Exploring the relationship between Grammar, Phonology, Music and Genes in Northern Asia and Greenland

Peter Ranacher and Balthasar Bickel

2018-07-27

Contents

List of Figures

In this document we explore the correlation between music, grammar, phonology and genetics for thirteen peoples in Northern Asia and Greenland.

1 Packages and functions

For the analysis we mainly use the following three packages:

- `ade4` provides tools for multivariate data analysis (mostly for ecological data)
- `adespatial` provides tools for multiscale spatial analysis of multivariate data
- `vegan` provides tools for ordination methods and diversity analysis

```
library(adespatial)
library(ade4)
library(broom)
library(vegan)
library(knitr)
library(ape)
library(gclus)
library(sp)
library(dplyr)
library(ggplot2)
library(leaflet)
library(reshape2)
library(extrafont)
library(grid)
library(rgdal)
library(missMDA)
library(pcaMethods)
library(FactoMineR)
library(spdep)
library(kableExtra)
library(ggpolypath)
library(RColorBrewer)
library(cowplot)
library(xtable)
font_import(pattern ="FTLC", prompt=F)
```

1.1 Functions

In this section we document all custom-defined functions used in the analysis, including functions for performing the redundancy analysis, helper functions for data preprocessing and functions for visualizing the results.

1.1.1 Redundancy Analysis

```
# Distance-based Moran's Eigenvector (dbMEM) Analysis
random_points_to_dbmem <- function(r_points, print_count=FALSE){
#' This function computes the dbMEMs for each random point sample in r_points
#' @param r_points: the random points (SpatialPointsDataFrame)
#' @param print_count: Print the number of processed samples when iterating over r_points?
```

```r
#' @return a list comprising the dbMEMs for each sample

  if (class(r_points)[1] != "SpatialPointsDataFrame") {
    stop("please provide a SpatialPointsDataFrame")
  }
  n_sample <- max(r_points$sample_id)

  epsilon <- 0.1
  geo_pco  <- list()

  for (j in 1:n_sample) {

      # Get all points of sample j
      points <- r_points[r_points$sample_id==j, ]

      # Compute distances between all points in the sample
      mat <- spDists(points, points)

      # Compute the mst, find its longest edge and use as a threshold
      mst_1 <- spantree(mat)
      mst_le <- max(mst_1$dist)

      # Add a small epsilon (for numerical stability)
      thresh <- mst_le + epsilon

      # Find all nearest neighbors within the distance threshold
      nb <- dnearneigh(points, 0, thresh)

      # Normalize the data
      spwt <- lapply(nbdists(nb, points), function(x) 1 - (x/(4 * thresh))^2)

      # Compute weighted neighbor list
      lw <- nb2listw(nb, style = "B", glist = spwt, zero.policy = TRUE)

      # Compute MEMs which correspond to positive autocorrelation
      res <- scores.listw(lw, MEM.autocor = "positive")

      rownames(res) <- points$nam_label
      colnames(res) <- paste("geo_pco_", seq(1,ncol(res)), sep="")
      res <- res[order(rownames(res)), ]
      geo_pco[[paste('sample_', j, sep="")]] <- as.data.frame(res)

      if (print_count) {
        if (j%%1000 == 0) {
          print(paste(j, " samples processed"))}}}
  return (geo_pco)}

# Redundancy Analysis
rda_wrapper <- function (response, explanatory, random_geo_pco=NULL, print_count=FALSE) {
#' This function computes a (spatially constrained) RDA
#' @param response: the repsonse variable
#' @param explanatory: the explanatory variable
#' @param random_geo_pco dbMEMs of random spatial point patterns
```

```
#' @param print_count: Print the number of processed samples when iterating over r_points?
#' @return a list comprising the rda results for each sample

  if (is.null(rownames(response)) & is.null(rownames(explanatory))) {
    stop("Row names of the response and the explanatory variable must be defined!")
  }

  if (any(rownames(response) != rownames(explanatory))) {
    stop("Row names of the response and the
         explanatory variable must be identical and match in order!")
  }

  ex_name <- sub('\\_.*', '', colnames(explanatory)[1])
  re_name <- sub('\\_.*', '', colnames(response)[1])


  if (is.null(random_geo_pco)) {

    # There is no geo-constraint, hence perform regular RDA
    rda <- rda(X = explanatory, Y = response)
    r2 <- RsquareAdj(rda)$r.squared

    # Compute the adjusted explained variance
    r2_adj <- RsquareAdj(rda)$adj.r.squared
    sig <- anova.cca(rda, step = 10000)$`Pr(>F)`[1]

    # Perform permutations
    perm <- permute_rda(10000, response, explanatory)
    rda_results <- list(r2=r2, r2_adj=r2_adj,
                        perm_r2=perm$r2, perm_r2_adj=perm$r2_adj,
                        explanatory=ex_name, response=re_name, sig=sig, geo=FALSE)}

  # There is a geo-constraint, perform partial RDA
  else{
    rda_results <- list()

  for (i in 1:n_sample) {
    sample <- paste("sample_", i, sep="")
    constraint <- random_geo_pco[[sample]]
    if (is.null(constraint)) {
      rda_results[[sample]] <- NULL}

    else {
      rda <- rda(X = explanatory, Y = response, Z = constraint)

      # Compute the (adjusted) explained variance
      r2 <- RsquareAdj(rda)$r.squared
      r2_adj <- RsquareAdj(rda)$adj.r.squared

      # Perform permutations
      perm <- permute_rda(10, response, explanatory, constraint)

      rda_results[[sample]] <- list(r2=r2, r2_adj=r2_adj,
```

```
                                    perm_r2=perm$r2, perm_r2_adj=perm$r2_adj,
                                    explanatory=ex_name, response=re_name, geo=TRUE)}
    if (print_count){
     if (i%%100 == 0) {
       print(paste(i, " samples processed"))}}}}
 return(rda_results)}

# Permute RDA
permute_rda <- function(n_perm, explanatory, response, constraint=NULL) {
#' This function runs n_perm RDAs with permuted data
#' @param n_perm: the number of permutations
#' @param response: the repsonse variable
#' @param explanatory: the explanatory variable
#' @param constraint the constraint
#' @return a list comprising the RDA results for each permutation

  if (!is.numeric(n_perm)){stop("The number of permutations must
                                be .. well... a number.")}
  permutation_results <- data.frame(r2_adj=rep(NA, n_perm))

  for (i in 1:n_perm){
    # permute the response
    permutation_order <- sample(1:nrow(response))
    response <- response[permutation_order, ]
    # Geo-contraint?
    if (is.null(constraint)) {rda <- rda(X = explanatory, Y = response)}
    else {rda <- rda(X = explanatory, Y = response, Z = constraint)}

    r2 <-  RsquareAdj(rda)$r.squared
    r2_adj <- RsquareAdj(rda)$adj.r.squared

    permutation_results[i, c("r2")] <- r2
    permutation_results[i, c("r2_adj")] <- r2_adj}

  return(permutation_results)}
```

1.1.2 Helper functions for data preprocessing

```
trim_data <- function(data.list, trim.to=siberia_metadata_all, extra.coverage=.7) {
 #' This function trims the linguistic input data and only keeps variables
 #' in the study area with one data point per language and non-constant values

 #' @param data.list the data to be trimmed
 #' @param trim.to contains the ids of those languuages that are retained after trimming
 #' @param extra.coverage the percentage of covered data
 #' @return the trimmed data

  lgs <- lapply(data.list, function(l) {
      l$UULID <- ifelse(l$isocode %in% c('bxm','bxr'),
                '[i-bua][a-1095][g-buri1258]',
               paste(l$UULID))
      subset(l, UULID %in% trim.to$UULID)
```

```
    })
  vars <- lgs[sapply(lgs, function(l) {
       length(l$UULID)==length(unique(l$UULID)) &
       length(unique(l[,1]))>1 &
         length(unique(l$UULID)) >= floor(extra.coverage*length(trim.to$UULID))
      })]
  return(lapply(vars, function(l) l[,c(1,3)]))}


compute_coverage <- function(data.list, gg=siberia_metadata) {
 #' This function computes the coverage of all languages
 #' @param data.list the input data
 #' @param gg a data.frame comprising the languages for which the coverage is computed
 #' @return the input data with the coverage added as a column for each language

  x <- sapply(gg$UULID, function(l) {
          coverage <- round(mean(sapply(data.list, function(v) { l %in% v$UULID })),2)*100
        })
  df <- data.frame(UULID=names(x), Coverage=x)
  gg$Language <- rownames(gg)
  df.g <- merge(df, gg)
  return(df.g)}


flatten <- function(data.list, gg=siberia_metadata) {
 #' This function flattens the nested linguistic data
 #' @param data.list: the input data to be flattened
 #' @param gg: a data.frame comprising the languages which are flattened
 #' @return the flattened data

  df.list <- lapply(seq_along(data.list), function(v) {
    df <- data.list[[v]]
    var.name <- gsub('(.*\\$)', '\\2', names(data.list)[v])
    names(df)[1] <- var.name
    return(dplyr::select(df, UULID, dplyr::everything()))
    })
  df.flat <- Reduce(function(x,y) dplyr::full_join(x, y, by='UULID'), df.list)
  rownames(df.flat) <- sapply(df.flat$UULID, function(x) rownames(gg[gg$UULID %in% x,]))
  return(df.flat)}


pairwise_correlation <- function (data) {
  #' This function computes the pairwise correlation between all variables
  #' @param data: the input data
  #' @return the pairwise correlations

  # Collect all PCs and PCos in one data.frame
  all_pcs <- as.data.frame(data)
  colnames(all_pcs) <- sub('.*\\.', '', colnames(all_pcs))
  colnames(all_pcs)  <- gsub("pc", "PC", colnames(all_pcs))

  results <- data.frame(var1 = character(),
                        var2 = character(),
```

```
                        r = numeric(),
                        sig = character(),
                        stringsAsFactors=FALSE)
  done <- character()

  for (i in colnames(all_pcs))
    for (j in colnames(all_pcs)){
      cell_id <- paste (i, j, " ")

      if (i != j & !cell_id %in% done){

        rev_cell_id <- paste (j, i, " ")
        done <- c(done, cell_id, rev_cell_id)
        res <- cor.test(all_pcs[,i], all_pcs[,j])
        results[nrow(results)+1, c(1,2)] <- c(gsub("_"," ", i),
                                              gsub("_"," ", j))
        results[nrow(results), 3] <- abs(res$estimate)

        if (res$p.value > 0.05) {sig <- ""}
        else if (res$p.value <= 0.01) {sig <- "**"}
        else if (res$p.value > 0.01 & res$p.value <= 0.05) {sig <- "*"}

        results[nrow(results), 4] <- sig}}

  return (results)}
```

