# UCI MLREPO CANCER GENE DATA

Read in the data. The load() function reads in a dataset that has 20532 columns and may take some time. You may want to clear your environment (or open a new RStudio window) if you have other work open.

```
> load('geneCancerUCI.RData')
> table(cancerlabels$Class)
```

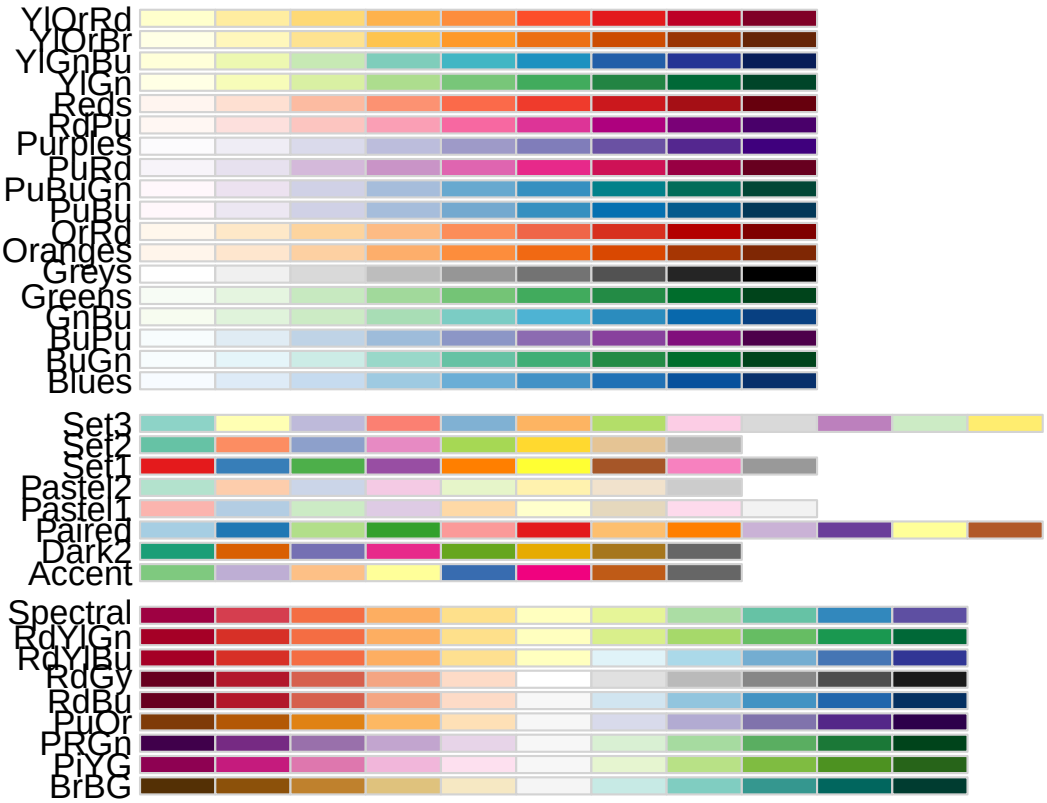
```
BRCA COAD KIRC LUAD PRAD
 300   78  146  141  136
```

Original Source: *The cancer genome atlas pan-cancer analysis project*

- BRCA = Breast Invasive Carcinoma
- COAD = Colon Adenocarcinoma
- KIRC = Kidney Renal clear cell Carcinoma
- LUAD = Lung Adenocarcinoma
- PRAD = Prostate Adenocarcinoma

