Analysis of Tissue Specimen for the Identification of Malignant Tumors

Predictors

Background

Modern diagnosis of breast cancer involves histologic examination of tissue specimen from the suspicious mass. Traditionally, a pathologist would examine a breast biopsy and compose a feature pattern for a particular case and then based on that pattern, associate a probability of malignancy (and a possible diagnosis of breast cancer). Attempts are currently being made to automate this process, using imaging equipment and machine learning (Maglogiannis & Zafiropoulos, 2004).

Origin of Data

Part of the histologic examination is examination of cytologic features which are features of individual cells in the breast biopsy. Three important cytologic features are cell nuclear size, nuclear membrane irregularity, and nuclear chromatin. 569 tissue specimens (from breast biopsies) were collected (Wolberg, 1995). Image analysis looked at 10-20 cells per specimen and classified each on: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. These variables are representations of nuclear size, nuclear membrane irregularity, and nuclear chromatin. The final measurement was an average of each sample's 10-20 cells.

Imputation

Missing data is a serious issue. It can bias parameter estimation, weaken generalizability, and decrease statistical power (Dong & Peng, 2013). Additionally, the techniques used in this section (PCA, Classification, etc.) are not naturally equipped to deal with missing data so imputation is a necessary step to pre-processing the data.

Missing Data Mechanism

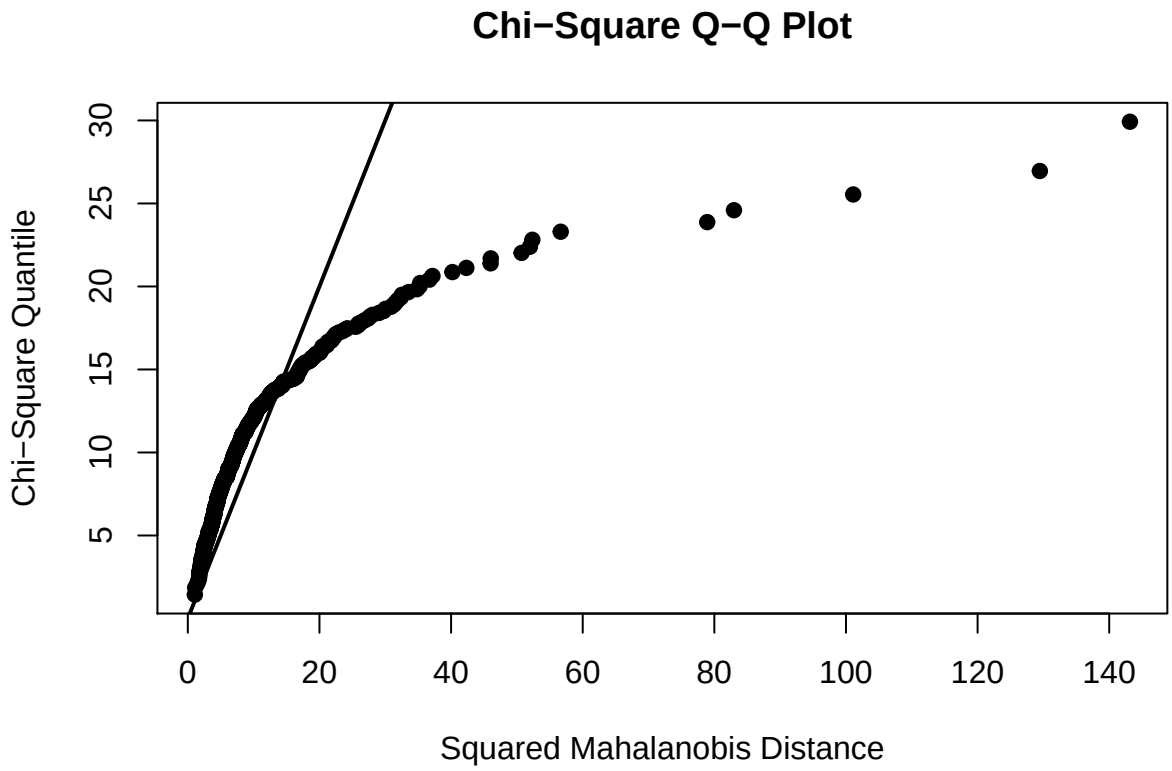
Imputation technique is conditional on the missing data mechanism at work in our dataset. If the data are missing completely at random (MCAR), then most imputation techniques won't violate the validity of any inferences we may make. Jamshidian and Jalal (2010) recommend two tests of the null hypothesis that data are MCAR: a normal theory test and a non-parametric test. Testing Mardia's multivariate skew coefficients, we found that the data are not multivariate normal, $p<0.05$. Applying the non-parametric MCAR test, we cannot reject the null hypothesis that our data are MCAR, $p>0.05$.

Non-Parameteric Imputation

Another important factor is the prevalence of missingness – this also determines what fix to apply. There is one pattern of missingness (Compactness, Concavity, and Concave Points) in the data for two observations: clients with IDs 843786 and 844359. Since the proportion is so low (0.4%), the go-to method would usually be list-wise deletion (Izenman, 2013). Additionally, since the proportion is so low, if we elect for some form of imputation, the difference between list-wise deletion and between different forms of imputation will be minimal in terms of analyses we run. In our case, the choice of how we deal with the missing data is very low-impact/risk. Simple imputation has the easiest implementation but it may bias standard error estimates

to make our models look more precise than they actually are (Izenman, 2013). Thus I elected to use the distribution-free imputation method of Srivastava and Dolatabadi (2009; implemented using the MissMech R package) for the two cases with missing data.

```
###2B. Missingness and Imputation
#check normality
mdtest = mardiaTest(cancer_raw[-c(6,7),-c(1,2)],qqplot=TRUE)
```




```
print(mdtest)
```

```
##    Mardia's Multivariate Normality Test
## ---------------------------------------
##    data : cancer_raw[-c(6, 7), -c(1, 2)]
##
##    g1p            : 70.46685
##    chi.skew       : 6659.118
##    p.value.skew   : 0
##
##    g2p            : 255.4102
##    z.kurtosis     : 104.0657
##    p.value.kurt   : 0
##
##    chi.small.skew : 6700.786
##    p.value.small  : 0
##
##    Result          : Data are not multivariate normal.
## ---------------------------------------
```

```
#not normal

#check MCAR status
mcartest = TestMCARNormality(data=cancer_raw[,-c(1,2)],del.lesscases=1,method="Nonparametric")
print(mcartest$pnormality)
```

```
## [1] 0.3934481
```

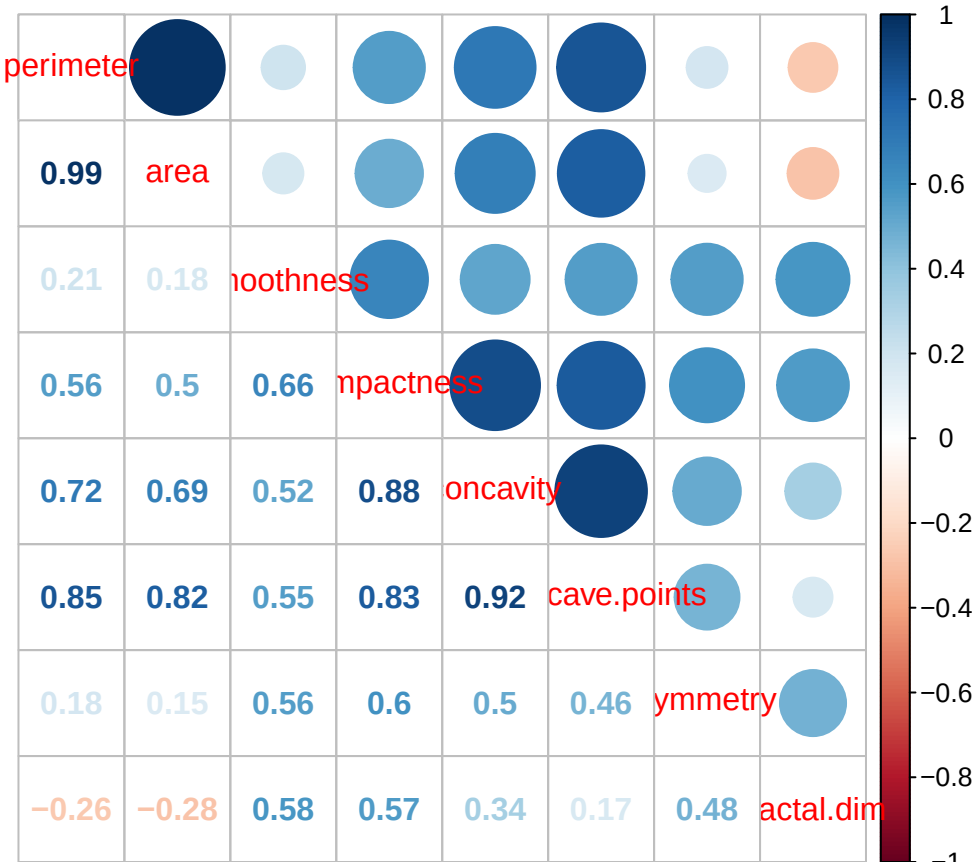
```
#can't reject MCAR at alpha=.05

#we can use a variety of imputation techniques without compromising inference
#since our missingness is  0.35%, the imputation technique we use most likely doesn't matter.

cancer<-TestMCARNormality(data=cancer_raw[,-c(1,2)],del.lesscases=1,imputation.method="Dist.Free",metho
```

Dimension Reduction through PCA

Are the predictors related?

Let's investigate the relationship between the predictors. Below is a correlogram, a visualization of the correlation matrix of the predictors. We can see strong positive relationships between the variables including a near perfect correlation between perimeter and area ($r = 0.99$).

```
###2C. PCA------
####predictors related?
#correlogram
corrplot.mixed(cor(cancer[,-c(1,2)]))
```




```
#smc
smc_data = as.data.frame(round(smc(cor(cancer[,-c(1,2)])),2))
names(smc_data) = "Squared Multiple Correlation"
knitr::kable(smc_data)
```

	Squared Multiple Correlation
perimeter	0.99
area	0.98
smoothness	0.65
compactness	0.93
concavity	0.91
concave.points	0.95
symmetry	0.44
fractal.dim	0.84

```
#multicollinearity looks like

#condition number
kappa(cor(cancer[,-c(1,2)]))
```

```
## [1] 445.6621
```

Further, the squared multiple correlations (SMCs; tabulated above) between the predictors are incredibly high. For example, 98.7% of the variance in Perimeter is accounted for by the other predictors; four other predictors have SMC's above 90% There is a lot of redundancy in these predictors. A lot of the variance that exists in the Perimeter dimension also exists in the Area dimension also exists in the Concave Points

dimension and so on. We don't need to keep a full, 8-dimensional predictor space because the data don't occupy the whole space. Geometrically, there may be an opportunity to reduce the predictor space in such a way as to preserve much of the original variance of the predictors.