1.1.3 Visualizations (Themes and Functions)

```
# Language maps
# Theme
theme_map <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    legend.position ="none",
    plot.background = element_rect(fill = "#f9f9f9", size=1,linetype="solid",
                                   color="black"), #"#f5f5f2"
    panel.background = element_rect(fill = "#f9f9f9", color = NA),
    ...)}

# Plot function
plot_language_map <- function(lang_poly, lang_labels_pos, languages) {
  #' This function plots the language polygons
```

```
  #' @param lang_poly: the language polygons
  #' @param lang_labels _pos: the geogrpahic positions of the language labels
  #' @param languages: the languages displayed in the map
  #' @return a ggplot of the map

  cols <- brewer.pal(length(languages),"Set1")
  lang_poly <- lang_poly[lang_poly$nam_label %in% languages, ]
  lang_labels <- lang_labels_pos[lang_labels_pos$nam_label %in% languages, ]

  gg <- ggplot()
  gg <- gg + coord_map('orthographic',
                       orientation=c(120,0,180),
                       ylim=c(33,80))
  gg <- gg + geom_polygon(data=world, aes(x=long, y=lat, group=group), color=NA,
                          fill='lightgrey')
  gg <- gg + geom_polypath(data=lang_poly, aes(x=long, y=lat,
                           group=group, fill=factor(nam_label)), size=2, colour=NA)
  gg <- gg + geom_point()
  gg <- gg + geom_text(data=lang_labels,
                       aes(x=long, y=lat, label=nam_label,
                           vjust=-1.1, hjust = .3), size=2)
  gg <- gg + scale_fill_manual(values=alpha(cols, 0.8))
  gg <- gg + theme_map()
  return(gg)}

# Scree plot
# Theme
theme_scree <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    axis.line = element_blank(),
    legend.background = element_rect(fill = "#f5f5f2", color = NA),
    panel.border = element_blank(),
    ...)}

# Plot function
plot_pc_scree <- function(eigenval, title, type="PC") {
#' This function generates the scree plot for either the PCs or the PCos
#' @param eigenval: the eigenvalues of the PC(o)
#' @param title: the title of the plot
#' @param type: the type of the plot (either PC or PCo)
#' @return the scree plot (ggplot)

  if (type == "PC") {
    pc_index = seq(1:nrow(eigenval))
    eigenval = data.frame(val = eigenval[, 1],
                          idx = pc_index,
                          rel = eigenval[, 2],
                          cum = eigenval[, 3])
    pc_legend = "Principal Components"
    subtitle= paste("Explained variance by ", pc_legend)
    pc_labels = paste(rep(c("PC"), nrow(eigenval)), seq(1:nrow(eigenval)))
```

```
    ann_offset_y = 6}

  else if (type == "PCo"){
    pc_index = seq(1:length(eigenval))
    eigenval = data.frame(val = eigenval,
                          idx = pc_index,
                          rel = eigenval/sum(eigenval)*100,
                          cum = cumsum(eigenval/sum(eigenval)*100))
    pc_legend = "Principal Coordinates"
    subtitle= paste("Explained variance by ", pc_legend)
    pc_labels = paste(rep(c("PCo"), nrow(eigenval)), seq(1:nrow(eigenval)))
    ann_offset_y = 10}

  else stop("Type must be either PC or PCo")
  p <- ggplot(data=eigenval,aes(x=idx, y=rel))
  p <- p + geom_bar(stat="identity")
  p <- p + xlab(pc_legend) + ylab("Eigenvalues (%)")
  p <- p + geom_line(data=eigenval, aes(x=idx, y=cum), colour="orange")
  p <- p + geom_point(data=eigenval, aes(x=idx, y=cum), colour="orange")
  p <- p + scale_x_discrete(limits=pc_labels)
  p <- p + labs(title=title, subtitle=subtitle)
  p <- p + annotate ("label", x = tail(pc_index, n=1)-2.3,
                     y = tail(eigenval$cum, n=1) - ann_offset_y,
                     label="Cumulative eigenvalues",
                     colour ="orange", label.size=NA, size=3.5)
  p <- p + theme_scree()
  return(p)}

# Heat maps
# Theme
theme_heat_maps <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    legend.background = element_rect(fill = NA, color = NA),
    legend.direction = "horizontal",
    legend.position = "bottom",
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_blank(),
    axis.title = element_blank(),
    axis.text = element_text(size=15),
    axis.text.x = element_text(vjust = 0, angle = 90, hjust=0),
    plot.margin=unit(c(3,1,1.5,1.2),"cm"),

    ...)}

# Plot function
heat_map_pc <- function(pcs, type="PCo"){
  #' This function plots a heat map of the PCs/PCos
  #' @param pcs: the principal components/coordinates
```

```
  #' @param type: either PC or PCo
  #' @return the heat map (ggplot)

  reorder <- c("Ainu", "Japanese", "Korean", "Buryat", "Even", "Evenki", "Yakut", "Selkup",
               "Nganasan", "Chukchi", "Koryak", "Yukagir", "West Greenland")
  pcs <- pcs[reorder, ]

  cnames <- colnames(pcs)

  # Normalize the PCs/PCos for each factor
  pcs_norm <- sapply(cnames, function (y) {
    x <- pcs[, y]
    x_norm <- (x-min(x))/(max(x)-min(x))
    return (x_norm)}, USE.NAMES = T)

  if (type == "PC"){
    colnames(pcs_norm) <- paste("PC", seq(1, ncol(pcs_norm)))}
  if (type == "PCo"){
    colnames(pcs_norm) <- paste("PCo", seq(1, ncol(pcs_norm)))}

  rownames(pcs_norm) <- rownames(pcs)
  melted <- melt(pcs_norm)

  # Create heat map
  h <- ggplot(data = melted, aes(Var1, ordered(Var2, levels=rev(levels(Var2))),
                                 fill=value))

  h <- h + geom_tile(colour="grey")
  h <- h + scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                                limit = c(0,1), midpoint=0.5, name = NULL)
  h <- h + theme_heat_maps()
  h <- h + coord_fixed()
  h <- h + scale_x_discrete(position = "top")
  return (h)}


# Correlation matrices
# Theme
theme_corr <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    legend.background = element_rect(fill = NA, color = NA),
    legend.direction = "horizontal",
    legend.position = c(0.65, 0.15),
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_blank(),
    axis.title = element_blank(),
    axis.text.y = element_text(vjust = 0.5, size = 10),
    axis.text.x = element_text(angle=90, hjust=0.5, vjust = 0.5, size = 10),
```

```
    plot.margin=unit(c(3,1,1.5,1.2),"cm"),
    ...)}

# Plot function
plot_correlation <- function(data){
  #' This function visualizes the correlation matrix
  #' @param data: the correlation matrix
  #' @return the plot of the correlation matrix
p <- ggplot(data, aes(var1, var2, fill = r))
p <- p + geom_tile(color = "grey")
p <- p + scale_fill_gradient(low = "white", high = "red", limit = c(0, 0.9),
                             name="Pearson's Correlation Coefficient \n(absolute values)")
p <- p + geom_text(aes(var1, var2, label = sig))
p <- p + annotate("text", label = "Significance: * p \u2264 0.05, ** p \u2264 0.01",
                  x = 13.2, y = 1.5,
                  family = "Frutiger Light Condensed")
p <- p + scale_x_discrete(position="top")
p <- p + theme_corr()
p <- p + coord_fixed()
return(p)}

# Density Plots
# Theme
theme_density <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    legend.background = element_rect(fill = NA, color = NA),
    legend.direction = "horizontal",
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.text.y = element_blank(),
    axis.text.x = element_text(angle=90, hjust=0.5, vjust = 0.5, size = 10),
    plot.margin=unit(c(3,1,1.5,1.2),"cm"),
    ...
  )
}

# RDA matrix
# Theme
theme_corr_rda <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Frutiger Light Condensed", color = "#22211d"),
    legend.background = element_rect(fill = NA, color = NA),
    legend.direction = "horizontal",
    legend.position = c(0.2, 1.1),
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_blank(),
    axis.title = element_text(size = 15, hjust=0.5, vjust=0.4),
```

```
    axis.text = element_text(vjust = 1, size = 10),
    plot.margin=unit(c(3,1,1.5,1.2),"cm"),
    ...
  )
}

# Plot function
plot_rda_matrix <- function(rda_matrix){
  #' This function plots the results from the redundancy analysis
  #' @param rda_matrix: a matrix comprising the rda results
  #
  c <- ggplot(data = rda_mat, aes(response, explanatory, fill = r2_adj))
  c <- c + geom_tile(color = "grey")
  c <- c + scale_fill_gradient(low = "white", high = "red",
                               limit = c(0,0.81),
  name=expression(paste("Explained variance \nin response (adjusted ",R^{2},")    ")))
  c <- c + geom_text(aes(response, explanatory,
                         label = paste(round(r2_adj,2), sig_level)),
                     color = "black", size = 3)
  c <- c + xlab("\nresponse") + ylab("explanatory\n")
  c <- c + theme_corr_rda()
  c <- c + coord_fixed()
  return(c)}

# Plot function
density_plot <- function(rda_res, r2_type) {
  #' This function computes a density plot of the observed and permuted
  #' (adjusted) R squared in an RDA
  #' @param rda_res: a list comprising the RDA results (output of function rda wrapper)
  #' @param r2_type: either "r2" or "r2_adj"
  #' @return the density plot

  if (!r2_type %in% c("r2", "r2_adj")) {
    stop("r2_type must be r2 or r2_adj")}
  else if (r2_type=="r2"){label="R squared"}
  else if (r2_type=="r2_adj"){label="Adjusted R squared"}


  ex <- paste(toupper(substr(rda_res$explanatory, 1, 1)),
              substr(rda_res$explanatory, 2,
                     nchar(rda_res$explanatory)), sep="")

  re <- paste(toupper(substr(rda_res$response, 1, 1)),
              substr(rda_res$response, 2,
                     nchar(rda_res$response)), sep="")

  perm_distr <- as.data.frame(rda_res[paste("perm", "r2", sep="_")])
  colnames(perm_distr) <- "permuted"
  p_below <- ecdf(perm_distr$permuted)(rda_res[[r2_type]])
  p_above<- 1- p_below

  p_below <- paste(format(round(p_below, 3) *100, nsmall=1), "%")
  p_above <- paste(format(round(p_above, 3) *100, nsmall=1), "%")
```

```
  d <- ggplot()
  d <- d + geom_density(data=perm_distr, aes(x=permuted), alpha=.3,
                        colour=NA, fill="grey")
  d <- d + geom_segment(aes(x = rda_res[[r2_type]], y = 0 ,
                            xend = rda_res[[r2_type]], yend = 4),
                        linetype="dotted", color="red", size=0.75)
  d <- d + labs(subtitle=paste ("Significance of", label, sep=" "),
                title=paste("Predicting ", ex, "with ", re, sep=" "))
  d <- d + xlab(paste(label, "(permutations)")) + ylab("Density")
  d <- d + annotate("text", x=rda_res[[r2_type]] -0.06, y=2.8,
                    label=p_below, hjust=0, size=3)
  d <- d + annotate("text", x=rda_res[[r2_type]] +0.01, y=2.8,
                    label=p_above, hjust=0, size=3)
  d <- d + annotate("text", x=rda_res[[r2_type]], y=5,
                    label="observed", size=3, color="red")
  d <- d + theme_density()

  return (d)}

spatial_density_plot <- function(sp_rda_res, r2_type) {
  #' This function computes a density plot of the observed and permuted (adjusted)
  #' R squared for an RDA where space has been partialled out
  #' @param sp_rda_res: a list comprising the spatial RDA results
  #' @param r2_type: either "r2" or "r2_adj"
  #' @return the density plot

  if (!r2_type %in% c("r2", "r2_adj")) {
    stop("r2_type must be r2 or r2_adj")}

  else if (r2_type=="r2"){label="R squared"}
  else if (r2_type=="r2_adj"){label="Adjusted R squared"}

  ex <- paste(toupper(substr(sp_rda_res$sample_1$explanatory, 1, 1)),
              substr(sp_rda_res$sample_1$explanatory, 2,
                     nchar(sp_rda_res$sample_1$explanatory)), sep="")

  re <- paste(toupper(substr(sp_rda_res$sample_1$response, 1, 1)),
              substr(sp_rda_res$sample_1$response, 2,
                     nchar(sp_rda_res$sample_1$response)), sep="")


  rda <- sapply(sp_rda_res, function(x) {return (x[[r2_type]])}, simplify=TRUE)
  observed_distr = data.frame(x=na.omit(rda))

  perm_rda <- sapply(sapply(sp_rda_res,
                            function(x) {return (x[paste("perm_", r2_type, sep="")])},
                            simplify=TRUE),
                     function(y) {return (y[1])}, simplify=TRUE)

  permuted_distr <- data.frame(x=na.omit(perm_rda))

  peak_perm_idx <- which.max(density(permuted_distr$x)$y)
```

```
  peak_perm_x <- density(permuted_distr$x)$x[peak_perm_idx]
  peak_perm_y <- density(permuted_distr$x)$y[peak_perm_idx]

  peak_obs_idx <- which.max(density(observed_distr$x)$y)
  peak_obs_x <- density(observed_distr$x)$x[peak_obs_idx]
  peak_obs_y <- density(observed_distr$x)$y[peak_obs_idx]

  s <- ggplot()
  s <- s + geom_density(data=permuted_distr, aes(x=x), alpha=0.3, colour=NA, fill="blue")
  s <- s + geom_density(data=observed_distr, aes(x=x), alpha=0.3, colour=NA, fill="red")
  s <- s + theme_density()
  s <- s + xlab(label) + ylab("Density")
  s <- s + annotate("text", x=peak_obs_x+0.05, y=peak_obs_y+0.2,
                    label="Observed", color="red", size=3)
  s <- s + annotate("text", x=peak_perm_x+0.05, y=peak_perm_y+0.2,
                    label="Random permutations", color="blue", size=3)
  s <- s + labs(title=paste("Predicting", re, "with", ex, sep=" "),
                subtitle="The influence of Space has been partialled out")
  s <- s + xlim(-1, 1)
  return (s)}


# Mapping spatial patterns
map_low_r2 <- function(sp_rda_res, rand_points, lang_poly,
                       languages, lang_labels, r2_type, thr){
#' This function plots the spatial locations of models for which removing the effects
#' of space resulted in a particularily low (adjusted) R rsquared
#' @param sp_rda_res: a list comprising the spatial RDA results
#' @param rand_points: all random spatial locations
#' @param lang_poly: the language polygons
#' @param languages: the languages to be displayed in the map
#' @param lang_labels: the language labels
#' @param r2_type: either "r2" or "r2_adj"
#' @param thr: the percentile of what qualifies as a low (adjusted) R squared
#' @return the plot of the mapped random points

  if (!r2_type %in% c("r2", "r2_adj")) {
    stop("r2_type must be r2 or r2_adj")}

  else if (r2_type=="r2"){label="R squared"}
  else if (r2_type=="r2_adj"){label="Adjusted R squared"}

  if (!is.numeric(thr)) {stop("threshold must be numeric")}
  if (!(thr >= 0 &&  thr <=1)) {stop("threshold must be between 0 and 1")}

  ex <- paste(toupper(substr(sp_rda_res$sample_1$explanatory, 1, 1)),
              substr(sp_rda_res$sample_1$explanatory, 2,
                     nchar(sp_rda_res$sample_1$explanatory)), sep="")

  re <- paste(toupper(substr(sp_rda_res$sample_1$response, 1, 1)),
              substr(sp_rda_res$sample_1$response, 2,
                     nchar(sp_rda_res$sample_1$response)), sep="")
```

```
rda <- sapply(sp_rda_res, function(x) {return (x[[r2_type]])}, simplify=TRUE)
observed_distr = data.frame(x=rda)

# Get samples below threshold
low_ids <- sapply(names(sp_rda_res), function(y) {

  val <- ecdf(observed_distr$x)(sp_rda_res[[y]][[r2_type]])
  if (val <= thr) {
    names_samples <- names(sp_rda_res[y][1])
    ids <- sub('.*\\_', '', names_samples)
    return(as.numeric(ids))}
  })

low_ids <- as.vector(do.call(rbind, low_ids))

# Get locations corresponding to low ids
low_points <- rand_points[rand_points$sample_id %in% low_ids, ]
low_points <- low_points[low_points$nam_label %in% languages,  ]
low_points <- as.data.frame(low_points)
colnames(low_points) <- c("sample_id", "gid", "nam_label", "long", "lat")

lang_poly <- lang_poly[lang_poly$nam_label %in% languages, ]
lang_labels <- lang_labels[lang_labels$nam_label %in% languages, ]

# Plot all locations
cols <- brewer.pal(length(languages),"Set1")

m <- ggplot()
m <- m + coord_map('orthographic',
                   orientation=c(120,0,180),
                   ylim=c(33,80))
m <- m + geom_polygon(data=world, aes(x=long, y=lat, group=group),
                      color=NA, fill='lightgrey')
m <- m + geom_polypath(data=lang_poly, aes(x=long, y=lat,
                                           group=group), size=2, colour=NA, fill="darkgrey")
m <- m + geom_point(data=low_points, aes(x=long, y=lat,
                                         color=factor(nam_label)), stroke = 0,
                    alpha=0.7, shape=16, size=1)
m <- m + scale_color_manual(values=alpha(cols, 1))
m <- m + geom_text(data=lang_labels,
                   aes(x=long, y=lat, label=nam_label,
                       vjust=-1.1, hjust = .3), size=2)
m <- m + theme_map()
return(m)}
```

2 Data

2.1 Grammar and Phonology

Our data comprise numerical and partly categorical variables for grammar and phonology, distance matrices for genetics and music, and the geographical locations of peoples in Northern Asia and Greenland in the form

of language polygons.

Data on grammar and phonology are aggregated from the following sources:

- AUTOTYP (Bickel et al. 2017)
- WALS (Dryer and Haspelmath 2013), with recoding by Balthasar Bickel (https://fossils.ivs.uzh.ch/wals-recoding/home)[1]
- ANU Phonotactics database (Donohue et al. 2013)
- PHOIBLE (Moran, McCloy, and Wright 2014)

The geographical locations are taken from ethnologue (Simons and Fennig 2018).

