A Geometric Morphometric approach to lithic backed flake categories

Guillermo Bustos-Pérez

28/2/2022

A geometric morphometric approach to testing discrete categories of backed flakes from recurrent centripetal core reduction.

Guillermo Bustos-Pérez[(1)]
Brad Gravina[(2,3)] Michel Brenet[(3,4)] Francesca Romagnoli[(1)]

[(1)]Universidad Autónoma de Madrid. Departamento de Prehistoria y Arqueología, Campus de Cantoblanco, 28049 Madrid, Spain
[(2)]Musée national de Préhistoire, MC, 1 rue du Musée, 24260 Les Eyzies de Tayac, France
[(3)]UMR-5199 PACEA, Université de Bordeaux, Bâtiment B8, Allée Geoffroy Saint Hilaire, CS 50023, 33615 PESSAC CEDEX, France
[(4)]INRAP Grand Sud-Ouest, Centre mixte de recherches archéologiques, Domaine de Campagne, 242460 Campagne, France

Abstract
Paleolithic lithic assemblages are usually dominated by flakes, which display a high degree of morphological variability. When analyzing Paleolithic lithic assemblages, it is common to classify flakes into categories based on their morphological and technological features, which are linked to the position of the flake in a reduction sequence and how removals are organized in a given production method. For the analysis of Middle Paleolithic lithic assemblages, two categories of flakes are commonly used: core edge flakes and pseudo-Levallois points. A third type, core edge flakes with a limited back, is also commonly found in the archaeological literature, providing an alternative category with a definition that does not match the two previous types but shares many of their morphological and technological features. The present study addresses whether these three flakes constitute discrete categories based on their morphological and technological attributes. Geometric morphometrics are employed on an experimental set composed of the three categories of flakes to quantify morphological variation. Machine learning models and principal components biplots are used to test the discreteness of the categories. The results indicate that geometric morphometrics succeed in capturing the morphological and technological features that characterize each type of product. Pseudo-Levallois points have the highest discreteness of the three technological products, and while some degree of mixture exists between core edge flakes and core edge flakes with a limited back, they are also highly distinguishable. We conclude that the three categories are discrete and can be employed in technological lists of products for the analysis of lithic assemblages and that geometric morphometrics is useful for testing for the validity of categories.

Key words: lithic analysis; lithic technology; geometric morphometrics; machine learning; Middle Paleolithic; Levallois; discoidal

1. Introduction

Lithic artifacts constitute some of the most important and abundant remains in Paleolithic sites. When analyzing lithic assemblages, in addition to taking metric measurements and attributes it is common to classify unretouched flakes according to their morphology and technological features. This classification is a crucial part of lithic analysis since it classifies flakes into technological categories. These categories are technological because the retained features and morphology are indicative of the knapping method from which they were generated. These technological products usually reflect different knapping strategies, different stages of reduction, but also variations in the organization of removals and exploitation of the surface. Well known examples of technological classifications of flakes are the bipolar/on anvil flakes (Callahan, 1996; Hayden, 1980);

1.1 Load packages, data and procrustes analysis

```
# Load packages
list.of.packages <- c("tidyverse", "caret", "Morpho")
lapply(list.of.packages, library, character.only = TRUE)
```

```
## [[1]]
##  [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
##  [7] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [13] "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "caret"     "lattice"   "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "Morpho"    "caret"     "lattice"   "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
```

```
rm(list.of.packages)

load("Data/Flakes LM rotated.RData")
Att <- read.csv("Data/Attributes data.csv")
```

```
# PCA on rotated landmarks
pca <- prcomp(LM.DF, scale. = TRUE)
summary(pca)$importance[1:3, 1:25]
```

```
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     18.05288 16.62783 12.83087 10.83128 10.42072 8.299316
## Proportion of Variance  0.21385  0.18142  0.10803  0.07698  0.07125 0.045200
## Cumulative Proportion   0.21385  0.39527  0.50330  0.58028  0.65153 0.696730
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     7.73039 7.439897 6.710911 6.173021 5.368746 4.773021
## Proportion of Variance 0.03921 0.036320 0.029550 0.025000 0.018910 0.014950
## Cumulative Proportion  0.73594 0.772260 0.801810 0.826810 0.845730 0.860670
```

```
##                            PC13    PC14     PC15     PC16    PC17    PC18
## Standard deviation     4.562145 4.44228 3.803028 3.750857 3.54599 3.23142
## Proportion of Variance 0.013660 0.01295 0.009490 0.009230 0.00825 0.00685
## Cumulative Proportion  0.874330 0.88728 0.896770 0.906000 0.91425 0.92110
##                            PC19    PC20     PC21     PC22     PC23    PC24
## Standard deviation     2.956198 2.70170 2.649925 2.516422 2.466077 2.38239
## Proportion of Variance 0.005730 0.00479 0.004610 0.004160 0.003990 0.00372
## Cumulative Proportion  0.926840 0.93163 0.936240 0.940390 0.944380 0.94811
##                            PC25
## Standard deviation     2.327281
## Proportion of Variance 0.003550
## Cumulative Proportion  0.951660
```

```
# store PCA values in a dataframe and add ID's
PCA_Coord <- as.data.frame(pca$x)
PCA_Coord$ID <- filenames
PCA_Coord$Core <- str_sub(PCA_Coord$ID, end = 2)
```

```
# Left joined with the attribute database
PCA_Coord <- left_join(PCA_Coord, Att, by = "ID")
```