Is PCA necessary?

From a statistical point-of-view we can take two stances. On the one hand, high inter-predictor associations often result in estimation problems (which may propagate to interpretation problems) so we may want to perform PCA. The phenomenon of such high correlation between predictors is multicollinearity, which interferes with matrix inversion and inflates parameter standard errors. In addition to the high SMCs, the correlation matrix of the predictors has a condition number of $\kappa(R)$=446, indicating ill-conditioning and multicollinearity. The inflated standard errors ruin parameter interpretations as we can't tell the contribution of individual predictors. On the other hand, as long as we can get our matrix to invert and we don't care about our high standard errors (essentially making our method a black box), we may not want to perform a PCA as preprocessing to our main analysis. From a practical point-of-view we don't really have a large number of predictors to project to a smaller subspace and even if we were to do so, does it make sense to interpret composite indices of these predictors? For example, what does it mean to have a weighted difference between smoothness and area, and perimeter and fractal dimension? The typical application of PCA involves many, many predictors whose variance can be effectively reduced down to only a handful of linear combinations. In this case with small p, perhaps a better solution would be some penalized glm like a lasso which would shrink some collinear predictors to zero.

Should we standardize for PCA?

PCA of unstandardized variables (i.e. PCA of Σ) is not equivalent to PCA of standardized variables (i.e. PCA of R). In fact, the principal components derived in one situation are not a simple function of those derived in the other. When there is a disparity in the scales of the variables, the variables with larger variances dominate (have larger coefficients for) the principal components. It is apparent from the various measures of dispersion in the table below that the eight predictors have wildly different scales – to be exact Area (and to a lesser extent, Perimeter) have much larger variances and so would dominate the PCA. Therefore, the PCA should be performed on R.

```
#standardize? compare variances of predictors
std_table = as.data.frame(round(cbind(

  apply(cancer[,-c(1,2)],2,sd),
  apply(cancer[,-c(1,2)],2,IQR),
  apply(cancer[,-c(1,2)],2,min),
  apply(cancer[,-c(1,2)],2,max)
),2))
colnames(std_table) = c("SD","IQR","Min","Max")
knitr::kable(std_table)
```

	SD	IQR	Min	Max
perimeter	24.30	28.93	43.79	188.50
area	351.91	362.40	143.50	2501.00
smoothness	0.01	0.02	0.05	0.16
compactness	0.05	0.07	0.02	0.35
concavity	0.08	0.10	0.00	0.43
concave.points	0.04	0.05	0.00	0.20
symmetry	0.03	0.03	0.11	0.30
fractal.dim	0.01	0.01	0.05	0.10

To drive this point home, we can look at the difference in coefficients between PCA of standardized and unstandardized predictors. The coefficients are shown below. Notice that the PCA of unstandardized predictors results in a first principal component totally dominated by Area.

```
#differences in loadings between PCA of E and of R
PC1Sloadings<-princomp(cancer[,-c(1,2)],cor=FALSE)[["loadings"]][,1] #1st PC of Sigma
PC1Rloadings<-princomp(cancer[,-c(1,2)],cor=TRUE)[["loadings"]][,1] #1st PC of R
loading_tbl = round(-cbind(PC1Sloadings,PC1Rloadings),2)
colnames(loading_tbl) = c("PC1 S Loadings","PC1 R Loadings")
knitr::kable(loading_tbl)
```

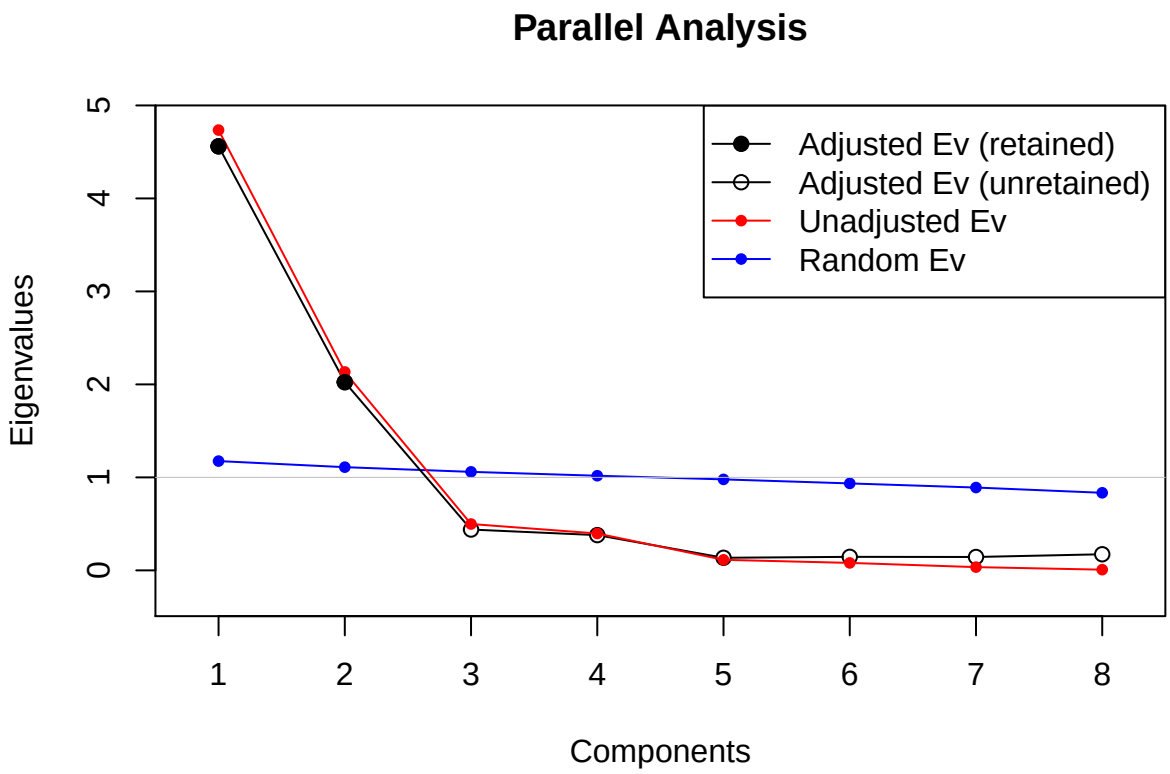
	PC1 S Loadings	PC1 R Loadings
perimeter	0.07	0.36
area	1.00	0.34
smoothness	0.00	0.30
compactness	0.00	0.42
concavity	0.00	0.43
concave.points	0.00	0.44
symmetry	0.00	0.28
fractal.dim	0.00	0.15

Perform PCA

To select the optimal number of principal components we will use Horn's Parallel Analysis and the amount of variation explained:

```
#HPA
paran(cancer[,-c(1,2)],quietly=TRUE,graph=TRUE)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
```



HPA tells us that we should stick with two principal components. As the table below shows, these correspond to 86% of the total variance in the predictors! Including an additional principal component only increases the total variance accounted for by 6%. Thus we stick with two principal components.

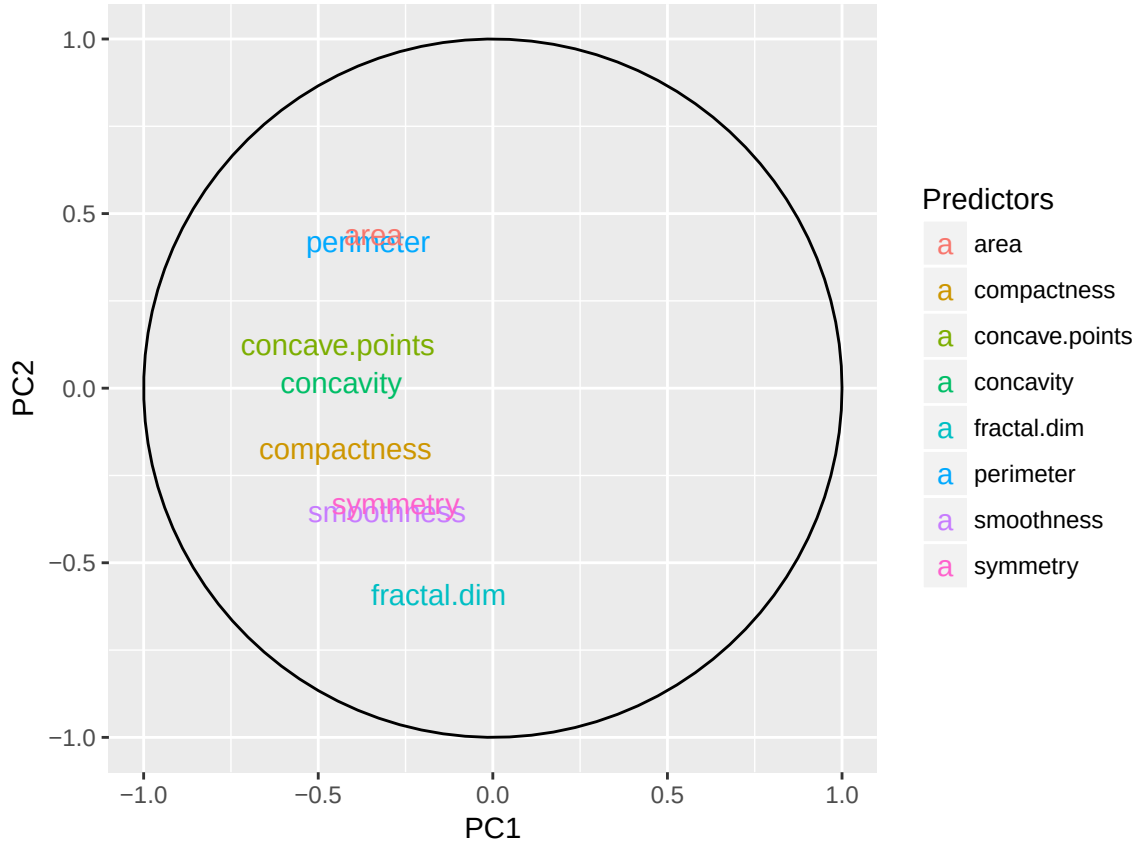
```
#prop variance
pca = princomp(cancer[,-c(1,2)],cor=TRUE)
PoV <- round(pca$sdev^2/sum(pca$sdev^2),2)
pov_tbl = rbind(PoV,cumsum(PoV))
rownames(pov_tbl) = c("Prop of Var.","Cum. Prop.")
knitr::kable(pov_tbl)
```

