Clustering Full

Isaac Baguisa

2025-03-31

K-means

- Run K-means on each dataset with k = number of types to determine if type can be recovered
- Tune for the optimal number of clusters
- Try K-means++ and weighted kmeans, and compare with standard kmeans
- Clustering results – visualization with all types and optimal k
- Compare accuracy with mode

Load libraries and setup

```
library(cluster)
library(factoextra)
library(caret)
library(ggplot2)
library(ggfortify)
library(tidyverse)
library(VIM)
library(gridExtra)
load("../Data/pokemon.RData")
load("../Data/dr_pokemon2.RData")
load("../DimensionReduction/umap_pokemon.rds")

stats_numeric <- stats %>%
  select(-c(abilities, capture_rate, classfication, japanese_name, name,
            name.simple, type1, type2, image_path, has_img))

#Imputation on missing data using KNN
stats_numeric_impKNN <- kNN(stats_numeric)
#Remove cols with variance = 0
# Check the variance of each column
variances <- apply(stats_numeric_impKNN, 2, var)
zero_var_col <- which(variances == 0)
stats_numeric_impKNN <- stats_numeric_impKNN[, -zero_var_col]
```

Helper functions

```
# Helper function lecture 7
scatterplot = function(X, M, cluster, label = FALSE){
  X_df <- data.frame(X, cluster = as.factor(cluster))
  M_df <- data.frame(M)
```

```r
  if (length(unique(cluster)) == 1) {
    plt <- ggplot(X_df, aes(x = PC1, y = PC2)) +
      geom_point() +
      geom_point(data = M_df, aes(x = PC1, y = PC2), shape = 4, size = 4, color = "red") +
      labs(title = "Scatterplot of Pokemon Clusters")

    if (label) {
      plt <- plt + geom_text(aes(label = stats$name), nudge_x = 0.1, size = 3)
    }
    return(plt)
  }
  else {
    ggplot(X_df, aes(x = PC1, y = PC2, color = cluster)) +
      geom_point(alpha = 0.7) +
      geom_point(data = M_df, aes(x = PC1, y = PC2), shape = 4, size = 4, color = "black") +
      scale_color_manual(values = rainbow(length(unique(cluster)))) +
      theme_minimal() +
      labs(title = "K-Means Clustering of Pokemon (PC1 vs PC2)", x = "PC1", y = "PC2") +
      theme(legend.position = "right")
  }
}
#Reference lecture 7 slide 10, and modifications from hw 3
weighted_kmeans <- function(X, K) {
  X <- as.matrix(X)
  n <- nrow(X)

  # Step 1: Initialize first centroid
  set.seed(2201)
  centroids <- X[sample(1:n, 1), , drop = FALSE]

  # Step 2: Compute the initial weights distance
  distances <- as.matrix(dist(rbind(centroids, X)))[2:(n+1), 1]
  weights <- distances^2 / sum(distances^2)

  # Step 3: Initialize the remaining K-1 centroids
  for (i in 2:K) {
    P <- weights / sum(weights)
    pick <- sample(1:n, 1, prob = P)
    new_centroid <- X[pick, , drop = FALSE]
    centroids <- rbind(centroids, new_centroid)

    # Compute new weights based on the distance from the new centroid
    new_distances <- as.matrix(dist(rbind(new_centroid, X)))[2:(n+1), 1]
    weights <- new_distances^2 / sum(new_distances^2)
  }

  converged <- FALSE
  clusters <- rep(0, n)
  while (!converged) {
    # Assign points to the nearest centroid based on weighted distances
    distances <- as.matrix(dist(rbind(centroids, X)))
    distances <- distances[1:K, (K+1):(n+K)]
```

```
    # Weight the distances using the computed weights (soft clustering)
    weighted_distances <- distances * weights

    # Assign each point to the nearest centroid based on weighted distances
    new_clusters <- apply(weighted_distances, 2, which.min)

    # Update centroids using weighted averages
    new_centroids <- matrix(NA, ncol = ncol(X), nrow = K)
    for (k in 1:K) {
      cluster_points <- X[new_clusters == k, ]
      cluster_weights <- weights[new_clusters == k]
      if (length(cluster_points) > 0) {
        new_centroids[k, ] <- colSums(cluster_points * cluster_weights) / sum(cluster_weights)
      }
    }

    # Check for convergence (if centroids don't change)
    if (all(centroids == new_centroids)) {
      converged <- TRUE
    } else {
      centroids <- new_centroids
      clusters <- new_clusters
    }
  }


  return(list(centroids = centroids, clusters = clusters))
}

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

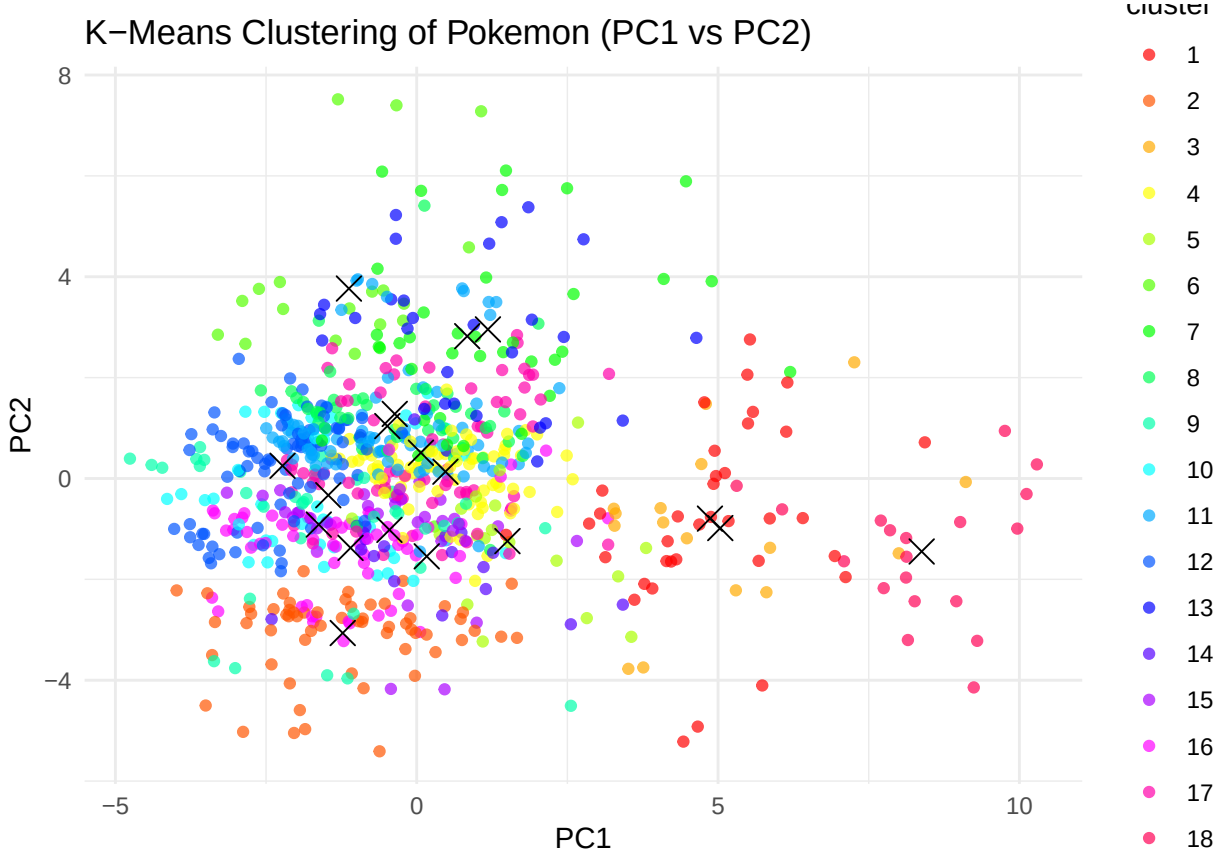
Clustering on Stats Dataset

PCA on stats

