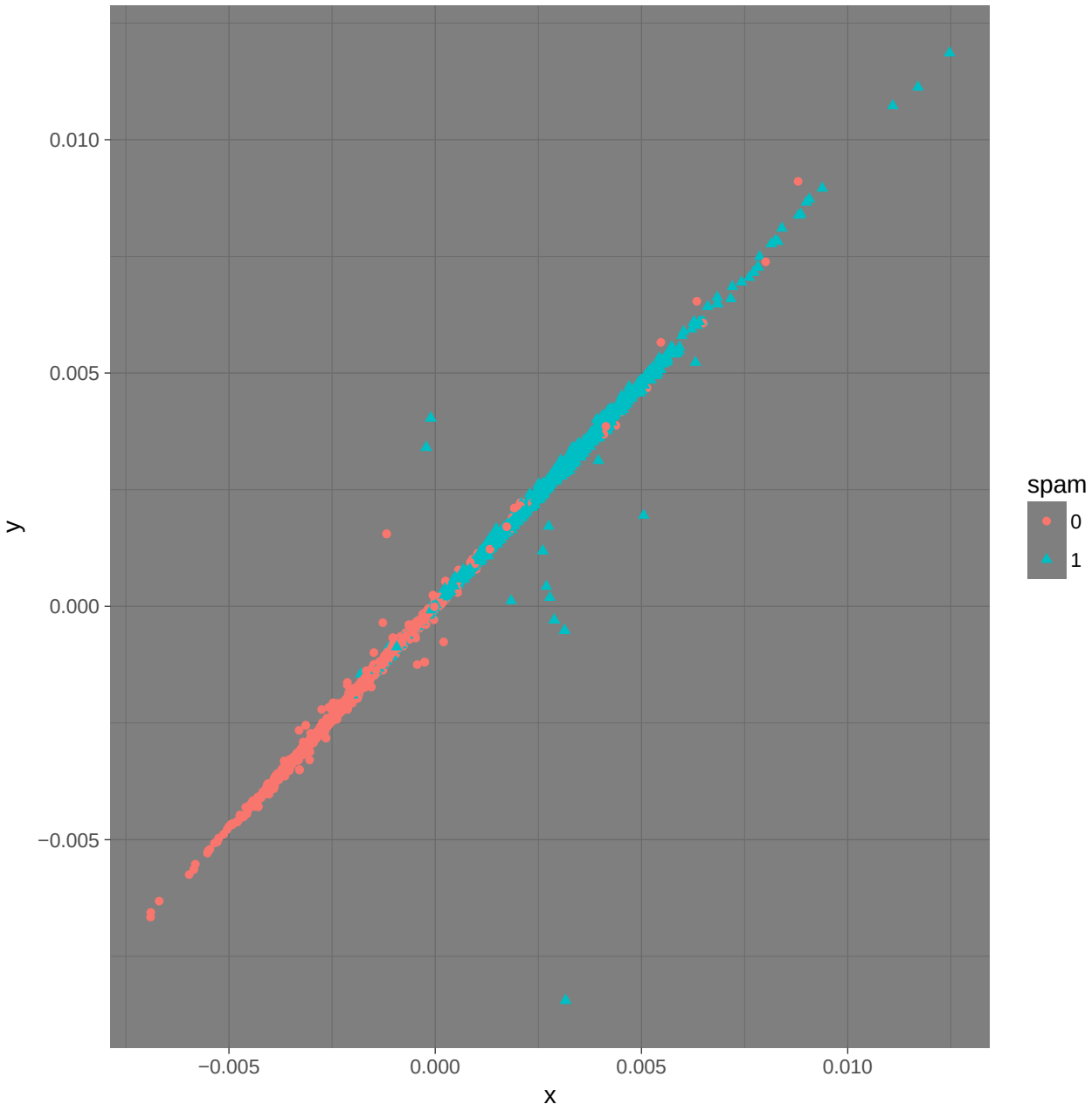
```r
#compute direction
dat_sd=rbind(train_sd,test_sd)
dat_log=rbind(train_log,test_log)
id1=which(train[,58]==1)
id0=which(train[,58]==0)
sig_sd=1/(n_train-2)*((sum(id0)-1)*cov(train_sd[id0,-58])+(sum(id1)-1)*cov(train_sd[id1,-58]))
sig_log=1/(n_train-2)*((sum(id0)-1)*cov(train_log[id0,-58])+(sum(id1)-1)*cov(train_log[id1,-58]))
dire_sd=solve(sig_sd) %*% (apply(train_sd[id1,-58],2,mean)-apply(train_sd[id0,-58],2,mean))
dire_log=solve(sig_log) %*% (apply(train_log[id1,-58],2,mean)-apply(train_log[id0,-58],2,mean))

train_sd2=data.frame(as.matrix(train_sd[,-58]) - as.matrix(train_sd[,-58]) %*% (dire_sd/as.numeric(sqrt(c
sig_sd2=1/(n_train-2)*((sum(id0)-1)*cov(train_sd2[id0,])+(sum(id1)-1)*cov(train_sd2[id1,]))
train_log2=data.frame(as.matrix(train_log[,-58]) - as.matrix(train_log[,-58]) %*% (dire_log/as.numeric(sc
sig_log2=1/(n_train-2)*((sum(id0)-1)*cov(train_log2[id0,])+(sum(id1)-1)*cov(train_log2[id1,]))
require(corpcor)
```

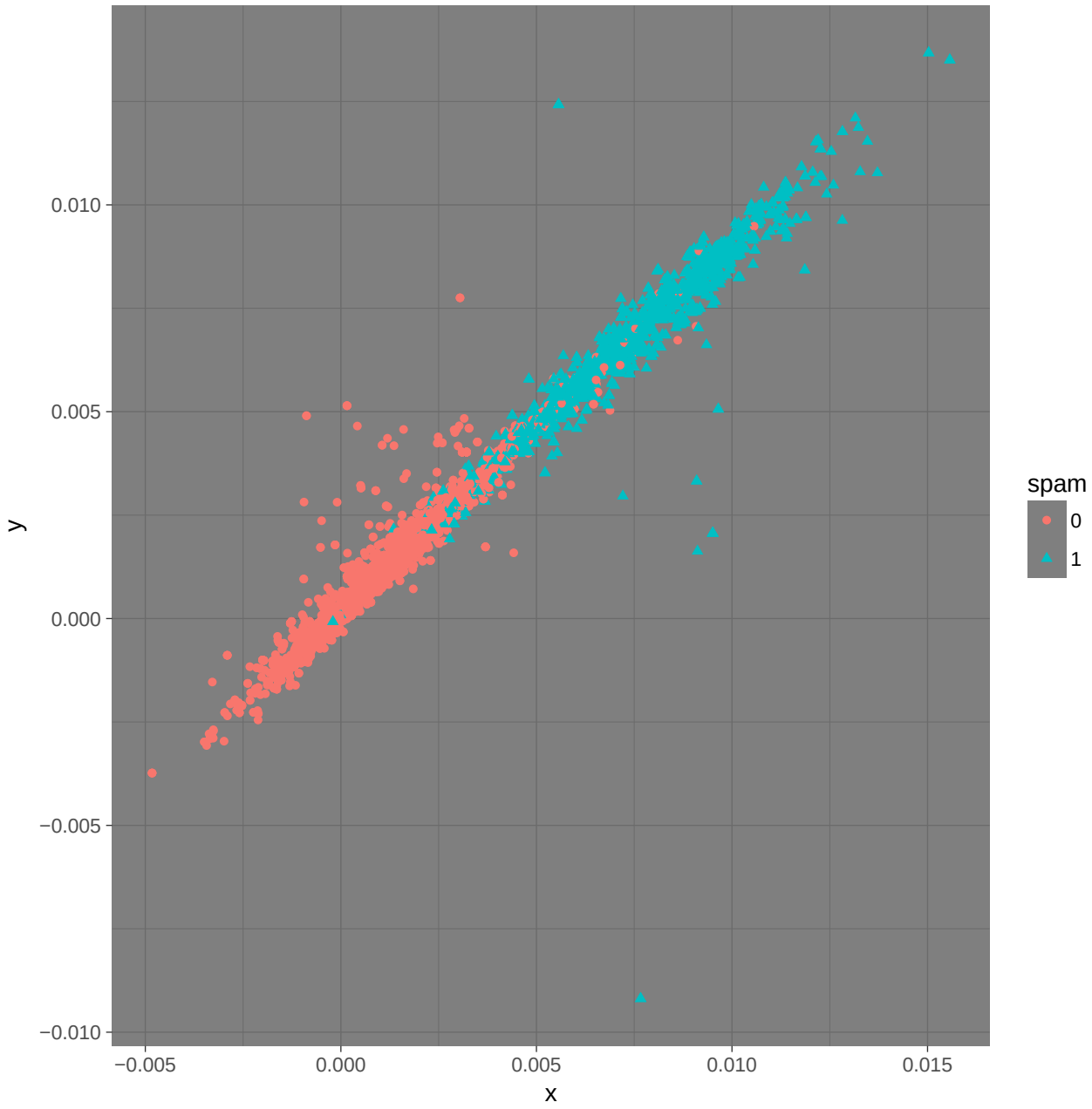
```
## Loading required package:  corpcor

dire_sd2=pseudoinverse(sig_sd2) %*% (apply(train_sd2[id1,],2,mean)-apply(train_sd2[id0,],2,mean))
dire_log2=pseudoinverse(sig_log2) %*% (apply(train_log2[id1,],2,mean)-apply(train_log2[id0,],2,mean))

dd_draw_sd=data.frame(as.matrix(dat_sd[,-58]) %*% dire_sd, as.matrix(dat_sd[,-58]) %*% dire_sd2, factor(d