The split into phonology vs. grammar is based on the narrow definition in `typology-data-checkout/split-data.R` because we are interested in the difference in distribution between segments (and their features) as opposed to all other aspects of linguistic structure. We extract a subset from the data comprising languages from thirteen different sites. Notes: - In AUTOTYP and WALS, Buriat is represented by ISO code `bua` (Buriat in general), while in PHOIBLE it is represented by ISO code `bxr` and in the ANU data by `bxm`. We map all of them below to `bua`.

```
# Define filepath
data_folder <- file.path("../data")   # Data folder

# Read the grammar, phonology and typology data (meta data)
phonology_list <- readRDS(file.path(data_folder, "phonology/phonology.list.RDS"))
grammar_list <- readRDS(file.path(data_folder, "grammar/grammar.list.RDS"))
typology_coverage_df <- readRDS(file.path (data_folder, "typology/typology.coverage.RDS"))

# Define subset
siberia_sample <- c(
 "[i-ain][a-12][g-ainu1240]",    # Ainu
 "[i-bua][a-1095][g-buri1258]",  # Buriat
 "[i-bxm][a-][g-mong1330]",      # Buriat (Mongolia)
 "[i-bxr][a-][g-russ1264]",      # Buriat (Russia)
 "[i-ckt][a-56][g-chuk1273]",    # Chukchi
 "[i-eve][a-738][g-even1260]",   # Even
 "[i-evn][a-527][g-even1259]",   # Evenki
 "[i-kal][a-511][g-kala1399]",   # West Greenlandic
 "[i-jpn][a-118][g-nucl1643]",   # Japanese
 "[i-kor][a-141][g-kore1280]",   # Korean
 "[i-kpy][a-1808][g-kory1246]",  # Koryak
 "[i-nio][a-2172][g-ngan1291]",  # Nganasan
 "[i-sel][a-2393][g-selk1253]",  # Selkup
 "[i-sah][a-2662][g-yaku1245]",  # Yakut
 "[i-ykg][a-423][g-nort2745]")   # Yukagir (Tundra)
 #"[i-yux][a-2797][g-sout2750]")  # Yukagir (Kolyma)

# Extract meta data for the above subset
siberia_metadata_all <- subset(typology_coverage_df, UULID %in% siberia_sample)
siberia_metadata <- subset(siberia_metadata_all, !isocode %in% c('bxm','bxr'))

counts <- xtabs(~autotyp.Stock, siberia_metadata, drop.unused.levels = T)
rownames(siberia_metadata) <- with(siberia_metadata,
                                   ifelse(autotyp.Stock %in% names(counts[counts>1]),
                                          paste(autotyp.Stock, autotyp.Language, sep="/"),
                                          paste(autotyp.Language)))
```

[1]This repository will be made public before publication

```
rownames(siberia_metadata) <- gsub('Yukagir/','', rownames(siberia_metadata))
```

We only use variables with one data point per language, and only variables with non-constant values (which otherwise can't deliver a distance signal). At the same time, we also remap `bxr` and `bxm` to `bua` (cf. above). We trim the linguistic data. This leaves the following number of variables. (For a detailed list, see the appendix; for the metadata and detailed explanations, see the original sources.)

```
# Trim the data
siberia_grammar_list <- trim_data(grammar_list)
siberia_phonology_list <- trim_data(phonology_list)
```

For each site we compute the coverage, i.e. the percentage of available variables per site.

```
# Compute the coverage for each variable and visualize it
grammar_coverage <- compute_coverage(siberia_grammar_list) %>%
  dplyr::select(Language, Coverage)
phonology_coverage <- compute_coverage(siberia_phonology_list) %>%
  dplyr::select(Language,Coverage)
```

We simplify and standardize the language names and visualize the coverage in a table.

```
# Change variable names (see next section)
# Grammar
grammar_coverage[grammar_coverage$Language == 'Chukchi-Kamchatkan/Chukchi',
                 "Language"] <-'Chukchi'
grammar_coverage[grammar_coverage$Language == 'Tungusic/Evenki', "Language"] <- 'Evenki'
grammar_coverage[grammar_coverage$Language == 'Greenlandic Eskimo (West)',
                 "Language"] <-'West Greenland'
grammar_coverage[grammar_coverage$Language == 'Uralic/Selkup', "Language"] <- 'Selkup'
grammar_coverage[grammar_coverage$Language == 'Yukagir (Tundra)', "Language"] <- 'Yukagir'
grammar_coverage[grammar_coverage$Language == 'Tungusic/Even', "Language"] <- 'Even'
grammar_coverage[grammar_coverage$Language == 'Buriat', "Language"] <- 'Buryat'
grammar_coverage[grammar_coverage$Language == 'Uralic/Nganasan', "Language"] <- 'Nganasan'
grammar_coverage[grammar_coverage$Language == 'Chukchi-Kamchatkan/Koryak',
                 "Language"] <- 'Koryak'

# Phonology
phonology_coverage[phonology_coverage$Language == 'Chukchi-Kamchatkan/Chukchi',
                   "Language"] <-'Chukchi'
phonology_coverage[phonology_coverage$Language == 'Tungusic/Evenki', "Language"] <- 'Evenki'
phonology_coverage[phonology_coverage$Language == 'Greenlandic Eskimo (West)',
                   "Language"] <-'West Greenland'
phonology_coverage[phonology_coverage$Language == 'Uralic/Selkup', "Language"] <- 'Selkup'
phonology_coverage[phonology_coverage$Language == 'Yukagir (Tundra)',
                   "Language"] <- 'Yukagir'
phonology_coverage[phonology_coverage$Language == 'Tungusic/Even', "Language"] <- 'Even'
phonology_coverage[phonology_coverage$Language == 'Buriat', "Language"] <- 'Buryat'
phonology_coverage[phonology_coverage$Language == 'Uralic/Nganasan',
                   "Language"] <- 'Nganasan'
phonology_coverage[phonology_coverage$Language == 'Chukchi-Kamchatkan/Koryak',
                   "Language"] <- 'Koryak'

# Visualize the coverage in a table
inner_join(grammar_coverage, phonology_coverage, by='Language') %>%
  arrange(desc(Coverage.x)) %>%
```

Table 1: Data coverage for all thirteen sites.

Language	Grammar (%)	Phonology (%)
Evenki	100	100
Japanese	100	99
West Greenland	100	100
Yakut	100	100
Ainu	98	99
Buryat	98	78
Chukchi	95	100
Even	95	100
Korean	95	100
Selkup	90	97
Nganasan	88	100
Koryak	86	100
Yukagir	81	100

```
kable (booktabs=T, linesep = "",
      caption ='Data coverage for all thirteen sites.',
      col.names=c('Language','Grammar (%)', 'Phonology (%)')) %>%
      kable_styling()
```

Finally, we flatten the nested linguistic data and convert them to data frames.

```
# Flatten the data
grammar <-flatten(siberia_grammar_list)
phonology <- flatten(siberia_phonology_list)
```

2.2 Genetics and Music

We read the distance matrices for genetics and music.

```
# Read the gentic data
genetics <- read.csv(file.path(data_folder,"genetics/SNPs13PopDist.csv"),sep=",",
                     header=TRUE, row.names=1)

# Read the music data
music <- read.csv(file.path(data_folder,"music/MusicAll13PopDist.csv"),sep=",",
                  header=TRUE, row.names=1)
```

2.3 Geographic locations

We read the language polygons from a geo-database. Moreover, we also retrieve 10,000 samples of point locations taken randomly from the language polygons. The random point samples were generated in PostGIS with the function `ST_GeneratePoints`.

```
# Fetch the language polygons from the DB
geo_polygons <- readOGR(dsn=secret_db_login,
                        "genetic_ling.language_polygons_simple")

# Fetch random spatial points in the polygons
```

```
geo_random_points <- readOGR(dsn=secret_db_login,
                             "genetic_ling.random_sample_points_languages")
```

Since the data are gathered from different sources, the names used for the thirteen sites differ. We standardise all names.

```
# The rownames and colnames of the dataframes differ. Of course.

# list_A <- list(genetics, music, grammar, phonology)
# differing_names <- lapply(list_A, function(y)
#   lapply(list_A, function (x) setdiff(rownames(y), rownames(x))))

# We update all non-matching names using the names in geo_random_points as a template
# Genetics
colnames(genetics)[colnames(genetics) == 'westGreenland'] <- 'West Greenland'
rownames(genetics)[rownames(genetics) == 'westGreenland'] <- 'West Greenland'
colnames(genetics)[colnames(genetics) == 'Evenk'] <- 'Evenki'
rownames(genetics)[rownames(genetics) == 'Evenk'] <- 'Evenki'

# Music
colnames(music)[colnames(music) == 'WestGreenland'] <- 'West Greenland'
rownames(music)[rownames(music) == 'WestGreenland'] <- 'West Greenland'
colnames(music)[colnames(music) == 'Nganasa'] <- 'Nganasan'
rownames(music)[rownames(music) == 'Nganasa'] <- 'Nganasan'
colnames(music)[colnames(music) == 'Evenk'] <- 'Evenki'
rownames(music)[rownames(music) == 'Evenk'] <- 'Evenki'

# Grammar
rownames(grammar)[rownames(grammar) == 'Chukchi-Kamchatkan/Chukchi'] <-'Chukchi'
rownames(grammar)[rownames(grammar) == 'Tungusic/Evenki'] <- 'Evenki'
rownames(grammar)[rownames(grammar) == 'Greenlandic Eskimo (West)'] <-'West Greenland'
rownames(grammar)[rownames(grammar) == 'Uralic/Selkup'] <- 'Selkup'
rownames(grammar)[rownames(grammar) == 'Yukagir (Tundra)'] <- 'Yukagir'
rownames(grammar)[rownames(grammar) == 'Tungusic/Even'] <- 'Even'
rownames(grammar)[rownames(grammar) == 'Buriat'] <- 'Buryat'
rownames(grammar)[rownames(grammar) == 'Uralic/Nganasan'] <- 'Nganasan'
rownames(grammar)[rownames(grammar) == 'Chukchi-Kamchatkan/Koryak'] <- 'Koryak'

# Phonology
rownames(phonology)[rownames(phonology) == 'Chukchi-Kamchatkan/Chukchi'] <-'Chukchi'
rownames(phonology)[rownames(phonology) == 'Tungusic/Evenki'] <- 'Evenki'
rownames(phonology)[rownames(phonology) == 'Greenlandic Eskimo (West)'] <-'West Greenland'
rownames(phonology)[rownames(phonology) == 'Uralic/Selkup'] <- 'Selkup'
rownames(phonology)[rownames(phonology) == 'Yukagir (Tundra)'] <- 'Yukagir'
rownames(phonology)[rownames(phonology) == 'Tungusic/Even'] <- 'Even'
rownames(phonology)[rownames(phonology) == 'Buriat'] <- 'Buryat'
rownames(phonology)[rownames(phonology) == 'Uralic/Nganasan'] <- 'Nganasan'
rownames(phonology)[rownames(phonology) == 'Chukchi-Kamchatkan/Koryak'] <- 'Koryak'
```

3 Dimensionality reduction

3.1 Factorial analysis of mixed data (FAMD) of Grammar and Phonology

In view of the fact that the grammar and phonology data are partly numerical and partly categorical, we use a balanced mix of PCA and MCA (Lê, Josse, and Husson 2008). Empty values are imputed using the methods developed by Josse and Husson (2016).

```
# Impute empty values
grammar_imputed <- imputeFAMD(grammar[,-1])
phonology_imputed <- imputeFAMD(phonology[,-1])

# Perform FAMD
grammar_famd <- FAMD(grammar[,-1],
                     tab.comp=grammar_imputed$tab.disj,
                     ncp=10,
                     graph=F)

phonology_famd <- FAMD(phonology[,-1],
                       ncp=10,
                       tab.comp=phonology_imputed$tab.disj,
                       graph=F)
```

We rescale the dimensions obtained through FAMD in relation to the explained variance:

```
for(i in 1:ncol(phonology_famd$ind$coord)) {
  phonology_famd$ind$coord[,i] <-
    scale(phonology_famd$ind$coord[,i])*
    phonology_famd$eig[i,"percentage of variance"]}

for(i in 1:ncol(grammar_famd$ind$coord)) {
  grammar_famd$ind$coord[,i]<-
    scale(grammar_famd$ind$coord[,i])*
    grammar_famd$eig[i,"percentage of variance"]}
```

3.2 Principal Coordinates Analysis (PCoA) of Music and Genes

We perform a principal coordinate analysis (PCoA) on the distance matrices for genetics and music. Similar to a PCA, a PCoA produces a set of orthogonal axes whose importance is measured by eigenvalues (Dray, Legendre, and Peres-Neto 2006). However, in contrast to the PCA, non-Euclidean distance matrices can be used. We correct for negative eigenvalues using the Cailliez procedure.

```
# Convert the matrices into dist objects
genetics_dist <- as.dist(genetics, diag = FALSE, upper = FALSE)
music_dist <- as.dist(music, diag = FALSE, upper = FALSE)


# Perform PCoA
genetics_pcoa <- pcoa(genetics_dist, correction = "cailliez")
music_pcoa <- pcoa(music_dist, correction = "cailliez")
```

We rescale the PCoA components in relation to the explained variance.

```
for(i in 1:ncol(genetics_pcoa$vectors)) {
genetics_pcoa$vectors[,i]<-scale(genetics_pcoa$vectors[,i])*genetics_pcoa$values$Rel_corr_eig[i]}
```

```
for(i in 1:ncol(music_pcoa$vectors)) {
music_pcoa$vectors[,i]<-scale(music_pcoa$vectors[,i])*music_pcoa$values$Rel_corr_eig[i]}
```

3.3 Distance-based Moran's Eigenvector Map Analysis (dbMEM) of the spatial locations

First, we visualize the polygons of all sites on a map.

```
# Convert the SpatialPolygonsDataFrame into a format that ggplot can interpret
geo_polygons_map <- tidy(geo_polygons, region="gid")
geo_polygons_map <- merge(geo_polygons_map, geo_polygons@data, by.x="id", by.y="gid")


# Specify the position of the language labels in the map
lang_labels <- rbind(c(147, 45, "Ainu"), c(105, 56, "Buryat"), c(-165, 64, "Chukchi"),
                     c(139, 65, "Even"), c(94, 55, "Evenki"), c(136, 28, "Japanese"),
                     c(123, 43, "Korean"), c(170, 57, "Koryak"), c(85, 76, "Nganasan"),
                     c(-44, 73, "West Greenland"), c(80, 55, "Selkup"),
                     c(123, 75, "Yakut"), c(160, 70, "Yukagir"))

lang_labels <- data.frame(long = as.numeric(lang_labels[, 1]),
                          lat = as.numeric(lang_labels[, 2]),
                          nam_label = lang_labels[, 3])

# Load background map
world = map_data("world")


# The polygons overlap., Therefore, we  plot the languages on two separate maps
languages <- c("Ainu", "Buryat", "Chukchi", "Even", "Evenki", "Japanese",
               "Korean", "Koryak", "Nganasan", "West Greenland", "Selkup",
               "Yakut", "Yukagir")
languages_1 <- c("Chukchi", "Nganasan", "Even", "Selkup", "Japanese",
                 "Yukagir", "Buryat")
languages_2 <- setdiff(languages,languages_1)

loc_1_plot <- plot_language_map(geo_polygons_map, lang_labels, languages_1)
loc_2_plot <- plot_language_map(geo_polygons_map, lang_labels, languages_2)
locations_plot <- plot_grid(ggplotGrob(loc_1_plot), ggplotGrob(loc_2_plot))
locations_plot
```

For each of the 10,000 samples, we compute the spherical distance between all random locations, which we store in a distance matrix. Then we perform a distance-based Moran's eigenvector map analysis (dbMEM) where we decompose the spatial structure of each of the resulting 10,000 distance matrices (Borcard and Legendre 2002). Similar to a PCoA, dbMEM reveals the principal coordinates of the spatial locations from which the distance matrix was generated. However, in contrast to PCoA dbMEM is primarily concerned with the interaction between spatial neighbours. Thus, only distances below a certain threshold feed directly into constructing the principal coordinates, whereas distances above the threshold are "truncated" (i.e. they are set to four times the threshold value). In the dbMEM, we use the length of the longest edge in the minimum spanning tree as a truncation threshold. Moreover, we only return those eigenfunctions that correspond to positive autocorrelation.