```
pca_stats <- prcomp(stats_numeric_impKNN, center = TRUE, scale. = TRUE)
pca_stats_df <- as.data.frame(pca_stats$x)
colnames(pca_stats_df) <- paste0("PC", 1:ncol(pca_stats_df))
var_explained <- summary(pca_stats)$importance[2, ]  # Proportion of variance explained
cumulative_var <- cumsum(var_explained)
# Keep 20 PCs (90% VE)
pca_stats_df <- pca_stats_df[,1:20]
```

Kmeans on stats k = 18

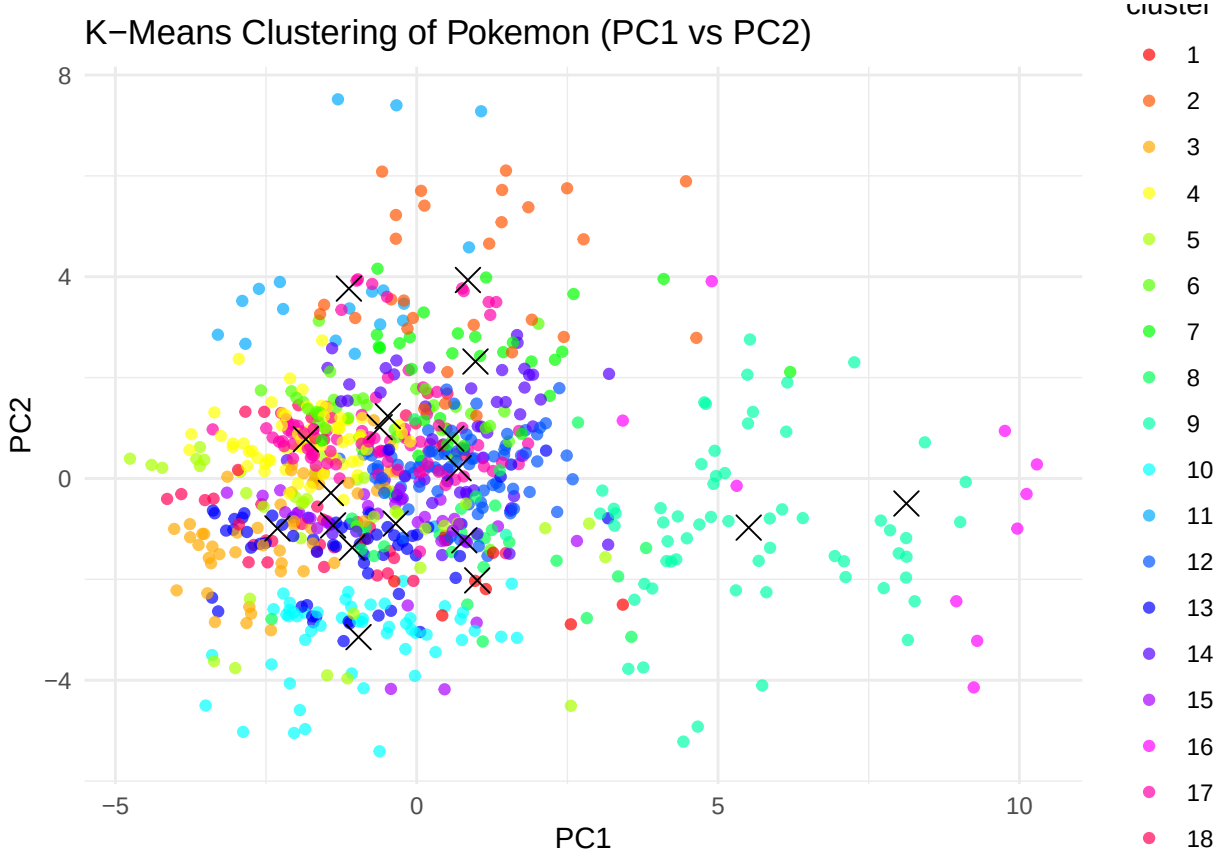
```
k_types <- length(unique(stats$type1))
kmeans_stats <- kmeans(pca_stats_df, centers = k_types)
scatterplot(pca_stats_df, kmeans_stats$centers, kmeans_stats$cluster)
```



Comparing standard kmeans with kmeans++ and weighted kmeans

Kmeans++

```
#Ref: lecture 7
# Randomly select the first centroid
n <- nrow(pca_stats_df)
M <- pca_stats_df[sample(1:n, 1), , drop = FALSE]

for (i in 2:k_types) {
  # Dist from each point to the nearest centroid
  D <- as.matrix(dist(rbind(M, pca_stats_df)))[2:(n+1), 1]

  # Probability for each point to be chosen as the next centroid
  P <- D^2 / sum(D^2)

  # Select the next centroid based on P
  pick <- sample(1:n, 1, prob = P)
  M <- rbind(M, pca_stats_df[pick, , drop = FALSE])
}

kmeans_stats_pp <- kmeans(pca_stats_df, centers = M, algorithm = "Lloyd")
scatterplot(pca_stats_df, kmeans_stats_pp$centers, kmeans_stats_pp$cluster)
```




```
#Lower withinss is better
kmeans_stats$tot.withinss
```

```
## [1] 12613.8
```

```
kmeans_stats_pp$tot.withinss
```

```
## [1] 12848.89
```

```
#Higher betweenss is better
kmeans_stats$betweenss
```

```
## [1] 14360.22
```

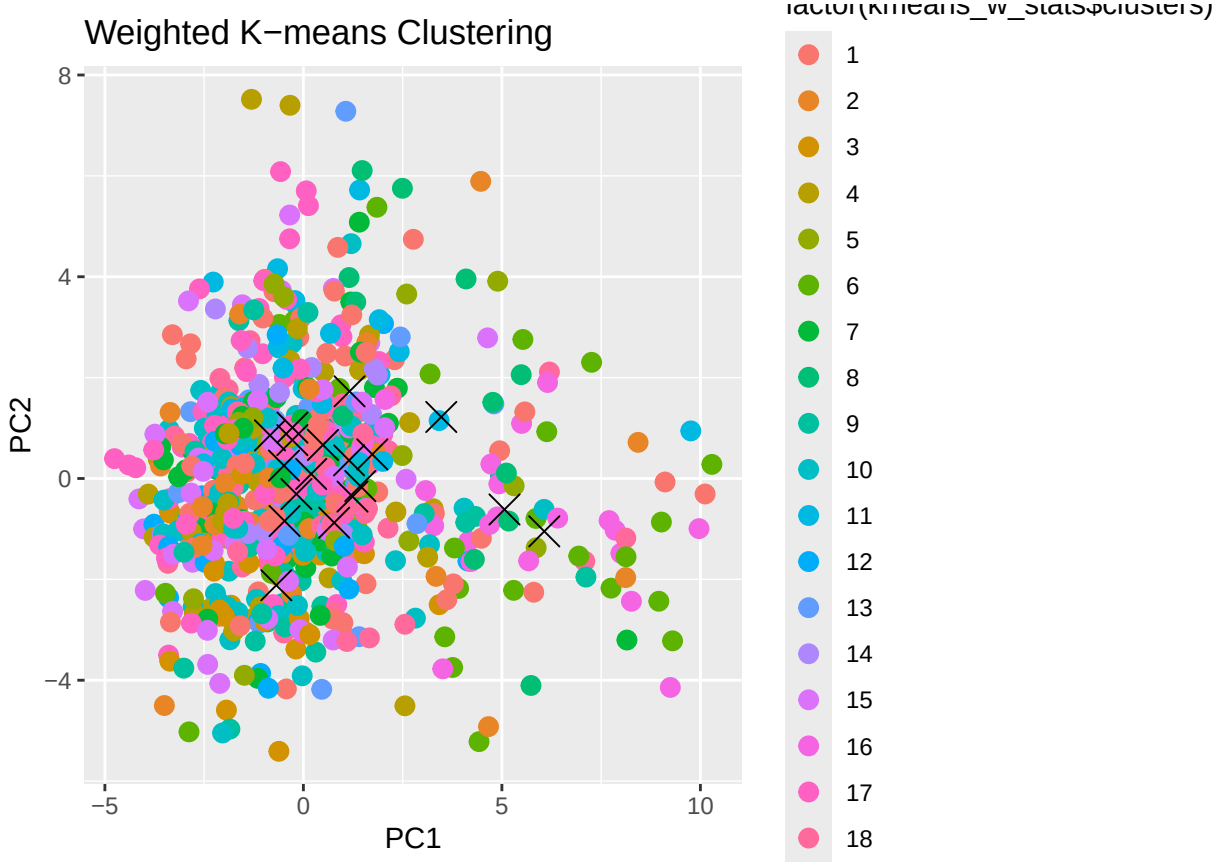
```
kmeans_stats_pp$betweenss
```

