TPS Test

```
# Load libraries
library(geomorph)
```

```
Loading required package: RRPP
```

```
Loading required package: rgl
```

```
Loading required package: Matrix
```

```
library(readxl)
library(ggplot2)
library(ggforce)
library(ggrepel)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.4     v tibble    3.2.1
v purrr     1.0.4     v tidyr     1.3.1
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x tidyr::expand() masks Matrix::expand()
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x tidyr::pack()   masks Matrix::pack()
x tidyr::unpack() masks Matrix::unpack()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
palette("Okabe-Ito")
```

```r
# Read Excel file (adjust path if needed)
file_path <- "TPS combined.xlsx"
raw_data <- read_excel(file_path, sheet = "Sheet1", col_names = FALSE)
```

```
Warning: Expecting numeric in B1628 / R1628C2: got 'CHECK'

Warning: Expecting logical in D1628 / R1628C4: got 'SPECIMEN'

Warning: Expecting logical in E1628 / R1628C5: got 'ID.png'

Warning: Expecting numeric in B1954 / R1954C2: got '1464_Talus_Dorsal_R.png'

Warning: Expecting numeric in B1979 / R1979C2: got '1476_Talus_Dorsal_L.png'

Warning: Expecting numeric in B2004 / R2004C2: got 'foot'

Warning: Expecting logical in C2004 / R2004C3: got 'talus'

Warning: Expecting logical in D2004 / R2004C4: got
'reconstruction_dorsal_L.png'

New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
```

```r
View(raw_data)
# Convert to matrix for easier processing
raw_matrix <- as.matrix(raw_data)
```

```r
# Initialize list to store each specimen's landmark matrix
landmark_list <- list()
i <- 1
specimen_counter <- 1

while (i < nrow(raw_matrix)) {
  if (!is.na(as.numeric(raw_matrix[i, 1]))) {
    # Attempt to extract 21 rows of coordinates
    block <- raw_matrix[i:(i+20), 1:2]
    if (all(!is.na(block))) {
      landmark_list[[specimen_counter]] <- as.matrix(apply(block, 2, as.numeric))
      specimen_counter <- specimen_counter + 1
      i <- i + 25  # Skip landmark rows + metadata + blank line
    } else {
      i <- i + 1
    }
  } else {
    i <- i + 1
  }
}
```

```
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
Warning: NAs introduced by coercion
```

```r
# Convert list to 3D array (p x k x n)
coords_array <- array(NA, dim = c(21, 2, length(landmark_list)))
for (j in 1:length(landmark_list)) {
  coords_array[,,j] <- landmark_list[[j]]
}

# Group by
group <- as.factor(c(
  "Ardipithecus",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
```

```
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Gorilla",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pan",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Pongo",
  "Homo sapiens",
```

```
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Homo sapiens",
  "Hylobates",
  "Hylobates",
  "Hylobates",
  "Hylobates",
  "Hylobates",
  "Hylobates",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "Mandrillus",
  "LB1",
  "AL288.1",
  "AL333-147",
  "STW 88",
  "KNM-ER 1464",
  "KNM-ER 1476",
  "Littlefoot",
  "MH2",
  "OH8",
  "Omo"
))

#Create Side Vector
side_vector <- c("L","L","L","L","L","L","L","R","L","R","L","L","L","R","R","R","R","L","L"
```

```r
# Flip right-sided specimens (reflect X coordinates)
coords_flipped <- coords_array  # make a copy to modify
for (i in 1:dim(coords_array)[3]) {
  if (side_vector[i] == "R") {
    coords_flipped[,1,i] <- -coords_array[,1,i]  # Flip X only
  }
}
# Now coords_flipped has all shapes oriented as left-sided
```