2. Methods

2.1 Experimental assemblage

```
# Count artifact type per class
PCA_Coord %>% group_by(ARTIFACTTYPE) %>%
  summarise(Count = n())
```

```
## # A tibble: 3 x 2
##   ARTIFACTTYPE                Count
##   <chr>                       <int>
## 1 Core Edge Flake                30
## 2 Core edge with limited back    93
## 3 pseudo-Levallois Point         16
```

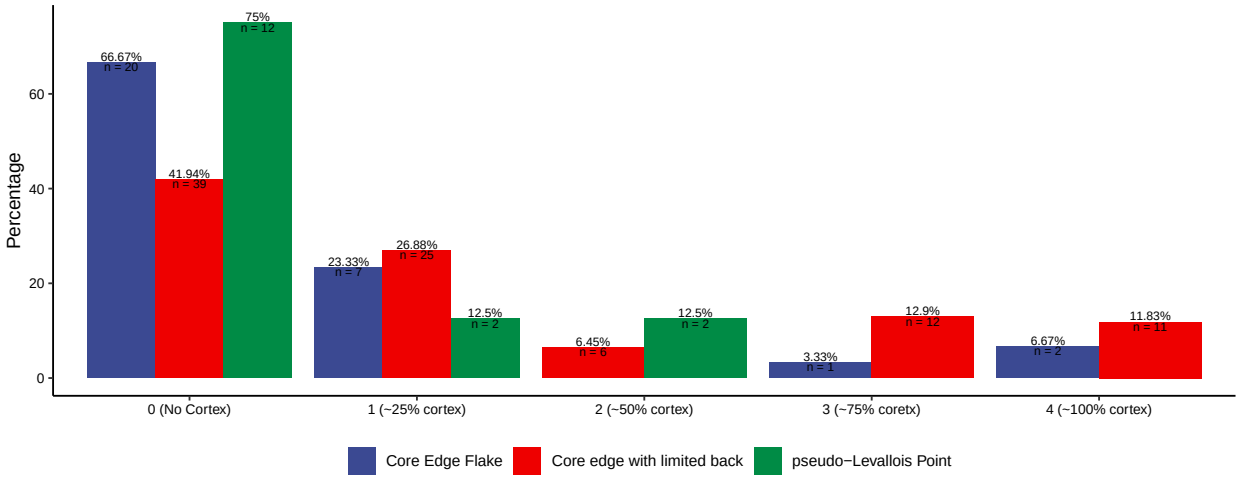
```
# Cortex per artefact type
PCA_Coord %>% group_by(ARTIFACTTYPE) %>%
  count(CORTEX) %>%

  mutate(Percentage = round(n/sum(n)*100, 2)) %>%

  ggplot(aes(CORTEX, Percentage, fill = ARTIFACTTYPE)) +
  geom_col(position = "dodge") +
  ggsci::scale_fill_aaas() +
  xlab(NULL) +
  geom_text(aes(label = paste0(Percentage, "%")),
            vjust= -0.2, size = 2.5,
            position = position_dodge(.9)) +
  geom_text(aes(label = paste("n =", n)),
            vjust= "top", size = 2.5,
```

```
            position = position_dodge(.9)) +
  labs(fill = NULL) +

  theme_classic() +
  theme(
    legend.position = "bottom",
    axis.text = element_text(color = "black", size = 8))
```