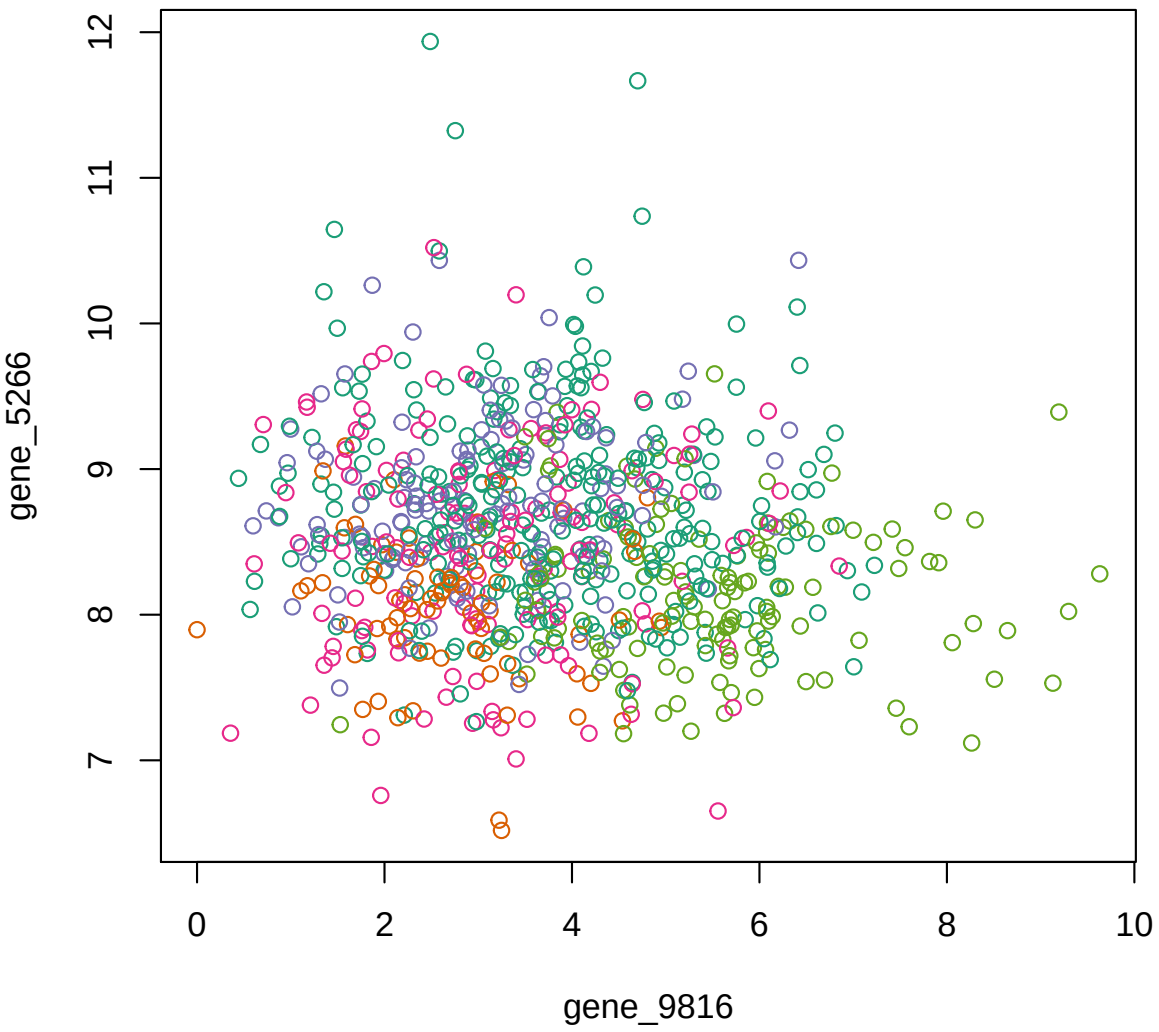
We are going to want to plot the data points according to their different classification labels. We should pick out a nice color palette for categorical attributes.

```
> library(RColorBrewer)
> display.brewer.all()
> palette(brewer.pal(n = 8, name = "Dark2"))
```