#PCA

```r
# Run GPA and PCA on flipped data
gpa <- gpagen(coords_flipped)
```

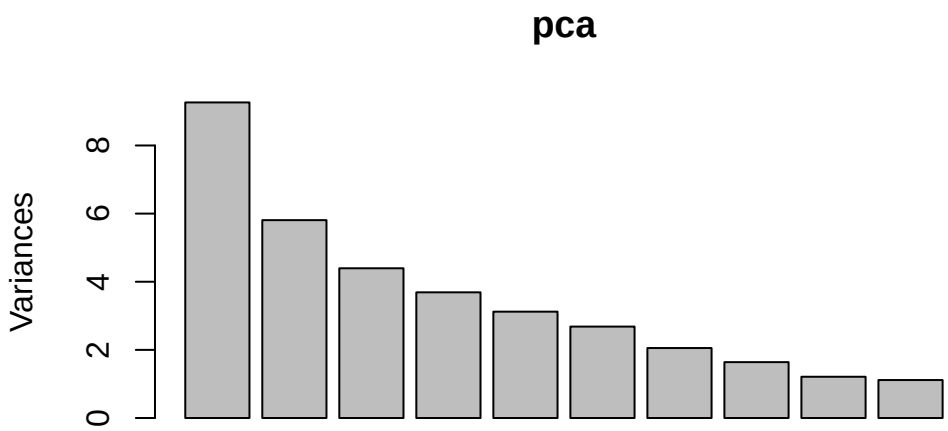
```
Performing GPA

  |
  |                                                                      |   0%
  |
  |==================                                                    |  25%
  |
  |===================================                                   |  50%
  |
  |====================================================                  |  75%
  |
  |======================================================================| 100%

Making projections... Finished!
```

```r
flat_coords <- two.d.array(gpa$coords)
pca <- prcomp(flat_coords, scale. = TRUE)

#PCA Summary
summary(pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5    PC6     PC7
Standard deviation     3.0436 2.4100 2.0965 1.92077 1.76679 1.6382 1.43326
Proportion of Variance 0.2206 0.1383 0.1047 0.08784 0.07432 0.0639 0.04891
```

```
Cumulative Proportion  0.2206 0.3589 0.4635 0.55134 0.62567 0.6896 0.73847
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     1.28048 1.10048 1.05587 0.97304 0.89015 0.83936 0.80340
Proportion of Variance 0.03904 0.02883 0.02654 0.02254 0.01887 0.01677 0.01537
Cumulative Proportion  0.77751 0.80635 0.83289 0.85543 0.87430 0.89107 0.90644
                          PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     0.75755 0.67914 0.63643 0.58489 0.53526 0.48992 0.47873
Proportion of Variance 0.01366 0.01098 0.00964 0.00815 0.00682 0.00571 0.00546
Cumulative Proportion  0.92011 0.93109 0.94073 0.94888 0.95570 0.96141 0.96687
                          PC22    PC23    PC24    PC25    PC26    PC27    PC28
Standard deviation     0.44610 0.41712 0.40245 0.38800 0.35328 0.32026 0.30509
Proportion of Variance 0.00474 0.00414 0.00386 0.00358 0.00297 0.00244 0.00222
Cumulative Proportion  0.97161 0.97575 0.97961 0.98319 0.98616 0.98861 0.99082
                          PC29    PC30    PC31    PC32    PC33    PC34    PC35
Standard deviation     0.2968 0.26927 0.25612 0.19649 0.18674 0.18101 0.15000
Proportion of Variance 0.0021 0.00173 0.00156 0.00092 0.00083 0.00078 0.00054
Cumulative Proportion  0.9929 0.99465 0.99621 0.99713 0.99796 0.99874 0.99927
                          PC36    PC37    PC38      PC39      PC40      PC41
Standard deviation     0.12621 0.10117 0.06623 4.842e-15 3.494e-15 3.378e-15
Proportion of Variance 0.00038 0.00024 0.00010 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  0.99965 0.99990 1.00000 1.000e+00 1.000e+00 1.000e+00
                           PC42
Standard deviation     1.617e-15
Proportion of Variance 0.000e+00
Cumulative Proportion  1.000e+00
```

```r
#Scree plot
plot(pca)
```



#PCA to Dataframe for Plot

```
# Plot with ggplot2
pca_df <- data.frame(pca$x,
                     Group = group)  # group vector as before

# Extract IMAGE= lines from raw data
ids <- c()
for (i in 1:nrow(raw_data)) {
  val <- raw_data[i, 1]
  if (!is.na(val) && str_starts(as.character(val), "IMAGE=")) {
    id <- str_remove(as.character(val), "IMAGE=")
    ids <- c(ids, id)
  }
}

#add ID's for PCA
pca_df$ID <- ids
```

```
# Clean the PCA data
pca_df <- pca_df %>%
  filter(!is.na(Group), !is.na(PC1), !is.na(PC2)) %>%
  mutate(Group = droplevels(factor(Group)))
```

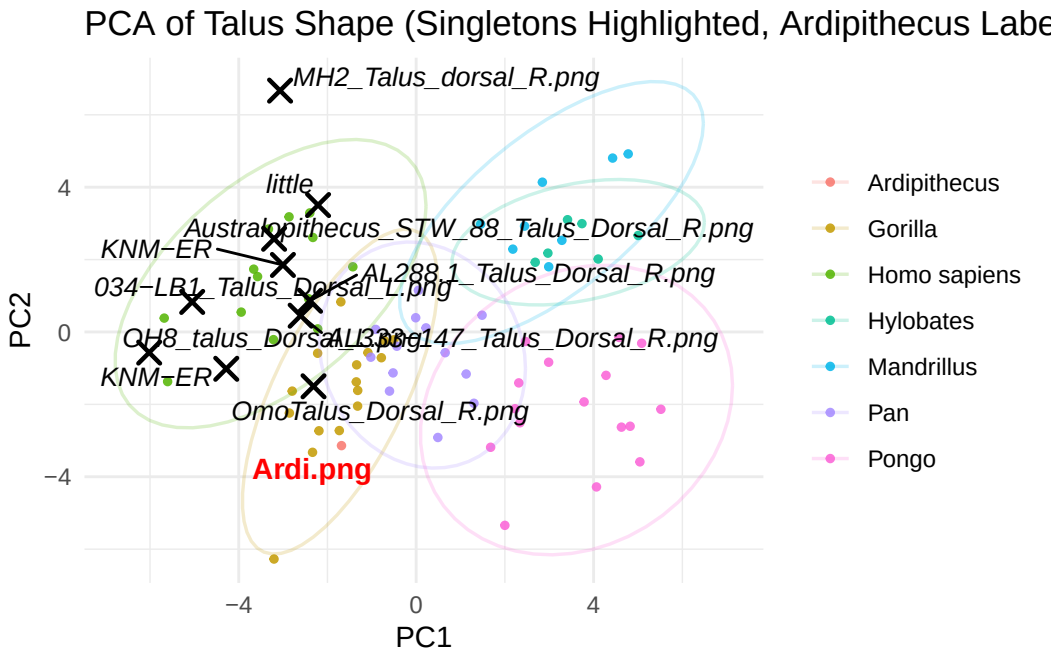
```r
# Identify singleton groups (excluding Ardipithecus)
singleton_df <- pca_df %>%
  group_by(Group) %>%
  filter(n() == 1 & Group != "Ardipithecus") %>%
  ungroup()

# Non-singleton or Ardipithecus specimens
nonsingleton_df <- pca_df %>%
  filter(!(ID %in% singleton_df$ID))
```

