Clustering is one of the most common unsupervised machine learning problems. Similarity between observations is defined using some inter-observation distance measures or correlation-based distance measures.

There are 5 classes of clustering methods:

+ Hierarchical Clustering  
+ Partitioning Methods (k-means, PAM, CLARA)  
+ Density-Based Clustering  
+ Model-based Clustering  
+ Fuzzy Clustering

My desire to write this post came mainly from reading about the [**clustree** package](https://github.com/lazappi/clustree), the [**dendextend**](https://cran.r-project.org/web/packages/dendextend/vignettes/Cluster_Analysis.html) documentation, and the [Practical Guide to Cluster Analysis in R](https://www.datanovia.com/en/product/practical-guide-to-cluster-analysis-in-r/) book written by Alboukadel Kassambara author of the [**factoextra**](http://www.sthda.com/english/wiki/r-packages)package.

**Data Set**

I will be using a lesser known data set from the **cluster** package: [all.mammals.milk.1956](https://www.rdocumentation.org/packages/cluster.datasets/versions/1.0-1/topics/all.mammals.milk.1956?tap_a=5644-dce66f&tap_s=10907-287229), one which I haven’t looked at before.

This small dataset contains a list of 25 mammals and the constituents of their milk (water, protein, fat, lactose, ash percentages) from [John Hartigan, Clustering Algorithms, Wiley, 1975](http://people.sc.fsu.edu/~jburkardt/datasets/hartigan/hartigan.html).

First let’s load the required packages.

library(tidyverse)  
library(magrittr)  
library(cluster)  
library(cluster.datasets)  
library(cowplot)  
library(NbClust)  
library(clValid)  
library(ggfortify)  
library(clustree)  
library(dendextend)  
library(factoextra)  
library(FactoMineR)  
library(corrplot)  
library(GGally)  
library(ggiraphExtra)  
library(knitr)  
library(kableExtra)

Now load the data.

data("all.mammals.milk.1956")  
raw\_mammals <- all.mammals.milk.1956

# subset dataset  
mammals <- raw\_mammals %>% select(-name) # set rownames  
mammals <- as\_tibble(mammals)

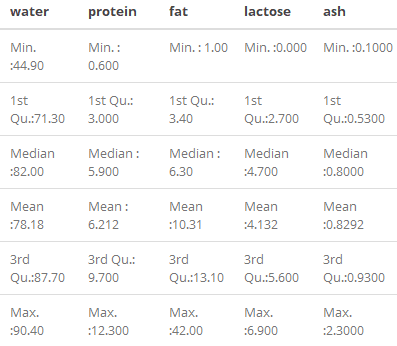
Let’s explore and visualize the data.

# Glimpse the data set  
glimpse(mammals)

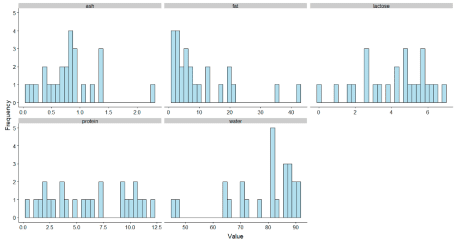
Observations: 25  
Variables: 5  
$ water 90.1, 88.5, 88.4, 90.3, 90.4, 87.7, 86.9, 82.1, 81.9, 81.6, 81.6, 86.5, 90.0,...  
$ protein 2.6, 1.4, 2.2, 1.7, 0.6, 3.5, 4.8, 5.9, 7.4, 10.1, 6.6, 3.9, 2.0, 7.1, 3.0, 5...  
$ fat 1.0, 3.5, 2.7, 1.4, 4.5, 3.4, 1.7, 7.9, 7.2, 6.3, 5.9, 3.2, 1.8, 5.1, 4.8, 6....  
$ lactose 6.9, 6.0, 6.4, 6.2, 4.4, 4.8, 5.7, 4.7, 2.7, 4.4, 4.9, 5.6, 5.5, 3.7, 5.3, 4....  
$ ash 0.35, 0.24, 0.18, 0.40, 0.10, 0.71, 0.90, 0.78, 0.85, 0.75, 0.93, 0.80, 0.47,...

All the variables are expressed as numeric. What about the statistical distribution?