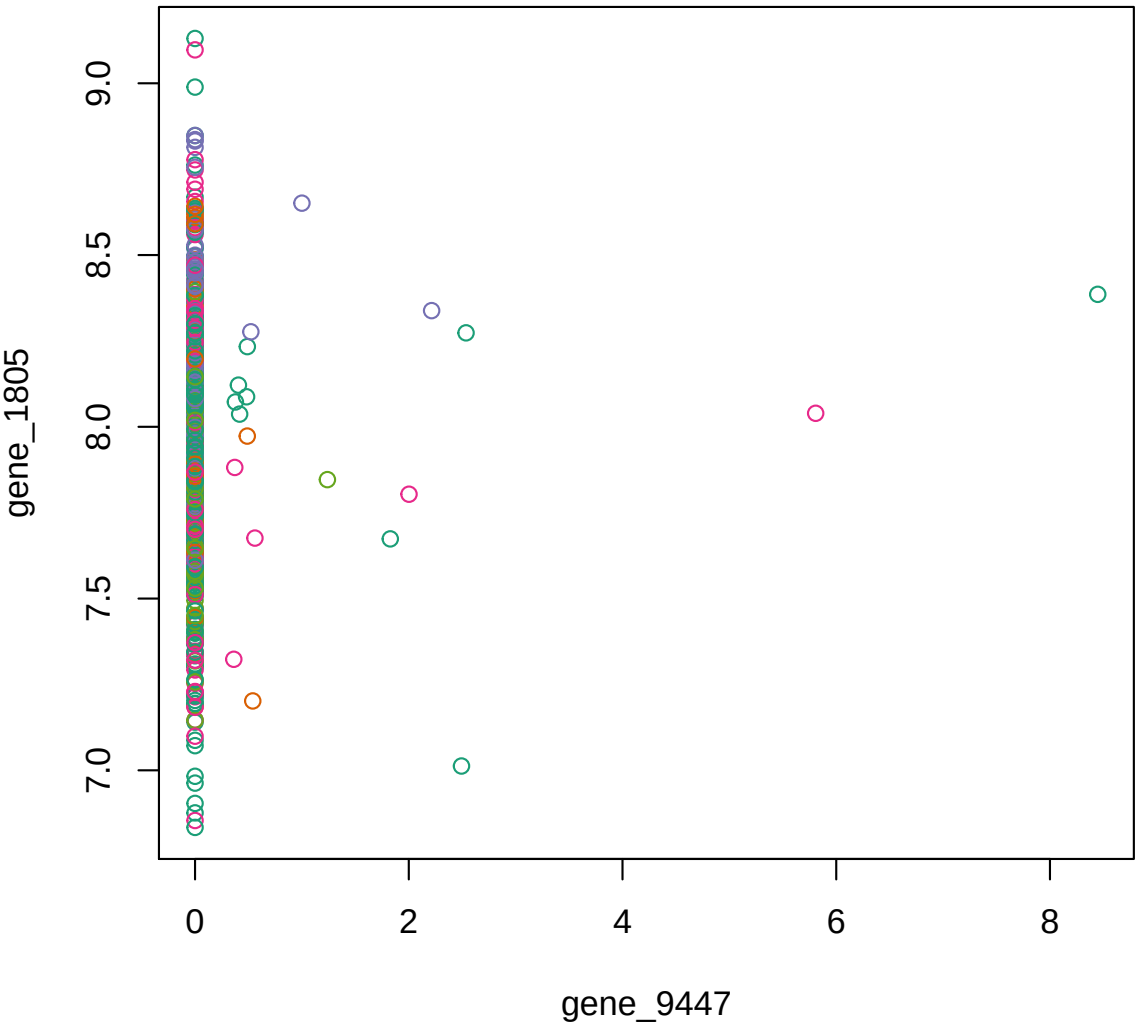



The first step is typically to explore the data. Obviously we can't look at ALL the scatter plots of input variables. For the fun of it, let's look at a few of these scatter plots which we'll pick at random. First pick two column numbers at random, then draw the plot, coloring by the label. Repeat these two lines of code a few times for your own amusement.

```
> randomColumns = sample(2:20532,2)
> plot(cancer[,randomColumns],col = cancerlabels$Class)
```




```
> randomColumns = sample(2:20532,2)
> plot(cancer[,randomColumns],col = cancerlabels$Class)
```




## Computing the PCA

The prcomp() function is the one I most often recommend for reasonably sized principal component calculations in R. This function returns a list with class "prcomp" containing the following components (from help prcomp):

1. sdev: the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix).