#Plot PCA

```r
# Plot
ggplot() +
  # Non-singletons and Ardipithecus: group-colored
  geom_point(data = nonsingleton_df, aes(x = PC1, y = PC2, color = Group), size = 1, alpha =

  # Ellipses only for groups with >1 specimen
  stat_ellipse(data = nonsingleton_df, aes(x = PC1, y = PC2, group = Group, color = Group),
               type = "norm", linetype = "solid", linewidth = 0.6, alpha = 0.2) +

  # Singleton specimens (excluding Ardipithecus): black X
  geom_point(data = singleton_df, aes(x = PC1, y = PC2), color = "black", shape = 4, size = 3

  # Label Ardipithecus (bold)
  geom_text_repel(data = filter(pca_df, Group == "Ardipithecus"), aes(x = PC1, y = PC2, label

  # Label remaining singletons (italic)
  geom_text_repel(data = singleton_df,
                  aes(x = PC1, y = PC2, label = ID),
                  color = "black", fontface = "italic", size = 3.5) +

  labs(title = "PCA of Talus Shape (Singletons Highlighted, Ardipithecus Labeled)",
       x = "PC1", y = "PC2") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

```
Too few points to calculate an ellipse
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_path()`).
```



#DFA

```
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```
library(dplyr)

# Step 1: Identify singleton groups and Ardipithecus
singleton_ids <- pca_df %>%
  group_by(Group) %>%
  filter(n() == 1 & Group != "Ardipithecus") %>%
  pull(ID)

ardi_ids <- pca_df %>%
  filter(Group == "Ardipithecus") %>%
  pull(ID)
```

```r
# Combine those to be predicted
to_predict_ids <- c(singleton_ids, ardi_ids)

# Step 2: Split data into training (non-singleton, non-Ardi) and testing (Ardi + singletons)
train_df <- pca_df %>% filter(!(ID %in% to_predict_ids))
test_df  <- pca_df %>% filter(ID %in% to_predict_ids)

# Step 3: Choose top PCs (e.g., PC1 to PC5)
train_data <- train_df %>% dplyr::select(Group, PC1, PC2, PC3, PC4, PC5)
test_data  <- test_df %>% dplyr::select(PC1, PC2, PC3, PC4, PC5)

# Step 4: Fit DFA model
dfa_model <- lda(Group ~ ., data = train_data)
```

```
Warning in lda.default(x, grouping, ...): groups AL288.1 AL333-147 Ardipithecus
KNM-ER 1464 KNM-ER 1476 LB1 Littlefoot MH2 OH8 Omo STW 88 are empty
```

```r
# Step 5: Predict group membership for Ardi + singletons
dfa_pred <- predict(dfa_model, newdata = test_data)

# Step 6: Add predicted groups to full data
pca_df$PredictedGroup <- pca_df$Group  # start by assuming true group
pca_df$PredictionType <- "True"        # label how each group was assigned

# Replace predicted group labels for Ardi + singletons
pca_df$PredictedGroup[pca_df$ID %in% to_predict_ids] <- dfa_pred$class
pca_df$PredictionType[pca_df$ID %in% to_predict_ids] <- "Predicted"

# Step 7: View predictions
predictions <- pca_df %>%
  filter(PredictionType == "Predicted") %>%
  dplyr::select(ID, Group, PredictedGroup)

print(predictions)
```

```
                                    ID        Group PredictedGroup
1                            Ardi.png Ardipithecus            Pan
74         034-LB1_Talus_Dorsal_L.png          LB1   Homo sapiens
75         AL288.1_Talus_Dorsal_R.png      AL288.1            Pan
76      AL333-147_Talus_Dorsal_R.png    AL333-147   Homo sapiens
```

