Dimension Reduction

Alex Faassen

2025-03-18

To Do: - PCA - Biplot - Loading vectors - Variance explained (Scree plot) - Compare with t-SNE/UMAP: is there a non-linear element? - Numeric comparisons (e.g. "silhouette score" - comparisons against labels)? - Save output

Setup

```r
## Load Data
# setwd("./Data")
load("../Data/pokemon.RData")

## Libraries
library(ggplot2)
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.4.2
```

```r
library(patchwork)
```

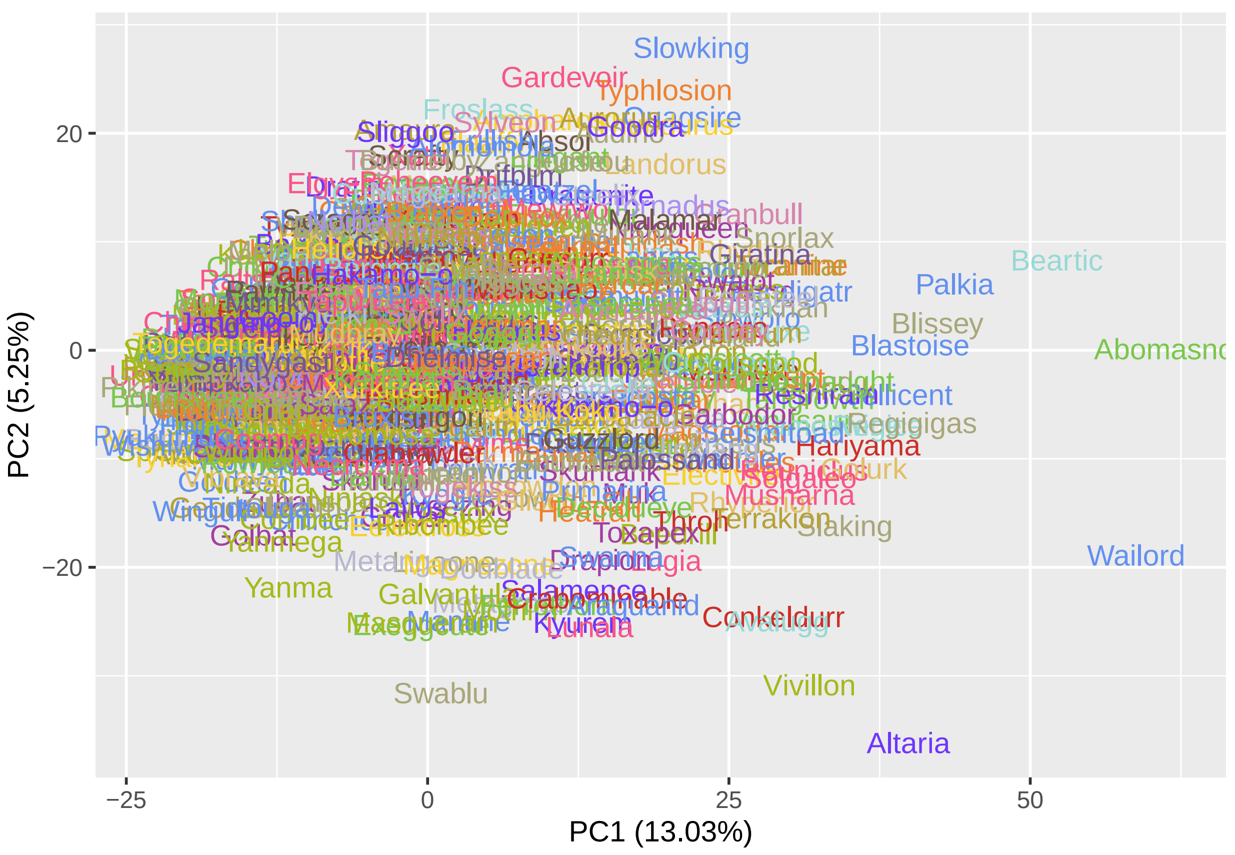
```
## Warning: package 'patchwork' was built under R version 4.4.2
```

```r
## Helper functions
barplot = function(values){
  n = length(values)
  df = data.frame(value = values,  index = 1:n)
  ggplot(df, aes(index, value, fill = value)) +
    geom_bar(color = "black", stat = "identity")+
    scale_fill_gradient2(low="#619CFF", mid="white", high="#F8766D")+
    theme_bw()
}

# heatmap = function(A){
#   n = nrow(A)
#   p = ncol(A)
#   df = data.frame(value = c(A),  i = 1:n, j = rep(1:p, rep(n, p)))
#   ggplot(df, aes(j, i, fill = value)) +
#     geom_tile(color = "black")+
#     scale_fill_gradient2(low="#619CFF", mid="white", high="#F8766D")+
#     scale_y_reverse()+
#     theme_void()
# }

getImg = function(flat_img){
  # matrix(unlist(gsvalues), 120, 120, 4, byrow = T)
  # rasterGrob(array(flat_img, dim = c(120, 120, 4)))
  array(as.numeric(flat_img), dim = c(120, 120, 3)) # (120, 120, 4)
}

plotImg <- function(img_raster) {
  # Plot using ggplot2
  ggplot() +
    annotation_raster(img_raster, xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf) +
    theme_void()  # Remove axes for clean visualization
}
```