colnames(dd_draw_sd)=c("x","y","spam")
ggplot(data=dd_draw_sd,aes(x=x,y=y,color=spam,shape=spam))+geom_point()+theme_dark()
```




```
dd_draw_log=data.frame(as.matrix(dat_log[,-58]) %*% dire_log, as.matrix(dat_log[,-58]) %*% dire_log2, fac
colnames(dd_draw_log)=c("x","y","spam")
ggplot(data=dd_draw_log,aes(x=x,y=y,color=spam,shape=spam))+geom_point()+theme_dark()
```




```
######Comment: The projected data clusters into a diagonal line shape, and two classes roughly
#stay at the two sides of the line.

#######3
require(nnet)
```

```
## Loading required package:  nnet

logit_sd = multinom(spam ~ ., data=train_sd)

## # weights:  59 (58 variable)
## initial  value 2125.882403
## iter  10 value 733.424941
## iter  20 value 636.722504
## iter  30 value 611.369175
## iter  40 value 604.567959
## iter  50 value 599.723654
## iter  60 value 591.644715
## iter  70 value 579.640495
## iter  80 value 578.766890
## iter  90 value 578.746002
## iter 100 value 578.718916
## final  value 578.718916
## stopped after 100 iterations

logit_log = multinom(spam ~ ., data=train_log)

## # weights:  59 (58 variable)
## initial  value 2125.882403
## iter  10 value 623.526887
## iter  20 value 493.758950
## iter  30 value 475.574885
## iter  40 value 466.822112
## iter  50 value 465.898731
## iter  60 value 465.432059
## iter  70 value 465.335251
## final  value 465.333549
## converged

logit_disc = multinom(spam ~ ., data=train_disc)