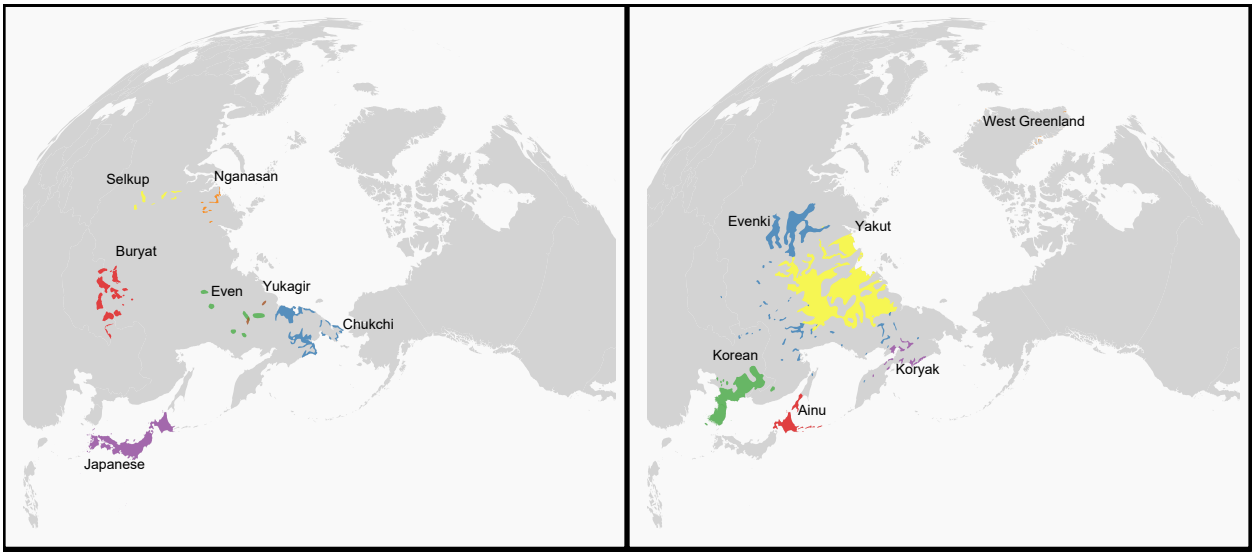

Figure 1: Geographic locations of the thirteen languages.

```
geo_mem <- random_points_to_dbmem(geo_random_points)
```

3.4 Visualizing the explained variance

We visualize the results of the PCA and PCoA in a scree plot. The plot shows the fraction of total variance in the data as explained by each PC/PCo in decreasing order. We extract the eigenvalues from the PCos/PCs and visualize the explained variance.

```
# Extract eigenvalues from PC/PCo results
genetics_ev <- genetics_pcoa$values$Corr_eig
music_ev <- music_pcoa$values$Corr_eig
grammar_ev <- grammar_famd$eig
phonology_ev <- phonology_famd$eig

# Generate Scree plots
plot_pc_scree(eigenval=genetics_ev, title="Genetics", type="PCo")
```

```
plot_pc_scree(eigenval=music_ev, title="Music", type="PCo")
```

```
plot_pc_scree(eigenval=grammar_ev, title="Grammar", type="PC")
```

```
plot_pc_scree(eigenval=phonology_ev, title="Phonology", type="PC")
```

3.5 Merging PCs and PCoAs

We computed the principal components (principal coordinates) for five factors: genetics, grammar, music, phonology and space. We match the order of sites for each factor and combine all factors in a list. We only retain the first k PCs/PCos which together account for at least 80% of the explained variance in the data.

```
# Extract the PCs and the PCos from the PCA and PCoA results
genetics_pco <- genetics_pcoa$vectors
```

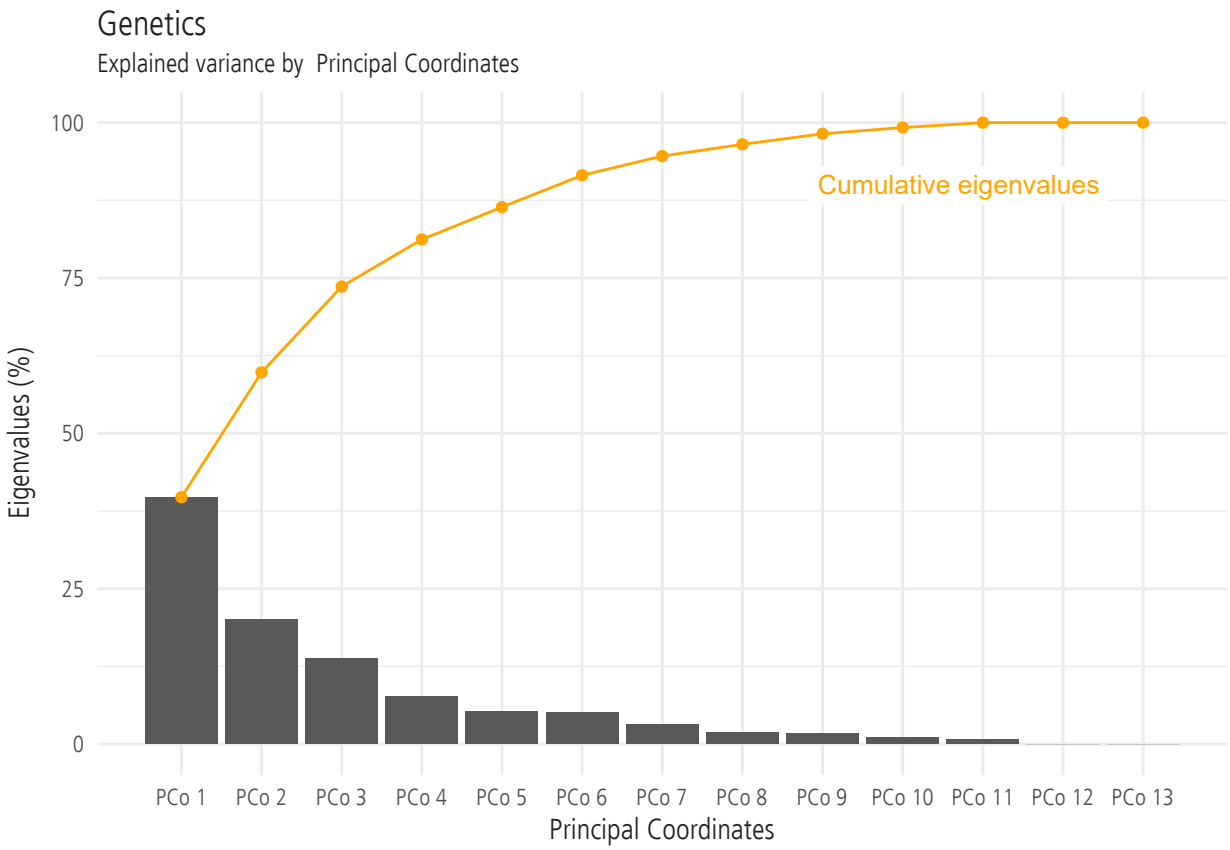

Figure 2: Scree plot of explained variance (Genetics).

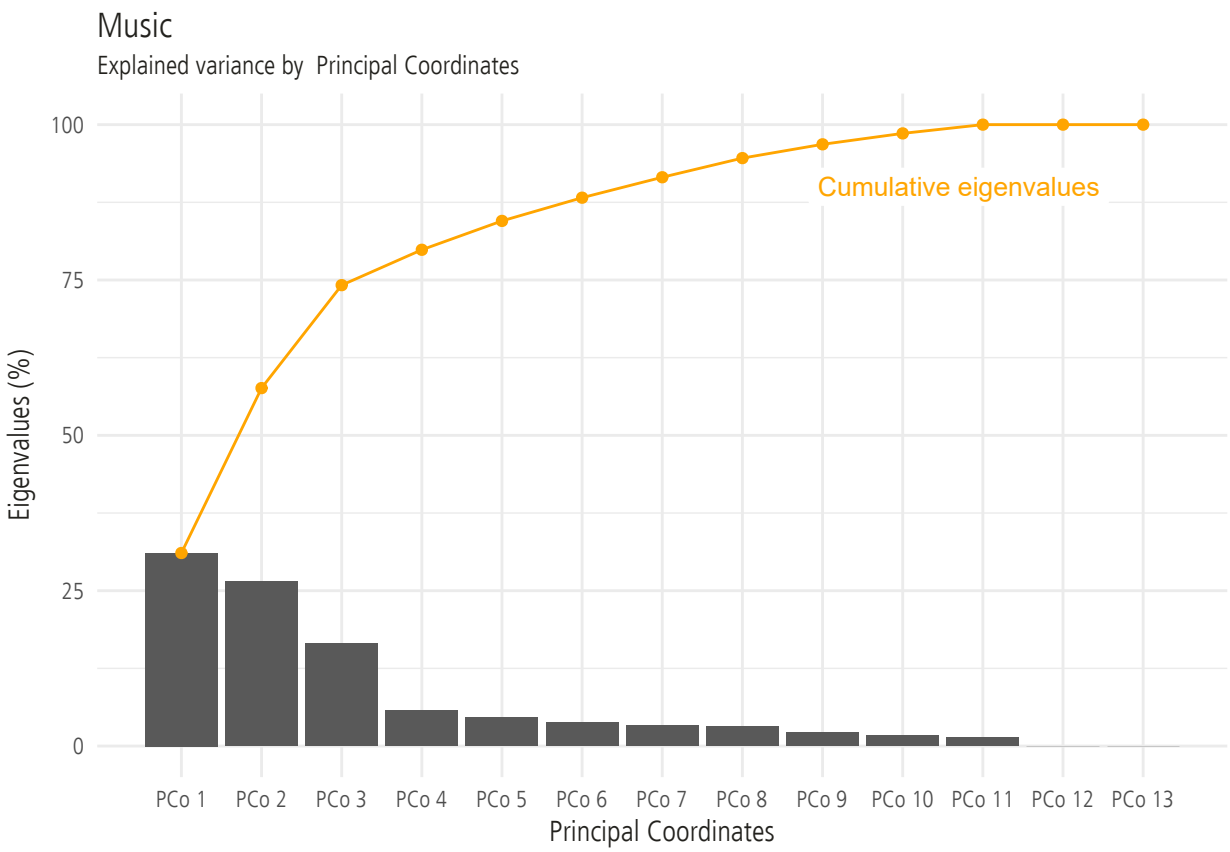

Figure 3: Scree plot of explained variance (Music).

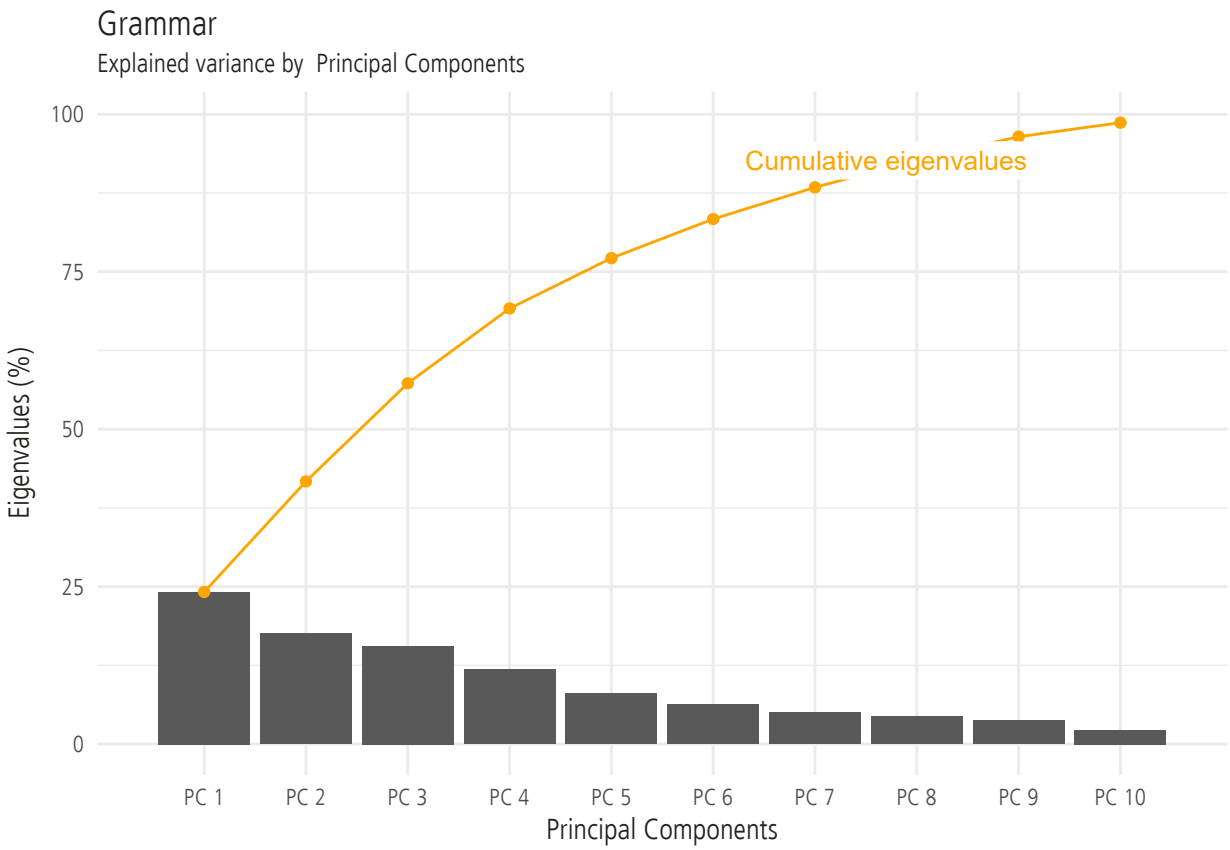

Figure 4: Scree plot of explained variance (Grammar).