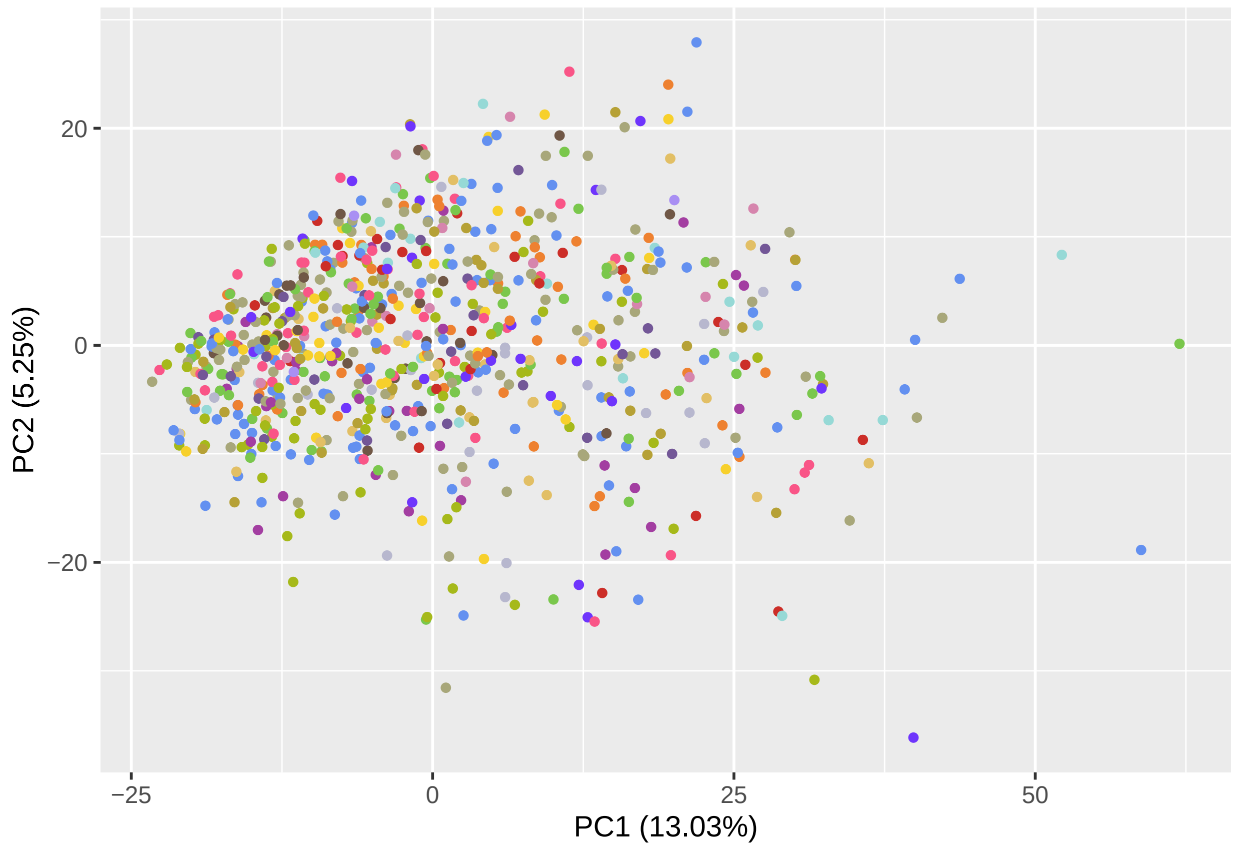
PCA

```r
## PCA
# Defaults: center = TRUE, scale. = FALSE
PCA = prcomp(images[,-1], center = TRUE, scale. = FALSE) #scale = TRUE)

sum_PCA = summary(PCA)
```

```r
## Biplot
# Colours
stats$type1 = factor(stats$type1)
# unique(stats$type1)
type_colours = c("#7AC74C", "#EE8130", "#6390F0", "#A6B91A", "#A8A77A", "#A33EA1",
            "#F7D02C", "#E2BF65", "#D685AD", "#CC2E28", "#F95587", "#B6A136", "#735797",
            "#96D9D6", "#6F35FC", "#705746", "#B7B7CE", "#A98FF3")
names(type_colours) = unique(stats$type1)

# library(ggfortify)
autoplot(PCA, data = cbind(stats$name, stats$type1, images[,-1]), shape = FALSE, color = "stats$type1",
  scale_colour_manual(values = type_colours) +
  theme(legend.position = "none")
```



```
# Without labels, just types.
autoplot(PCA, data = cbind(stats$name, stats$type1, images[,-1]), label = FALSE, shape = 16, color = "s
  scale_colour_manual(values = type_colours) +
  theme(legend.position = "none")
```