# Summary of data set  
summary(mammals) %>% kable() %>% kable\_styling()

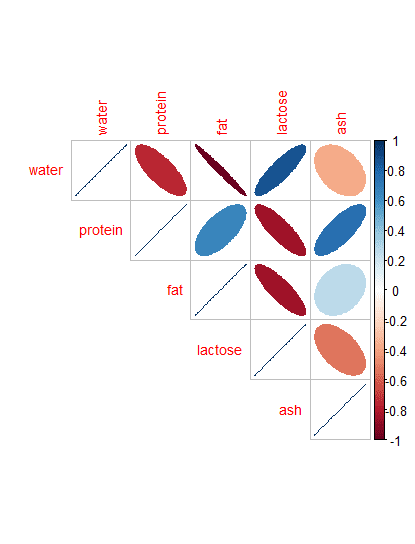


# Historgram for each attribute  
mammals %>%   
 gather(Attributes, value, 1:5) %>%   
 ggplot(aes(x=value)) +  
 geom\_histogram(fill = "lightblue2", color = "black") +   
 facet\_wrap(~Attributes, scales = "free\_x") +  
 labs(x = "Value", y = "Frequency")



What’s the relationship between the different attributes? Use `corrplot()` to create correlation matrix.

corrplot(cor(mammals), type = "upper", method = "ellipse", tl.cex = 0.9)



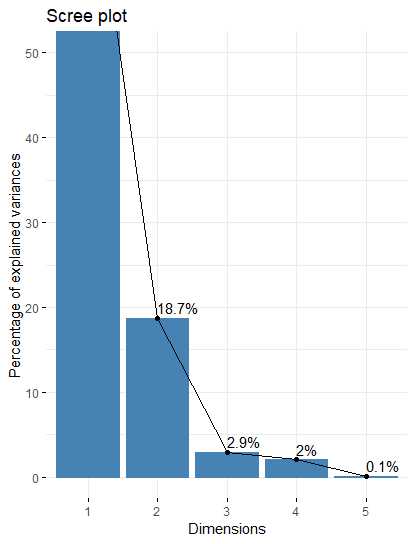
When you have variables which are measured in different scales it is useful to scale the data.

mammals\_scaled <- scale(mammals)  
rownames(mammals\_scaled) <- raw\_mammals$name

Dimensionality reduction can help with data visualization (*e.g.* PCA method).

res.pca <- PCA(mammals\_scaled, graph = FALSE)

# Visualize eigenvalues/variances  
fviz\_screeplot(res.pca, addlabels = TRUE, ylim = c(0, 50))



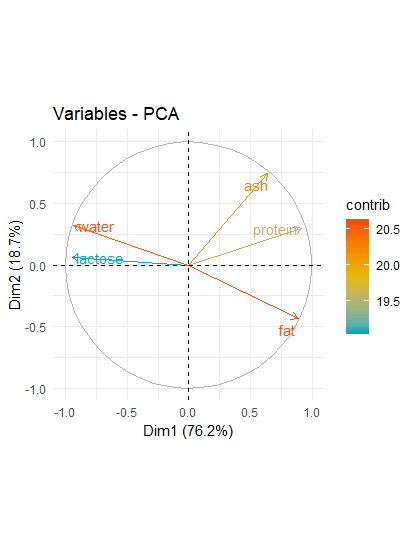
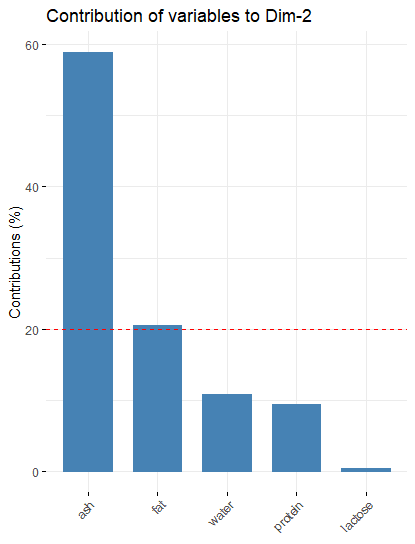
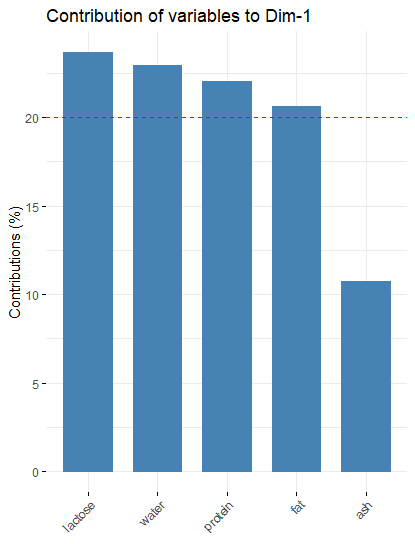
These are the **5 PCs that capture 80% of the variance**. The scree plot shows that **PC1 captured ~ 75% of the variance**.