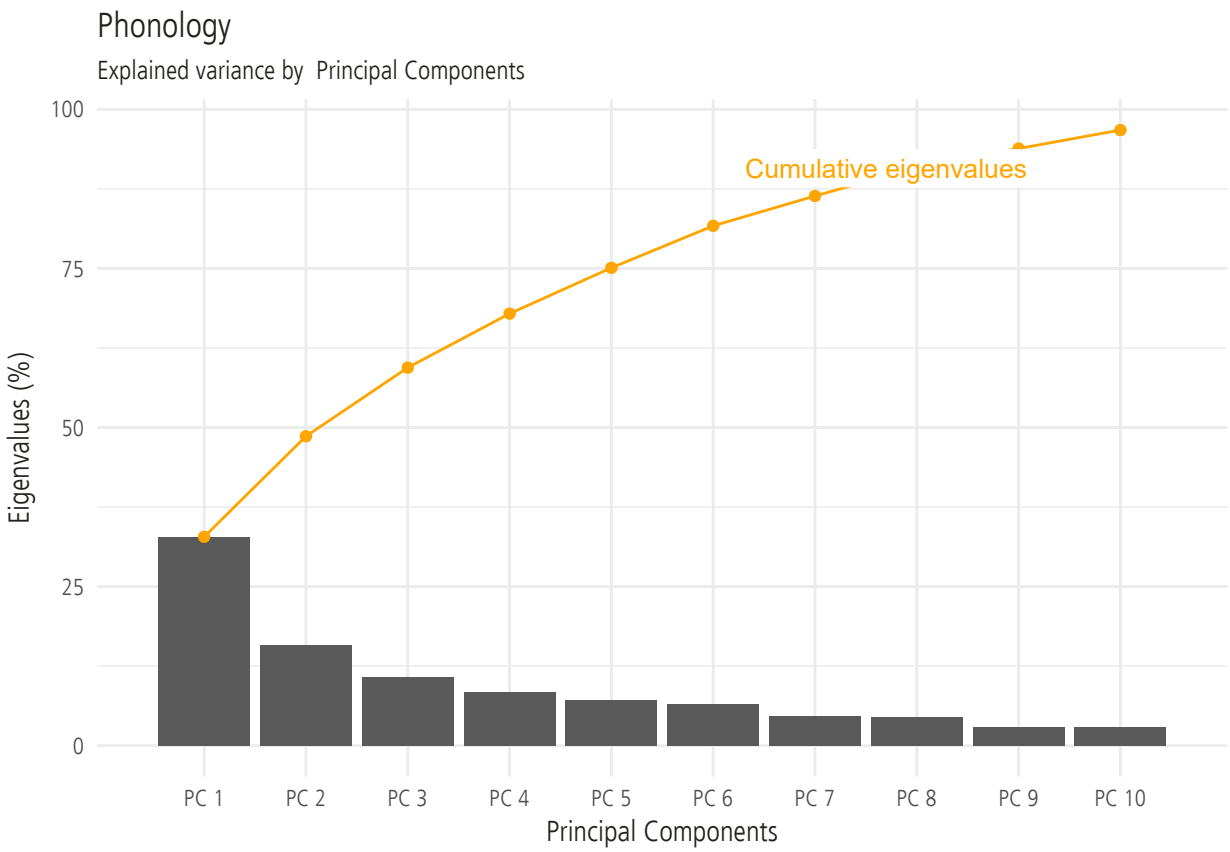

Figure 5: Scree plot of explained variance (Phonology).

```
music_pco <- music_pcoa$vectors
grammar_pc <- grammar_famd$ind$coord
phonology_pc <- phonology_famd$ind$coord

# Change the column names of all PCs and PCoAs
colnames(genetics_pco) <- paste("genetics_pco_", seq(1,ncol(genetics_pco)), sep="")
colnames(music_pco) <- paste("music_pco_", seq(1,ncol(music_pco)), sep="")
colnames(grammar_pc) <- paste("grammar_pc_", seq(1,ncol(grammar_pc)), sep="")
colnames(phonology_pc) <- paste("phonology_pc_", seq(1,ncol(phonology_pc)), sep="")


# Collect all factors in a list, order alphabetically
# (geo_mem is already in alphabetical order)
# Retain the first k PCs/Pcos which togheter account for 80% of the variance
var_th <- 0.8

genetics_pco_rel <- genetics_pco[, 1:min(which(genetics_pcoa$values$Cum_corr_eig>=var_th))]
grammar_pc_rel <- grammar_pc[, 1:min(which(grammar_famd$eig[ ,3]/100>=var_th))]
phonology_pc_rel <- phonology_pc[, 1:min(which(phonology_famd$eig[ ,3]/100>=var_th))]
music_pco_rel <- music_pco[, 1:min(which(music_pcoa$values$Cum_corr_eig>=var_th))]


factors = list(genetics = genetics_pco_rel[order(rownames(genetics_pco_rel)), ],
               music = music_pco_rel[order(rownames(music_pco_rel)), ],
               grammar = grammar_pc_rel[order(rownames(grammar_pc_rel)), ],
               phonology = phonology_pc_rel[order(rownames(phonology_pc_rel)), ],
               geo = geo_mem)
```

3.6 Heatmap of PCs and PCos

We plot a heat map of the PCs/PCos of each factor. We first normalize the PCs/PCos to range from 0 to 1 and then plot the heat map.

```
heat_map_pc(factors$genetics, type="PCo")
```

```
heat_map_pc(factors$music, type="PCo")
```

```
heat_map_pc(factors$grammar, type="PC")
```

```
heat_map_pc(factors$phonology, type="PC")
```

4 Redundancy Analysis (RDA)

Redundancy Analysis (RDA) extracts the variation in a set of explanatory variables that can be explained by a set of response variables (Legendre and Legendre 2012). In our case the response and explanatory variables comprise the principle components (coordinates) of a factor, e.g. the explanatory variable may comprise the PCs of grammar and the response those of phonology. RDA carries out a multiresponse multiple linear regression, i.e. regression of multiple response variables on multiple explanatory variables (Van Den Wollenberg 1977). The results generated through this regression are then “subject to”summarized" in aprincipal components analysis.

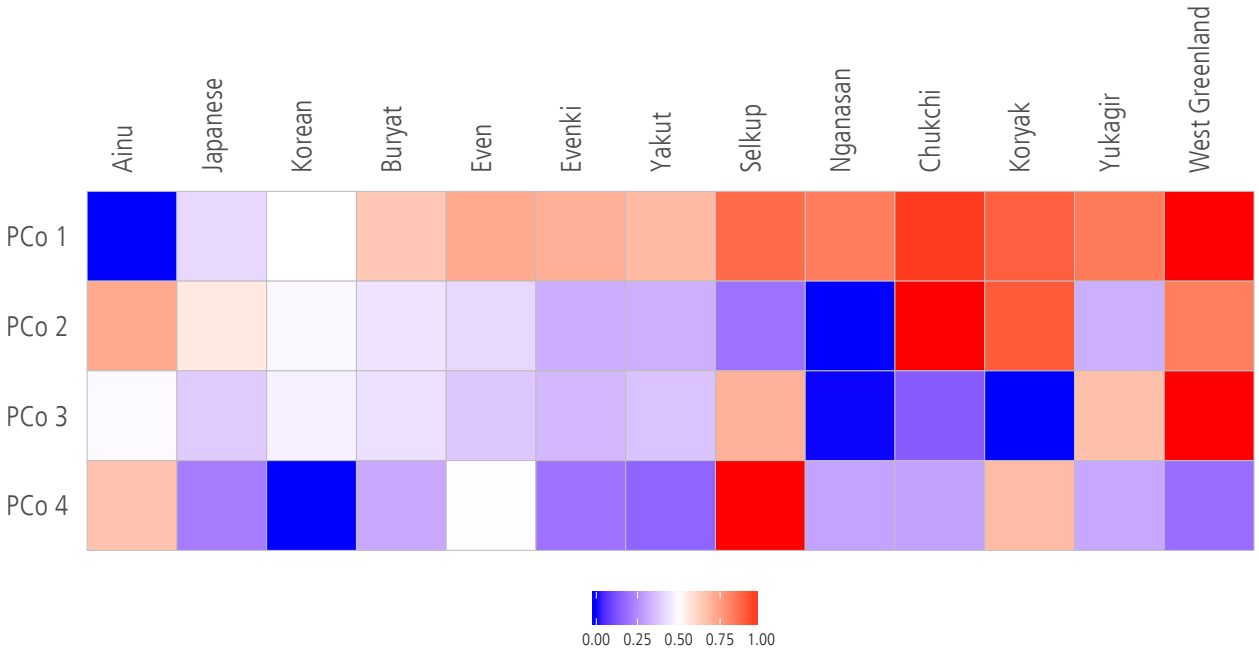

Figure 6: Heat plot of the first four PCos (normalized) for Genetics for each of the 13 populations.

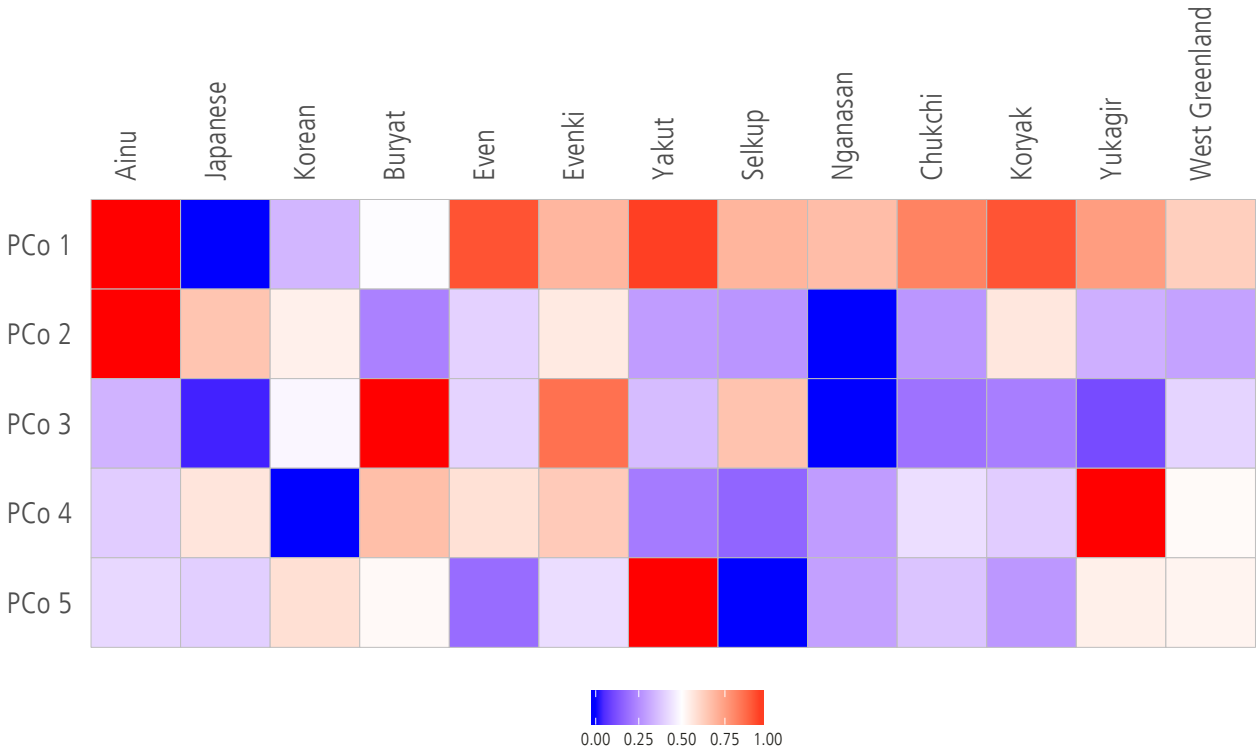

Figure 7: Heat plot of the first five PCos (normalized) for Music for each of the 13 populations.