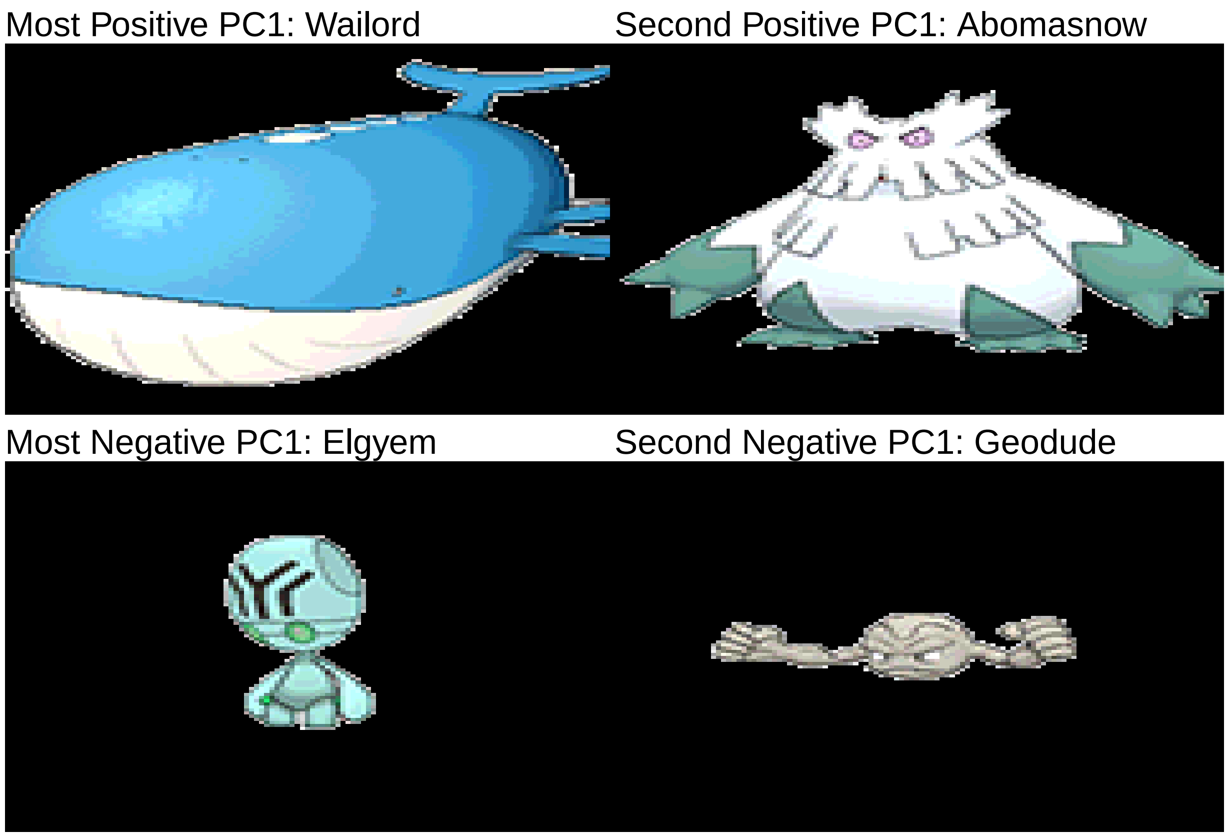
Loading Vectors

```r
## Smallest PC1 Pokemon
# Extract PCA-transformed data
pca_data <- as.data.frame(PCA$x)  # Convert PCA results to a data frame

# Add Pokémon names and types for reference
pca_data$name <- stats$name  # Ensure stats$name contains Pokémon names
pca_data$type1 <- stats$type1  # Primary type for reference

# Sort by PC1 in ascending order and select the two smallest
smallest_pc1_pokemon <- pca_data[order(pca_data$PC1), ][1:2, ]

# Print result
print(rownames(smallest_pc1_pokemon))
```

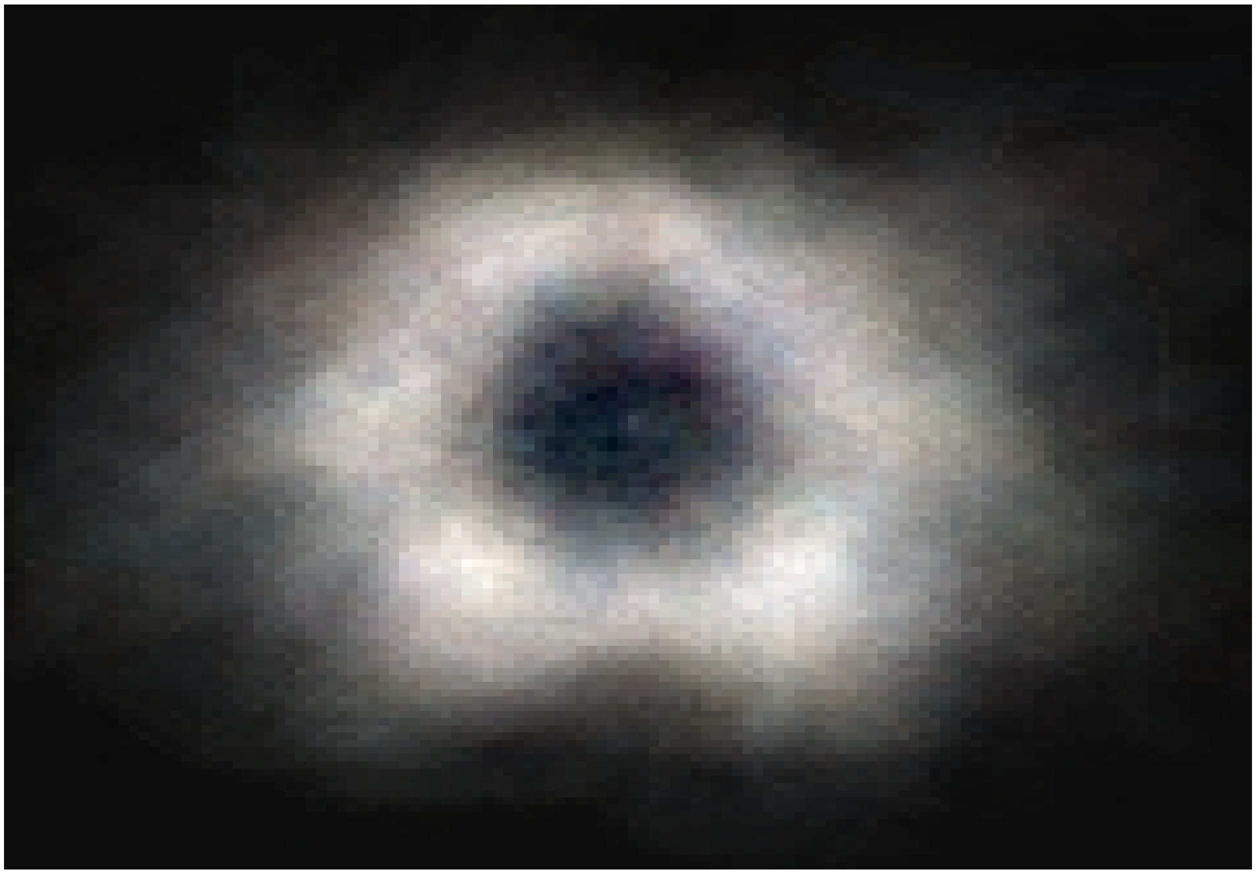
```
## [1] "images/pikipek.png" "images/unown.png"
```

```r
## PC1
# Extremes
pos1 = plotImg(getImg(images[stats$name == "Wailord", -1]))
pos2 = plotImg(getImg(images[stats$name == "Abomasnow", -1]))
neg1 = plotImg(getImg(images[stats$name == "Elgyem", -1])) # Pikipek
neg2 = plotImg(getImg(images[stats$name == "Geodude", -1])) # Unown
# exPlots = sapply(c(pos1, pos2, neg1, neg2), function(rgba) plotImg(getImg(rgba)))
# (pos1 + pos2) / (neg1 + neg2)
```