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8
Prop of Var.	0.59	0.27	0.06	0.05	0.01	0.01	0.00	0.00
Cum. Prop.	0.59	0.86	0.92	0.97	0.98	0.99	0.99	0.99

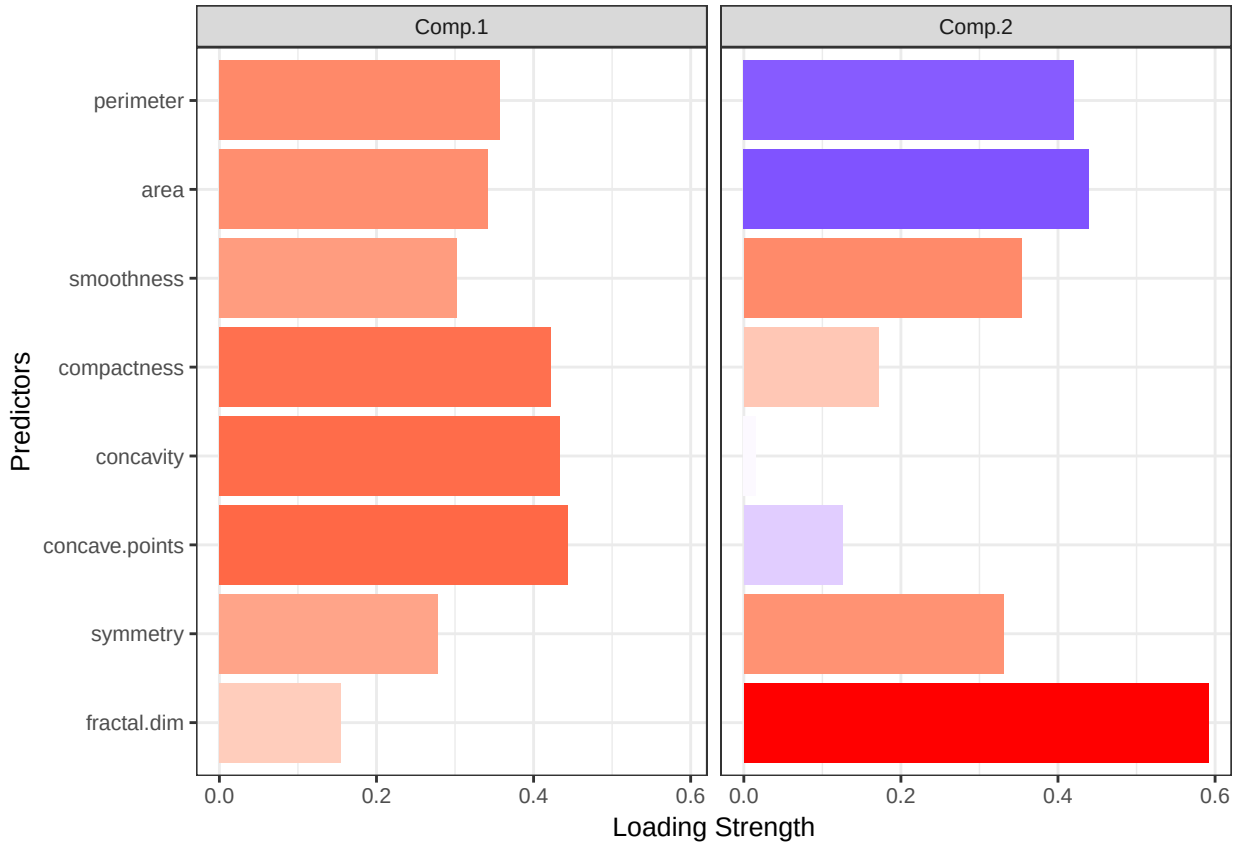
The two figures below aid us in interpreting the principal components as composite indices of our predictors:

```
#2-component PCA
load<-princomp(cancer[,-c(1,2)],cor=TRUE)[["loadings"]][,1:2]
cancer.pca <- prcomp(cancer[,-c(1,2)], scale. = TRUE)

#predictors on unit circle
theta <- seq(0,2*pi,length.out = 100)
circle <- data.frame(x = cos(theta), y = sin(theta))
p <- ggplot(circle,aes(x,y)) + geom_path()
loadings <- data.frame(cancer.pca$rotation,Predictors = row.names(cancer.pca$rotation))
```

```
p + geom_text(data=loadings,mapping=aes(x = PC1, y = PC2, label = Predictors, colour = Predictors)) + co
```




```
#plot of coefficients
ggplot(melt(load[8:1,]), aes(Var1, abs(value), fill=value)) +
  facet_wrap(~ Var2, nrow=1) + #place the factors in separate facets
  geom_bar(stat="identity") + #make the bars
  coord_flip() + #flip the axes so the test names can be horizontal
  #define the fill color gradient: blue=positive, red=negative
  scale_fill_gradient2(name = "Loading",
                       high = "blue", mid = "white", low = "red",
                       midpoint=0, guide=F) +
  ylab("Loading Strength") +xlab("Predictors")+ #improve y-axis label
  theme_bw(base_size=10) #use a black-and-white theme with set font size
```

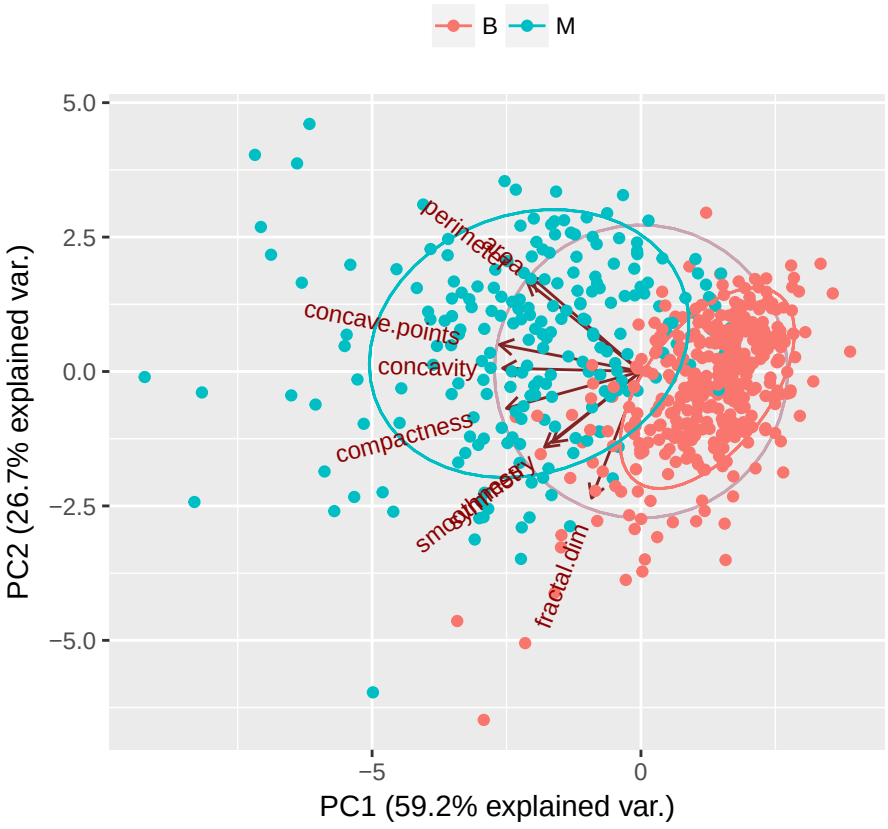


Our first principal component is a (nearly equal) weighted mean of all predictors; Fractal Dimension has the least weight. The second principal component is a weighted difference between Fractal Dimension, Symmetry, and Smoothness; and Perimeter and Area.

PCA and Classification

Can we see separation between the two diagnoses in our reduced subspace? In other words, is the 14% loss in fidelity of predictor variance worth it in terms of our ultimate goal of classification? The biplot below answers the question. The first principal component and the 59.2% variance it accounts for actually does a decent job of separating the diagnoses; incorporate the second principal component (86% cumulate variance) and the separation is even better. The question now is how much better this separation is than if we had used the original data. This will be answered in the next section.

```
#ggbiplot
cancer.pca <- prcomp(cancer[,-c(1,2)], scale. = TRUE)
ggbiplot(cancer.pca, obs.scale = 1, var.scale = 1,
         groups = cancer_raw$diagnosis, ellipse = TRUE, circle = TRUE) +
  scale_color_discrete(name = '') +
  theme(legend.direction = 'horizontal', legend.position = 'top')
```



Classification

Now we want to apply a selection of classifiers to the data to classify cases as benign vs. malignant. Building on our PCA, we must ask, can we perform classification on the reduced space formed by the first two principal components? In other words, is the variance we lost when performing data reduction very important in the prediction of diagnosis? If it is, we will see a large difference in classification metrics; if it isn't, the difference should be minimal. The advantage of a 2-component PCA is that we can visually assess if a classifier is performing well by looking at the decision boundary superimposed on a scatterplot of the first two principal components. Note that using PCA to deal with multicollinearity isn't really an advantage here because in this classification context, we don't care about the effects of multicollinearity, only about prediction precision. We may not be able to figure out how important each predictor is in terms of the prediction, but the prediction itself should not be affected. We use a validation set approach, randomly splitting the observations 50-50 into a training set and a test set. We train the classifiers on the training set and evaluate their performance with the test set.

```
train_index <- sample(569,285)

train <- cancer_raw[train_index, -1]
test <- cancer_raw[-train_index, -1]

cancer.pca <- prcomp(cancer[,-c(1,2)], scale. = TRUE)
pcscores<-as.data.frame(cancer.pca$x[,c(1,2)])
pcscores[["diagnosis"]]<-cancer_raw[,2]
train.pc <-pcscores[train_index,]
test.pc <-pcscores[-train_index,]
```

Logistic Regression

As mentioned above, we don't bother with subset selection as our focus is on prediction only. We fit all predictors. Logistic Regression fit very well with an overall classification rate of 93.3%, a sensitivity of 89.3% and a specificity of 96.7%. Individuals with benign masses were nearly classified perfectly. About 1 out of every 10 individuals with malignant were misdiagnosed. Sensitivity takes high precedence since the error of making a negative diagnosis when the client is positive is worse than making a positive diagnosis when the client is negative. It is better to administer cancer medication to someone without cancer (even if the medication has terrible side effects) than to give someone a clean bill of health and let someone slip past the point of no return.