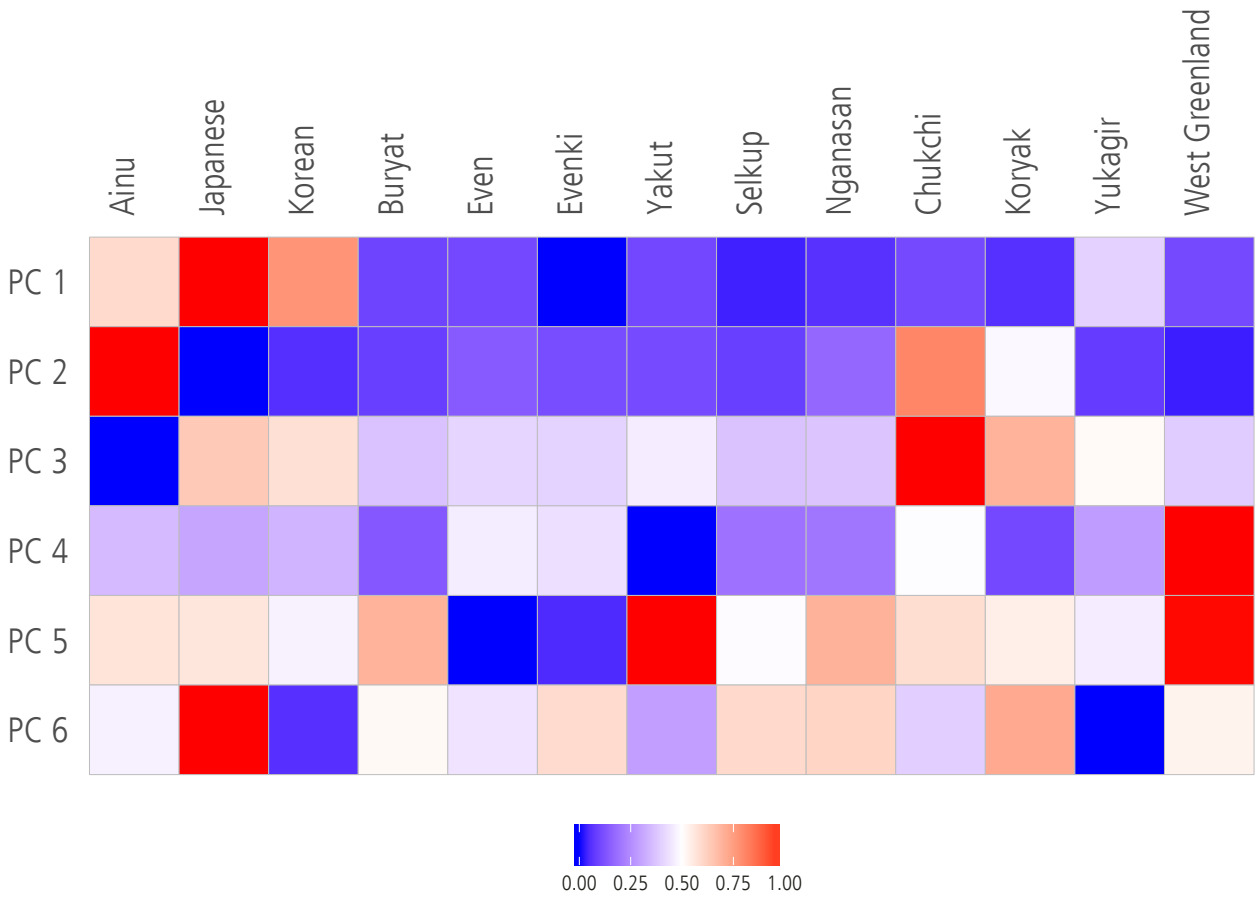

Figure 8: Heat plot of the first six PCs (normalized) for Grammar for each of the 13 populations.

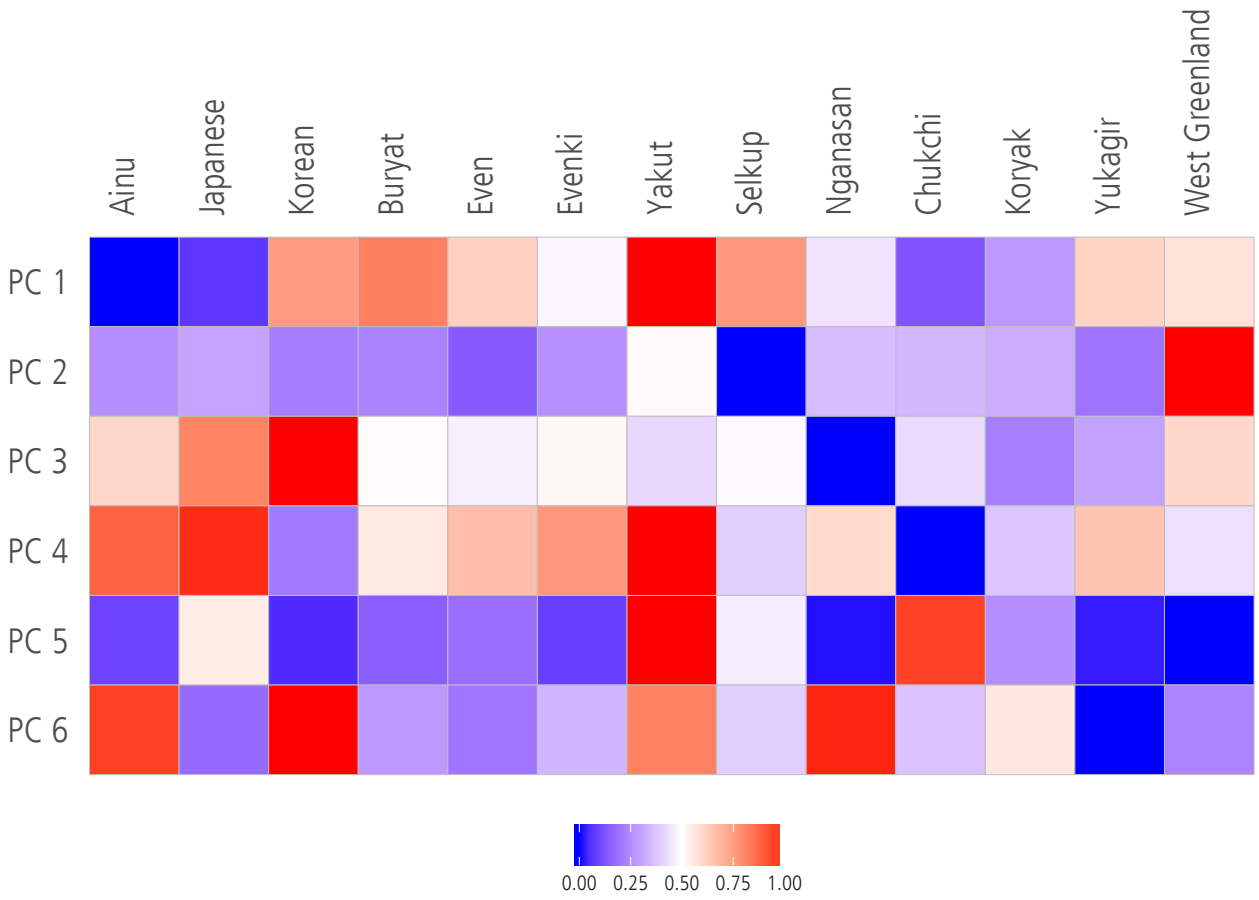

Figure 9: Heat plot of the first six PCs for Phonology (normalized) for each of the 13 populations.

4.1 Pearson Correlation Coefficient

Prior to RDA, we compute the simple Pearson correlation coefficient between all individual principal components/ prinicipal coordinates. We visualize the correlation coefficient and the significance for all pairs.

```
pwc <- pairwise_correlation(factors[c("genetics", "grammar", "music", "phonology")])
# to do pwc fix
plot_correlation(pwc)
```

4.2 Simple RDA

We carry out a simple RDA on genetics, grammar, music and phonology, where each of the factors is used as either the response or the explanatory variable in turns. This yields twelve RDA pairs for which we report the coefficient of determination (adjusted R squared) and the significance of the correlatation determined by an ANOVA like permutation test.

```
# All possible combinations for which we perfom RDA
all_comb <- expand.grid(factor_names, factor_names)
comb <- all_comb[!all_comb$Var1 == all_comb$Var2, ]
comb <- as.data.frame(t(comb), stringsAsFactors=FALSE)

# Perform RDA
rda_all <- lapply(comb, function (x) {
  rda_wrapper(factors[x[1]][[1]], factors[x[2]][[1]])})
```

```
# Extract statistics for each RDA result
rda_mat <- sapply(rda_all, function(x){
  simp <- c(r2=x$r2, r2_adj=x$r2_adj, sig=x$sig, explanatory=x$explanatory,
            response=x$response)
  return(simp)})

rda_mat <- data.frame(t(rda_mat))

rda_mat <- transform(rda_mat, sig=as.numeric(as.character(sig)),
           r2=as.numeric(as.character(r2)), r2_adj=as.numeric(as.character(r2_adj)))

rda_mat[rda_mat$sig > 0.05, "sig_level"] <- ""
rda_mat[rda_mat$sig <= 0.01 , "sig_level"] <- "**"
rda_mat[rda_mat$sig > 0.01 & rda_mat$sig <= 0.05 , "sig_level"] <- "*"
```

We plot the results of the simple pairwise RDA.

```
plot_rda_matrix(rda_mat)
grid.text("Significance: * p \u2264 0.05, ** p \u2264 0.01", x=unit(0.30, "npc"),
          y = unit(0.85, "npc"), just="left",
          gp=gpar(fontfamily = "Frutiger Light Condensed", cex=0.75))
```

4.3 Partial RDA

For all significant pairs we perform a partial RDA where we explore the influence of space on the observed relationships. Partial RDA removes the effects of one or more explanatory variables on the response prior to an ordinary RDA (Borcard, Legendre, and Drapeau 1992). We remove the effect of space and, thus, account for the influence of spatial autocorrelation. Since the languages occupy a large territory simple point locations might yield a misleading picture of the possible spatial interactions. Thus, we randomly sample

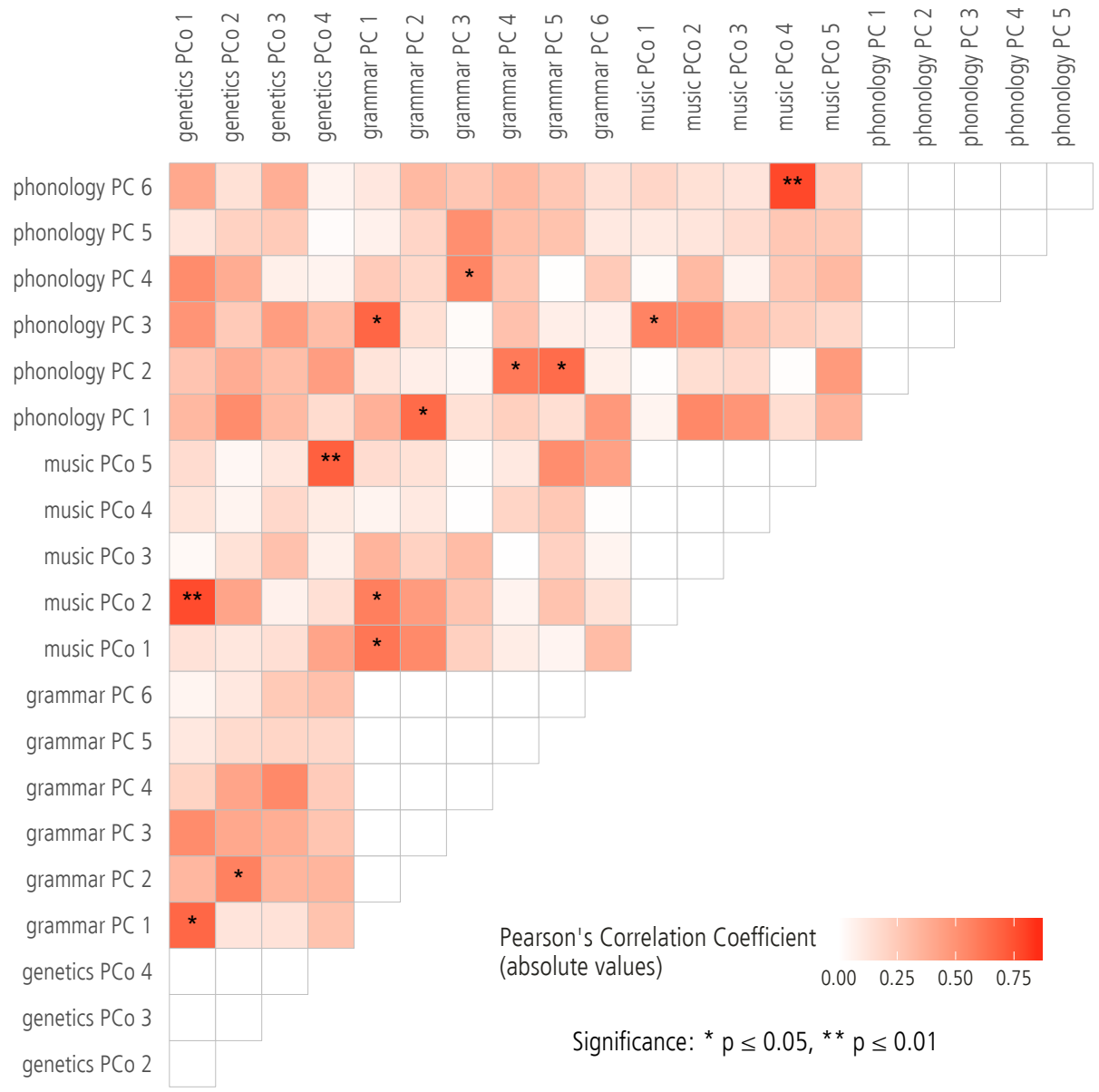

Figure 10: Pairwise correlation between all principal components / principal coordinates (except for space)

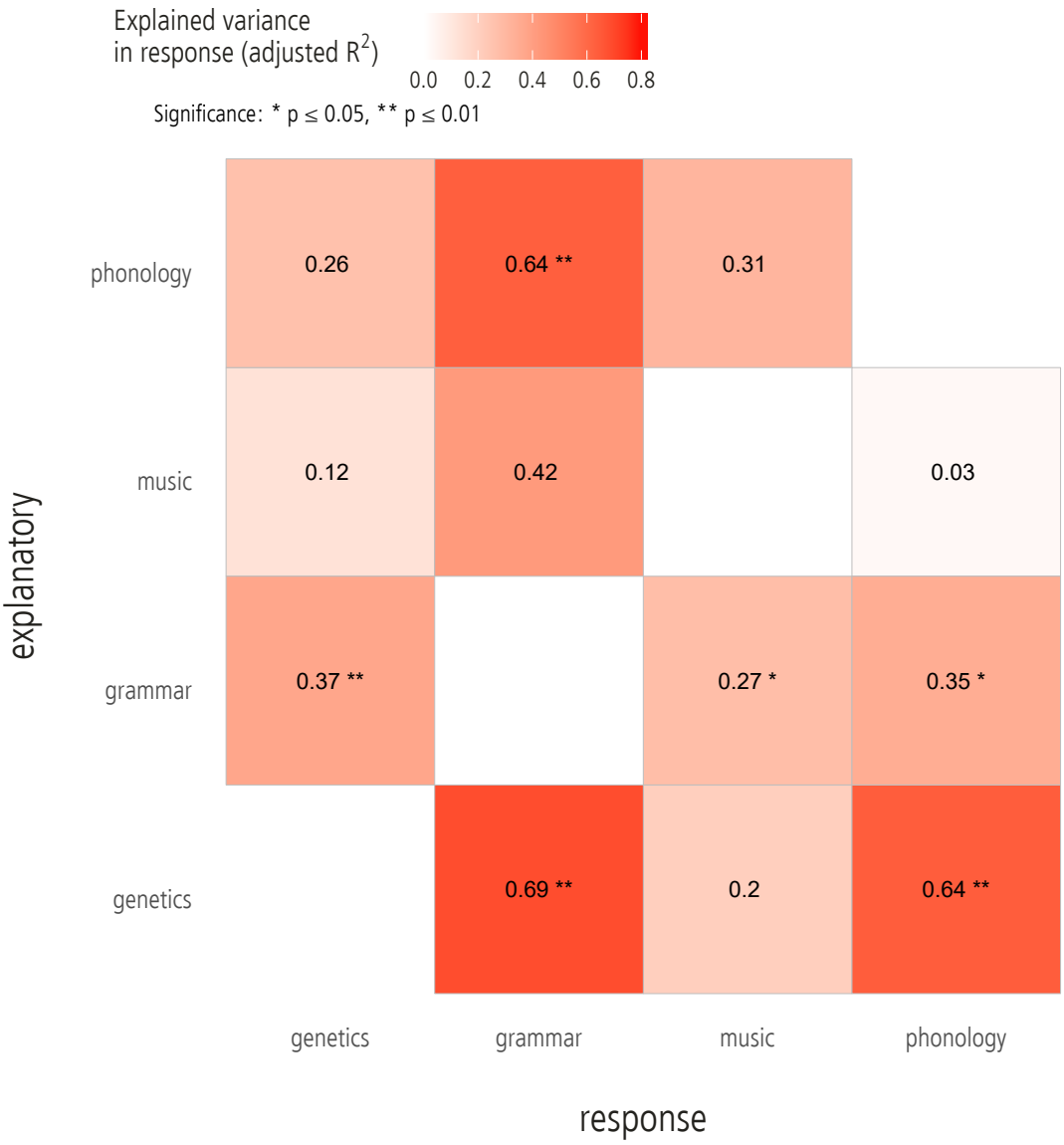

Figure 11: Redundancy Analysis: Variance in the response explained by each explanatory variable