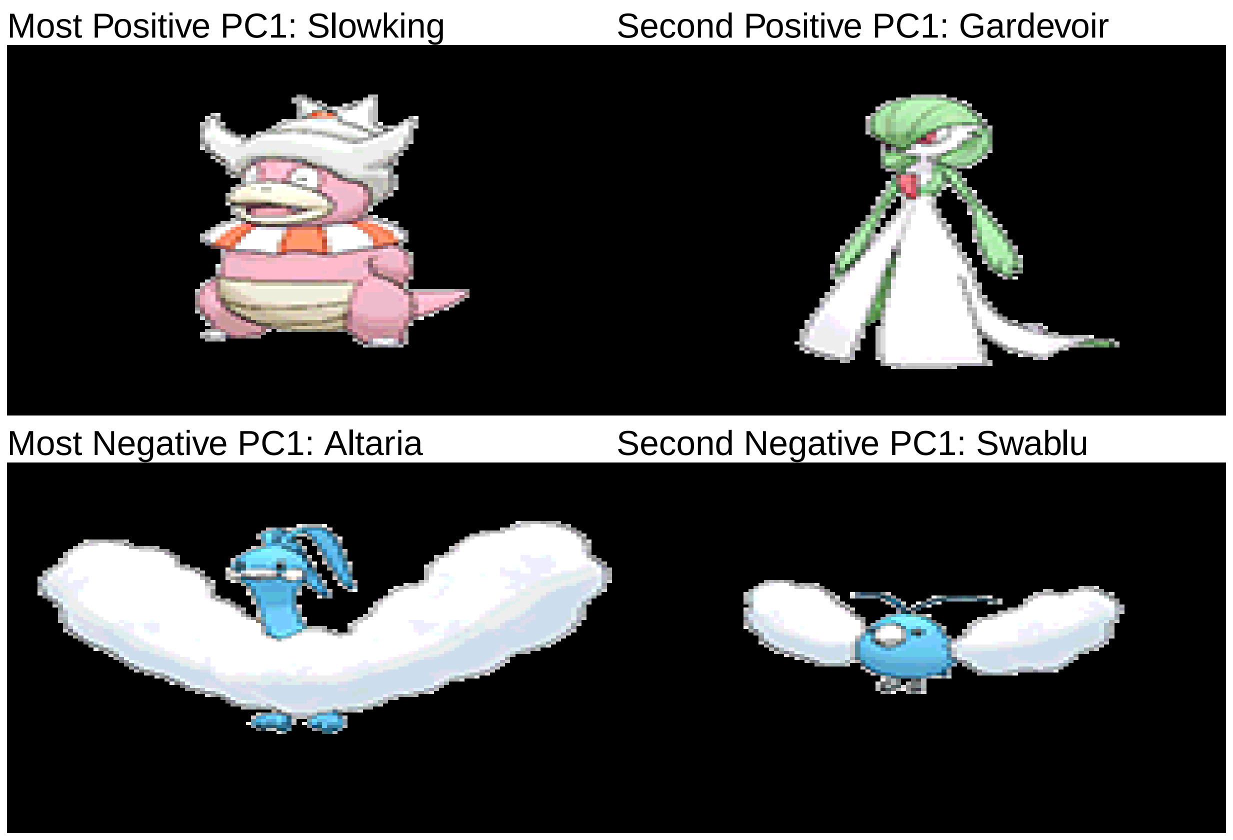
```r
final_plot = (pos1 + labs(title = "Most Positive PC1: Wailord") +
              pos2 + labs(title = "Second Positive PC1: Abomasnow")) /
             (neg1 + labs(title = "Most Negative PC1: Elgyem") +
              neg2 + labs(title = "Second Negative PC1: Geodude"))
print(final_plot)
```


Most Positive PC1: Wailord
Second Positive PC1: Abomasnow
Most Negative PC1: Elgyem
Second Negative PC1: Geodude