```
77 Australopithecus_STW_88_Talus_Dorsal_R.png       STW 88   Homo sapiens
78                                     KNM-ER  KNM-ER 1464   Homo sapiens
79                                     KNM-ER  KNM-ER 1476   Homo sapiens
80                                     little   Littlefoot        Gorilla
81                      MH2_Talus_dorsal_R.png          MH2   Homo sapiens
82                      OH8_talus_Dorsal_L.png          OH8   Homo sapiens
83                       OmoTalus_Dorsal_R.png          Omo            Pan
```

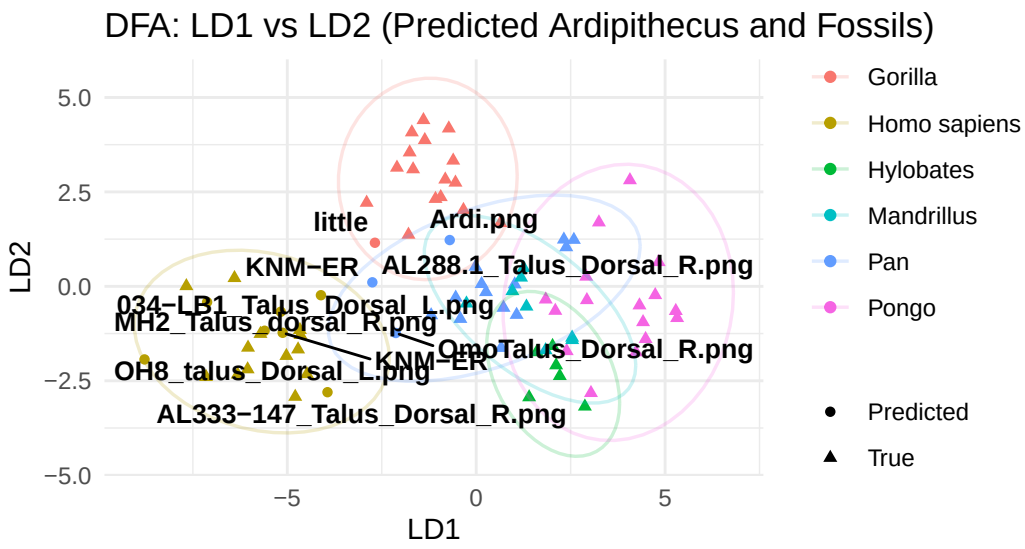
#Plot DFA

```
# Step 1: Get DFA projection (LD scores)
dfa_projection <- as.data.frame(predict(dfa_model)$x)
dfa_projection$ID <- train_df$ID  # IDs for training specimens
dfa_projection$Group <- train_df$Group
dfa_projection$PredictionType <- "True"

# Step 2: Project test specimens (Ardi + singletons) into DFA space
dfa_test_projection <- as.data.frame(predict(dfa_model, newdata = test_data)$x)
dfa_test_projection$ID <- test_df$ID
dfa_test_projection$Group <- dfa_pred$class  # use predicted groups
dfa_test_projection$PredictionType <- "Predicted"

# Step 3: Combine for plotting
dfa_plot_df <- rbind(dfa_projection, dfa_test_projection)

# Step 4: Plot
library(ggrepel)

# Identify Ardipithecus + singleton IDs
ardi_ids <- pca_df %>%
  filter(Group == "Ardipithecus") %>%
  pull(ID)

singleton_ids <- pca_df %>%
  group_by(Group) %>%
  filter(n() == 1 & Group != "Ardipithecus") %>%
  pull(ID)

# Combine labels to apply
label_ids <- c(ardi_ids, singleton_ids)

# Plot with labels
```

```r
ggplot(dfa_plot_df, aes(x = LD1, y = LD2, color = Group, shape = PredictionType)) +
  geom_point(aes(fill = Group)) +
  stat_ellipse(data = dfa_plot_df, aes(x = LD1, y = LD2, group = Group, color = Group),
               type = "norm", linetype = "solid", linewidth = 0.6, alpha = 0.2) +
  # Label Ardipithecus + singletons
  geom_text_repel(
    data = dfa_plot_df %>% filter(ID %in% label_ids),
    aes(label = ID),
    color = "black",
    size = 3.5,
    fontface = "bold"
  ) +
  scale_fill_manual(values=palette()) +
  labs(title = "DFA: LD1 vs LD2 (Predicted Ardipithecus and Fossils)",
       x = "LD1", y = "LD2", shape = "Type") +
  coord_equal() +
  theme_minimal() +
  theme(legend.title = element_blank())
```

```
Warning: The following aesthetics were dropped during statistical transformation: shape.
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```