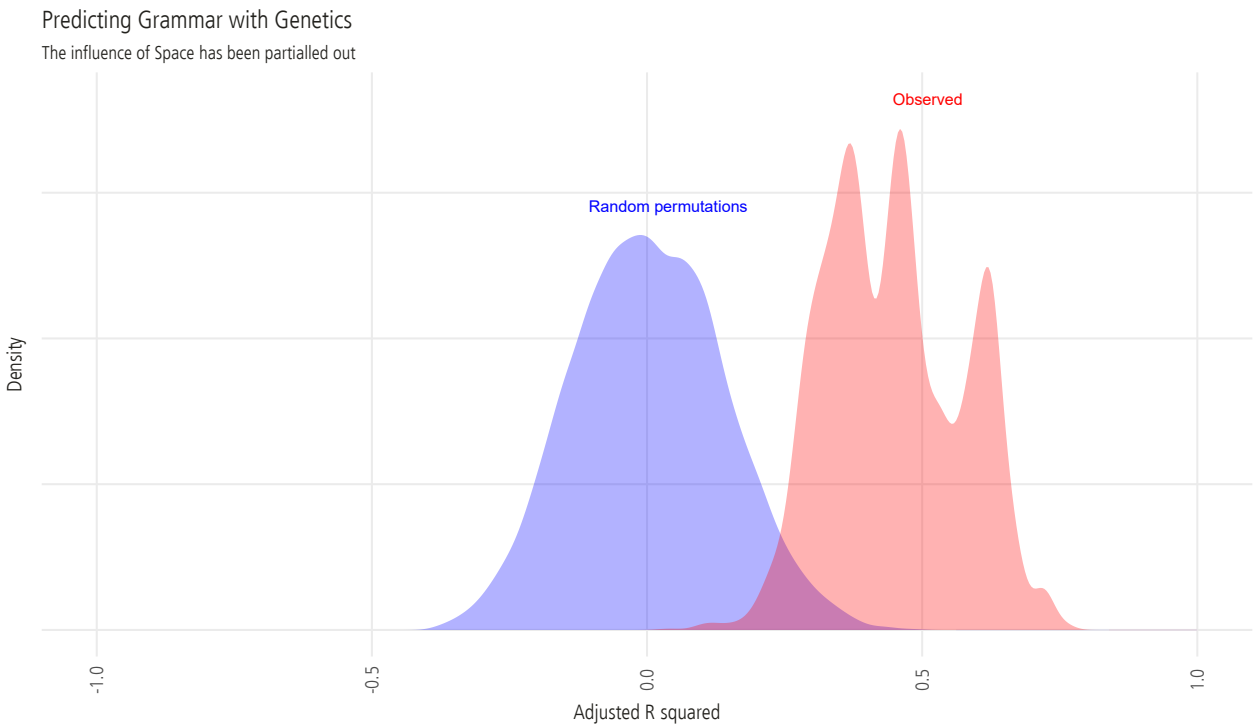

Figure 12: Partial RDA of Genetics (explanatory variable) and Grammar (response)

10'000 points from the language polygons and perform a partial RDA for each. With each sample we remove the influence for one possible scenario of spatial neighborhood and, consequently, the potential for spatial interaction. We report the observed adjusted R squared for each random sample. Finally we compare the resulting distribution of R squared against a distribution for R squared for random permutations, i.e. samples for which the rows of the explanatory variable were randomly permuted for each run of the partial RDA.

```
# Perform Partial RDA (remove the effects of space prior to the RDA)
rda_sp_all <- lapply(comb, function (x) {
  rda_wrapper(factors[x[1]][[1]], factors[x[2]][[1]], factors$geo)})
```

```
# Rename the list entries
names(rda_all) <- sapply(rda_all, function (q){
  return (paste(q$explanatory, q$response, sep="_"))})

names(rda_sp_all) <- sapply(rda_sp_all, function (q){
  return (paste(q$sample_1$explanatory, q$sample_1$response, sep="_"))})

# Visualize the results of the partial RDA
# Genetics --> Grammar
spatial_density_plot(rda_sp_all$genetics_grammar, "r2_adj")
```

```
# Genetics --> Phonology
spatial_density_plot(rda_sp_all$genetics_phonology, "r2_adj")
```

```
# Grammar --> Genetics
spatial_density_plot(rda_sp_all$grammar_genetics, "r2_adj")
```

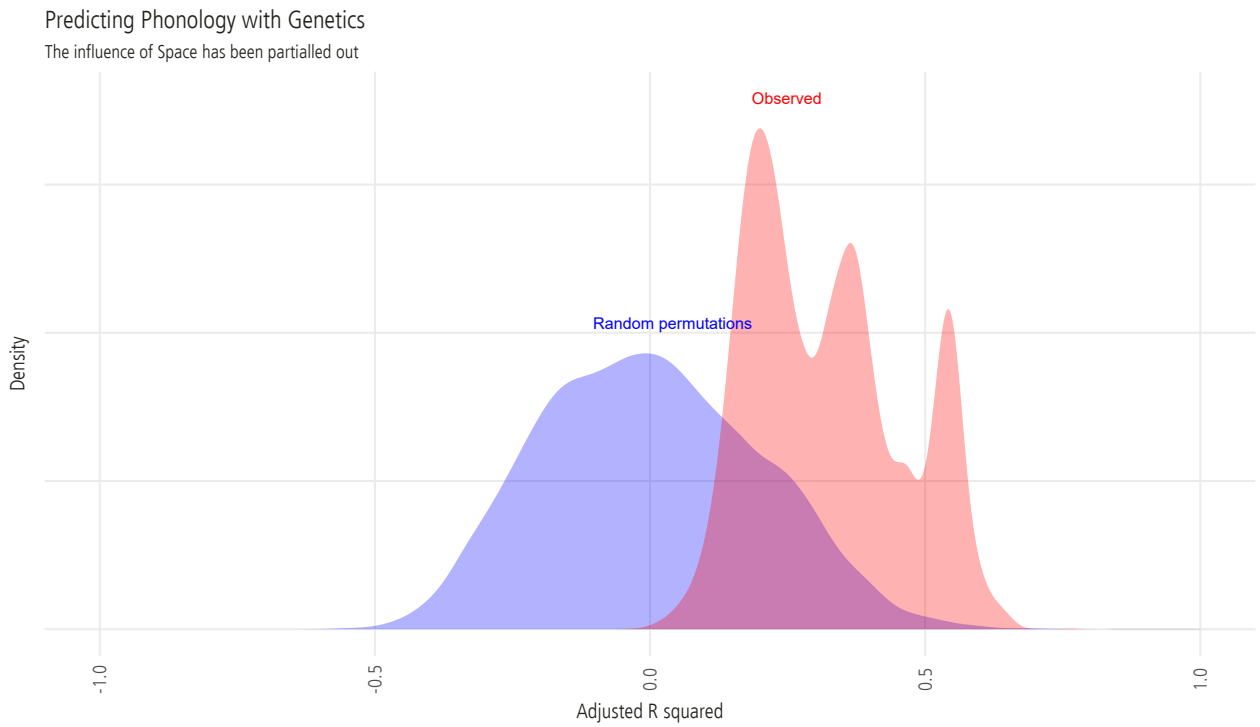

Figure 13: Partial RDA of Genetics (explanatory variable) and Phonology (response)

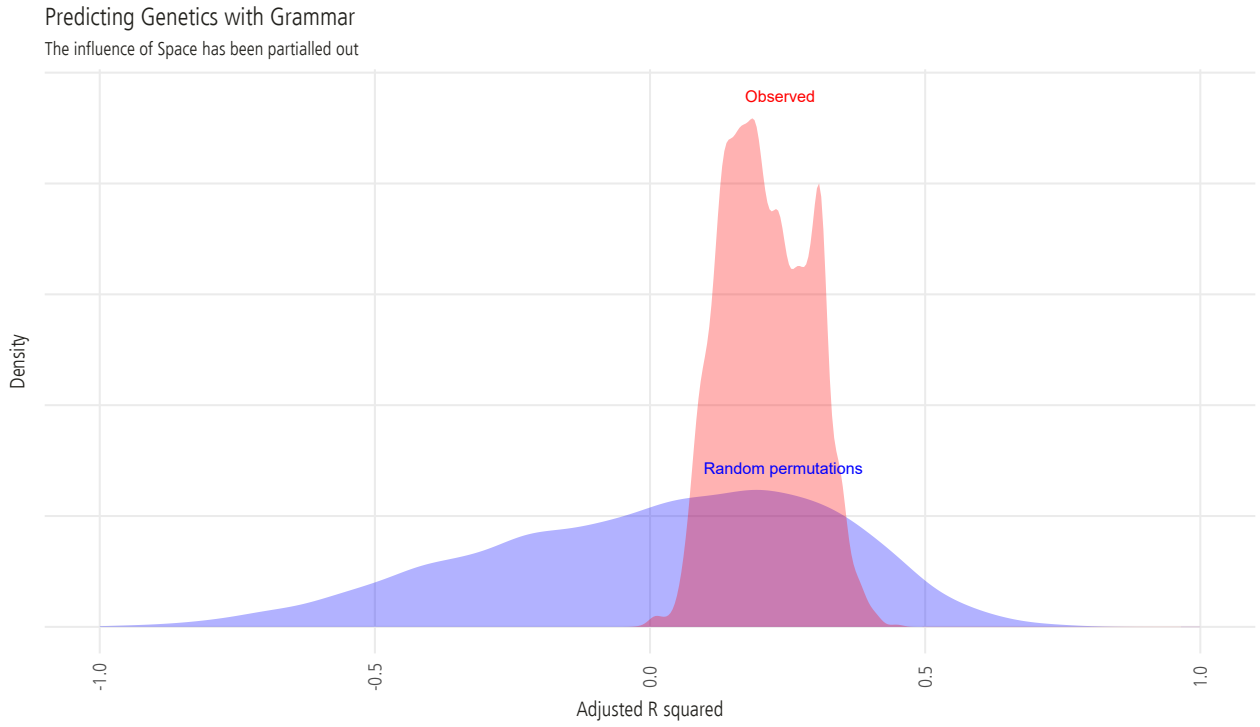

Figure 14: Partial RDA of Grammar (explanatory variable) and Phonology (response)

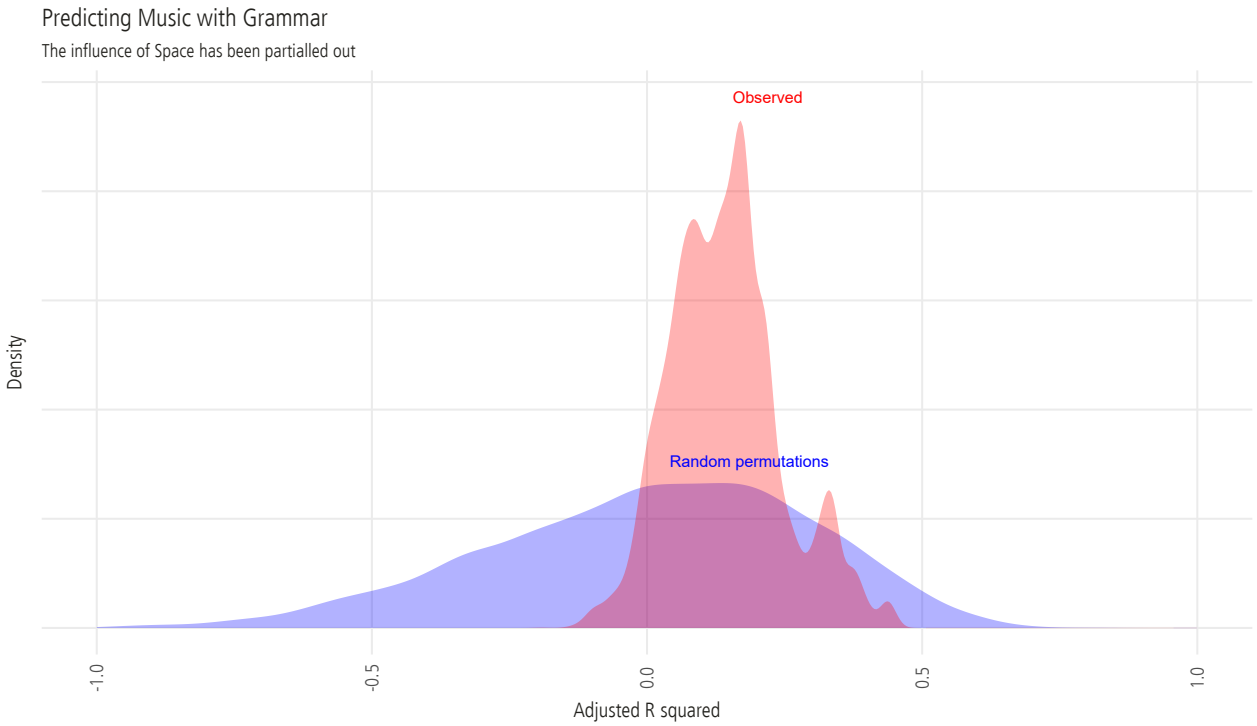

Figure 15: Partial RDA of Grammar (explanatory variable) and Music (response)

```
# Grammar --> Music
spatial_density_plot(rda_sp_all$grammar_music, "r2_adj")
```

```
# Grammar --> Phonology
spatial_density_plot(rda_sp_all$grammar_phonology, "r2_adj")
```

```
# Phonology --> Grammar
spatial_density_plot(rda_sp_all$phonology_grammar, "r2_adj")
```