```
## [1] 14125.13
```

```
#Standard kmeans outperforms kmeans++
```

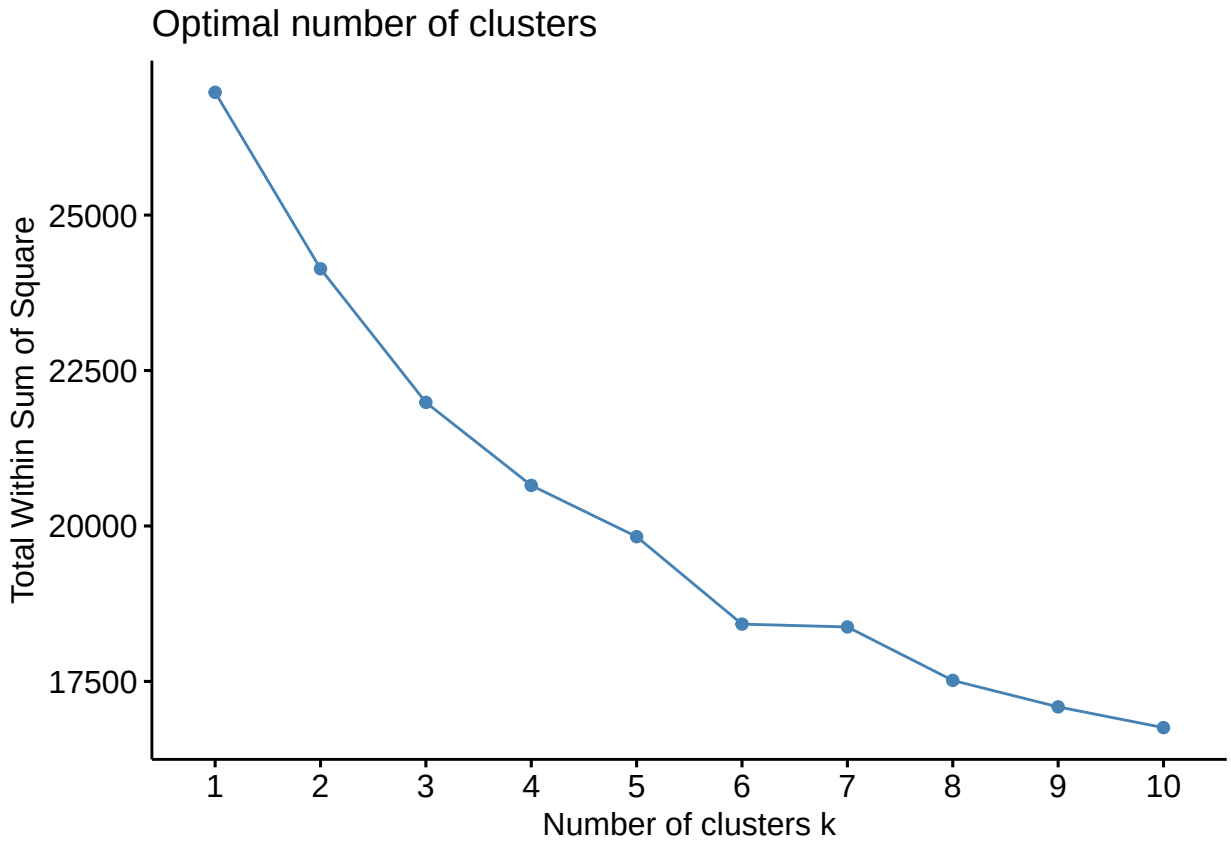
Weighted kmeans

```
K <- 18
kmeans_w_stats <- weighted_kmeans(pca_stats_df, K)
colnames(kmeans_w_stats$centroids) <- colnames(pca_stats_df)