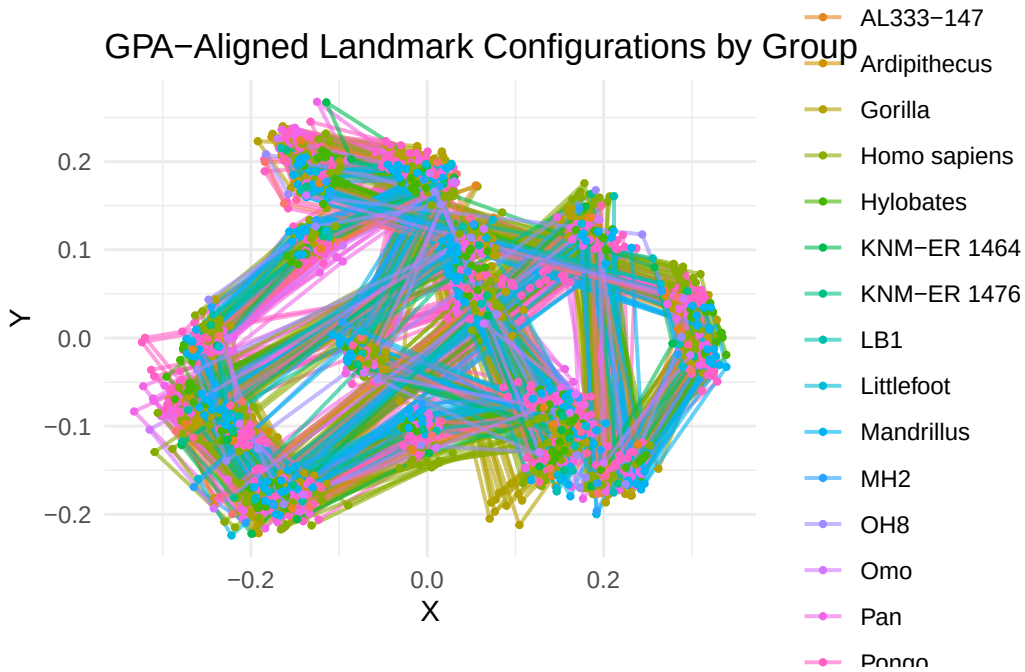
```
Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



#View Landmarks in a plot

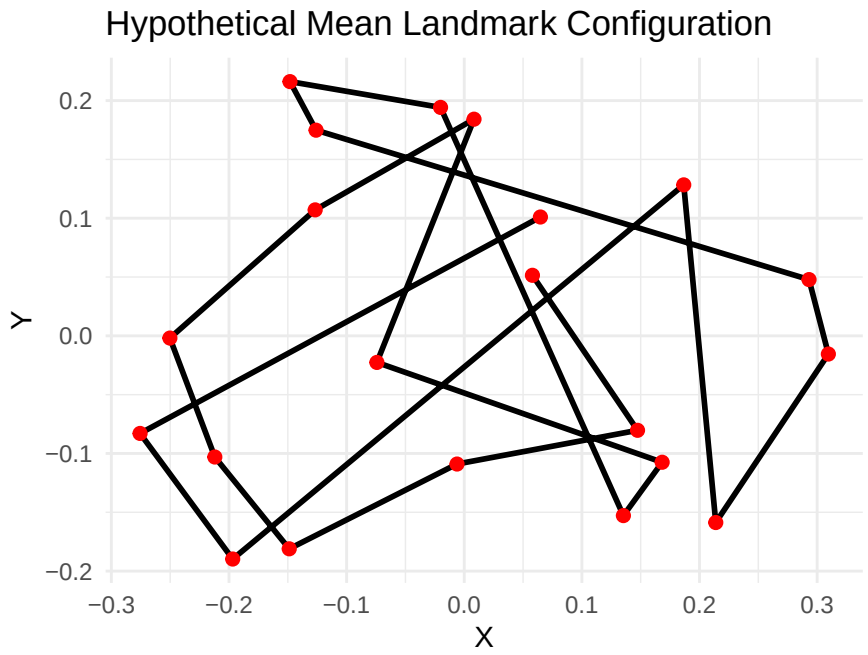
```
library(tidyverse)
library(geomorph)

# Step 1: Extract dimensions
n_landmarks <- dim(gpa$coords)[1]
n_specimens <- dim(gpa$coords)[3]

# Step 2: Build long-format data frame for plotting
coords_list <- lapply(1:n_specimens, function(i) {
  coords <- gpa$coords[,,i]
  df <- as.data.frame(coords)
  colnames(df) <- c("X", "Y")  # name the columns properly
  df$Landmark <- 1:n_landmarks
  df$Specimen <- i
  df$Group <- group[i]
  return(df)
})
land_df <- bind_rows(coords_list)