2. rotation: the matrix of *variable loadings* (i.e., a matrix whose columns contain the eigenvectors). The function princomp returns this in the element loadings.

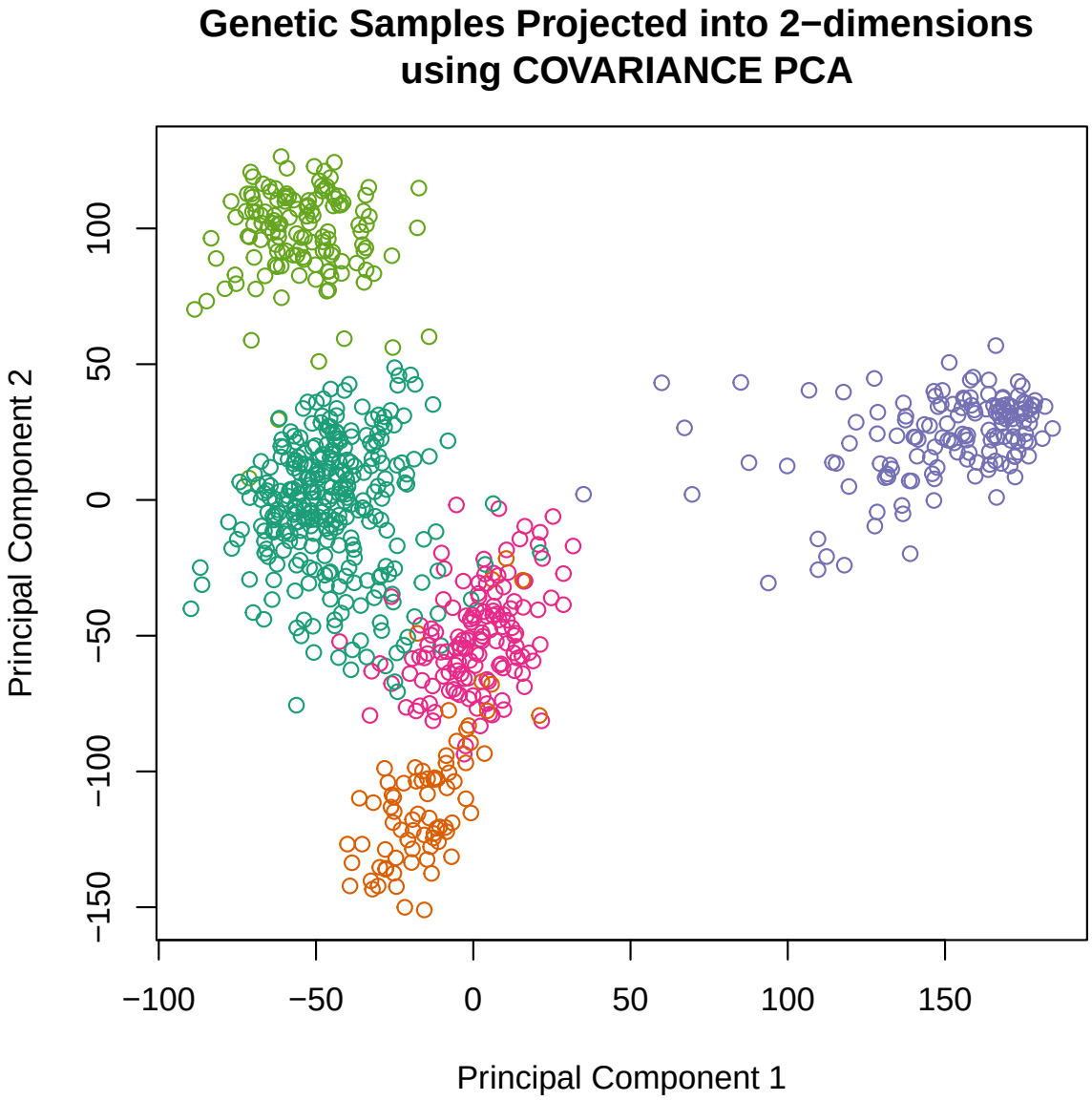
3. x: if retx is true *the value of the rotated data (i.e. the scores)* (the centred (and scaled if requested) data multiplied by the rotation matrix) is returned. Hence, cov(x) is the diagonal matrix $diag(sdev^2)$. For the formula method, napredict() is applied to handle the treatment of values omitted by the na.action.