ggplot(data.frame(pca_stats_df), aes(PC1, PC2, color = factor(kmeans_w_stats$clusters))) +
  geom_point(size = 3) +
  geom_point(data = data.frame(kmeans_w_stats$centroids), aes(PC1, PC2),
             color = "black", shape = 4, size = 5) +
  labs(title = "Weighted K-means Clustering", x = "PC1", y = "PC2")
```

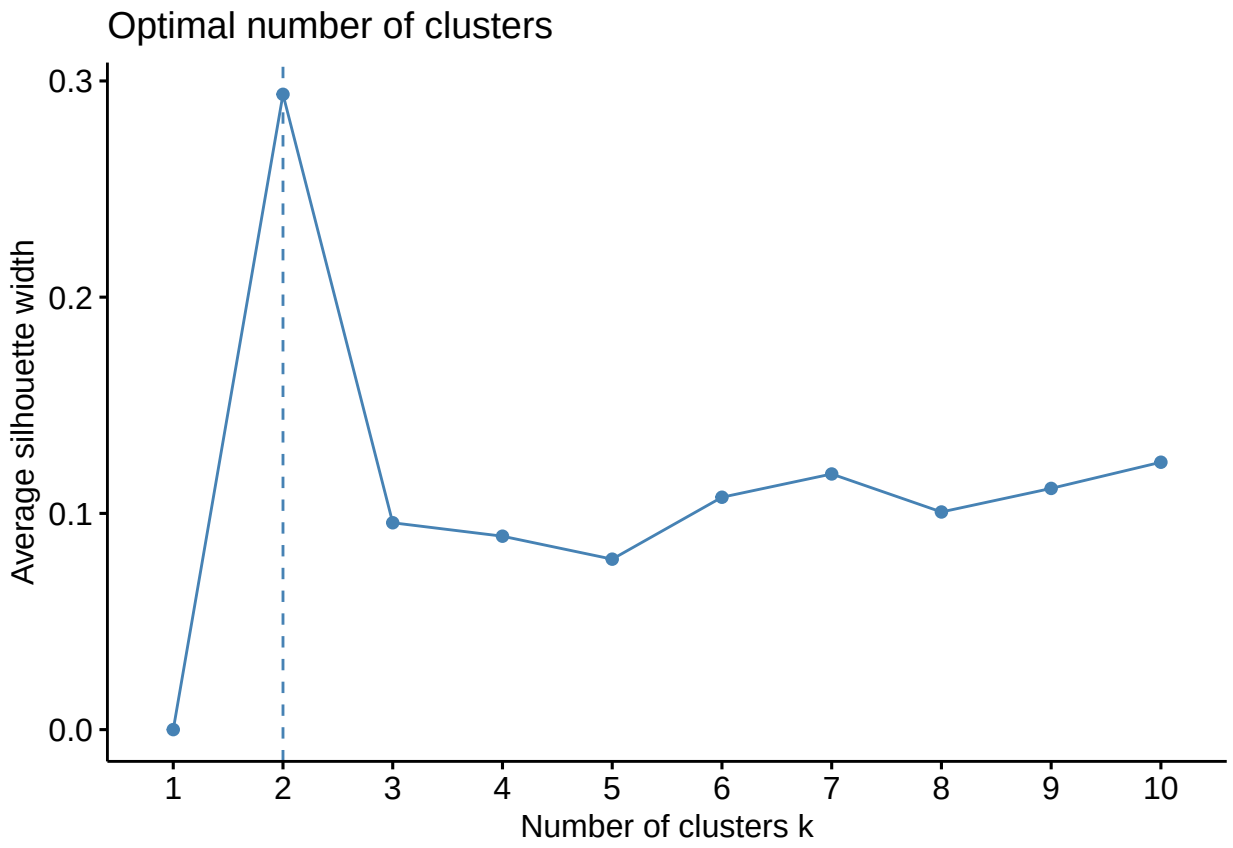


Tune for optimal K

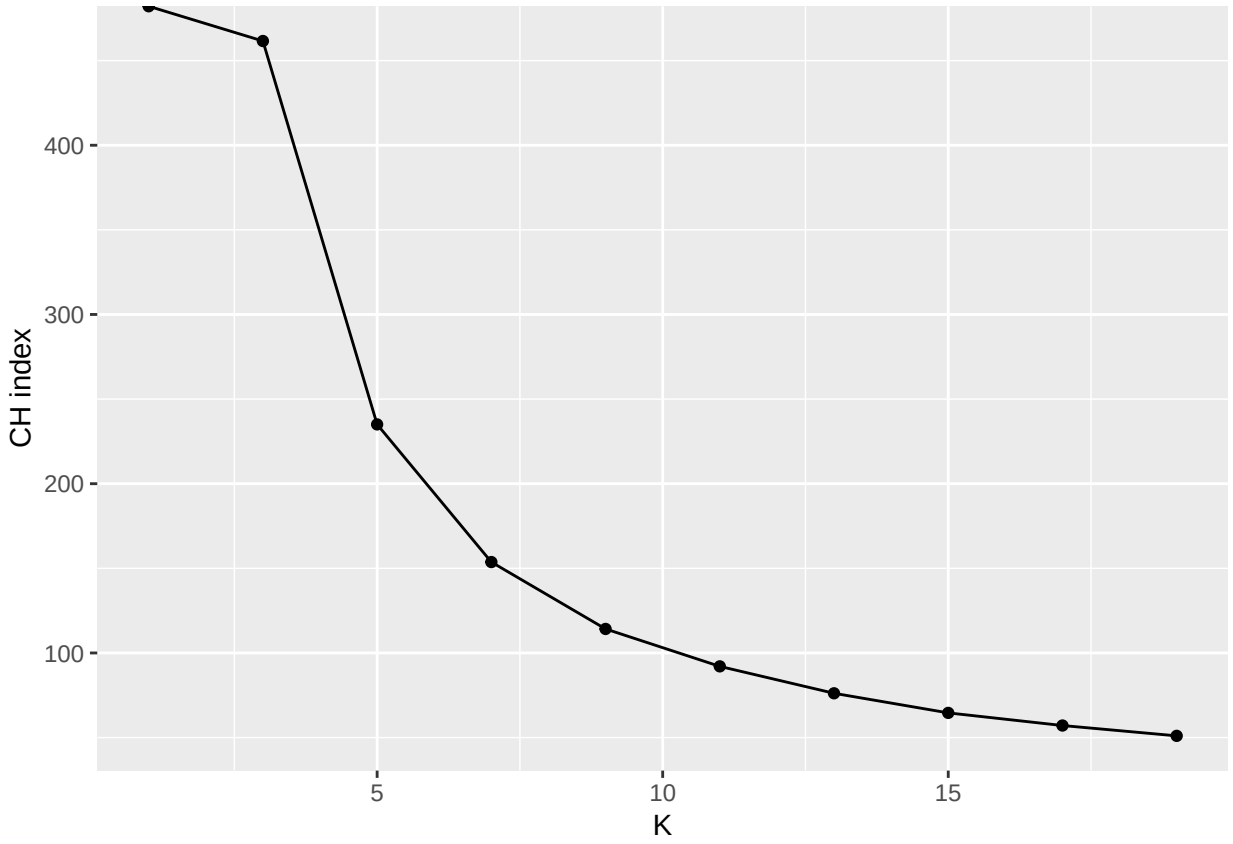
```
# Elbow method
fviz_nbclust(pca_stats_df, kmeans, method = "wss", algorithm = "Lloyd")
```



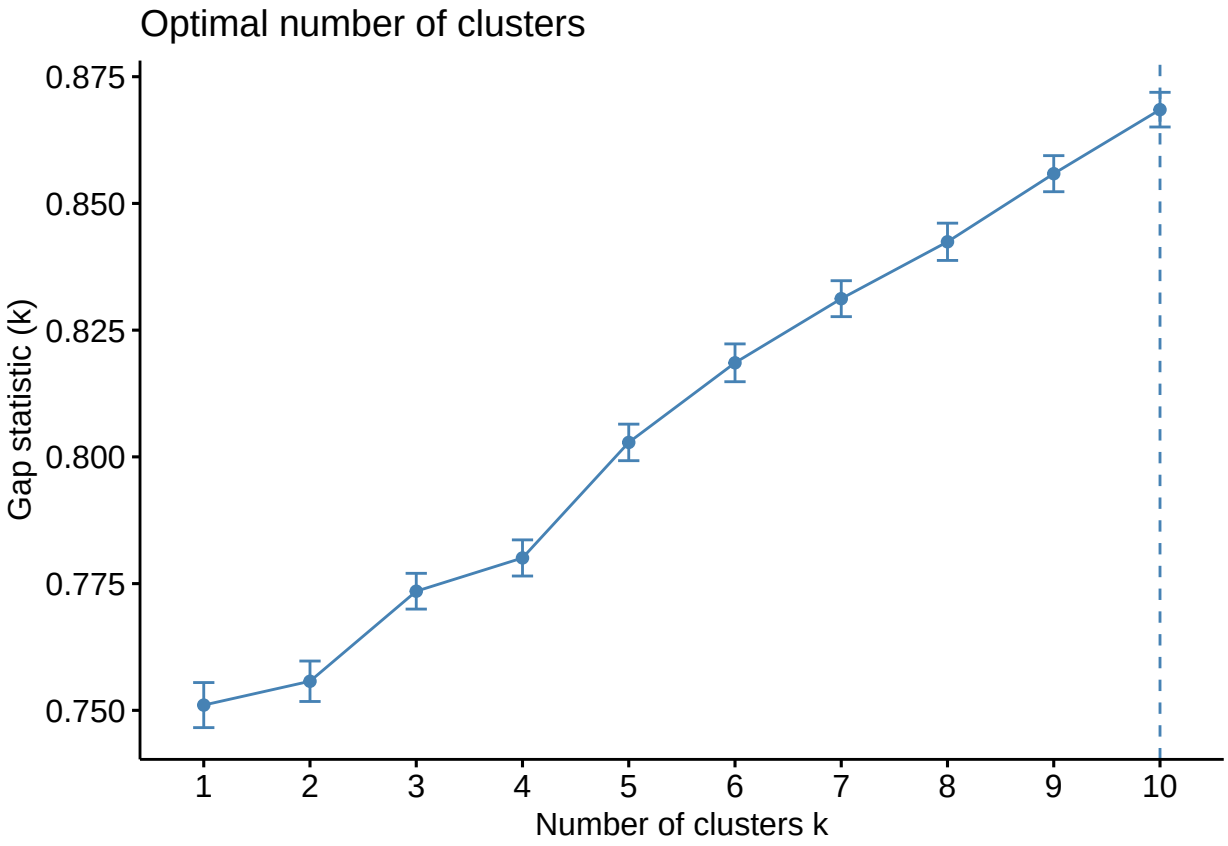

```
# Silhouette method
fviz_nbclust(pca_stats_df, kmeans, method = "silhouette")
```




```r
#CH index
CHs = c()
Ks = seq(1, 20, 2)

for(K in Ks){
  KM = kmeans(pca_stats_df, centers = k_types, nstart = 25)