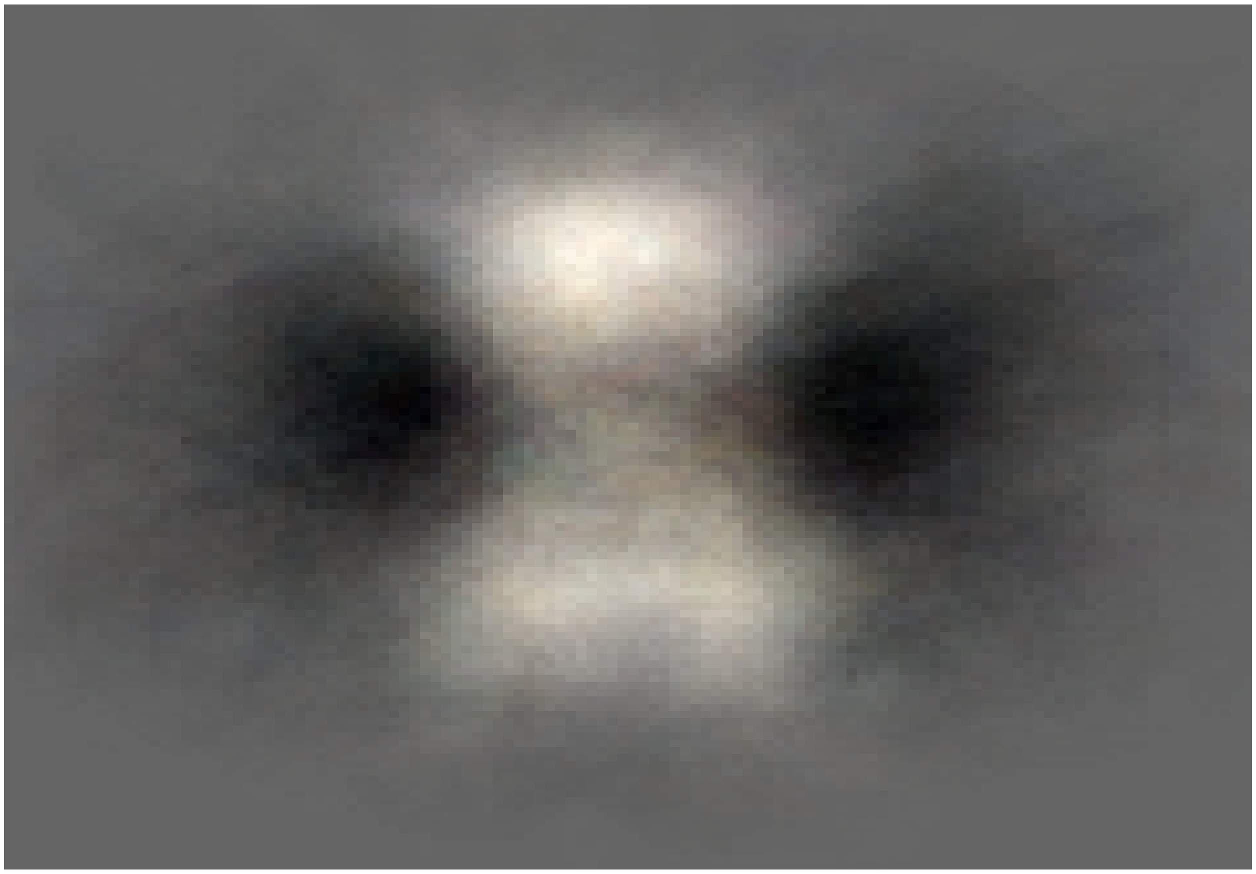
```r
# Loading Vector
pca = PCA$rotation[,1]
pca_norm = (pca - min(pca)) / (max(pca) - min(pca))
# pca_norm = (pca - mean(pca))/(2 * sd(pca))
# pca_clip = pmax(pmin(pca, 1), 0)
plotImg(getImg(pca_norm))
```



```r
## PC2
# Extremes
pos1 = plotImg(getImg(images[stats$name == "Slowking", -1]))
pos2 = plotImg(getImg(images[stats$name == "Gardevoir", -1]))
neg1 = plotImg(getImg(images[stats$name == "Altaria", -1]))
neg2 = plotImg(getImg(images[stats$name == "Swablu", -1]))
# exPlots = sapply(c(pos1, pos2, neg1, neg2), function(rgba) plotImg(getImg(rgba)))
# (pos1 + pos2) / (neg1 + neg2)
final_plot = (pos1 + labs(title = "Most Positive PC1: Slowking") +
               pos2 + labs(title = "Second Positive PC1: Gardevoir")) /
              (neg1 + labs(title = "Most Negative PC1: Altaria") +
               neg2 + labs(title = "Second Negative PC1: Swablu"))
print(final_plot)
```


Most Positive PC1: Slowking
Second Positive PC1: Gardevoir
Most Negative PC1: Altaria
Second Negative PC1: Swablu

```r
# Loading Vector
pca = PCA$rotation[,2]
pca_norm = (pca - min(pca)) / (max(pca) - min(pca))
plotImg(getImg(pca_norm))
```