# Extract the results for variables  
var <- get\_pca\_var(res.pca)

# Contributions of variables to PC1  
fviz\_contrib(res.pca, choice = "var", axes = 1, top = 10)

# Contributions of variables to PC2  
fviz\_contrib(res.pca, choice = "var", axes = 2, top = 10)

# Control variable colors using their contributions to the principle axis  
fviz\_pca\_var(res.pca, col.var="contrib",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Avoid text overlapping  
 ) + theme\_minimal() + ggtitle("Variables - PCA")



From these visualizations it’s apparrent that water and lactose tend to increase together and that protein, ash and fat increase together; the two groups being inversely related.

**Naïve K-means clustering**

Partitioning clustering methods, like k-means and Partitioning Around Medoids (PAM), require that you specify the number of clusters to be generated.

k-means clusters is probably one of the most well known partitioning methods. The idea behind k-means clustering consists of defining clusters the **total within-cluster variation** , which measures the compactness of the clusters is minimized.

We can compute k-means in R with the **kmeans()** function:

km2 <- kmeans(mammals\_scaled, centers = 2, nstart = 30)

The above example would group the data into two clusters, **centers = 2**, and attempt multiple initial configurations, reporting on the best one. For example, as this algorithm is sensitive to the initial positions of the cluster centroids adding **nstart = 30** will generate 30 initial configurations and then average all the centroid results.

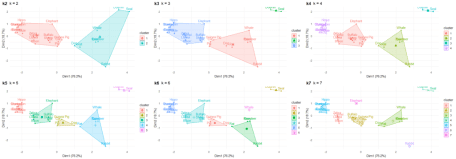
Because the number of clusters (**k**) needs to be set before we start it can be advantageous to examine several different values of **k**.

kmean\_calc <- function(df, ...){  
 kmeans(df, scaled = ..., nstart = 30)  
}

km2 <- kmean\_calc(mammals\_scaled, 2)  
km3 <- kmean\_calc(mammals\_scaled, 3)  
km4 <- kmeans(mammals\_scaled, 4)  
km5 <- kmeans(mammals\_scaled, 5)  
km6 <- kmeans(mammals\_scaled, 6)  
km7 <- kmeans(mammals\_scaled, 7)  
km8 <- kmeans(mammals\_scaled, 8)  
km9 <- kmeans(mammals\_scaled, 9)  
km10 <- kmeans(mammals\_scaled, 10)  
km11 <- kmeans(mammals\_scaled, 11)

p1 <- fviz\_cluster(km2, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 2")   
p2 <- fviz\_cluster(km3, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 3")  
p3 <- fviz\_cluster(km4, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 4")  
p4 <- fviz\_cluster(km5, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 5")  
p5 <- fviz\_cluster(km6, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 6")  
p6 <- fviz\_cluster(km7, data = mammals\_scaled, frame.type = "convex") + theme\_minimal() + ggtitle("k = 7")

plot\_grid(p1, p2, p3, p4, p5, p6, labels = c("k2", "k3", "k4", "k5", "k6", "k7"))



Although this visual assessment tells us where delineations occur between clusters, it does not tell us what the optimal number of clusters is.

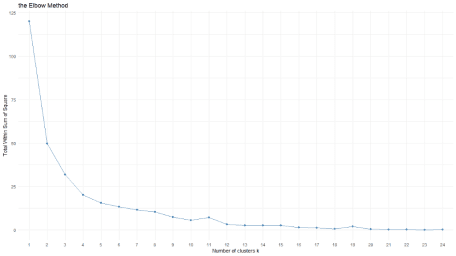
**Determining Optimal Number of Clusters**

A variety of measures have been proposed in the literature for evaluating clustering results. The term **clustering validation** is used to design the procedure of evaluating the results of a clustering algorithm. There are more than thirty indices and methods for identifying the optimal number of clusters so I’ll just focus on a few here including the very neat **clustree** package.

**The “Elbow” Method**

Probably the most well known method, the elbow method, in which the sum of squares at each number of clusters is calculated and graphed, and the user looks for a change of slope from steep to shallow (an elbow) to determine the optimal number of clusters. This method is inexact, but still potentially helpful.

set.seed(31)  
# function to compute total within-cluster sum of squares  
fviz\_nbclust(mammals\_scaled, kmeans, method = "wss", k.max = 24) + theme\_minimal() + ggtitle("the Elbow Method")



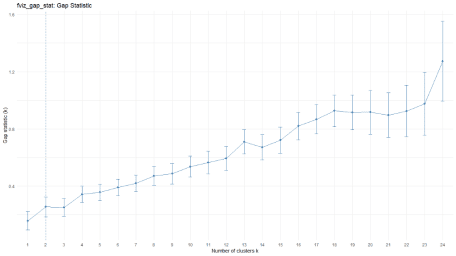
The Elbow Curve method is helpful because it shows how increasing the number of the clusters contribute separating the clusters in a meaningful way, not in a marginal way. The bend indicates that additional clusters beyond the third have little value. (See [[here](http://web.stanford.edu/~hastie/Papers/gap.pdf)] for a more mathematically rigorous interpretation and implementation of this method).