  # Between-cluster sum of squares
  B = KM$betweenss

  # Within-cluster sum of squares
  W = KM$tot.withinss

  # Number of data points
  n = nrow(pca_stats_df)

  # Calculate the Calinski-Harabasz index
  CH = (B / (K - 1)) / (W / (n - K))

  # Append the CH index for the current K to the list
  CHs = c(CHs, CH)
}
df = data.frame(K = Ks, CH = CHs)
ggplot(df, aes(K, CH)) +
  geom_point() +
  geom_line() +
  ylab("CH index")
```




```
# Gap statistics
gapstat_stats = clusGap(pca_stats_df, FUN = kmeans, nstart = 50, K.max = 10, B = 50)
fviz_gap_stat(gapstat_stats, maxSE = list(method = "Tibs2001SEmax", SE.factor = 1))
```



Elbow method: Optimal K = 3

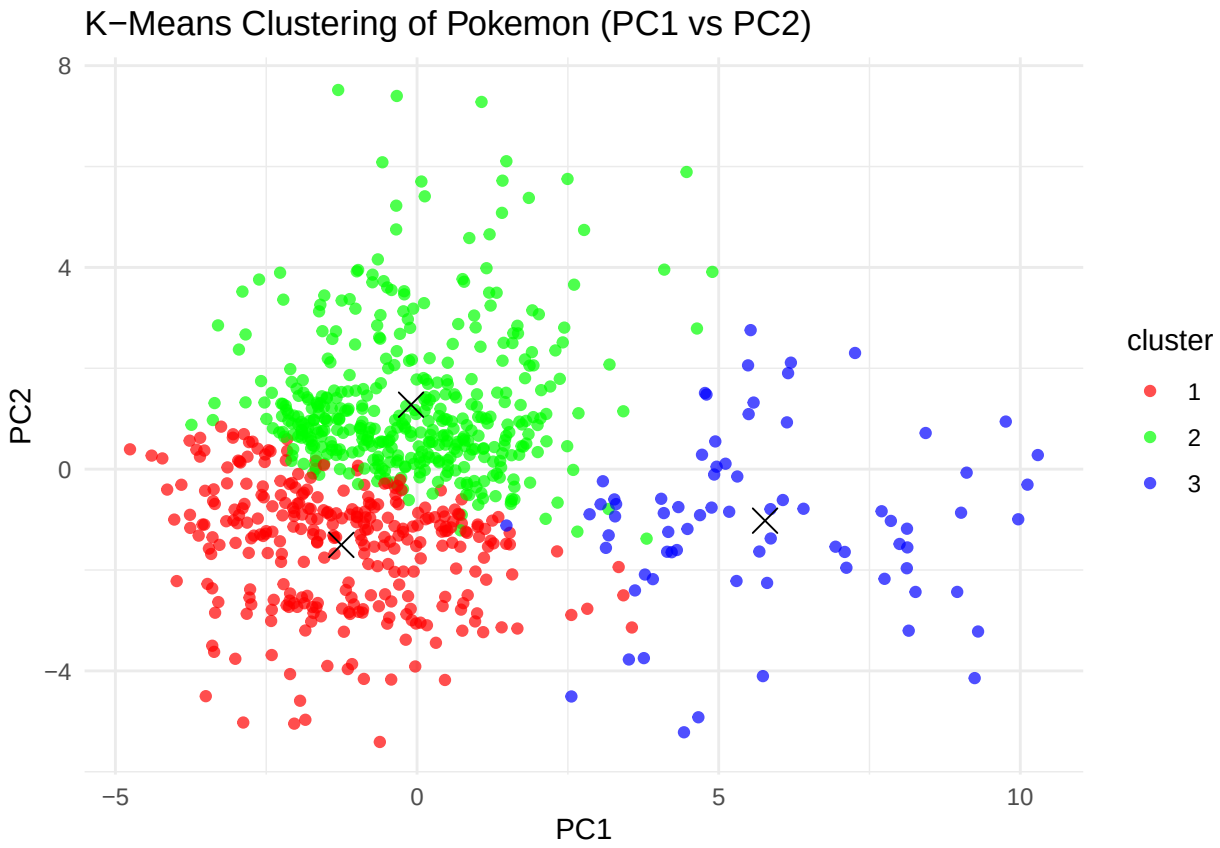
Silhouette method: Optimal K = 2

CH Index: Optimal K = 1

Gap stats: Optimal K = 10

Kmeans on optimal K = 3

```
kmeans_stats_optimal <- kmeans(pca_stats_df, centers = 3)
scatterplot(pca_stats_df, kmeans_stats_optimal$centers, kmeans_stats_optimal$cluster)
```



Comparing with true labels

```
comp_labels <- data.frame(type1 = stats$type1)
comp_labels$cluster <- as.factor(kmeans_stats$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_stats$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.5243446
```

```
comp_labels <- data.frame(type1 = stats$type1)
comp_labels$cluster <- as.factor(kmeans_w_stats$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
```

```
  mutate(cluster = factor(kmeans_w_stats$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy  (13 types)
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.2034956
```

```
comp_labels$cluster <- as.factor(kmeans_stats_optimal$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_stats_optimal$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
 mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.2309613
```

```
# pretty good considering grass water and psychic pokemon take up 30.5% of the dataset
```

Clustering on Image Dataset

Kmeans on images k = 18

```
pca_imgs_df<- dr_images[,-1]  # Remove image_path column
k_types <- length(unique(stats$type1))
kmeans_imgs <- kmeans(pca_imgs_df, centers = k_types)
```

Comparing standard kmeans with kmeans++ and weighted kmeans

Kmeans++

```
#Ref: lecture 7
# Randomly select the first centroid
n <- nrow(pca_imgs_df)
M <- pca_imgs_df[sample(1:n, 1), , drop = FALSE]

# Select remaining k-1 centroids using weighted probability
for (i in 2:k_types) {
  # Compute distance from each point to the nearest centroid
  D <- as.matrix(dist(rbind(M, pca_imgs_df)))[2:(n+1), 1]

  # Probability for each point to be chosen as the next centroid
  P <- D^2 / sum(D^2)

  # Select the next centroid based on P
  pick <- sample(1:n, 1, prob = P)
  M <- rbind(M, pca_imgs_df[pick, , drop = FALSE])
}