## # weights:  59 (58 variable)
## initial  value 2125.882403
## iter  10 value 673.340113
## iter  20 value 561.529882
## iter  30 value 515.767613
## iter  40 value 507.722233
## iter  50 value 507.336416
## iter  60 value 507.297897
## final  value 507.297224
## converged

#Train error of scaled dataset.
```

```r
tr1=mean(predict(logit_sd, train_sd) != train_sd$spam)
#Test error of scaled dataset.
te1=mean(predict(logit_sd, test_sd) != test_sd$spam)
#Train error of log-transformed dataset.
tr2=mean(predict(logit_log, train_log) != train_log$spam)
#Test error of log-transformed dataset.
te2=mean(predict(logit_log, test_log) != test_log$spam)
#Train error of discretized dataset.
tr3=mean(predict(logit_disc, train_disc) != train_disc$spam)
#Test error of discretized dataset.
te3=mean(predict(logit_disc, test_disc) != test_disc$spam)
# Mean train set error
mean(c(tr1,tr2,tr3))
```

```
## [1] 0.06216716
```

```r
# Mean test set error
mean(c(te1,te2,te3))
```

```
## [1] 0.06953498
```

```r
#######4
######Comment: As we know the third dataset is binary, so I would like to use Enclidean for the
#first two datasets, and binary metric for the third one. However, the knn function doesn't allow
#us for changing metric except Euclidean, and I tried KODAMA package, but it's no longer in
#service. Thus I used Enclidean for the actual situation.