**The Gap Statistic**

The gap statistic compares the total within intra-cluster variation for different values of **k** with their expected values under null reference distribution of the data. The estimate of the optimal clusters will be value that maximize the gap statistic (*i.e.*, that yields the largest gap statistic). This means that the clustering structure is far away from the random uniform distribution of points.

gap\_stat <- clusGap(mammals\_scaled, FUN = kmeans, nstart = 30, K.max = 24, B = 50)

fviz\_gap\_stat(gap\_stat) + theme\_minimal() + ggtitle("fviz\_gap\_stat: Gap Statistic")

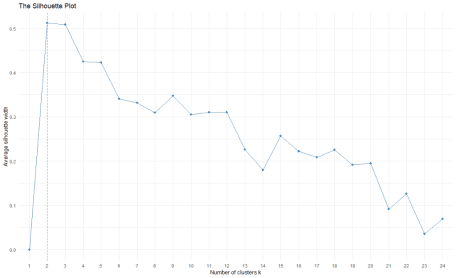


The gap stats plot shows the statistics by number of clusters (**k**) with standard errors drawn with vertical segments and the optimal value of **k** marked with a vertical dashed blue line. According to this observation **k = 2** is the optimal number of clusters in the data.

**The Silhouette Method**

Another visualization that can help determine the optimal number of clusters is called the a silhouette method. Average silhouette method computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for **k**.

fviz\_nbclust(mammals\_scaled, kmeans, method = "silhouette", k.max = 24) + theme\_minimal() + ggtitle("The Silhouette Plot")



This also suggests an optimal of 2 clusters.

**The Sum of Squares Method**

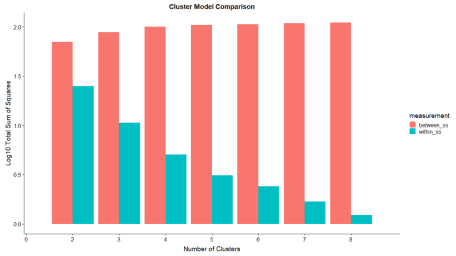
Another clustering validation method would be to choose the optimal number of cluster by minimizing the within-cluster sum of squares (a measure of how tight each cluster is) and maximizing the between-cluster sum of squares (a measure of how seperated each cluster is from the others).

ssc <- data.frame(  
 kmeans = c(2,3,4,5,6,7,8),  
 within\_ss = c(mean(km2$withinss), mean(km3$withinss), mean(km4$withinss), mean(km5$withinss), mean(km6$withinss), mean(km7$withinss), mean(km8$withinss)),  
 between\_ss = c(km2$betweenss, km3$betweenss, km4$betweenss, km5$betweenss, km6$betweenss, km7$betweenss, km8$betweenss)  
)

ssc %<>% gather(., key = "measurement", value = value, -kmeans)

#ssc$value <- log10(ssc$value)

ssc %>% ggplot(., aes(x=kmeans, y=log10(value), fill = measurement)) + geom\_bar(stat = "identity", position = "dodge") + ggtitle("Cluster Model Comparison") + xlab("Number of Clusters") + ylab("Log10 Total Sum of Squares") + scale\_x\_discrete(name = "Number of Clusters", limits = c("0", "2", "3", "4", "5", "6", "7", "8"))



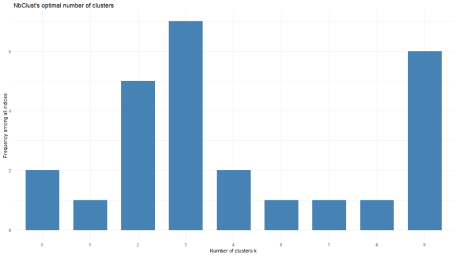
From this measurement it appears 7 clusters would be the appropriate choice.

**NbClust**

The **NbClust** package provides 30 indices for determining the relevant number of clusters and proposes to users the best clustering scheme from the different results obtained by varying all combinations of number of clusters, distance measures, and clustering methods.

res.nbclust <- NbClust(mammals\_scaled, distance = "euclidean",  
 min.nc = 2, max.nc = 9,   
 method = "complete", index ="all")

factoextra::fviz\_nbclust(res.nbclust) + theme\_minimal() + ggtitle("NbClust's optimal number of clusters")



This suggest the optimal number of clusters is 3.