kmeans_imgs_pp <- kmeans(pca_imgs_df, centers = M, algorithm = "Lloyd")
```

```
#Lower withinss is better
kmeans_imgs$tot.withinss
```

```
## [1] 677488.2
```

```
kmeans_imgs_pp$tot.withinss
```

```
## [1] 696824.9
```

```
#Higher betweenss is better
kmeans_imgs$betweenss
```

```
## [1] 288142.1
```

```
kmeans_imgs_pp$betweenss
```

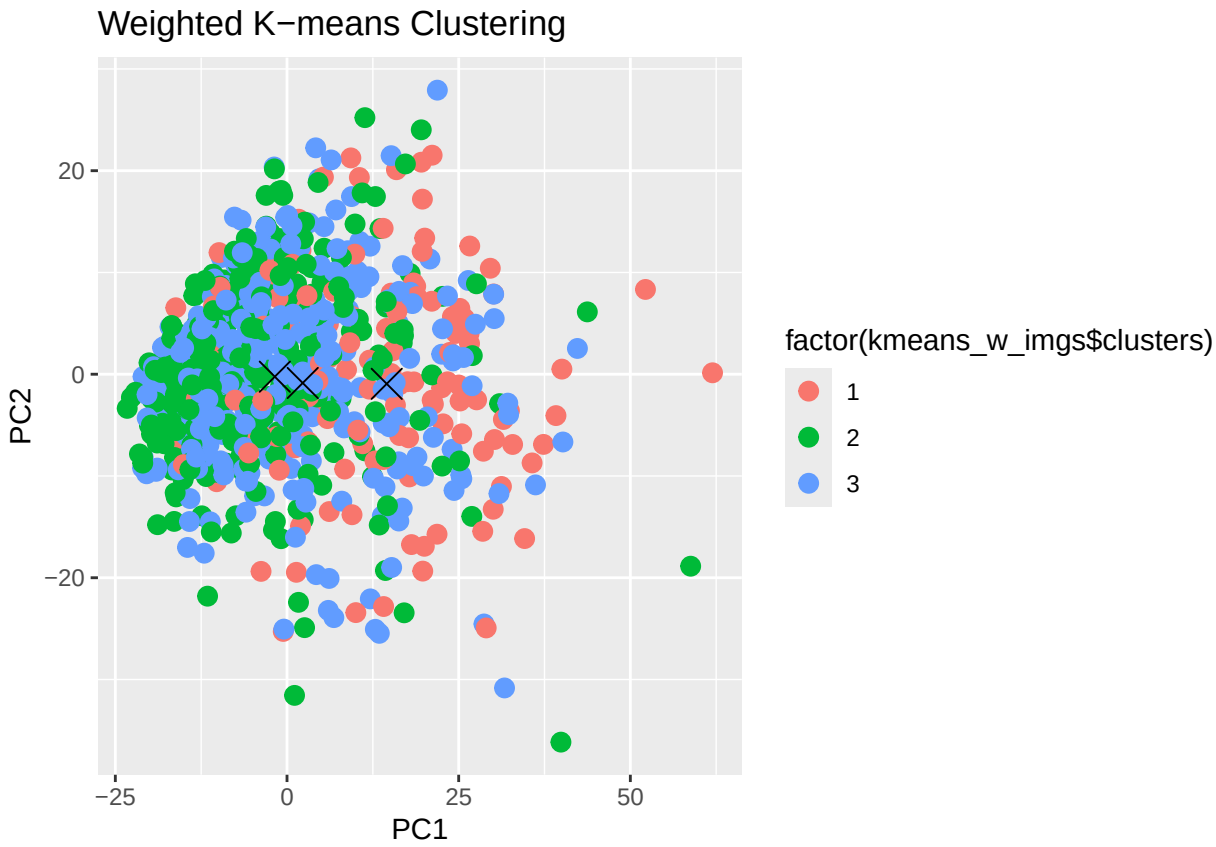
```
## [1] 268805.4
```

```
#Standard kmeans outperforms kmeans++
```

Weighted K-means

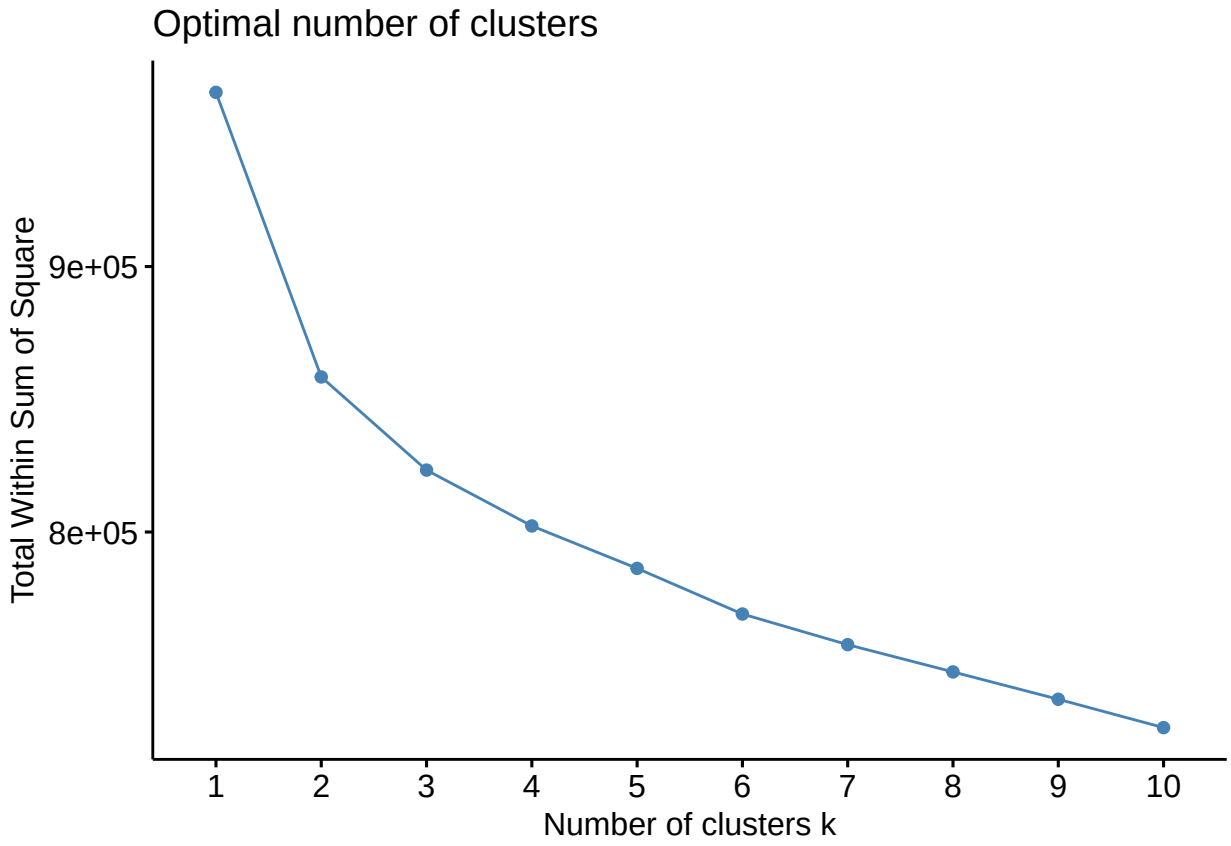
```
K <- 3
kmeans_w_imgs <- weighted_kmeans(pca_imgs_df, K)
colnames(kmeans_w_imgs$centroids) <- colnames(pca_imgs_df)

ggplot(data.frame(pca_imgs_df), aes(PC1, PC2, color = factor(kmeans_w_imgs$clusters))) +
```

```
  geom_point(size = 3) +
  geom_point(data = data.frame(kmeans_w_imgs$centroids), aes(PC1, PC2),
             color = "black", shape = 4, size = 5) +
  labs(title = "Weighted K-means Clustering", x = "PC1", y = "PC2")
```

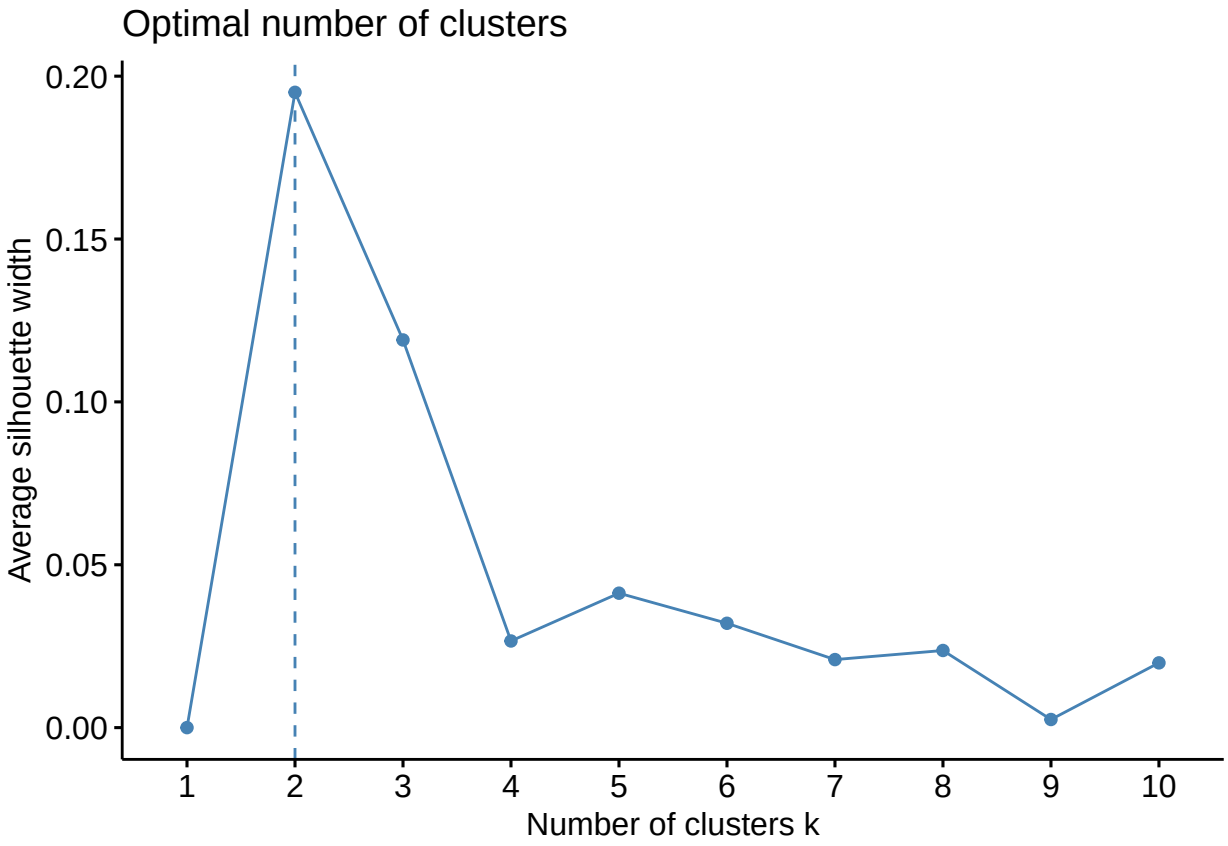


Tune for optimal k