Logistic Regression (Reduced Space)

With the reduced information, logistic regression performance fell across overall classification rate (1% drop), sensitivity (~4% drop), and specificity (1.6% drop). While this isn't a big deal in a statistical sense, from a practical sense any increase in error can come at the cost of human lives.

```
##### LR #####
glm.fit=glm(diagnosis~.,data=train ,family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm.probs =predict(glm.fit, test[,-1]  ,type="response")
glm.pred=rep("B" ,284)
glm.pred[glm.probs >0.5]= "M"

#test confusion matrix
tab<-table(actual=test[,1],glm.pred)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9119718
```

```
#ROC
lr.roccurve<-roc(test[,1]~glm.probs)
#AUC
auc(lr.roccurve)
```

```
## Area under the curve: 0.9808
```

```
###LR for reduced space
glm.fit=glm(diagnosis~.,data=train.pc ,family = binomial)
glm.probs =predict(glm.fit, test.pc[,-3]  ,type="response")
glm.pred=rep("B" ,284)
glm.pred[glm.probs >0.5]= "M"

#test confusion matrix
tab<-table(actual=test.pc[,3],glm.pred)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9190141
```

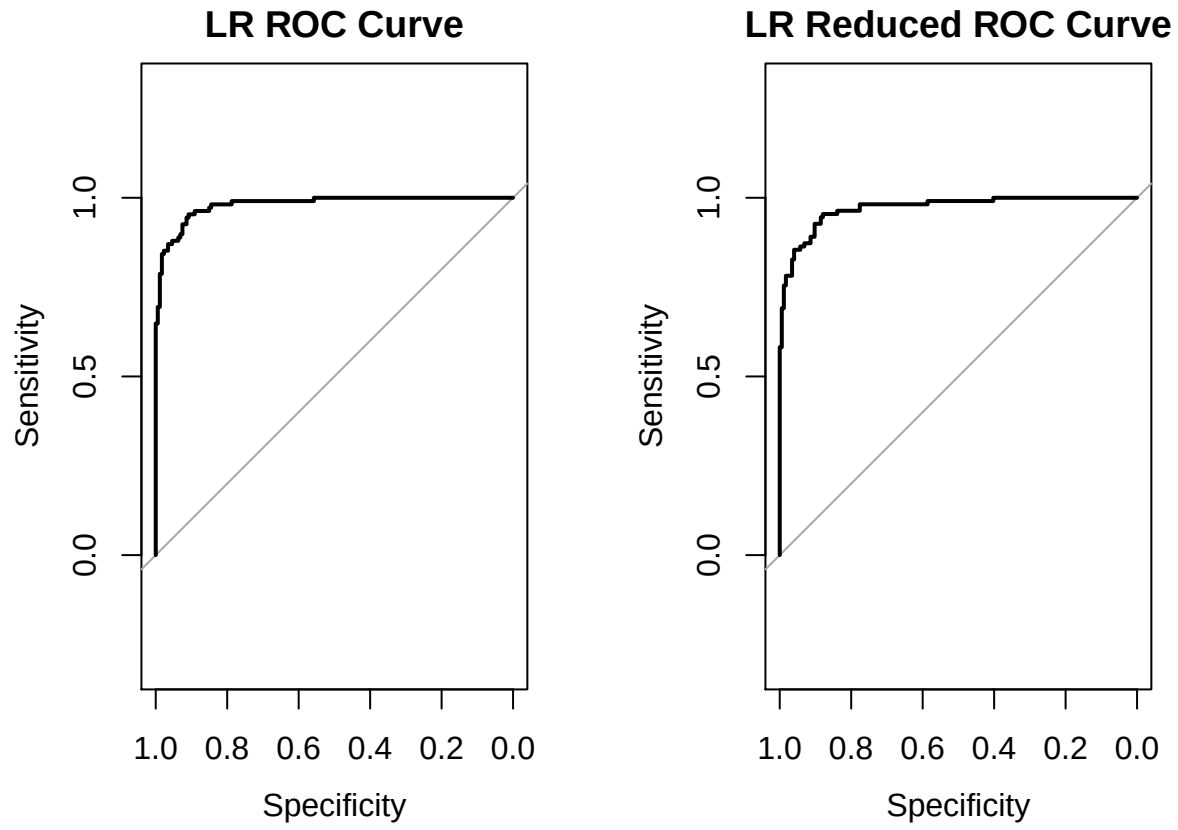
```
#ROC
lr.red.roccurve<-roc(test.pc[,3]~glm.probs)
#AUC
auc(lr.red.roccurve)
```

```
## Area under the curve: 0.9713
```

```
par(mfrow=c(1,2))
plot(lr.roccurve,ylim=c(0,1),main="LR ROC Curve")
```

```
##
## Call:
## roc.formula(formula = test[, 1] ~ glm.probs)
##
## Data: glm.probs in 174 controls (test[, 1] B) < 108 cases (test[, 1] M).
## Area under the curve: 0.9808
```

```
plot(lr.red.roccurve,ylim=c(0,1),main="LR Reduced ROC Curve")
```




```
##
## Call:
## roc.formula(formula = test.pc[, 3] ~ glm.probs)
##
## Data: glm.probs in 174 controls (test.pc[, 3] B) < 110 cases (test.pc[, 3] M).
## Area under the curve: 0.9713
```

Linear Discriminant Analysis

LDA performed better than logistic regression overall (94.7% overall classification rate). It outperformed logistic regression in specificity (99.4%) but at the cost of worse sensitivity (86.3%) which is the error rate that we care about. Therefore logistic regression is still the superior classifier.

Linear Discriminant Analysis (Reduced Space)

Like logistic regression, LDA on the reduced data decreased performance across the board; unlike logistic regression, the decrease was marked. This is very unacceptable from a practical standpoint.

```
##### LDA #####
lda.fit=lda(diagnosis~., data=train )
lda.pred = predict(lda.fit, test)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
lda.class =lda.pred$class

#test confusion matrix
tab<-table(actual=test[,1],lda.class)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9255319
```

```
#ROC
lda.roccurve<-roc(test[,1]~lda.pred$posterior[,2])
#AUC
auc(lda.roccurve)
```

```
## Area under the curve: 0.9784
```

```
###LDA reduced
lda.fit=lda(diagnosis~., data=train.pc )
lda.pred = predict(lda.fit, test.pc)
lda.class =lda.pred$class

#test confusion matrix
tab<-table(actual=test.pc[,3],lda.class)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.8978873
```

```
tab
```

```
##       lda.class
## actual   B   M
##      B 171   3
##      M  26  84
```

```
#ROC
lda.red.roccurve<-roc(test.pc[,3]~lda.pred$posterior[,2])
#AUC
auc(lda.roccurve)
```

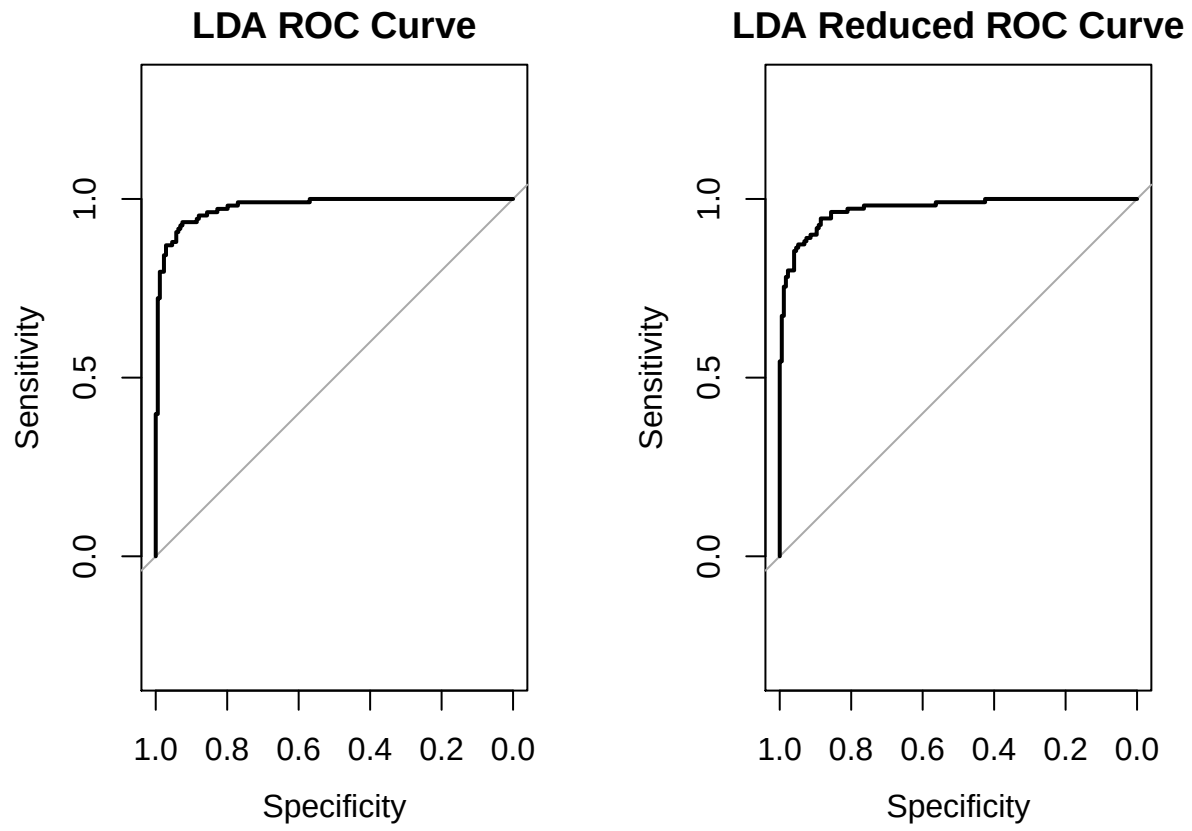
```
## Area under the curve: 0.9784
```

```
par(mfrow=c(1,2))
plot(lda.roccurve,ylim=c(0,1),main="LDA ROC Curve")
```

```
##
## Call:
```

```
## roc.formula(formula = test[, 1] ~ lda.pred$posterior[, 2])
##
## Data: lda.pred$posterior[, 2] in 174 controls (test[, 1] B) < 108 cases (test[, 1] M).
## Area under the curve: 0.9784
```

```
plot(lda.red.roccurve,ylim=c(0,1),main="LDA Reduced ROC Curve")
```




```
##
## Call:
## roc.formula(formula = test.pc[, 3] ~ lda.pred$posterior[, 2])
##
## Data: lda.pred$posterior[, 2] in 174 controls (test.pc[, 3] B) < 110 cases (test.pc[, 3] M).
## Area under the curve: 0.9716
```

Quadratic Discriminant Analysis

QDA performed the same as LDA in terms of sensitivity (86.3%) but was 2.7% worse in terms of specificity. So far it is the worst classifier of the three.