**Clustree**

The statistical method above produce a single score that only considers a single set of clusters at a time. The [**clustree**](https://github.com/lazappi/clustree) R package takes an alternative approach by considering how samples change groupings as the number of clusters increases. This is useful for showing which clusters are distinct and which are unstable. It doesn’t explicitly tell you which choice of *optimal* clusters is but it is useful for exploring possible choices.

Let’s take a look at 1 to 11 clusters.

tmp <- NULL  
for (k in 1:11){  
 tmp[k] <- kmeans(mammals\_scaled, k, nstart = 30)  
}

df <- data.frame(tmp)

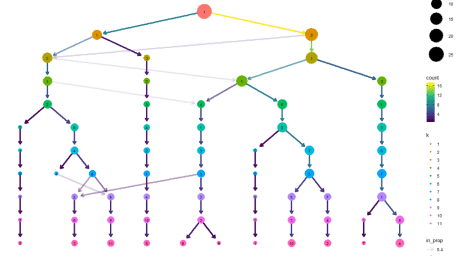
# add a prefix to the column names  
colnames(df) <- seq(1:11)  
colnames(df) <- paste0("k",colnames(df))

# get individual PCA  
df.pca <- prcomp(df, center = TRUE, scale. = FALSE)

ind.coord <- df.pca$x  
ind.coord <- ind.coord[,1:2]

df <- bind\_cols(as.data.frame(df), as.data.frame(ind.coord))

clustree(df, prefix = "k")



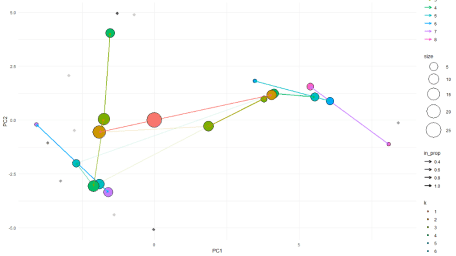
In this figure the size of each node corresponds to the number of samples in each cluster, and the arrows are coloured according to the number of samples each cluster receives. A separate set of arrows, the transparent ones, called the incoming node proportion, are also coloured and shows how samples from one group end up in another group — an indicator of cluster instability.

In this graph we see that as we move from **k=2** to **k=3** a number of species from the lookers-left cluster are reasigned to the third cluster on the right. As we move from **k=8** to **k=9** we see one node with multiple incoming edges an indicator that we over-clustered the data.

It can also be useful to overlay this dimension on other dimensions in the data, particularly those that come from dimensionality reduction techniques. We can do this using the **clustree\_overlay()** function:

df\_subset <- df %>% select(1:8,12:13)

clustree\_overlay(df\_subset, prefix = "k", x\_value = "PC1", y\_value = "PC2")

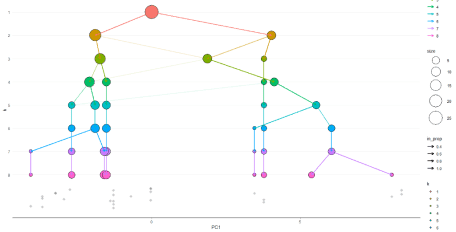
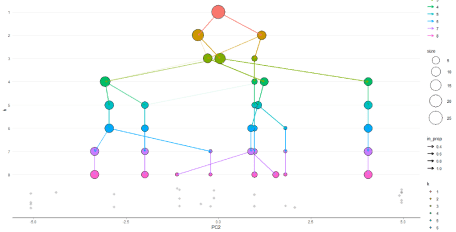


I prefer to see it from the side, showing one of the x or y dimensions against the resolution dimension.

overlay\_list <- clustree\_overlay(df\_subset, prefix = "k", x\_value = "PC1",  
 y\_value = "PC2", plot\_sides = TRUE)

overlay\_list$x\_side

overlay\_list$y\_side



This shows that we can an indication of the correct clustering resolution by examining the edges and we can overly information to assess the quality of the clustering.

**Choosing the appropriate algorithm**

What about choice of appropriate clustering algorithm? The **cValid** package can be used to simultaneously compare multiple clustering algorithms, to identify the best clustering approach and the optimal number of clusters. We will compare k-means, hierarchical and PAM clustering.

intern <- clValid(mammals\_scaled, nClust = 2:24,   
 clMethods = c("hierarchical","kmeans","pam"), validation = "internal")