4. center, scale: the centering and scaling used, or FALSE.

The option scale = TRUE inside the prcomp() function instructs the program to use **correlation PCA**. The **default is covariance PCA**.

Now let's compute the *first three* principal components and examine the data projected onto the first 2 axes. We can then look in 3 dimensions.

```
> pcaOut = prcomp(cancer,rank = 3, scale = F)
> plot(pcaOut$x[,1], pcaOut$x[,2],
+      col = cancerlabels$Class,
+      xlab = "Principal Component 1",
+      ylab = "Principal Component 2",
+      main = 'Genetic Samples Projected into 2-dimensions \n using COVARIANCE PCA')
```



## 3D plot with plotly package

Make sure the plotly package is installed for the 3d plot. To get the plot points colored by group, we need to execute the following command that creates a vector of colors (specifying a color for each observation).

```
> colors = factor(palette())
> colors = colors[cancerlabels$Class]
```

```
> library(plotly)
> graph = plot_ly(x = pcaOut$x[,1],
+                 y = pcaOut$x[,2],
+                 z= pcaOut$x[,3],
+                 type='scatter3d',
+                 mode="markers",
+                 marker = list(color=colors))
> graph
```

## 3D plot with rgl package

Make sure the rgl package is installed for the 3d plot.

```
> library(rgl)
> plot3d(x = pcaOut$x[,1],
+        y = pcaOut$x[,2],
+        z= pcaOut$x[,3],
+        col = colors,
+        xlab = "Principal Component 1",
+        ylab = "Principal Component 2",
+        zlab = "Principal Component 3")
```

## Variance explained

Proportion of Variance explained by 2,3 components:

```
> summary(pcaOut)
```

```
Importance of first k=3 (out of 801) components:
                          PC1     PC2      PC3
Standard deviation     75.7407 61.6805 58.57297
Proportion of Variance  0.1584  0.1050  0.09472
Cumulative Proportion   0.1584  0.2634  0.35815
```

```
> # Alternatively, if you had computed the ALL the principal components (omitted the rank=3 option) then
> # you could directly compute the proportions of variance explained using what we know about the
> # eigenvalues:
>
> # sum(pcaOut$sdev[1:2]^2)/sum(pcaOut$sdev^2)
> # sum(pcaOut$sdev[1:3]^2)/sum(pcaOut$sdev^2)
>
```

## Using Correlation PCA

The data involved in this exercise are actually on the same scale, and normalizing them may not be in your best interest because of this. However, it's always a good idea to explore both decompositions if you have time.