# Step 3: Plot with ggplot2
ggplot(land_df, aes(x = X, y = Y, group = Specimen, color = Group)) +
```

```
  geom_path(alpha = 0.6, linewidth = 0.7) +
  geom_point(size = 0.8) +
  coord_equal() +
  theme_minimal() +
  labs(title = "GPA-Aligned Landmark Configurations by Group",
       x = "X", y = "Y")
```



#Mean landmarks

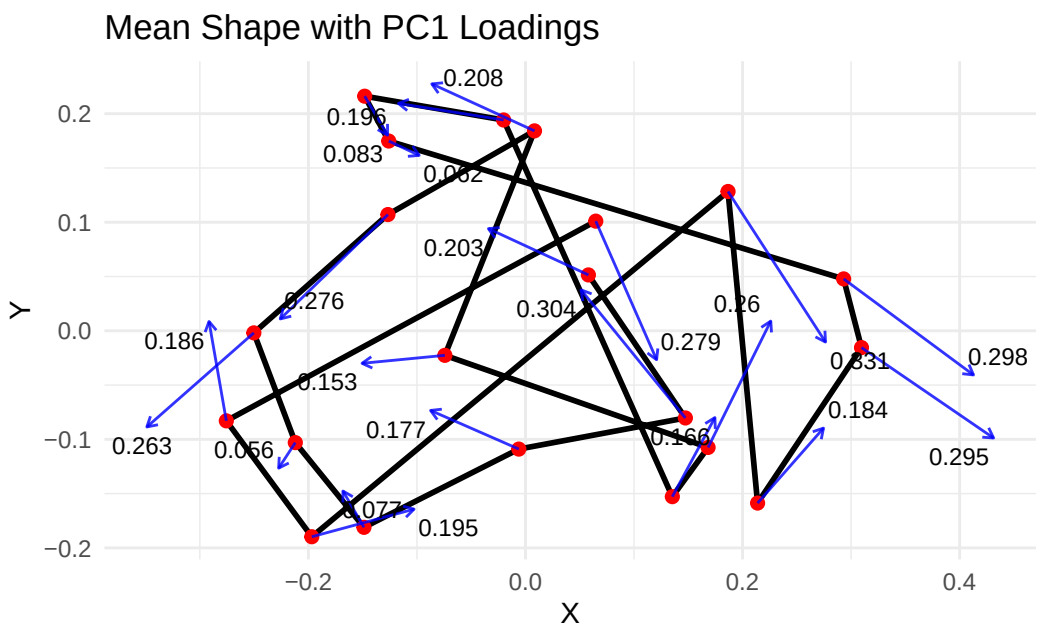
```
# Step 1: Extract consensus shape from GPA
mean_shape <- as.data.frame(gpa$consensus)
colnames(mean_shape) <- c("X", "Y")
mean_shape$Landmark <- 1:nrow(mean_shape)

# Step 2: Plot
ggplot(mean_shape, aes(x = X, y = Y)) +
  geom_path(color = "black", linewidth = 1) +
  geom_point(size = 2, color = "red") +
  coord_equal() +
  theme_minimal() +
  labs(title = "Hypothetical Mean Landmark Configuration",
       x = "X", y = "Y")
```



#Mean Landmarks and their PC1 Loadings

```
# Step 1: Extract mean shape
mean_shape <- as.data.frame(gpa$consensus)
colnames(mean_shape) <- c("X", "Y")
mean_shape$Landmark <- 1:nrow(mean_shape)

# Step 2: Extract PC1 loadings
pc1_vector <- pca$rotation[, 1]

# Step 3: Assign X and Y loadings per landmark
pc1_loadings <- data.frame(
  Landmark = 1:nrow(mean_shape),
  LD_X = pc1_vector[seq(1, length(pc1_vector), by = 2)],
  LD_Y = pc1_vector[seq(2, length(pc1_vector), by = 2)]
)

# Step 4: Calculate magnitude and scaled vectors
pc1_loadings <- pc1_loadings %>%
  mutate(
    magnitude = sqrt(LD_X^2 + LD_Y^2),
    LD_X = LD_X / magnitude,
    LD_Y = LD_Y / magnitude,
```

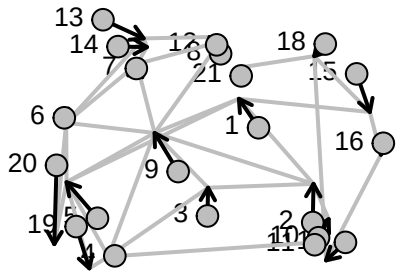
```
    scaled_X = LD_X * magnitude * 0.5,
    scaled_Y = LD_Y * magnitude * 0.5,
    label = round(magnitude, 3)
  )

# Step 5: Merge with mean shape
mean_shape <- left_join(mean_shape, pc1_loadings, by = "Landmark")

# Step 6: Plot with labels
ggplot(mean_shape, aes(x = X, y = Y)) +
  geom_path(color = "black", linewidth = 1) +
  geom_point(size = 2, color = "red") +
  geom_segment(
    aes(xend = X + scaled_X, yend = Y + scaled_Y),
    arrow = arrow(length = unit(0.15, "cm")),
    color = "blue", alpha = 0.8
  ) +
  ggrepel::geom_text_repel(
    aes(x = X + scaled_X, y = Y + scaled_Y, label = label),
    size = 3,
    color = "black",
    box.padding = 0.2
  ) +
  coord_equal() +
  theme_minimal() +
  labs(
    title = "Mean Shape with PC1 Loadings",
    x = "X", y = "Y"
  )
```



#Lowest Pc1 highest Pc2 Wireframe

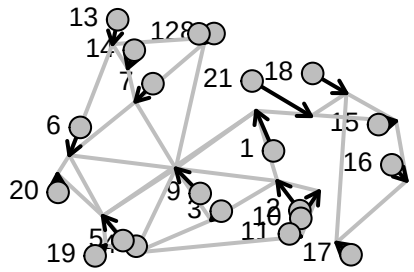
```
library(geomorph)

# Step 1: Identify specimens
pc1_scores <- pca$x[, 1]
pc2_scores <- pca$x[, 2]

lowest_pc1_index <- which.min(pc1_scores)
highest_pc2_index <- which.max(pc2_scores)