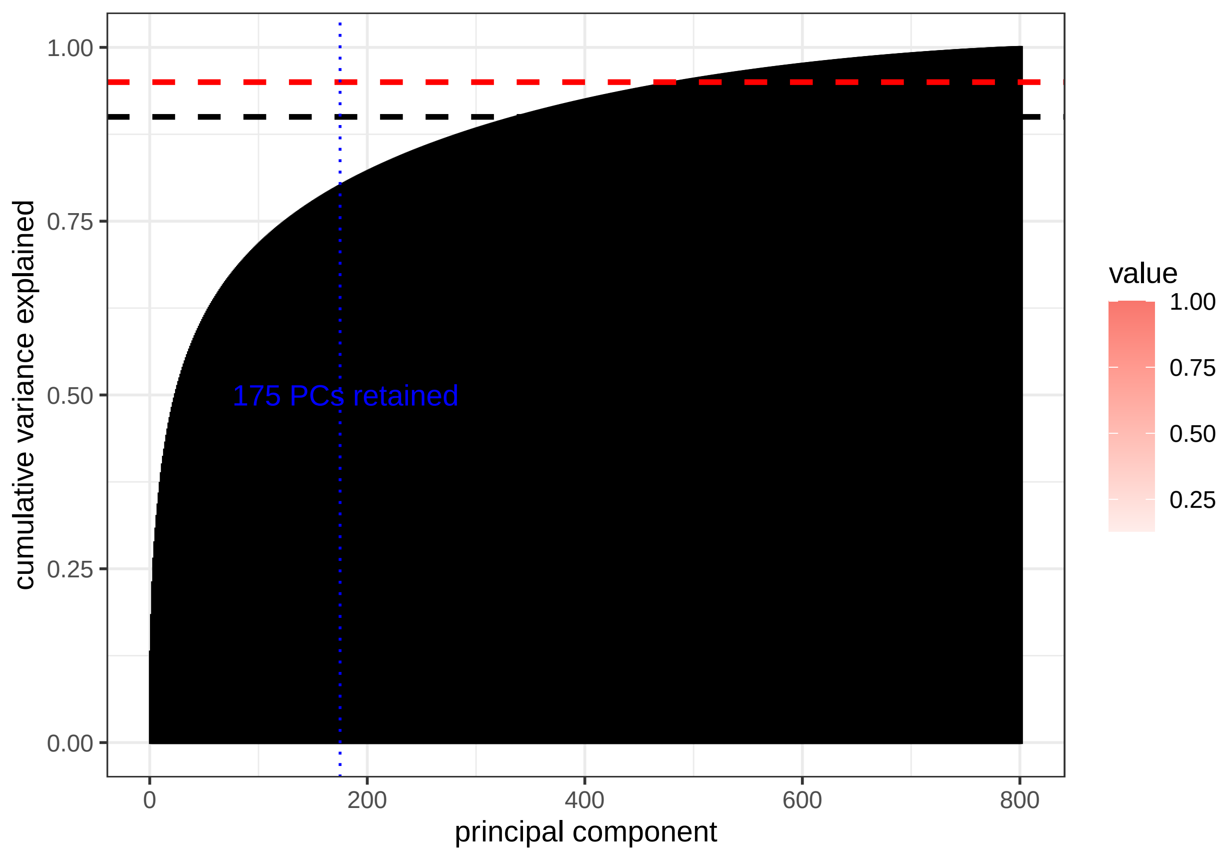


Variance Explained

```r
## Variance Explained
ve = summary(PCA)$importance[3,]

## Retain PCs
var_retain = 0.8 # % of VE
num_pcs = min(which(ve >= var_retain)); num_pcs
```

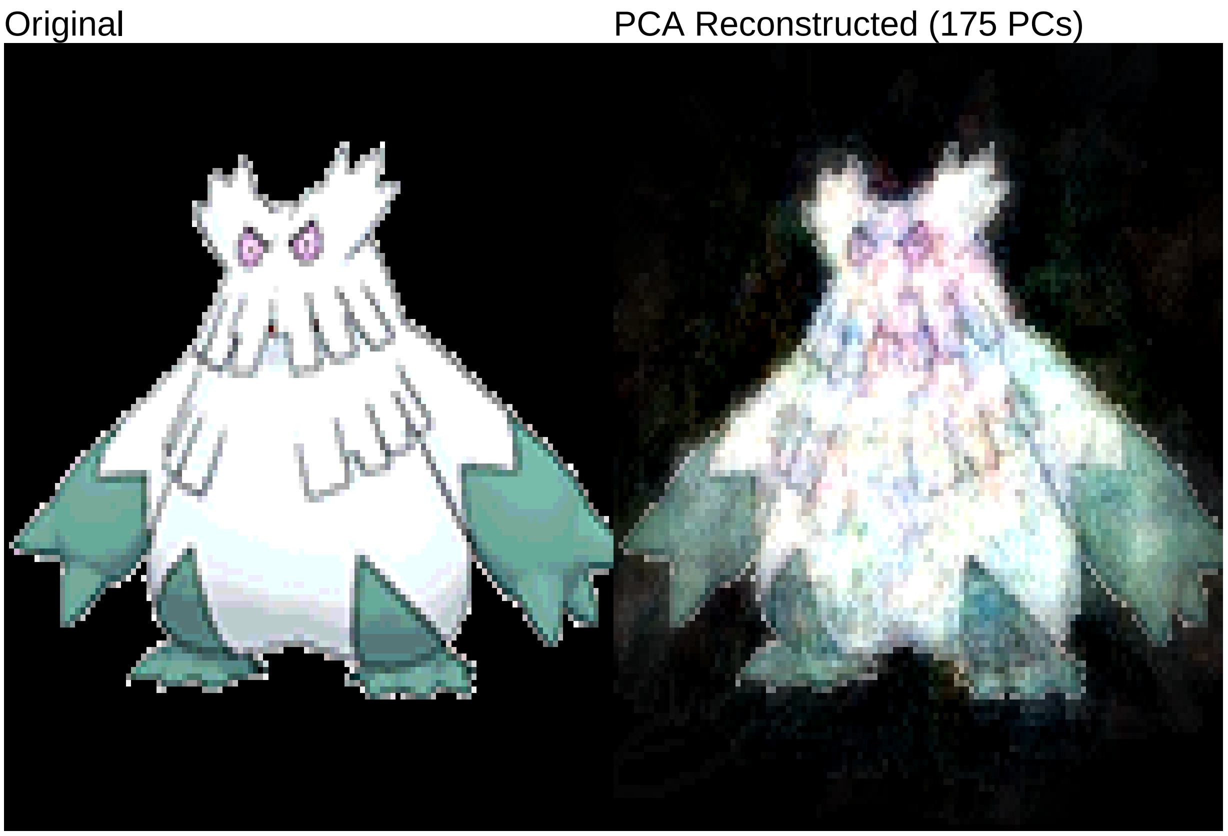
```
## [1] 175
```

```r
dr_images = data.frame(
  image_path = images$image_path,
  PCA$x[,1:num_pcs]
)

# Plot
barplot(ve)+
  xlab("principal component")+
  ylab("cumulative variance explained")+
  ylim(0, 1)+
  geom_hline(aes(yintercept = 0.9), linewidth = 1, linetype = "dashed") +
  geom_hline(aes(yintercept = 0.95), linewidth = 1, linetype = "dashed", color = "red") +
  geom_vline(xintercept = num_pcs, linetype = "dotted", color = "blue") +  # Highlight selected PC coun
  annotate("text", x = num_pcs + 5, y = 0.5, label = paste0(num_pcs, " PCs retained"), color = "blue")
```