```
# Elbow method
fviz_nbclust(pca_imgs_df, kmeans, method = "wss")
```

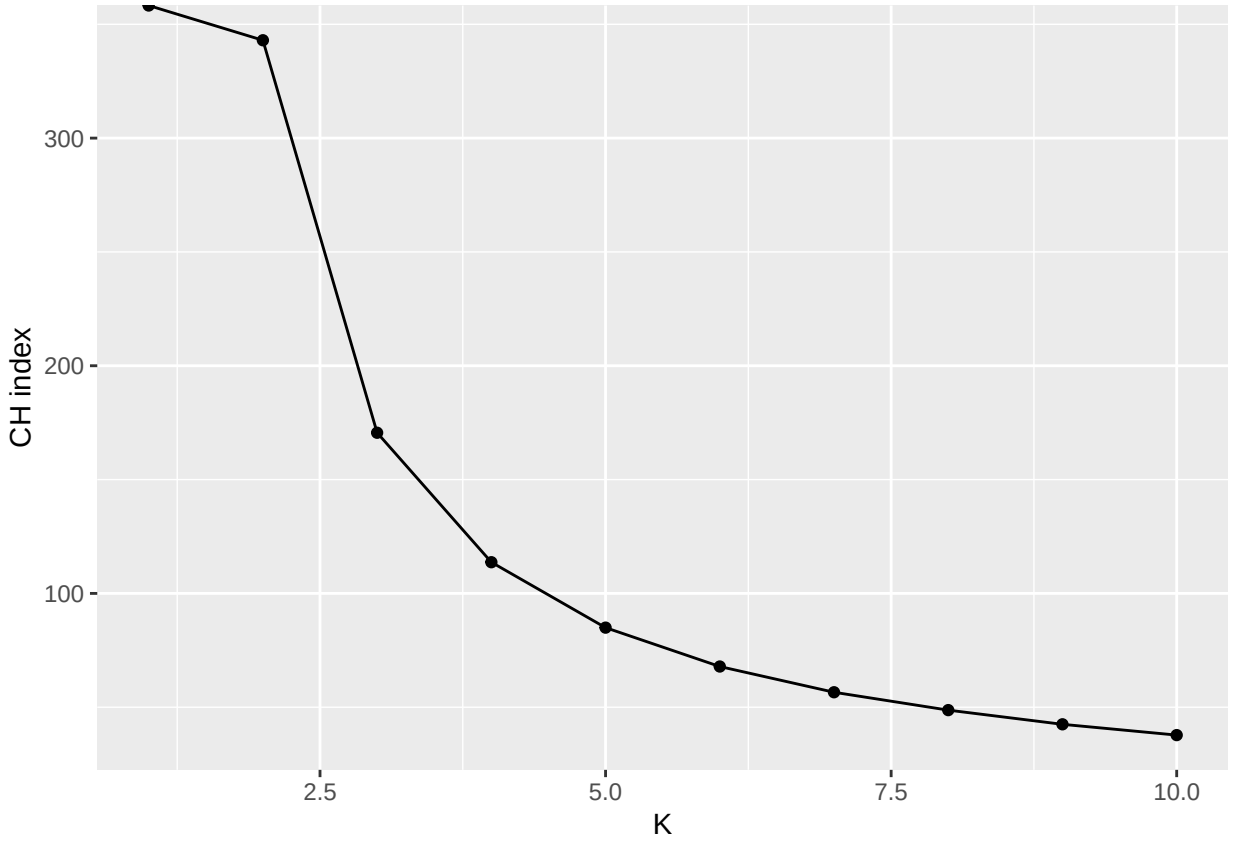



```
# Silhouette method
fviz_nbclust(pca_imgs_df, kmeans, method = "silhouette")
```

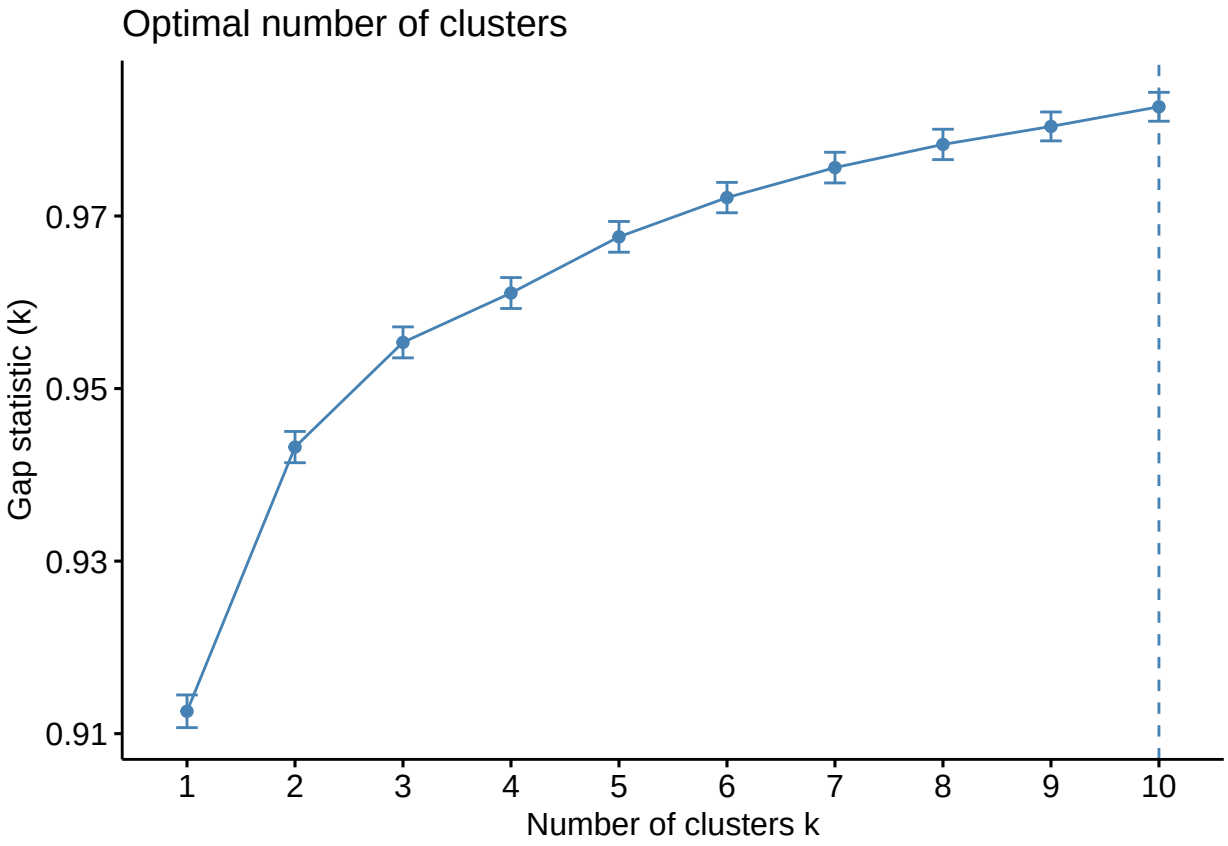