# Summary  
summary(intern) %>% kable() %>% kable\_styling()

Clustering Methods:  
 hierarchical kmeans pam   
  
Cluster sizes:  
 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   
  
Validation Measures:  
 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
   
hierarchical Connectivity 4.1829 10.5746 13.2579 20.1579 22.8508 25.8258 32.6270 35.3032 38.2905 39.2405 41.2405 45.7742 47.2742 50.6075 52.6075 55.8575 58.7242 60.7242 63.2242 65.2242 67.2242 69.2242 71.2242  
 Dunn 0.3595 0.3086 0.3282 0.2978 0.3430 0.3430 0.4390 0.4390 0.5804 0.5938 0.5938 0.8497 0.8497 0.5848 0.5848 0.4926 0.9138 0.9138 0.8892 0.9049 0.9335 1.0558 2.1253  
 Silhouette 0.5098 0.5091 0.4592 0.4077 0.4077 0.3664 0.3484 0.4060 0.3801 0.3749 0.3322 0.3646 0.3418 0.2650 0.2317 0.2166 0.2469 0.2213 0.1659 0.1207 0.1050 0.0832 0.0691  
kmeans Connectivity 7.2385 10.5746 15.8159 20.1579 22.8508 25.8258 33.5198 35.3032 38.2905 39.2405 41.2405 45.7742 47.2742 51.8909 53.8909 57.1409 58.7242 60.7242 63.2242 65.2242 67.2242 69.2242 71.2242  
 Dunn 0.2070 0.3086 0.2884 0.2978 0.3430 0.3430 0.3861 0.4390 0.5804 0.5938 0.5938 0.8497 0.8497 0.5866 0.5866 0.5725 0.9138 0.9138 0.8892 0.9049 0.9335 1.0558 2.1253  
 Silhouette 0.5122 0.5091 0.4260 0.4077 0.4077 0.3664 0.3676 0.4060 0.3801 0.3749 0.3322 0.3646 0.3418 0.2811 0.2478 0.2402 0.2469 0.2213 0.1659 0.1207 0.1050 0.0832 0.0691  
pam Connectivity 7.2385 14.1385 17.4746 24.0024 26.6857 32.0413 33.8913 36.0579 38.6607 40.6607 42.7869 45.7742 47.2742 51.7242 53.7242 56.9742 58.7242 60.7242 62.7242 64.7242 66.7242 69.2242 71.2242  
 Dunn 0.2070 0.1462 0.2180 0.2180 0.2978 0.2980 0.4390 0.4390 0.4390 0.4390 0.4390 0.8497 0.8497 0.5314 0.5314 0.4782 0.9138 0.9138 0.8333 0.8189 0.7937 1.0558 2.1253  
 Silhouette 0.5122 0.3716 0.4250 0.3581 0.3587 0.3318 0.3606 0.3592 0.3664 0.3237 0.3665 0.3646 0.3418 0.2830 0.2497 0.2389 0.2469 0.2213 0.1758 0.1598 0.1380 0.0832 0.0691  
  
Optimal Scores:  
  
 Score Method Clusters  
Connectivity 4.1829 hierarchical 2   
Dunn 2.1253 hierarchical 24   
Silhouette 0.5122 kmeans 2

Connectivity and Silhouette are both measurements of connectedness while the Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance.

**Extracting Features of Clusters**

We would like to answer questions like “what is it that makes this cluster unique from others?” and “what are the clusters that are similar to one another”

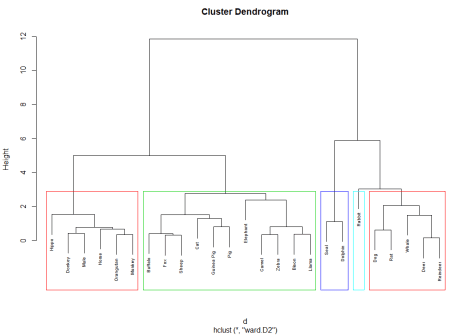
As mentioned earlier it’s difficult to assess the quality of results from clustering. We don’t have true labels so clustering is a good EDA starting point for exploring the differences between these clusters in greater detail. Let’s select five clusters and interrogate the features of these clusters.

# Compute dissimilarity matrix with euclidean distances  
d <- dist(mammals\_scaled, method = "euclidean")

# Hierarchical clustering using Ward's method  
res.hc <- hclust(d, method = "ward.D2" )

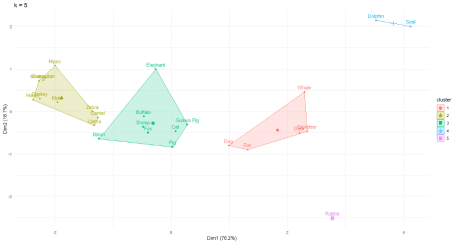
# Cut tree into 5 groups  
grp <- cutree(res.hc, k = 5)

# Visualize  
plot(res.hc, cex = 0.6) # plot tree  
rect.hclust(res.hc, k = 5, border = 2:5) # add rectangle



# Execution of k-means with k=5  
final <- kmeans(mammals\_scaled, 5, nstart = 30)

fviz\_cluster(final, data = mammals\_scaled) + theme\_minimal() + ggtitle("k = 5")