2.2 Geometric Morphometrics

2.3 Machine Learning and resampling techniques

2.4 Pre processing and training the models

```
#### Pre processing data
# Change syntax of output
PCA_Coord <- PCA_Coord %>% mutate(
  New_Art.Type =
    case_when(
      ARTIFACTTYPE == "Core Edge Flake" ~ "ED",
      ARTIFACTTYPE == "Core edge with limited back" ~ "EDlb",
      ARTIFACTTYPE == "pseudo-Levallois Point" ~ "p_Lp"
    ))

# Set factors
PCA_Coord$New_Art.Type <- factor(
  PCA_Coord$New_Art.Type,
  levels = c("ED", "EDlb", "p_Lp"),
  labels = c("ED", "EDlb", "p_Lp"))
```

```
# Set formula and validation
# Formula
frmla <- as.formula(
  paste("New_Art.Type", paste(colnames(PCA_Coord[,1:25]), collapse = " + "), sep = " ~ "))
```

```
# Validation
trControl <- trainControl(method  = "repeatedcv",
                          verboseIter = TRUE,
                          number  = 10,
                          repeats = 50,
                          savePredictions = "final",
                          classProbs = TRUE)
```

As mentioned previously, this is a unbalanced data set. Balancing to train the models correctly is done using the `groupdata2` package using the parameter **"mean"** for size. This means that each group will be up or down sampled to the result of dividing data set size between number of groups (46.3333333).

- A `tibble()` called `All_Results` is set to store the results from each model training.

Each loop works in the following steps:

1. The original data set is randomly up and down sampled for each target category.
2. Model is trained following the provided hyperparameters and data pre-processing.
3. Results from the trained model are extracted and binned to the `All_Results` tibble.
4. Steps 1 to 3 are repeated 30 times per model.

```
#### Train the models
# Set tibble in which results of each model will be stored
All_Results <- tibble()

# LDA model training
repeat {
  # Balance the dataset
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")
  # Train the model
  Model <- train(frmla,
                 Balanced,
                 method = "lda",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')

  # Bind model results and type of model
  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "LDA"))
```

```
  x = nrow(All_Results)
  if (x >= 30){
    break
  }
}

# Knn model training
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                     Balanced,
                     method = "knn",
                     trControl = trControl,
                     preProc = c("center", "scale"),
                     tuneGrid   = expand.grid(k = 2:30),
                     metric = "Accuracy")

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]],
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]],
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "KNN"))

  x = nrow(All_Results)
  if (x >= 60){
    break
  }
}

# Logistic regression
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "glmnet",
                 family = 'multinomial',
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy")

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
```

```
                          AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]],
                          AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]],
                          AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                          Model = "Log. Reg."))

  x = nrow(All_Results)
  if (x >= 90){
    break
  }
}

# Random forest
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "ranger",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]],
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]],
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                             Model = "Random Forest"))

  x = nrow(All_Results)
  if (x >= 120){
    break
  }
}

# SVM linear
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "svmLinear",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')
```

```
  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "SVM Linear"))

  x = nrow(All_Results)
  if (x >= 180){
    break
  }
}

# SVM radial
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "svmRadial",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "SVM Radial"))

  x = nrow(All_Results)
  if (x >= 210){
    break
  }
}

# SVM Poly
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "svmPoly",
```

```
                  trControl = trControl,
                  preProc = c("center", "scale"),
                  metric = "Accuracy",
                  importance = 'impurity')

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "SVM Poly"))

  x = nrow(All_Results)
  if (x >= 240){
    break
  }
}

# C5.0 Tree
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "C5.0",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "C5.0 Tree"))

  x = nrow(All_Results)
  if (x >= 270){
    break
  }
}

# Naïve Bayes
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
```

```
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "nb",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy",
                 importance = 'impurity')

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "Naïve Bayes"))

  x = nrow(All_Results)
  if (x >= 300){
    break
  }
}

# Boosted Tree
repeat {
  Balanced <- groupdata2::balance(PCA_Coord,
                                  size = "mean",
                                  cat_col = "New_Art.Type")

  Model <- train(frmla,
                 Balanced,
                 method = "gbm",
                 trControl = trControl,
                 preProc = c("center", "scale"),
                 metric = "Accuracy")

  All_Results <- rbind(All_Results,
                       tibble(
                         Accuracy = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[1]],
                         Kappa = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[2]],
                         AccuracyLower  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[3]]
                         AccuracyUpper  = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[4]]
                         AccuracyNull = confusionMatrix(Model$pred$pred, Model$pred$obs)$overall[[5]],
                         Model = "Boosted Tree"))

  x = nrow(All_Results)
  if (x >= 300){
    break
  }
}
```

3. Results

7. References

Callahan, E., 1996. The bipolar technique: The simplest way to make stone tools for survival. Bulletin of Primitive Technology 12, 16–20.

Hayden, B., 1980. Confusion in the Bipolar World: Bashed Pebbles and Splintered Pieces. Lithic Technology 9, 2–7. https://doi.org/10.1080/01977261.1980.11754456