require(class)
```

```
## Loading required package:  class
```

```r
require(sparsediscrim)
```

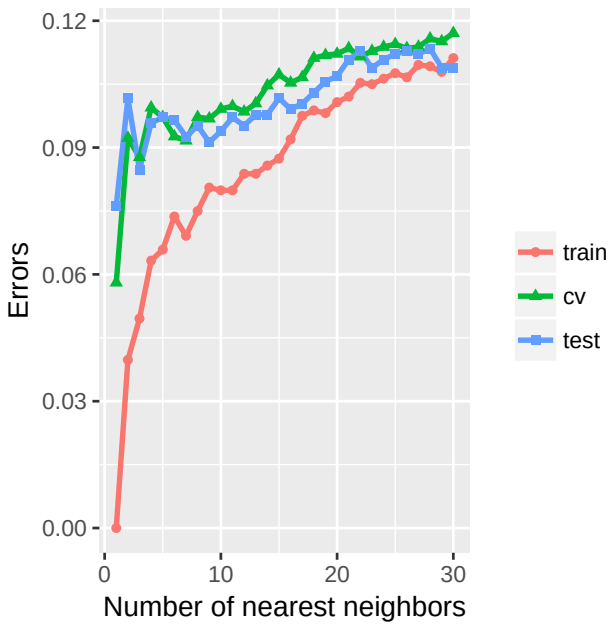
```
## Loading required package:  sparsediscrim
```

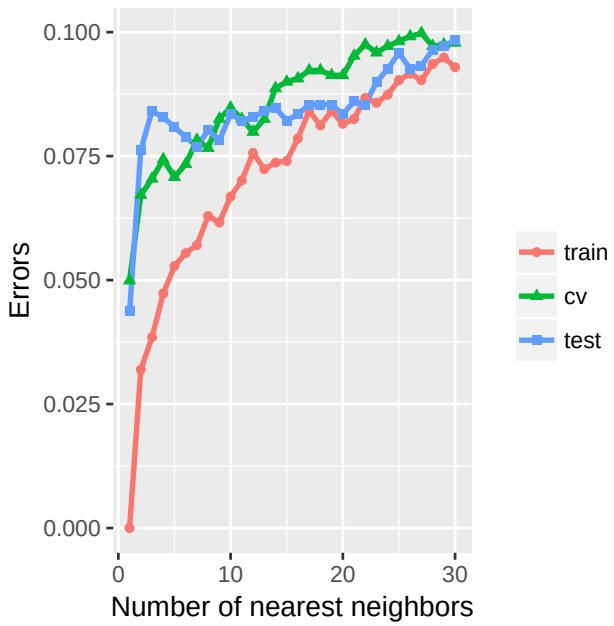
```r
knn_error = function(K,traindata,testdata) {
  set.seed(123)
  folds = cv_partition(traindata$spam, num_folds = 20)
  spam.knn = knn(train = traindata[,-ncol(traindata)], test = traindata[,-ncol(traindata)],
                 cl = traindata$spam, k = K)
  train_error = sum(spam.knn != traindata$spam) / n_train
  spam.cverr = sapply(folds, function(fold) {
    sum(traindata$spam[fold$test] != knn(train = traindata[fold$training,-ncol(traindata)], cl = traindat
  cv_error = mean(spam.cverr)
  spam.knn.test = knn(train = traindata[,-ncol(traindata)], test = testdata[,-ncol(testdata)],
                 cl = traindata$spam, k = K)
  test_error = sum(spam.knn.test != testdata$spam) / n_test
  return(c(train_error, cv_error, test_error))
}
```

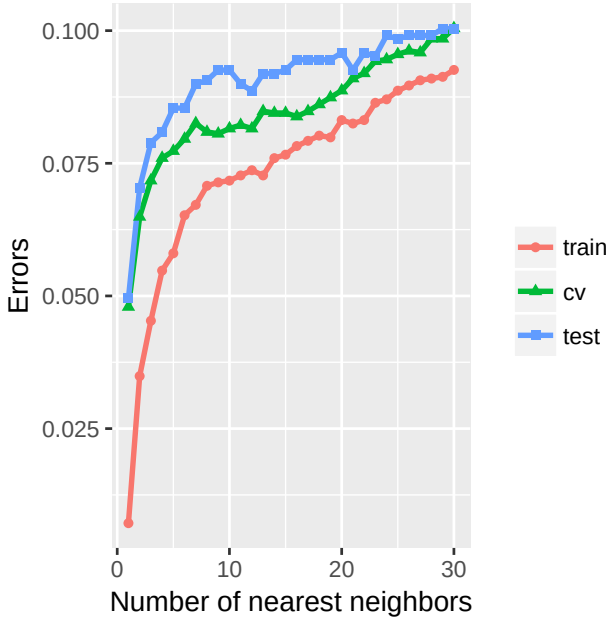
```
errors_sd = sapply(1:30, function(k) knn_error(k,train_sd,test_sd))