# Step 2: Extract GPA-aligned coordinates
specimen_low_pc1  <- gpa$coords[,,lowest_pc1_index]
specimen_high_pc2 <- gpa$coords[,,highest_pc2_index]

# Optional: Define connections between landmarks
# (Example: connect 1-2, 2-3, ..., or use your anatomical connection map)
connections <- matrix(c(
  1,2,
  2,3,
  3,4,
  4,5,
  5,6,
```

```
  6,7,
  7,8,
  8,9,
  9,1,
  9,2,
  9,3,
  9,4,
  9,5,
  9,6,
  9,7,
  2,10,
  10,11,
  11,4,
  8,12,
  12,13,
  13,14,
  14,7,
  13,6,
  1,15,
  15,16,
  16,17,
  17,18,
  18,21,
  4,19,
  19,5,
  5,20,
  20,6,
  1,5,
  15,18), ncol = 2, byrow = TRUE)

# Step 3: Plot wireframe comparison
plotRefToTarget(M1 = specimen_low_pc1,
                M2 = specimen_high_pc2,
                method = "vector",
                links = connections,
                mag = 1,
                label = TRUE,
                col.resid = "blue",
                main = "Wireframe: Lowest PC1 vs Highest PC2")
```



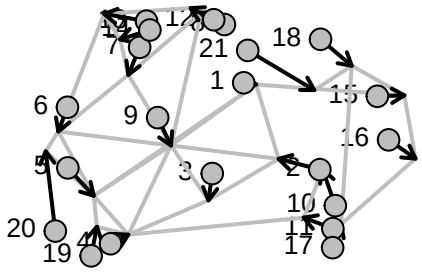
#Highest PC1 vs Lowest PC2

```
library(geomorph)

# Step 1: Identify specimens
lowest_pc2_index <- which.min(pc2_scores)
highest_pc1_index <- which.max(pc1_scores)

# Step 2: Extract GPA-aligned coordinates
specimen_low_pc2  <- gpa$coords[,,lowest_pc2_index]
specimen_high_pc1 <- gpa$coords[,,highest_pc1_index]

# Step 3: Plot wireframe comparison
plotRefToTarget(M1 = specimen_low_pc2,
                M2 = specimen_high_pc1,
                method = "vector",
                links = connections,
                mag = 1,
                label = TRUE,
                col.resid = "blue",
                main = "Wireframe: Highest PC1 vs Lowest PC2")
```



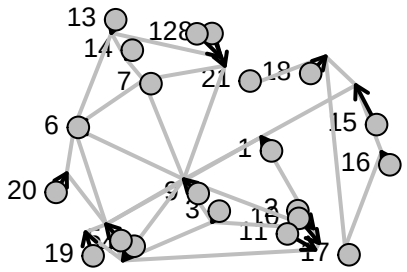
#Lowest Pc1 Lowest Pc2

```
plotRefToTarget(M1 = specimen_high_pc2,
                M2 = specimen_high_pc1,
                method = "vector",
                links = connections,
                mag = 1,
                label = TRUE,
                col.resid = "blue",
                main = "Wireframe: Highest PC1 vs Highest PC2")
```



#Highest Pc1 Highest Pc2

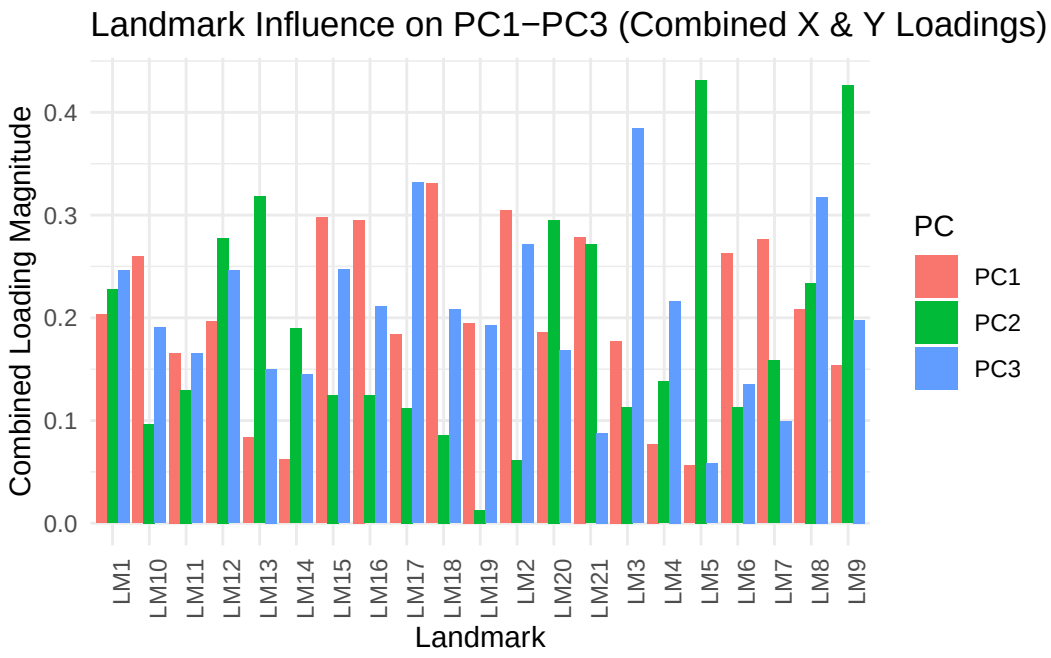
```
plotRefToTarget(M1 = specimen_low_pc2,
                M2 = specimen_low_pc1,
                method = "vector",
                links = connections,
                mag = 1,
                label = TRUE,
                col.resid = "blue",
                main = "Wireframe: Lowest PC1 vs Lowest PC2")
```



#PCA Loadings