Quadratic Discriminant Analysis (Reduced Space)

Curiously, QDA on the reduced space saw a huge decrease in sensitivity (drop of 4.4%) and a small increase in specificity (plus 1.6%). Overall, it's still unimpressive.

```
##### QDA #####
qda.fit=qda(diagnosis~., data=train )
qda.pred = predict(qda.fit, test)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
qda.class =qda.pred$class

#test confusion matrix
tab<-table(actual=test[,1],qda.class)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9219858
```

```
#ROC
qda.roccurve<-roc(test[,1]~qda.pred$posterior[,2])
#AUC
auc(qda.roccurve)
```

```
## Area under the curve: 0.9739
```

```
##QDA reduced
qda.fit=qda(diagnosis~., data=train.pc )
qda.pred = predict(qda.fit, test.pc)
qda.class =qda.pred$class
#test confusion matrix
tab<-table(actual=test.pc[,3],qda.class)
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9190141
```

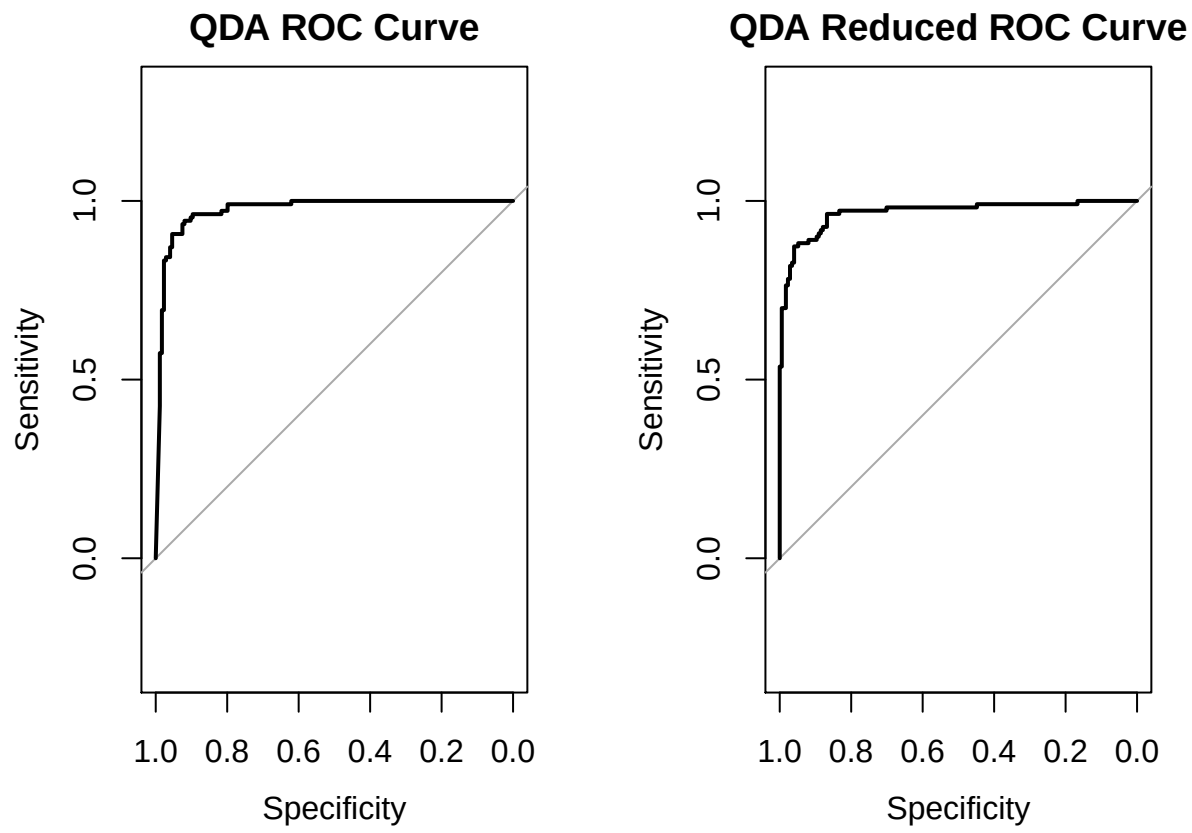
```
#ROC
qda.red.roccurve<-roc(test.pc[,3]~qda.pred$posterior[,2])
#AUC
auc(qda.roccurve)
```

```
## Area under the curve: 0.9739
```

```
par(mfrow=c(1,2))
plot(qda.roccurve,ylim=c(0,1),main="QDA ROC Curve")
```

```
##
## Call:
## roc.formula(formula = test[, 1] ~ qda.pred$posterior[, 2])
##
## Data: qda.pred$posterior[, 2] in 174 controls (test[, 1] B) < 108 cases (test[, 1] M).
## Area under the curve: 0.9739
```

```
plot(qda.red.roccurve,ylim=c(0,1),main="QDA Reduced ROC Curve")
```



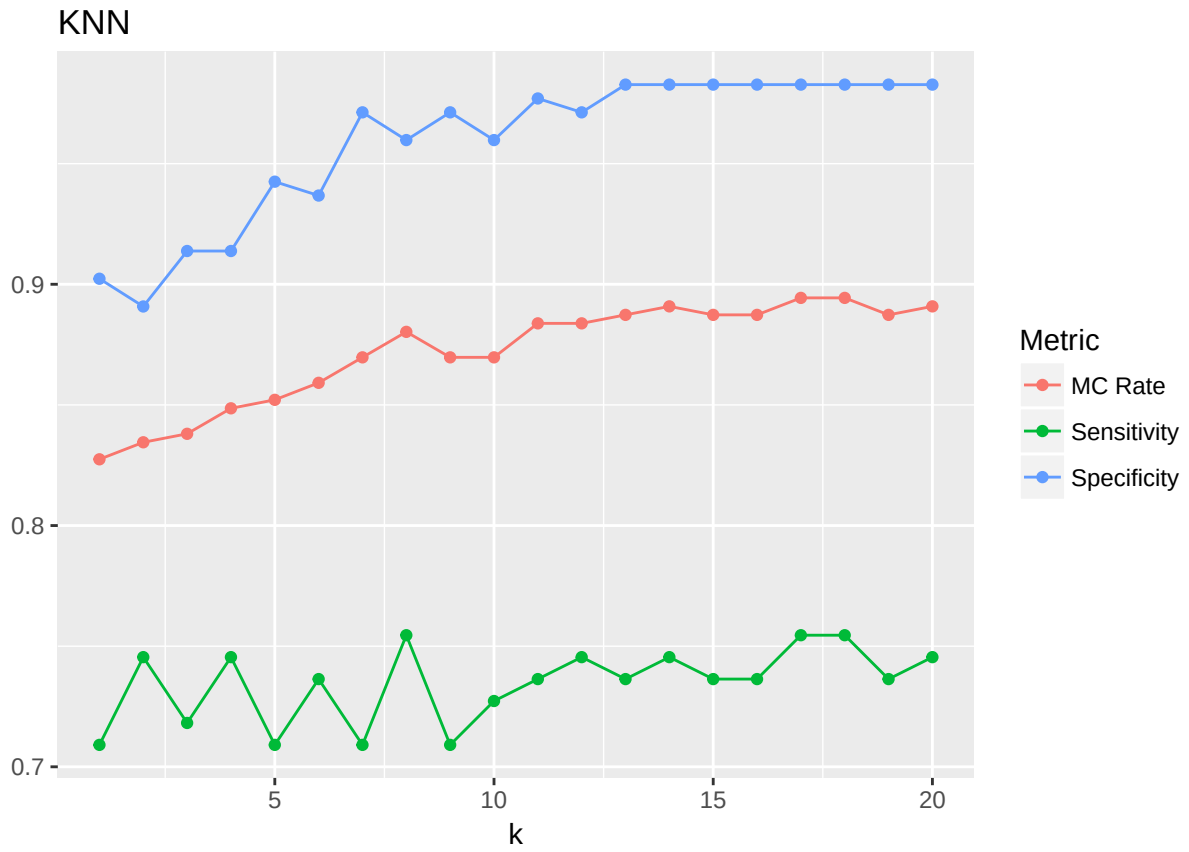

```
##
## Call:
## roc.formula(formula = test.pc[, 3] ~ qda.pred$posterior[, 2])
##
## Data: qda.pred$posterior[, 2] in 174 controls (test.pc[, 3] B) < 110 cases (test.pc[, 3] M).
## Area under the curve: 0.9675
```

K-Nearest-Neighbors

To pick the optimal K we tune it with the total misclassification rate (TCR), sensitivity, and specificity:

```
##### KNN #####
standardized.X=scale(cancer, scale =FALSE)
train.X <- standardized.X[train_index, ]
train.Y <- cancer_raw$diagnosis[train_index]
test.X <- standardized.X[-train_index, ]
test.Y <- cancer_raw$diagnosis[-train_index]
train.X.red <- pcscores[train_index,]
test.X.red <- pcscores[-train_index,]

set.seed(1) # for tie breakers

mcr<-numeric(20)
sens<-numeric(20)
spec<-numeric(20)
for(i in 1:20){
```

```
  knn.pred=knn(train.X,test.X,train.Y,k=i)
  tab<-table(knn.pred ,test.Y)
  mcr[i]<-sum(diag(tab))/sum(tab) #MCR
  sens[i]<-1-tab[1,2]/sum(tab[,2]) #sensitivity - true positive
  spec[i]<-1-tab[2,1]/sum(tab[,1]) #specificity - true negative
}

knn_plot = data.frame(
  k = rep(1:20,3),
  measure = c(mcr,sens,spec),
  Metric = rep(c("MC Rate","Sensitivity","Specificity"),each=20))
ggplot(data=knn_plot, aes(x=k,y=measure,colour=Metric)) +
  geom_line() +
  geom_point() +
  ylab("") +
  ggtitle("KNN")
```