Let’s extract the clusters and add them back to our initial data to do some descriptive statistics at the cluster level:

as.data.frame(mammals\_scaled) %>% mutate(Cluster = final$cluster) %>% group\_by(Cluster) %>% summarise\_all("mean") %>% kable() %>% kable\_styling()



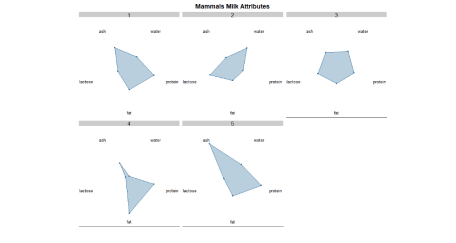
We see that cluster 2, composed solely of the Rabbit has a high ash content. Group 3 composed of the seal and dolphin are high in fat, which makes sense given the harsh demands of such a cold climate whil group 4 has a large lactose content.

mammals\_df <- as.data.frame(mammals\_scaled) %>% rownames\_to\_column()

cluster\_pos <- as.data.frame(final$cluster) %>% rownames\_to\_column()  
colnames(cluster\_pos) <- c("rowname", "cluster")

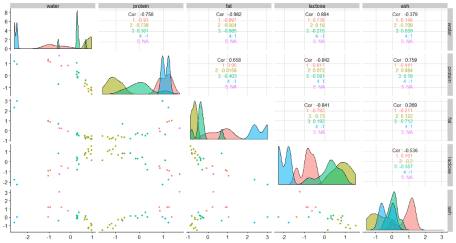
mammals\_final <- inner\_join(cluster\_pos, mammals\_df)

ggRadar(mammals\_final[-1], aes(group = cluster), rescale = FALSE, legend.position = "none", size = 1, interactive = FALSE, use.label = TRUE) + facet\_wrap(~cluster) + scale\_y\_discrete(breaks = NULL) + # don't show ticks  
theme(axis.text.x = element\_text(size = 10)) + scale\_fill\_manual(values = rep("#1c6193", nrow(mammals\_final))) +  
scale\_color\_manual(values = rep("#1c6193", nrow(mammals\_final))) +  
ggtitle("Mammals Milk Attributes")



mammals\_df <- as.data.frame(mammals\_scaled)  
mammals\_df$cluster <- final$cluster  
mammals\_df$cluster <- as.character(mammals\_df$cluster)

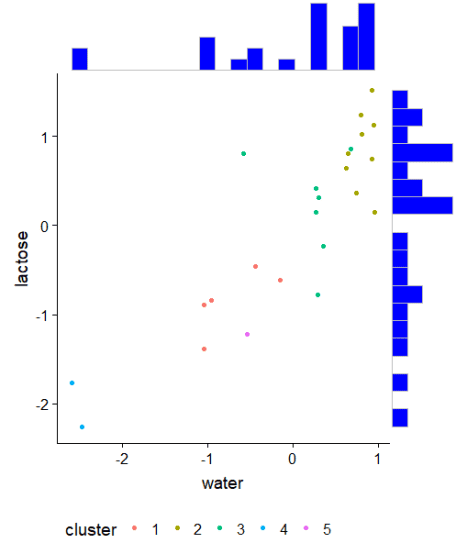
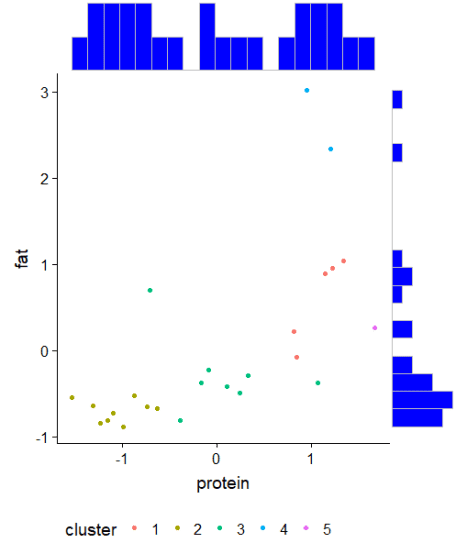
ggpairs(mammals\_df, 1:5, mapping = ggplot2::aes(color = cluster, alpha = 0.5),   
 diag = list(continuous = wrap("densityDiag")),   
 lower=list(continuous = wrap("points", alpha=0.9)))



# plot specific graphs from previous matrix with scatterplot

g <- ggplot(mammals\_df, aes(x = water, y = lactose, color = cluster)) +  
 geom\_point() +  
 theme(legend.position = "bottom")  
ggExtra::ggMarginal(g, type = "histogram", bins = 20, color = "grey", fill = "blue")

b <- ggplot(mammals\_df, aes(x = protein, y = fat, color = cluster)) +  
 geom\_point() +  
 theme(legend.position = "bottom")  
ggExtra::ggMarginal(b, type = "histogram", bins = 20, color = "grey", fill = "blue")