```
# Step 1: Extract loadings for PC1-PC3
loadings <- as.data.frame(pca$rotation[, 1:3])
colnames(loadings) <- c("PC1", "PC2", "PC3")

# Step 2: Compute magnitude of XY loadings for each landmark
n_landmarks <- nrow(loadings) / 2
landmark_ids <- 1:n_landmarks

# Combine X and Y loading magnitudes per landmark
landmark_loadings <- data.frame(
  Landmark = landmark_ids,
  PC1 = sqrt(loadings$PC1[seq(1, nrow(loadings), by = 2)]^2 +
             loadings$PC1[seq(2, nrow(loadings), by = 2)]^2),
  PC2 = sqrt(loadings$PC2[seq(1, nrow(loadings), by = 2)]^2 +
             loadings$PC2[seq(2, nrow(loadings), by = 2)]^2),
  PC3 = sqrt(loadings$PC3[seq(1, nrow(loadings), by = 2)]^2 +
             loadings$PC3[seq(2, nrow(loadings), by = 2)]^2)
)

# Step 3: Pivot to long format
long_landmark_loadings <- landmark_loadings %>%
```

```r
  pivot_longer(cols = c(PC1, PC2, PC3), names_to = "PC", values_to = "Loading") %>%
  mutate(Landmark = paste0("LM", Landmark))

# Step 4: Plot
ggplot(long_landmark_loadings, aes(x = Landmark, y = Loading, fill = PC)) +
  geom_col(position = "dodge") +
  theme_minimal() +
  labs(title = "Landmark Influence on PC1-PC3 (Combined X & Y Loadings)",
       x = "Landmark",
       y = "Combined Loading Magnitude") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Step 1: Extract loadings from PCA
loadings <- as.data.frame(pca$rotation[, 1:4])  # PC1 to PC4
n_landmarks <- nrow(loadings) / 2

# Step 2: Calculate combined loading (magnitude of X and Y) for each landmark per PC
landmark_loadings <- data.frame(
  Landmark = paste0("LM", 1:n_landmarks),
  PC1 = sqrt(loadings$PC1[seq(1, nrow(loadings), 2)]^2 + loadings$PC1[seq(2, nrow(loadings),
  PC2 = sqrt(loadings$PC2[seq(1, nrow(loadings), 2)]^2 + loadings$PC2[seq(2, nrow(loadings),
  PC3 = sqrt(loadings$PC3[seq(1, nrow(loadings), 2)]^2 + loadings$PC3[seq(2, nrow(loadings),
```

```
  PC4 = sqrt(loadings$PC4[seq(1, nrow(loadings), 2)]^2 + loadings$PC4[seq(2, nrow(loadings),
)

# Step 3: View or export the result
print(landmark_loadings)
```

```
   Landmark        PC1        PC2        PC3        PC4
1       LM1 0.20303032 0.22739525 0.24628928 0.02958021
2       LM2 0.30449587 0.06120997 0.27124058 0.10285083
3       LM3 0.17749042 0.11317217 0.38441908 0.13606742
4       LM4 0.07728118 0.13781018 0.21617561 0.21781714
5       LM5 0.05618633 0.43157888 0.05828334 0.18428020
6       LM6 0.26256970 0.11242523 0.13550860 0.20364233
7       LM7 0.27633580 0.15826329 0.09918498 0.04739513
8       LM8 0.20809904 0.23386176 0.31780685 0.11075438
9       LM9 0.15331679 0.42611615 0.19782290 0.07660442
10     LM10 0.25983918 0.09665410 0.19036378 0.07314410
11     LM11 0.16553119 0.12993233 0.16499322 0.05719622
12     LM12 0.19630115 0.27777737 0.24617728 0.12150536
13     LM13 0.08341320 0.31794228 0.15014577 0.46428121
14     LM14 0.06202318 0.18971576 0.14500780 0.61154922
15     LM15 0.29767041 0.12499788 0.24692073 0.11427150
16     LM16 0.29497730 0.12459474 0.21116571 0.18314181
17     LM17 0.18387985 0.11235516 0.33163333 0.05299314
18     LM18 0.33051144 0.08573383 0.20823958 0.24455986
19     LM19 0.19517409 0.01266147 0.19237333 0.28931670
20     LM20 0.18583228 0.29500614 0.16822559 0.07336712
21     LM21 0.27885845 0.27158157 0.08763555 0.12465760
```