errors_log = sapply(1:30, function(k) knn_error(k,train_log,test_log))
errors_disc = sapply(1:30, function(k) knn_error(k,train_disc,test_disc))
errors_sd = data.frame(t(errors_sd), 1:30)
errors_log = data.frame(t(errors_log), 1:30)
errors_disc = data.frame(t(errors_disc), 1:30)
colnames(errors_sd) = c('train', 'cv', 'test', 'K')
colnames(errors_log) = c('train', 'cv', 'test', 'K')
colnames(errors_disc) = c('train', 'cv', 'test', 'K')
require(reshape2)
```

```
## Loading required package:  reshape2
```

```
errors_sd=melt(errors_sd, id="K")
errors_log=melt(errors_log, id="K")
errors_disc=melt(errors_disc, id="K")
par(mfrow=c(2,2))
x=ggplot(errors_sd, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1) +
y=ggplot(errors_log, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1)
z=ggplot(errors_disc, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1)
grid.arrange(x,y,z,ncol=2)
```

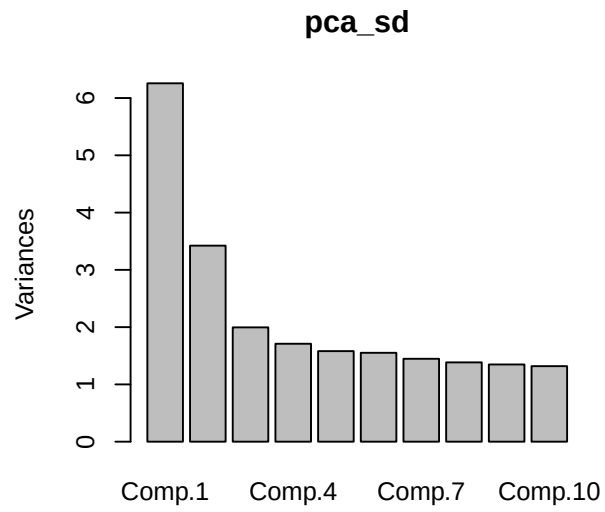


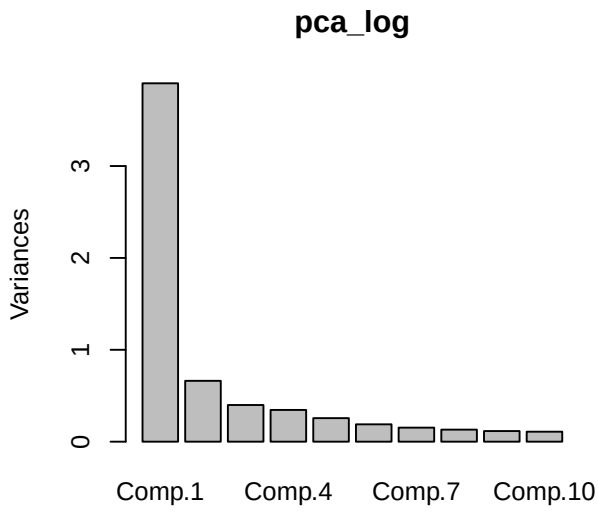


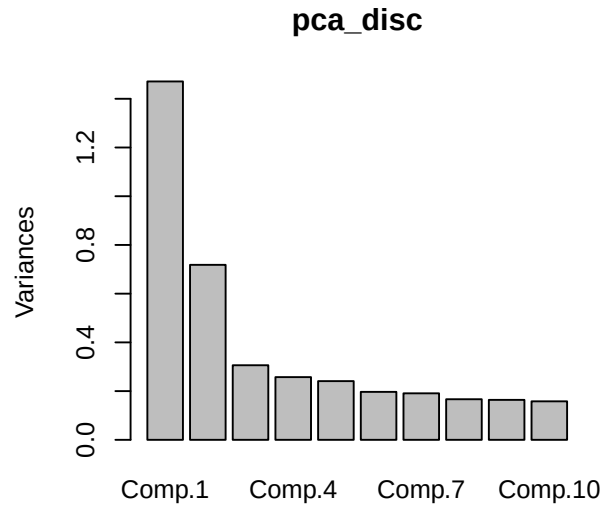



```
########Comment: error plots of original data.