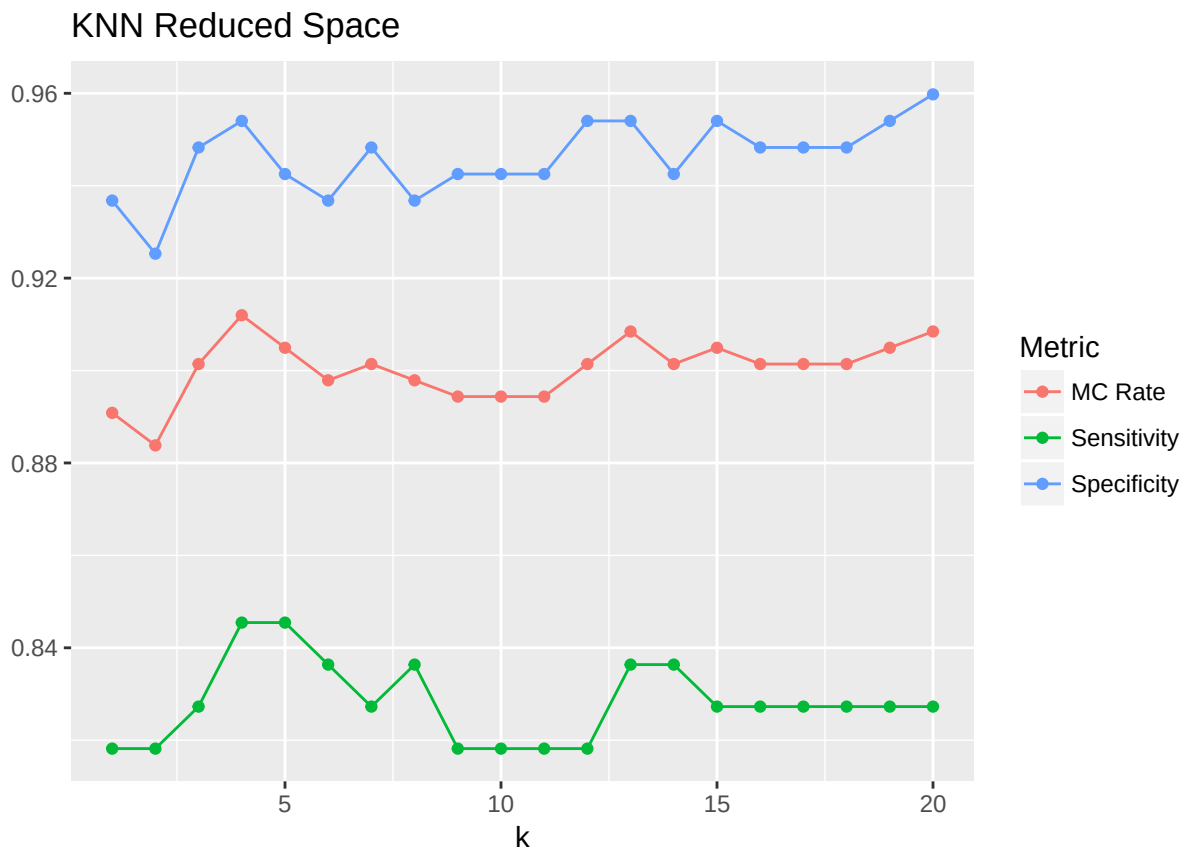
Total classification rate is highest for k=14. The k with best tradeoff between optimum sensitivity and total classification rate is k=3. So, we will use the 3-Nearest-Neighbors approach when comparing to other classifiers. K-Nearest-Neighbors appears to be the worst so far.

```
#KNN for reduced space

mcr<-numeric(20)
sens<-numeric(20)
spec<-numeric(20)
```

```
for(i in 1:20){
  knn.pred=knn(train.X.red[,c(1,2)],test.X.red[,c(1,2)],train.Y,k=i)
  tab<-table(knn.pred ,test.Y)
  mcr[i]<-sum(diag(tab))/sum(tab) #MCR
  sens[i]<-1-tab[1,2]/sum(tab[,2]) #sensitivity - true positive
  spec[i]<-1-tab[2,1]/sum(tab[,1]) #specificity - true negative
}
knn_tbl = as.data.frame(round(cbind(mcr,sens,spec),3))
row.names(knn_tbl) = 1:20
knitr::kable(knn_tbl,row.names=TRUE)
```

	mcr	sens	spec
1	0.891	0.818	0.937
2	0.884	0.818	0.925
3	0.901	0.827	0.948
4	0.912	0.845	0.954
5	0.905	0.845	0.943
6	0.898	0.836	0.937
7	0.901	0.827	0.948
8	0.898	0.836	0.937
9	0.894	0.818	0.943
10	0.894	0.818	0.943
11	0.894	0.818	0.943
12	0.901	0.818	0.954
13	0.908	0.836	0.954
14	0.901	0.836	0.943
15	0.905	0.827	0.954
16	0.901	0.827	0.948
17	0.901	0.827	0.948
18	0.901	0.827	0.948
19	0.905	0.827	0.954
20	0.908	0.827	0.960

```
knn_plot = data.frame(
  k = rep(1:20,3),
  measure = c(mcr,sens,spec),
  Metric = rep(c("MC Rate","Sensitivity","Specificity"),each=20))
ggplot(data=knn_plot, aes(x=k,y=measure,colour=Metric)) +
  geom_line() +
  geom_point() +
  ylab("") +
  ggtitle("KNN Reduced Space")
```



Prioritizing sensitivity, the optimal k for the reduced space is k=4. What's interesting here is that classification for KNN in the reduced space is better in every way than it is for the full space. Why should this be if we lost information (variation) when we reduced dimensions? My guess is that the nature of this algorithm is to classify based on neighbors and with reduced dimensions, there is less noise in the distances between neighbors. An observation may have been close to another observation with a different diagnosis in the full space but once the space was reduced, the new projected observation is now closer to a projected observation with the same diagnosis. Compare Classifiers To compare the performance of classifiers we will use five measures. The first is the misclassification rate which is simply, "when did the classifier get it right?" Out of all of the cases, how many cases were correctly diagnosed as benign or malignant? Sensitivity is our true positive rate – out of those cases with malignant tumors, how many were correctly diagnosed as malignant? Specificity is our true negative rate – out of those cases with benign tumors, how many were correctly diagnosed as benign? Finally, we look at the ROC curve for each. The ROC curve tells us the relationship between sensitivity and specificity for a classifier as we vary the threshold. For example, for a poor classifier, decreasing the threshold of classification to increase the sensitivity will tank the specificity. A great classifier will have a threshold at which both specificity and sensitivity are maximized at a high value (close to 1). The area under the ROC curve is a measure of the discrimination power of the test. It is bounded between 1 and 0.5 with the former being a perfect classifier and the latter being random chance (a coin flip for a diagnosis). Note that KNN in its original form is a non-probabilistic classifier so it does not have ROC curve or corresponding AUC.

```
#performancemetrics
```

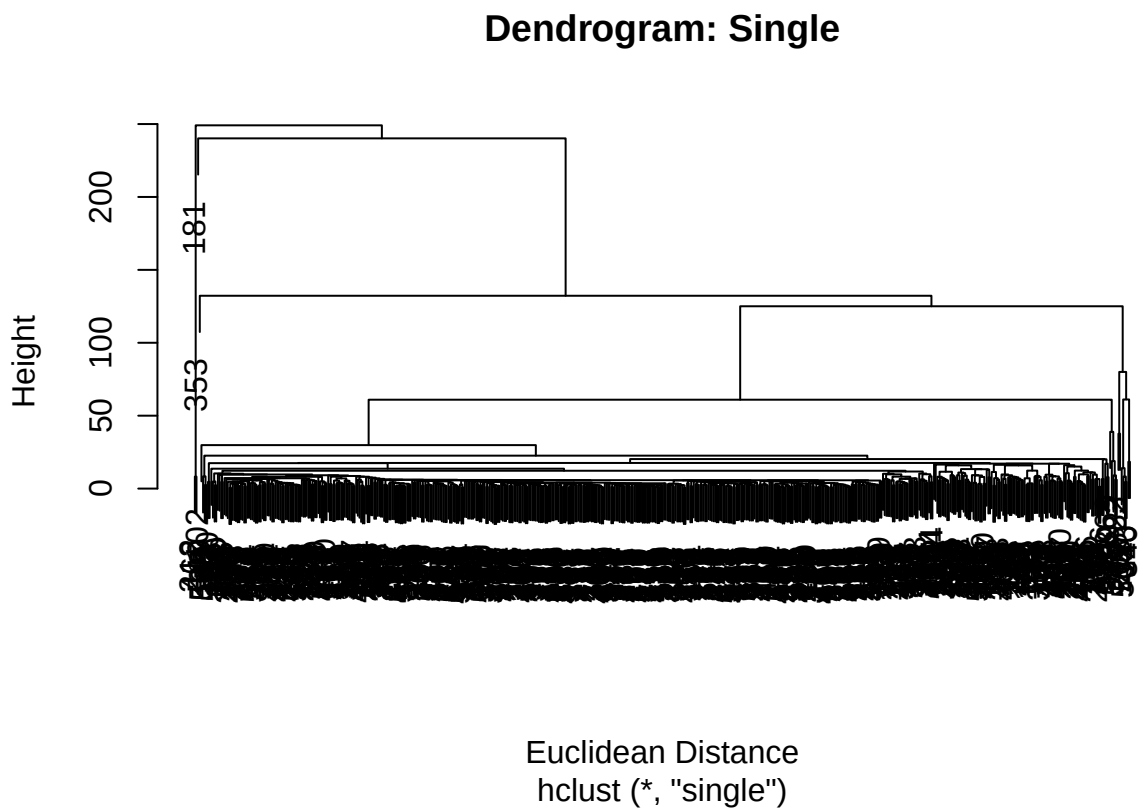
In terms of overall classification, LDA is the best, correctly diagnosing nearly 19 out of 20 patients. It also has the highest specificity, with nearly perfect classification of patients with benign tumors. The problem is its sensitivity is a full 3 percentage points lower than the two classifiers with the highest sensitivity: KNN (reduced) and logistic regression. KNN is a surprise because when applied to the original data it is easily the worst classifier. However, when applied to the reduced data, it's sensitivity improves markedly. Unfortunately it doesn't have a ROC curve because it's a nonprobabilistic classifier. Logistic regression is the best overall

performer (besides LDA) with high sensitivity, decent specificity, and large AUC. If specificity was important, LDA would be the top classifier; but in this context, sensitivity is mostly what matters and that makes Logistic Regression the best classifier.