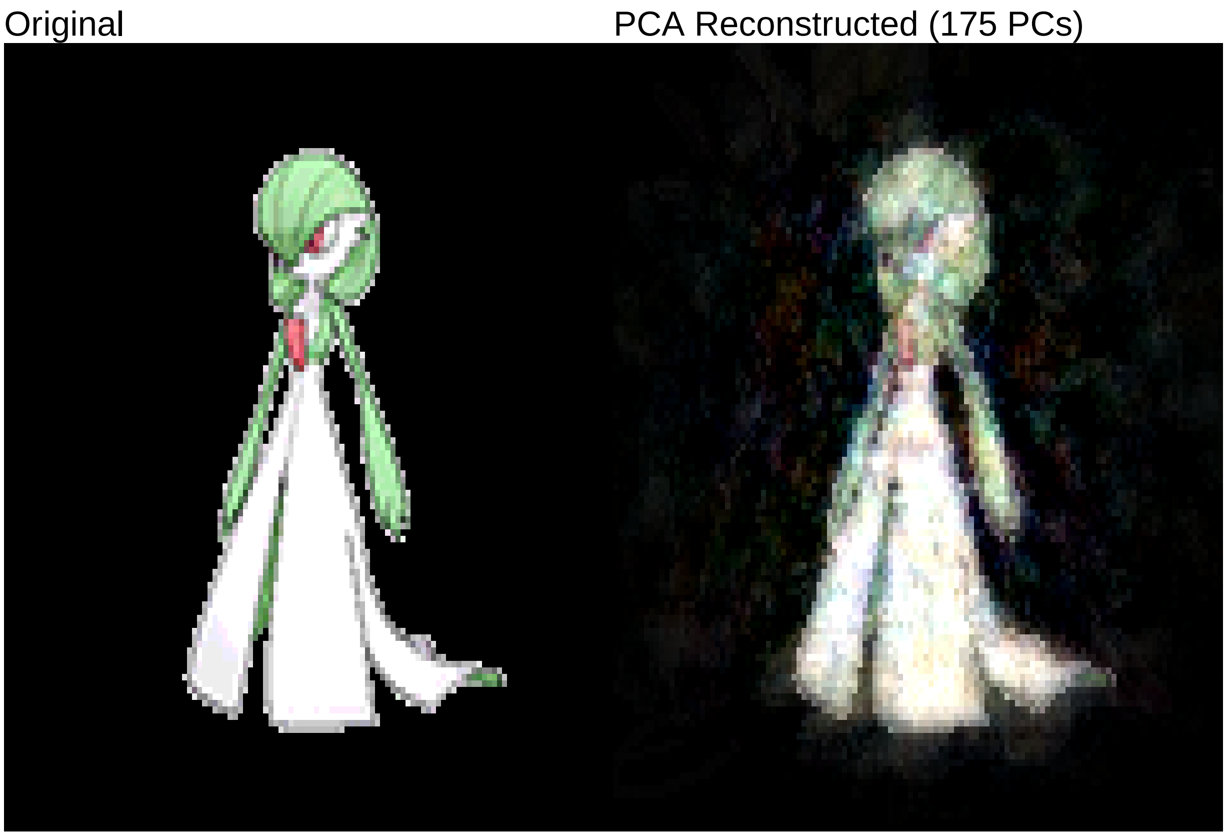


Visualize Compressed Images

```
## Abomasnow
pokemon = which(stats$name == "Abomasnow")
# Original image
og = plotImg(getImg(images[pokemon, -1])) + labs(title = "Original")
# Compressed image
scores = (as.numeric(images[pokemon, -1]) - PCA$center) %*% PCA$rotation
scores_95VE = scores[, 1:num_pcs, drop = FALSE] %*% t(PCA$rotation[, 1:num_pcs]) + PCA$center
# scores_95VE_norm = (scores_95VE - min(scores_95VE)) / (max(scores_95VE) - min(scores_95VE))
scores_95VE_clip = pmax(pmin(scores_95VE, 1), 0)
compressed = plotImg(getImg(scores_95VE_clip)) +
  labs(title = paste("PCA Reconstructed (", num_pcs, " PCs)", sep=""))
# Side by side
sbs = og + compressed
print(sbs)
```



```r
## Gardevoir
pokemon = which(stats$name == "Gardevoir")
# Original image
og = plotImg(getImg(images[pokemon, -1])) + labs(title = "Original")
# Compressed image
scores = (as.numeric(images[pokemon, -1]) - PCA$center) %*% PCA$rotation
scores_95VE = scores[, 1:num_pcs, drop = FALSE] %*% t(PCA$rotation[, 1:num_pcs]) + PCA$center
# scores_95VE_norm = (scores_95VE - min(scores_95VE)) / (max(scores_95VE) - min(scores_95VE))
scores_95VE_clip = pmax(pmin(scores_95VE, 1), 0)
compressed = plotImg(getImg(scores_95VE_clip)) +
  labs(title = paste("PCA Reconstructed (", num_pcs, " PCs)", sep=""))
# Side by side
sbs = og + compressed
print(sbs)
```



Save Dataset

```
## Save Dataset
# save(stats, dr_images, file = "../Data/dr_pokemon2.RData")
```