```r
# CH Index

CHs = c()
Ks = seq(1, 10, 1)

for(K in Ks){
  KM = kmeans(pca_imgs_df, centers = k_types, nstart = 25)

  # Between-cluster sum of squares
  B = KM$betweenss

  # Within-cluster sum of squares
  W = KM$tot.withinss

  # Number of data points
  n = nrow(pca_imgs_df)

  # Calculate the Calinski-Harabasz index
  CH = (B / (K - 1)) / (W / (n - K))

  # Append the CH index for the current K to the list
  CHs = c(CHs, CH)
}

df = data.frame(K = Ks, CH = CHs)
ggplot(df, aes(K, CH)) +
  geom_point() +
```

```
  geom_line() +
  ylab("CH index")
```




```
#Gap statistics
gapstat_img = clusGap(pca_imgs_df, FUN = kmeans, nstart = 50, K.max = 10, B = 50)
fviz_gap_stat(gapstat_img, maxSE = list(method = "Tibs2001SEmax", SE.factor = 1))
```



Elbow method: Optimal K = 3

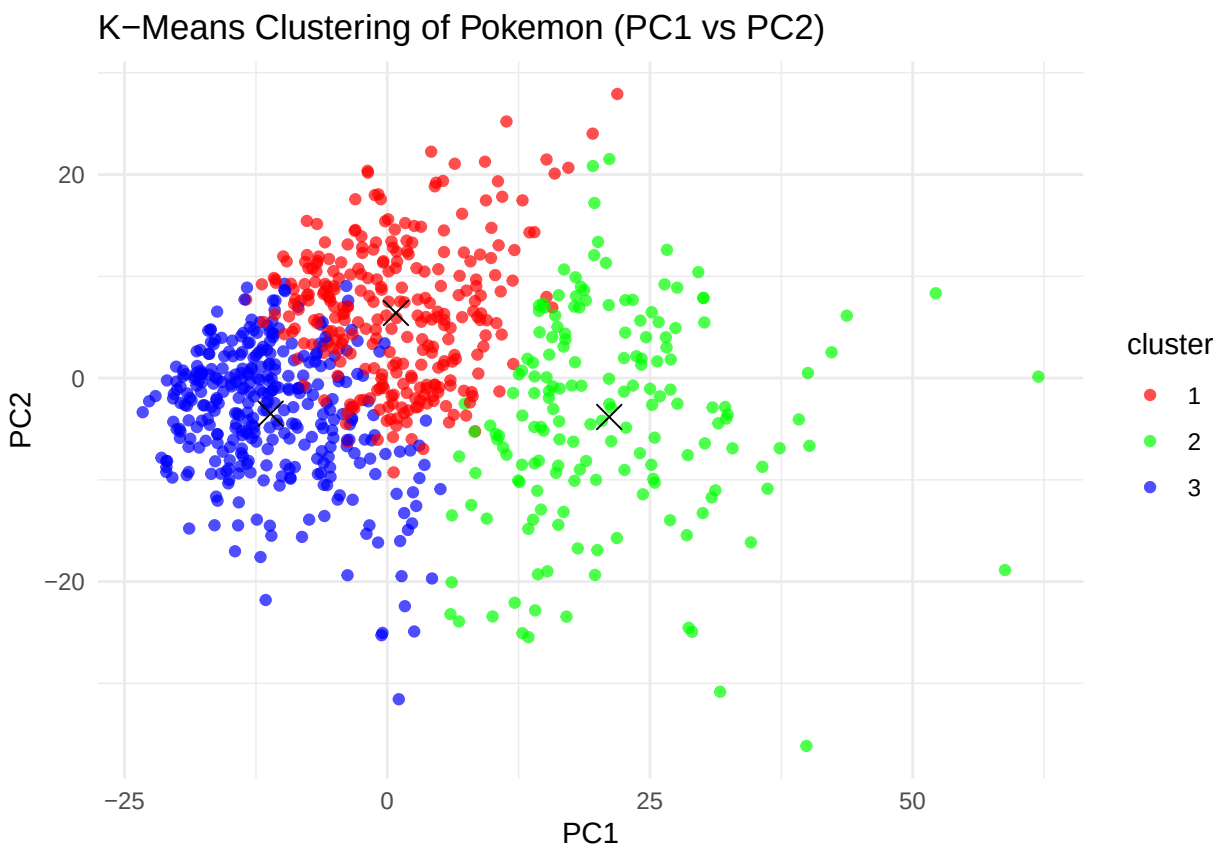
Silhouette method: Optimal K = 2

CH Index: Optimal K = 1

Gap statistics: Optimal K = 10

Kmeans on optimal K = 3

```
kmeans_imgs_optimal <- kmeans(pca_imgs_df, centers = 3, nstart = 25)
scatterplot(pca_imgs_df, kmeans_imgs_optimal$centers, kmeans_imgs_optimal$cluster)
```



Comparing with true labels

```
comp_labels$cluster <- as.factor(kmeans_imgs$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_imgs$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.196005
```

```
comp_labels <- data.frame(type1 = stats$type1)
comp_labels$cluster <- as.factor(kmeans_w_imgs$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_w_imgs$cluster)) %>%
```

```
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.1423221
```

```
comp_labels$cluster <- as.factor(kmeans_imgs_optimal$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_imgs_optimal$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.1473159
```

Clustering with UMAP

```
umap_data <- UMAP_imgs$layout
colnames(umap_data) <- c("PC1","PC2") #Change col name later
k_types <- length(unique(stats$type1))
kmeans_imgs_umap <- kmeans(umap_data, centers = k_types, nstart = 25)
```

Kmeans++ with UMAP data

```
set.seed(1234)
n <- nrow(umap_data)
M <- umap_data[sample(1:n, 1), , drop = FALSE]

# Select remaining k-1 centroids using weighted probability
for (i in 2:k_types) {
  # Compute distance from each point to the nearest centroid
  D <- as.matrix(dist(rbind(M, umap_data)))[2:(n+1), 1]

  # Probability for each point to be chosen as the next centroid
  P <- D^2 / sum(D^2)
```

```
  # Select the next centroid based on P
  pick <- sample(1:n, 1, prob = P)
  M <- rbind(M, umap_data[pick, , drop = FALSE])
}

kmeans_umap_pp <- kmeans(umap_data, centers = M, algorithm = "Lloyd")
```

```
comp_labels$cluster <- as.factor(kmeans_umap_pp$cluster)

# Map clusters to true labels
cluster_label_mapping <- comp_labels %>%
  mutate(cluster = factor(kmeans_umap_pp$cluster)) %>%
  group_by(cluster) %>%
  summarise(mode_type = Mode(type1), .groups = 'drop')


# Map cluster labels to the most common actual type
labelled_clusters <- comp_labels %>%
  left_join(cluster_label_mapping, by = "cluster") %>%
  select(type1, cluster, mode_type)

# Calculate accuracy
mean(labelled_clusters$type1 == labelled_clusters$mode_type)
```

```
## [1] 0.1822722
```