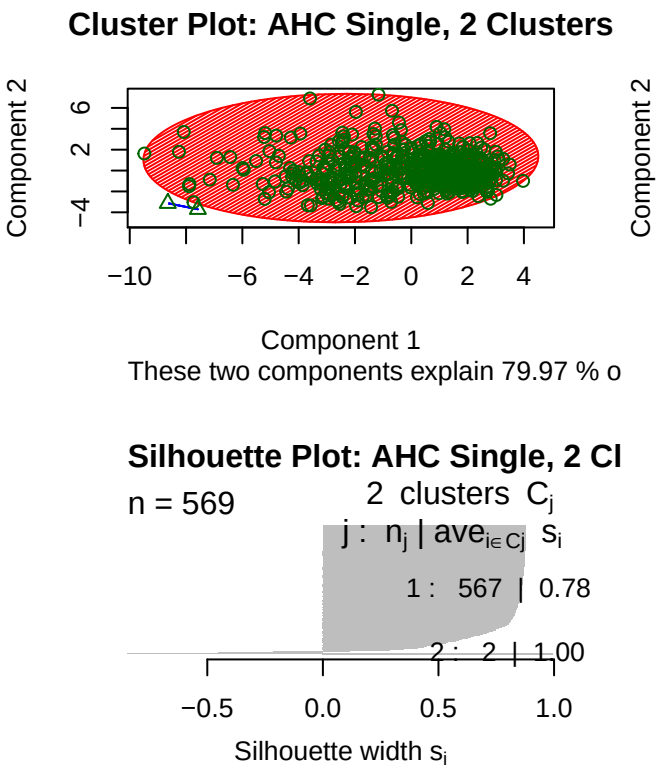
```
#LRCurves
```

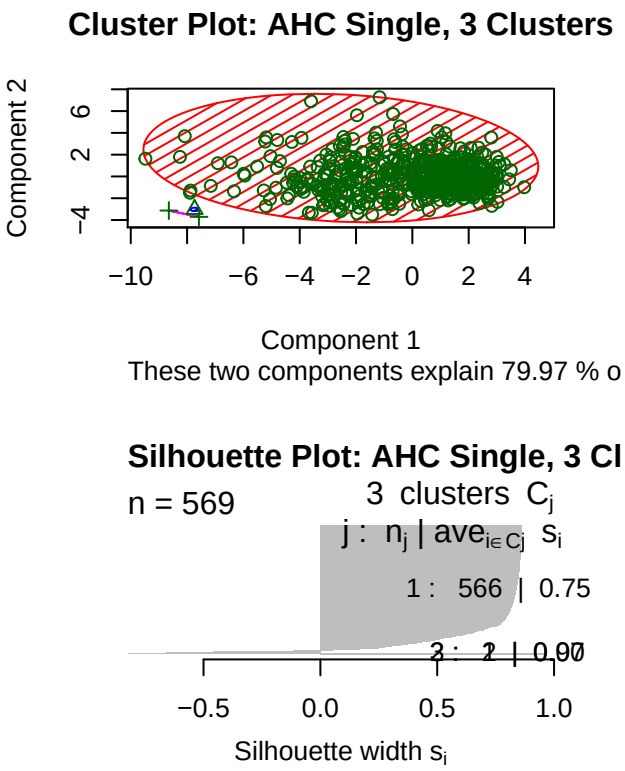
Cluster Analysis

```
###5. Cluster Analysis------

#euclidean
d <- dist(cancer)

#single-linkage
hc.s <- hclust(d,method="single")
plot(hc.s,main="Dendrogram: Single",xlab="Euclidean Distance")
```

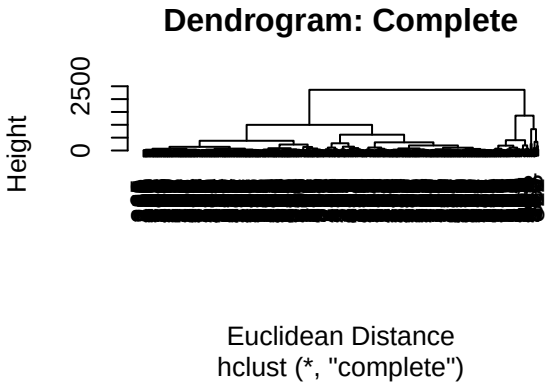


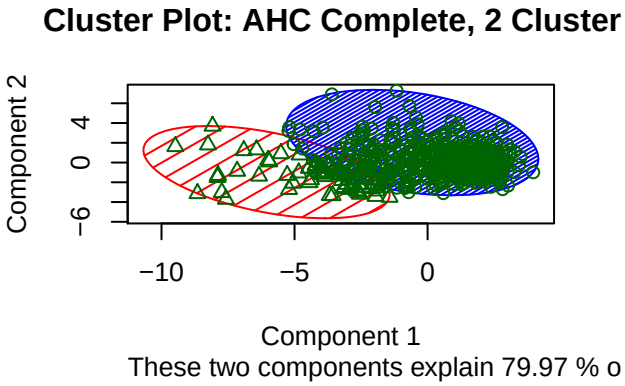

```
windows()
par(mfrow=c(2,2))
hc.s.2 <- cutree(hc.s, 2)
hc.s.3 <- cutree(hc.s, 3)
clusplot(cancer, hc.s.2,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Single, 2 Clusters")
clusplot(cancer, hc.s.3,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Single, 3 Clusters")
plot(silhouette(hc.s.2,d),main="Silhouette Plot: AHC Single, 2 Clusters")
plot(silhouette(hc.s.3,d),main="Silhouette Plot: AHC Single, 3 Clusters")
```

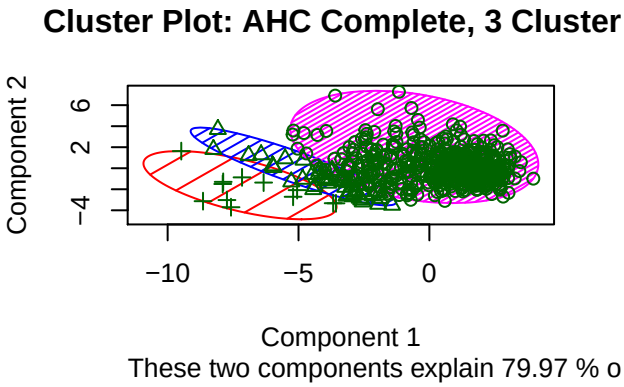


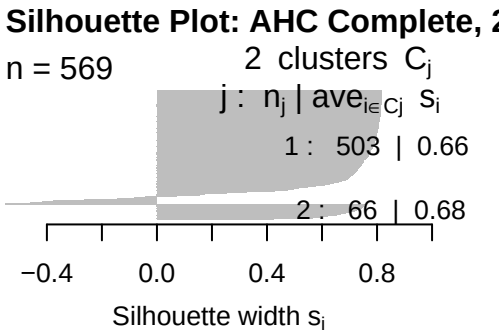



```
#complete-linkage
hc.comp <- hclust(d,method="complete")
plot(hc.comp,main="Dendrogram: Complete",xlab="Euclidean Distance")

hc.comp.2 <- cutree(hc.comp, 2)
hc.comp.3 <- cutree(hc.comp, 3)
clusplot(cancer, hc.comp.2,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Complete, 2 Clusters
clusplot(cancer, hc.comp.3,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Complete, 3 Clusters
plot(silhouette(hc.comp.2,d),main="Silhouette Plot: AHC Complete, 2 Clusters")
```

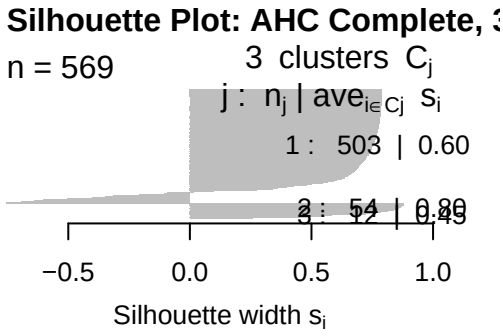


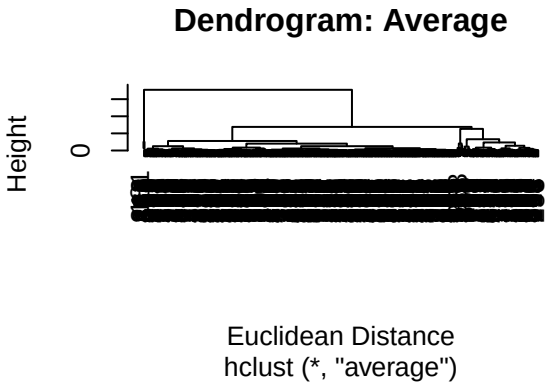


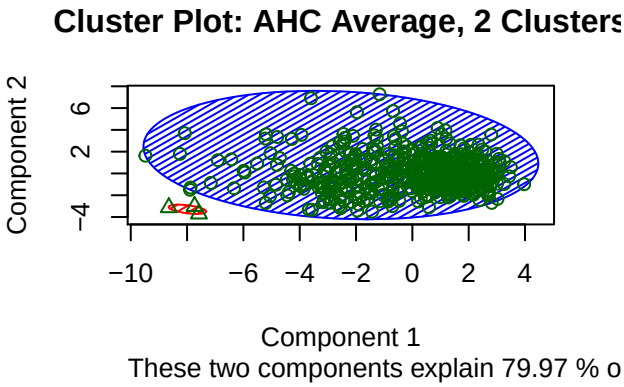


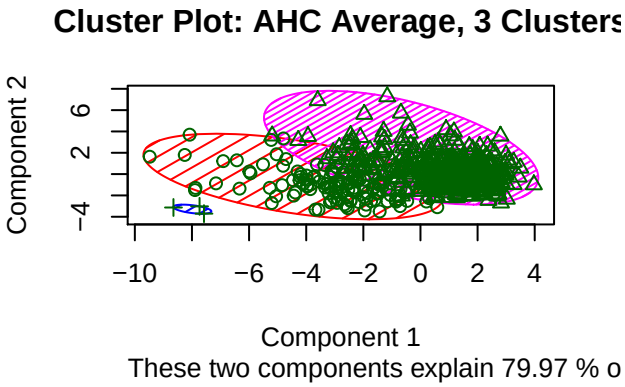



```
plot(silhouette(hc.comp.3,d),main="Silhouette Plot: AHC Complete, 3 Clusters")


#average-linkage
hc.a <- hclust(d,method="average")
plot(hc.a,main="Dendrogram: Average",xlab="Euclidean Distance")

hc.a.2 <- cutree(hc.a, 2)
hc.a.3 <- cutree(hc.a, 3)
clusplot(cancer, hc.a.2,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Average, 2 Clusters")
clusplot(cancer, hc.a.3,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Average, 3 Clusters")
```