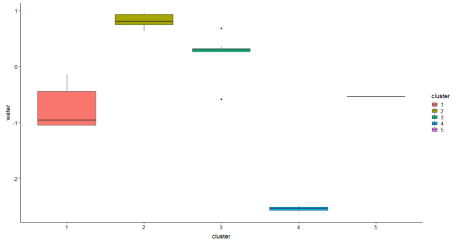
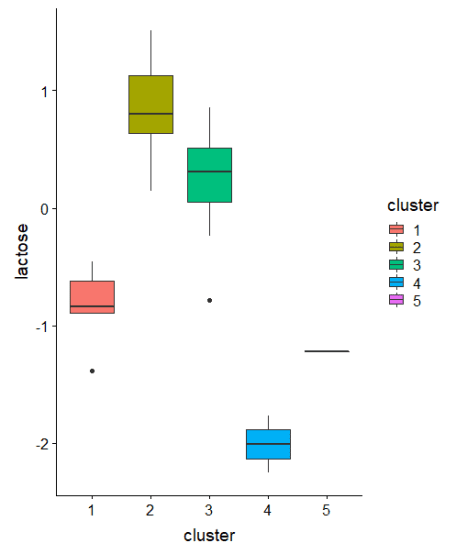
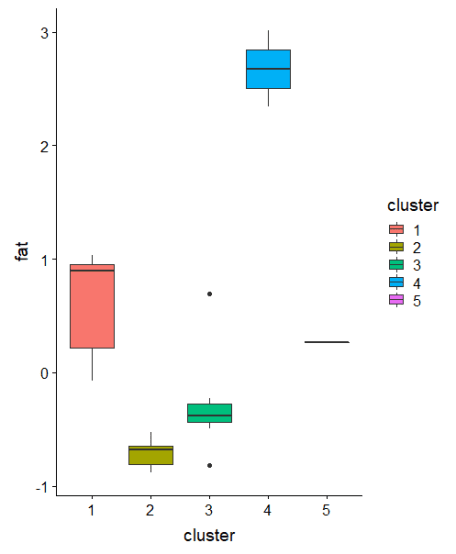
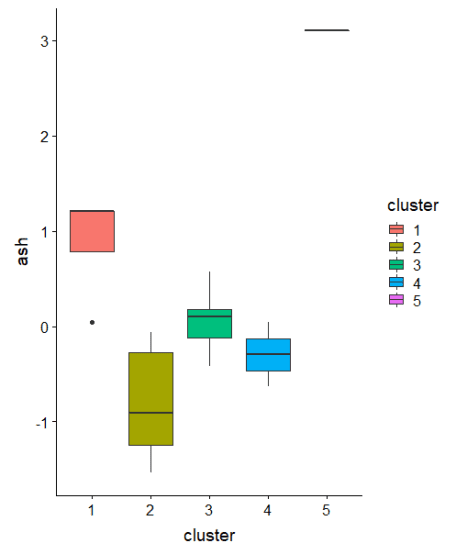
ggplot(mammals\_df, aes(x = cluster, y = protein)) +   
 geom\_boxplot(aes(fill = cluster))

ggplot(mammals\_df, aes(x = cluster, y = fat)) +   
 geom\_boxplot(aes(fill = cluster))

ggplot(mammals\_df, aes(x = cluster, y = lactose)) +   
 geom\_boxplot(aes(fill = cluster))

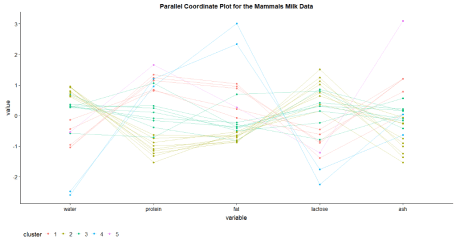
ggplot(mammals\_df, aes(x = cluster, y = ash)) +   
 geom\_boxplot(aes(fill = cluster))

ggplot(mammals\_df, aes(x = cluster, y = water)) +   
 geom\_boxplot(aes(fill = cluster))



# Parallel coordiante plots allow us to put each feature on seperate column and lines connecting each column

ggparcoord(data = mammals\_df, columns = 1:5, groupColumn = 6, alphaLines = 0.4, title = "Parallel Coordinate Plot for the Mammals Milk Data", scale = "globalminmax", showPoints = TRUE) + theme(legend.position = "bottom")



If you find this article useful feel free to share it with others or recommend this article!