par(mfrow=c(2,2))
pca_sd = princomp(train_sd[,-58], cor=FALSE)
plot(pca_sd)
pca_log = princomp(train_log[,-58], cor=FALSE)
```

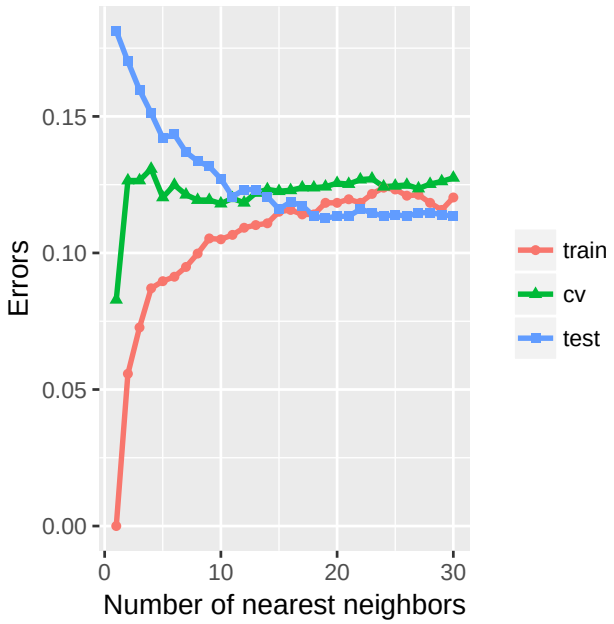
```
plot(pca_log)
pca_disc = princomp(train_disc[,-58], cor=FALSE)
plot(pca_disc)
test_sd_pca=as.matrix(test_sd[,-58]) %*% as.matrix(pca_sd$loadings[,1:2])
test_log_pca=as.matrix(test_log[,-58]) %*% as.matrix(pca_log$loadings[,1:2])
test_disc_pca=as.matrix(test_disc[,-58]) %*% as.matrix(pca_disc$loadings[,1:2])
######Comment: From scree plot, I choose 2 pincipal components for all 3 data sets.
errors_sd = sapply(1:30, function(k) knn_error(k,data.frame(pca_sd$scores[,1:2],spam=train_sd[,58]),data.
errors_log = sapply(1:30, function(k) knn_error(k,data.frame(pca_log$scores[,1:2],spam=train_log[,58]),da
errors_disc = sapply(1:30, function(k) knn_error(k,data.frame(pca_disc$scores[,1:2],spam=train_disc[,58])
errors_sd = data.frame(t(errors_sd), 1:30)
errors_log = data.frame(t(errors_log), 1:30)
errors_disc = data.frame(t(errors_disc), 1:30)
colnames(errors_sd) = c('train', 'cv', 'test', 'K')
colnames(errors_log) = c('train', 'cv', 'test', 'K')
colnames(errors_disc) = c('train', 'cv', 'test', 'K')
require(reshape2)
errors_sd=melt(errors_sd, id="K")
errors_log=melt(errors_log, id="K")
errors_disc=melt(errors_disc, id="K")
par(mfrow=c(2,2))
```

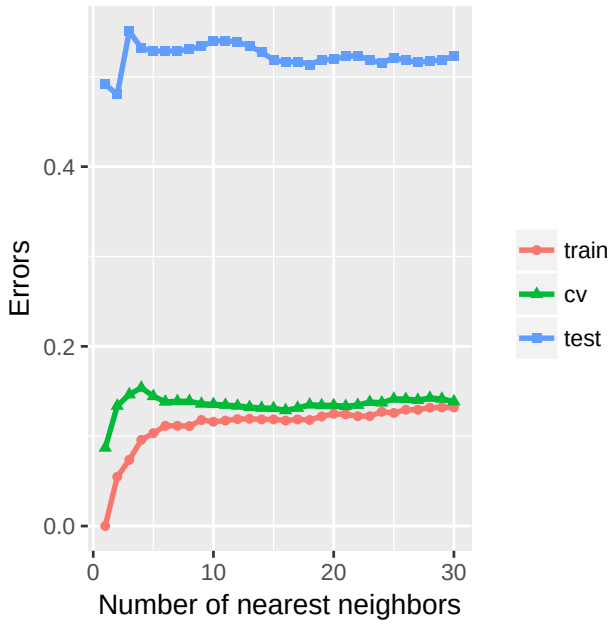


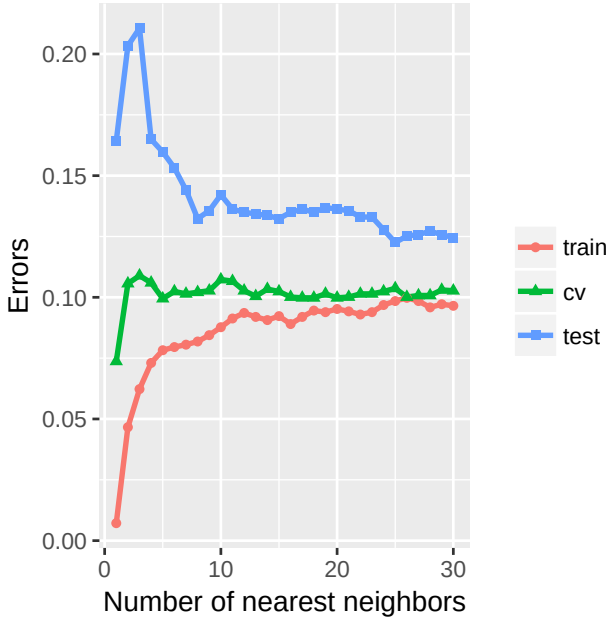





```
x=ggplot(errors_sd, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1) +
y=ggplot(errors_log, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1)
z=ggplot(errors_disc, aes(x=K, y=value, color=variable,group=variable, shape=variable))+geom_line(size=1)
grid.arrange(x,y,z,ncol=2)
```








```
######Comment:from the plots, and requirement of choosing k by CV error, we choose k=1 here.

########5
knn_sd=knn(train = train_sd[,-58], cl = train_sd$spam, test = test_sd[,-58], k = 1)
knn_log=knn(train = train_log[,-58], cl = train_log$spam, test = test_log[,-58], k = 1)