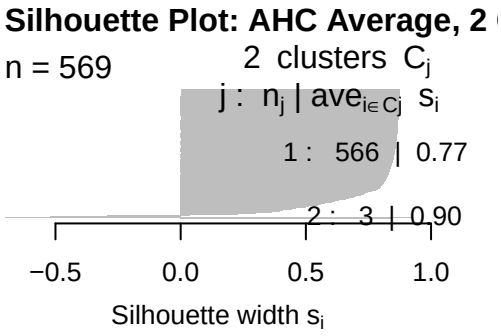




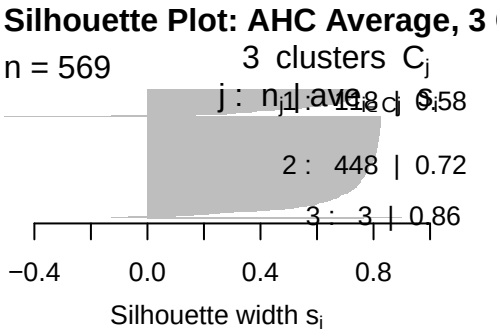




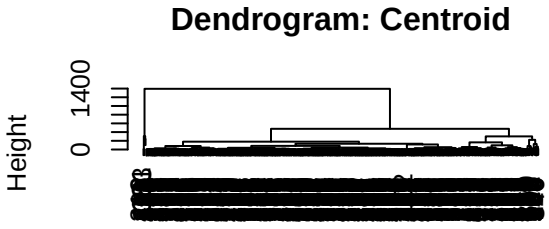

```r
plot(silhouette(hc.a.2,d),main="Silhouette Plot: AHC Average, 2 Clusters")
plot(silhouette(hc.a.3,d),main="Silhouette Plot: AHC Average, 3 Clusters")


#centroid-linkage
hc.cent <- hclust(d,method="centroid")
plot(hc.cent,main="Dendrogram: Centroid",xlab="Euclidean Distance")

hc.cent.2 <- cutree(hc.cent, 2)
hc.cent.3 <- cutree(hc.cent, 3)
clusplot(cancer, hc.cent.2,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Centroid, 2 Clusters
```



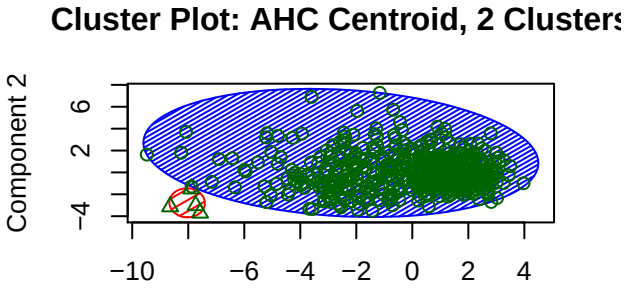
Average silhouette width : 0.77



Average silhouette width : 0.69

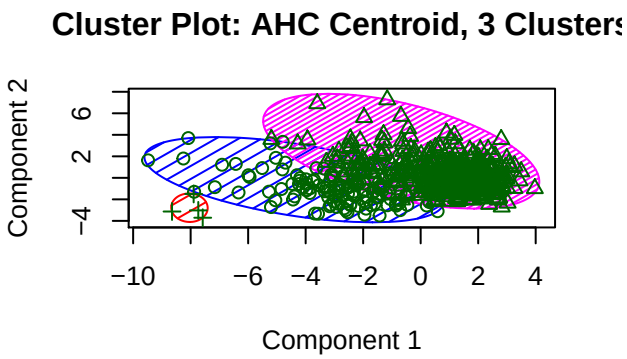


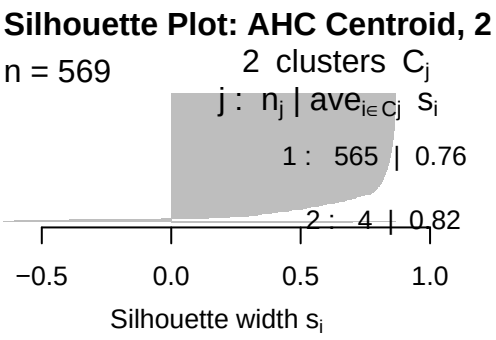
Euclidean Distance
hclust (*, "centroid")

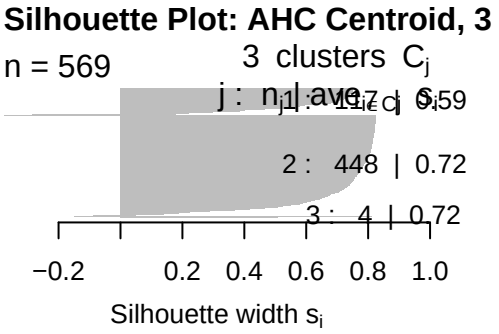


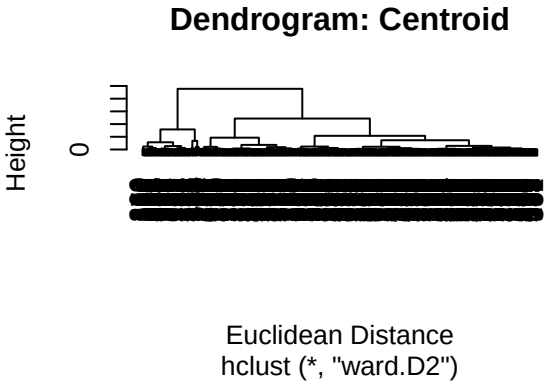
Component 1
These two components explain 79.97 % o

```
clusplot(cancer, hc.cent.3,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Centroid, 3 Clusters
plot(silhouette(hc.cent.2,d),main="Silhouette Plot: AHC Centroid, 2 Clusters")
plot(silhouette(hc.cent.3,d),main="Silhouette Plot: AHC Centroid, 3 Clusters")


#Ward-linkage
hc.w <- hclust(d,method="ward.D2")
plot(hc.w,main="Dendrogram: Centroid",xlab="Euclidean Distance")
```

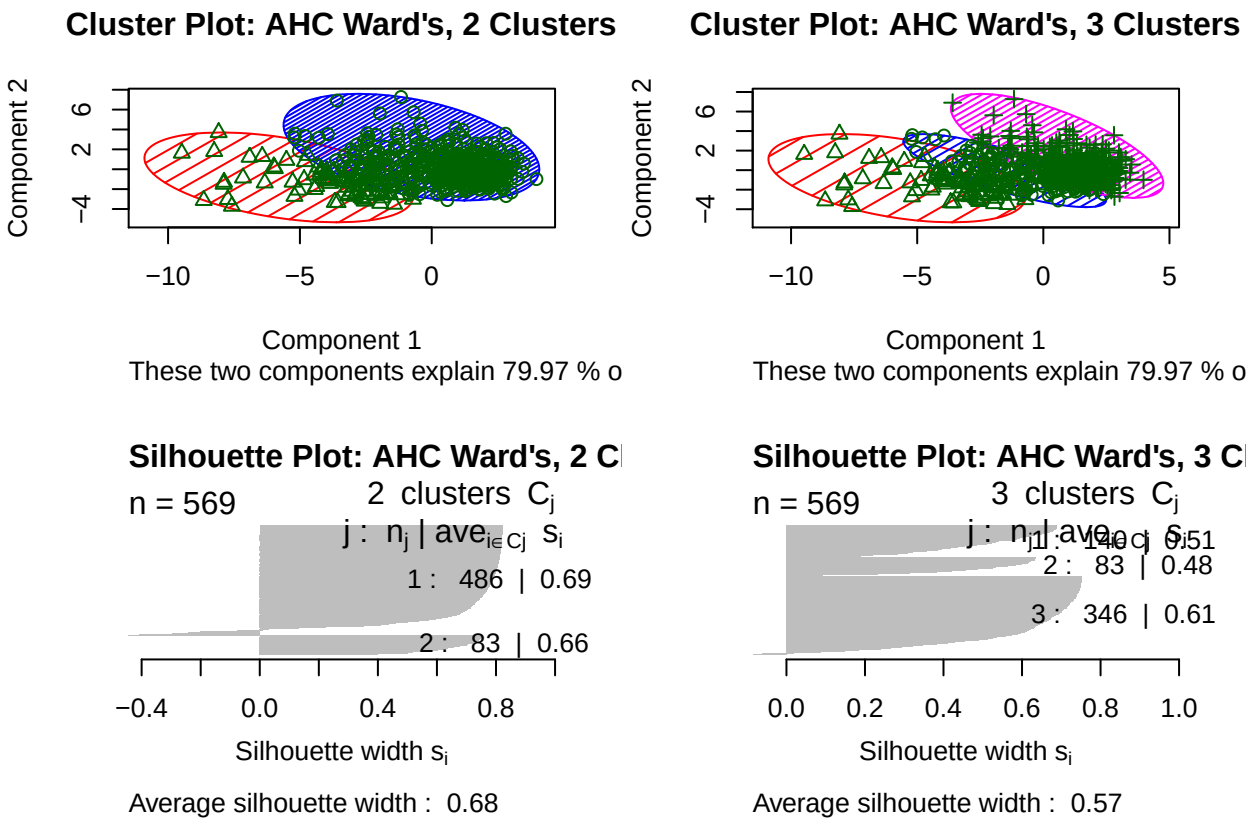









```
hc.w.2 <- cutree(hc.w, 2)
hc.w.3 <- cutree(hc.w, 3)
clusplot(cancer, hc.w.2,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Ward's, 2 Clusters")
clusplot(cancer, hc.w.3,color=TRUE, shade=TRUE, lines=0,main="Cluster Plot: AHC Ward's, 3 Clusters")
plot(silhouette(hc.w.2,d),main="Silhouette Plot: AHC Ward's, 2 Clusters")
plot(silhouette(hc.w.3,d),main="Silhouette Plot: AHC Ward's, 3 Clusters")
```




```
#Pick Optimal # of Clusters
n <- nrow(cancer)
Cstat <- numeric(7)
dstat <- numeric(7)
hc.w <- hclust(d,method="ward.D2")

for(k in 2:7){
  hc.cut <- cutree(hc.w, k)

  fit <- manova(cancer~hc.cut)

    B <- summary(fit)$SS$hc.cut
    W <- summary(fit)$SS$Residuals

    C <- (tr(B)/(k-1))/(tr(W)/(n-k))
  d <- k^2*det(W)
  Cstat[k] <- C
  dstat[k] <- d
}

Cstat
```

```
## [1]     0.000000 1031.812047  119.372362   26.623414   19.537295   11.948207
## [7]     8.653449
```

```
dstat
```

```
## [1] 0.000000e+00 3.603018e+10 1.370742e+11 2.581346e+11 2.682608e+11
```

```
## [6] 6.438436e+11 1.043928e+12
```

```
#Using K-means and the Clustergram to validate optimal # of clusters
source("https://raw.githubusercontent.com/talgalili/R-code-snippets/master/clustergram.r")
clustergram(cancer,1:7,line.width=.3)
```

```
## Loading required package: colorspace
```

```
## Warning: package 'colorspace' was built under R version 3.3.2
```

```
##
## Attaching package: 'colorspace'
```

```
## The following object is masked from 'package:pROC':
##
##     coords
```