Non-linear methods

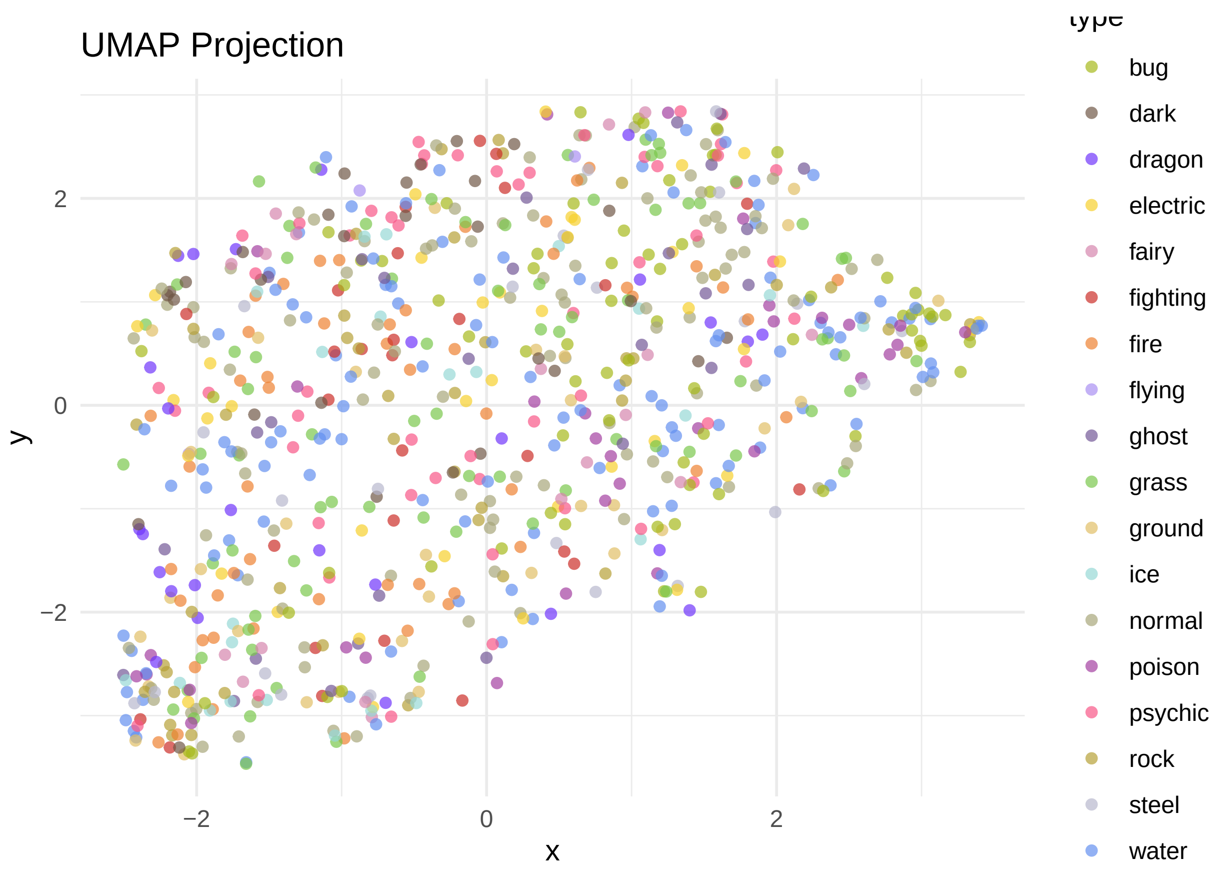
See whether non-linear clustering might work better.

t-SNE

```
## COULDN'T RUN: Kept hitting stack overflow protection

# ## t-SNE
# library(Rtsne)
#
# TSNE_imgs = Rtsne(images[,-1], perplexity = 5) # Adjust perplexity for different structures
#   # Original call: perplexity = 30
#
# # Plot
# type_colours = type_colours[match(levels(stats$type1), names(type_colours))] # Colours
#
# tsne_df = data.frame(x = TSNE_imgs$Y[,1], y = TSNE_imgs$Y[,2], type = stats$type1)
# ggplot(tsne_df, aes(x, y, color = type)) +
#   geom_point(alpha = 0.7) +
#   theme_minimal() +
#   scale_colour_manual(values = type_colours) +
#   ggtitle("t-SNE Projection")
#
# ## Save object
# save(TSNE_imgs, file = "DimensionReduction/tsne_pokemon.rds")
```

UMAP

```
# ## UMAP
# library(umap)
#
# UMAP_imgs = umap(images[,-1])

## Load object
load("umap_pokemon.rds")

# Plot
type_colours = type_colours[match(levels(stats$type1), names(type_colours))] # Colours

umap_df = data.frame(x = UMAP_imgs$layout[,1], y = UMAP_imgs$layout[,2], type = stats$type1)
ggplot(umap_df, aes(x, y, color = type)) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  scale_colour_manual(values = type_colours) +
  ggtitle("UMAP Projection")
```



```
## Save object
# save(UMAP_imgs, file = "DimensionReduction/umap_pokemon.rds")
```

Kernel PCA

```
## COULDN'T RUN: Kept hitting stack overflow protection

# ## Kernel PCA
# library(kernlab)
#
# # Perform Kernel PCA with RBF kernel
# kpca_result = kpca(~., data = images[,-1], kernel = "rbfdot", features = 20)
#
# # Convert KPCA to data frame
# kpca_df = data.frame(kpca_result@rotated, type = stats$type1)
#
# # Plot first 2 Kernel PCA components
# ggplot(kpca_df, aes(PC1, PC2, color = type)) +
#   geom_point(alpha = 0.7) +
#   theme_minimal() +
#   ggtitle("Kernel PCA Projection")
```