```

```
knn_disc=knn(train = train_disc[,-58], cl = train_disc$spam, test = test_disc[,-58], k = 1)
knn_sd1=knn(train = train_sd[,-58], cl = train_sd$spam, test = train_sd[,-58], k = 1)
knn_log1=knn(train = train_log[,-58], cl = train_log$spam, test = train_log[,-58], k = 1)
knn_disc1=knn(train = train_disc[,-58], cl = train_disc$spam, test = train_disc[,-58], k = 1)
a=table(test_sd$spam, knn_sd)
b=table(test_log$spam, knn_log)
c=table(test_disc$spam, knn_disc)
d=table(train_sd$spam, knn_sd1)
e=table(train_log$spam, knn_log1)
f=table(train_disc$spam, knn_disc1)
m_sd=array(c(d[1,2]/sum(d[1,]),a[1,2]/sum(a[1,]),d[2,1]/sum(d[2,]),a[2,1]/sum(a[2,])),c(2,2))
rownames(m_sd)=c("train","test")
colnames(m_sd)=c("0","1")
m_log=array(c(e[1,2]/sum(e[1,]),b[1,2]/sum(b[1,]),e[2,1]/sum(e[2,]),b[2,1]/sum(b[2,])),c(2,2))
rownames(m_log)=c("train","test")
colnames(m_log)=c("0","1")
m_disc=array(c(f[1,2]/sum(f[1,]),c[1,2]/sum(c[1,]),f[2,1]/sum(f[2,]),c[2,1]/sum(c[2,])),c(2,2))
rownames(m_disc)=c("train","test")
colnames(m_disc)=c("0","1")
m_sd
```

```
##                0          1
## train 0.00000000 0.00000000
## test  0.08624454 0.06148867
```

```
m_log
```

```
##                0          1
## train 0.00000000 0.00000000
## test  0.02620087 0.06957929
```

```
m_disc
```

```
##                 0          1
## train 0.004326663 0.01149425
## test  0.044759825 0.05663430
```

```
#####Comment:As we know, for continuous data, when k=1, the training error should be 0. But the
#training error for the discretized data is not 0; I feel this is because of inappropiate choice
#of meric in this case. Besides, the performance of knn is related to the transformation methods
#used for the data. The scaled version has the worst performance. And the performance of knn for
#different classes also varies according to different transformation. Log-version performs better
#for class 0, and discretized version performs better for class 1.
```