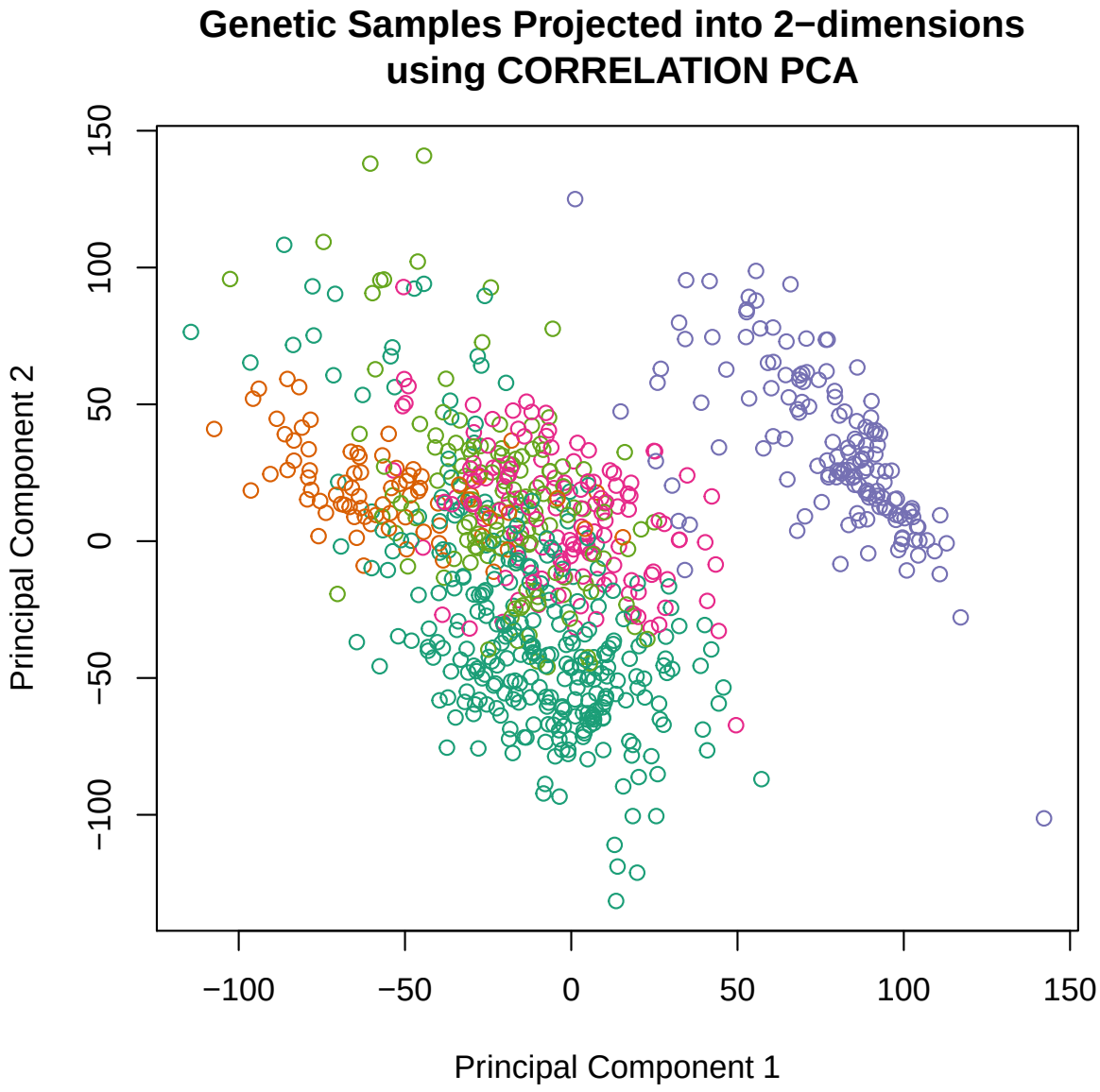
```
> pca = prcomp(cancer, rank=3, scale =T)
```

An error message! Cannot rescale a constant/zero column to unit variance. Solution: check for columns with zero variance and remove them. Recheck dimensions of the matrix to see how many columns we lost.

```
> cancer = cancer[,apply(cancer, 2, sd)>0 ]
> dim(cancer)
```

```
[1]   801 20264
```

```
> pca.cor = prcomp(cancer, rank=3, scale =T)
> plot(pca.cor$x[,1], pca.cor$x[,2],
+      col = cancerlabels$Class,
+      xlab = "Principal Component 1",
+      ylab = "Principal Component 2",
+      main = 'Genetic Samples Projected into 2-dimensions \n using CORRELATION PCA')
```



And it's clear just from the 2-dimensional projection that correlation PCA does not seem to work as well as covariance PCA when it comes to separating the 4 different types of cancer.

Indeed, we can confirm this from the proportion of variance explained, which is substantially lower than that of covariance PCA:

```
> summary(pca.cor)
```

```
Importance of first k=3 (out of 801) components:
                          PC1      PC2     PC3
Standard deviation     46.2145 42.11838 39.7823
Proportion of Variance  0.1054  0.08754  0.0781
Cumulative Proportion   0.1054  0.19294  0.2710
```