4.4 Plotting locations with low adjusted R squared

Spatial neighborhood or unlikely to explain the observed relationship between factors if the adjusted R squared for all spatial samples is well above zero and if it does not overlap with the adjusted R squared for random permutations.To further rule out possible influence of space on the observed correlation we plot those location samples for which the adjusted R-squared is particularily low, i.e. in the 0.05 percentile. If all locations with a low R squared cluster in distinct regions of the polygons this gives rise to the assumption that under a particular pattern of spatial neighborhood, spatial neighborhood explains the observed relationship between two factors. If the spatial locations distribute randomly, however, the low adjusted-R squared values are random negative outliers.

```
# Grammar --> Phonology
low_loc_1_grammar_phon <- map_low_r2(rda_sp_all$grammar_phonology, geo_random_points,
                           geo_polygons_map, languages_1, lang_labels, "r2", 0.05)
low_loc_2_grammar_phon  <- map_low_r2(rda_sp_all$grammar_phonology, geo_random_points,
                           geo_polygons_map, languages_2, lang_labels, "r2", 0.05)
```

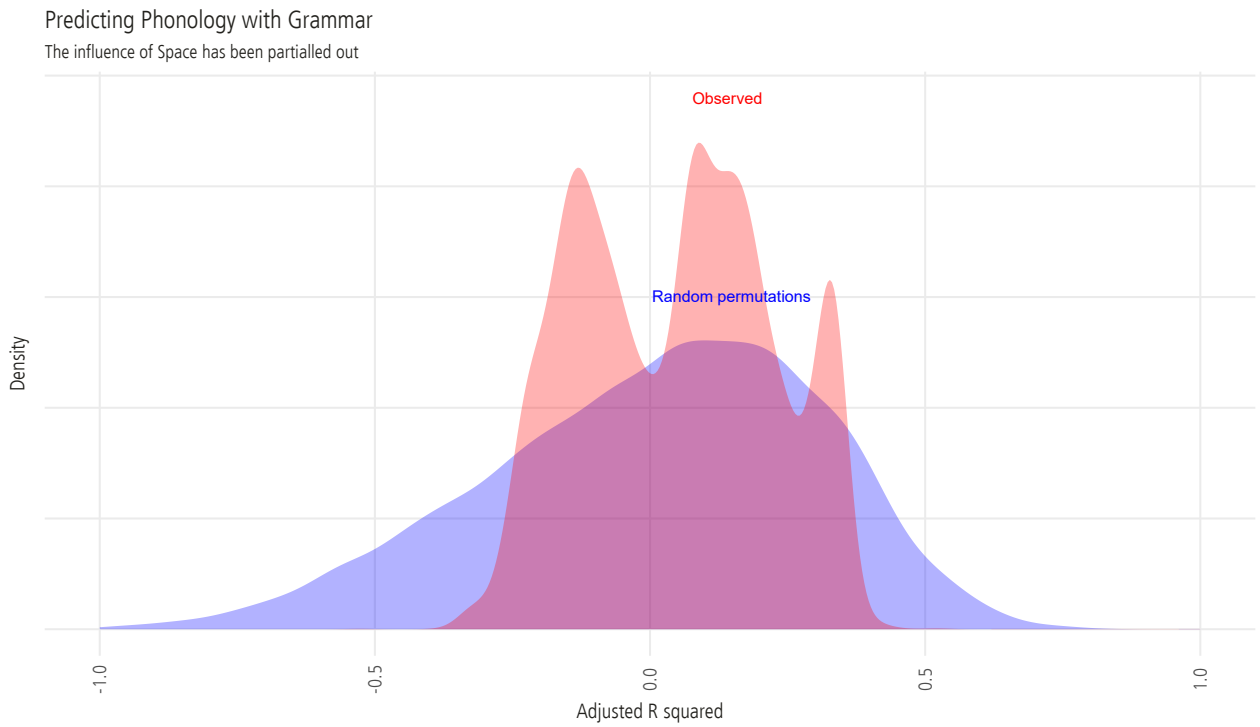

Figure 16: Partial RDA of Grammar (explanatory variable) and Phonology (response)

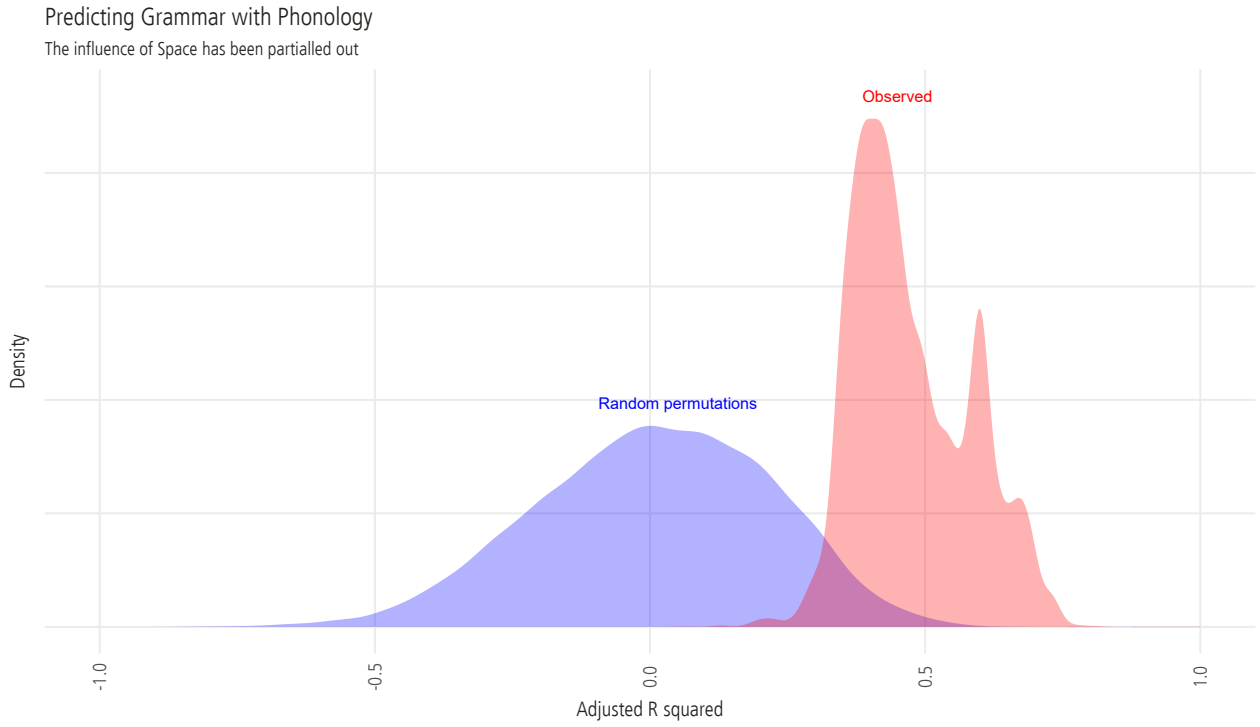

Figure 17: Partial RDA of Phonology (explanatory variable) and Grammar (response)

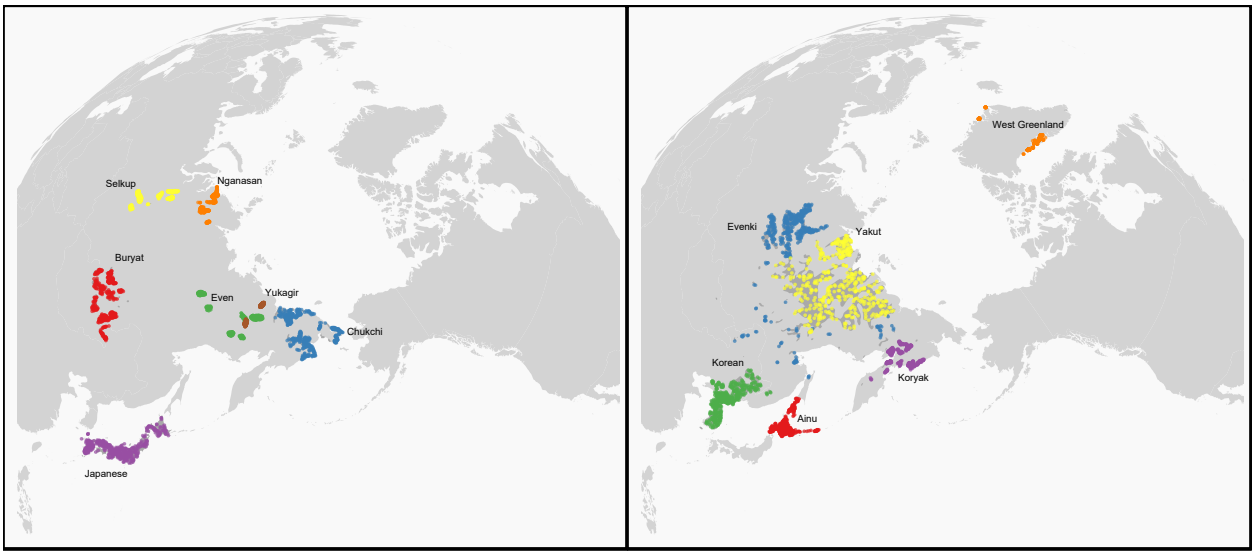

Figure 18: Point samples used for removing the influence of space in the partial RDA between Grammar (explanatory variable) and Phonology (response) with a low R squared

```
plot_grid(ggplotGrob(low_loc_1_grammar_phon), ggplotGrob(low_loc_2_grammar_phon))
```

```
# Genetics --> Grammar
low_loc_1_gen_grammar  <- map_low_r2(rda_sp_all$genetics_grammar, geo_random_points,
                            geo_polygons_map, languages_1, lang_labels, "r2", 0.05)
low_loc_2_gen_grammar <- map_low_r2(rda_sp_all$genetics_grammar, geo_random_points,
                            geo_polygons_map, languages_2, lang_labels, "r2", 0.05)

plot_grid(ggplotGrob(low_loc_1_gen_grammar), ggplotGrob(low_loc_2_gen_grammar))
```

5 Discussion and Interpretation

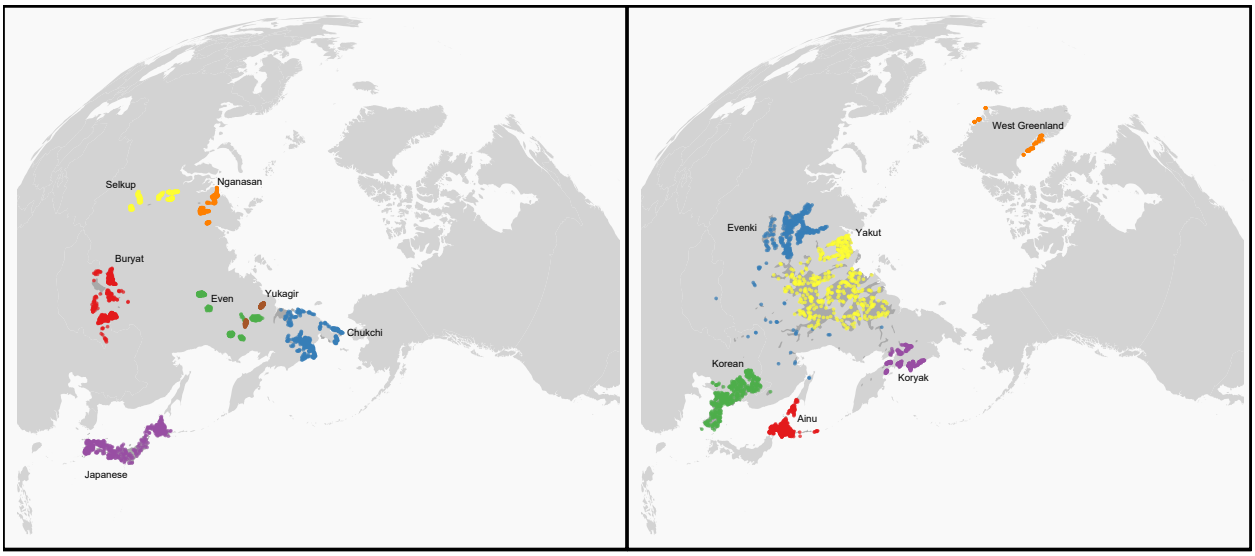

Figure 19: Point samples used for removing the influence of space in the partial RDA between Genetics (explanatory variable) and Grammar (response) with a low R squared

References

Bickel, Balthasar, Johanna Nichols, Taras Zakharko, Alena Witzlack-Makarevich, Kristine Hildebrandt, Michael Rießler, Lennard Bierkandt, Fernando Zúñiga, and John B Lowe. 2017. *The AUTOTYP Typological Databases, Version 0.1.0*. GitHub [https://github.com/autotyp/autotyp-data/tree/0.1.0].

Borcard, Daniel, and Pierre Legendre. 2002. "All-Scale Spatial Analysis of Ecological Data by Means of Principal Coordinates of Neighbour Matrices." *Ecological Modelling* 153 (1-2). Elsevier: 51–68.

Borcard, Daniel, Pierre Legendre, and Pierre Drapeau. 1992. "Partialling Out the Spatial Component of Ecological Variation." *Ecology* 73 (3). Wiley Online Library: 1045–55.

Donohue, Mark, Rebecca Hetherington, James McElvenny, and Virginia Dawson. 2013. *World Phonotactics Database*. Canberra: Department of Linguistics, The Australian National University. http://phonotactics.anu.edu.au.

Dray, Stéphane, Pierre Legendre, and Pedro R Peres-Neto. 2006. "Spatial Modelling: A Comprehensive Framework for Principal Coordinate Analysis of Neighbour Matrices (Pcnm)." *Ecological Modelling* 196 (3-4). Elsevier: 483–93.

Dryer, Matthew S., and Martin Haspelmath, eds. 2013. *WALS Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology. http://wals.info/.

Josse, Julie, and François Husson. 2016. "missMDA: A Package for Handling Missing Values in Multivariate Data Analysis." *Journal of Statistical Software* 70 (1): 1–31. doi:10.18637/jss.v070.i01.

Legendre, Pierre, and Loic FJ Legendre. 2012. *Numerical Ecology*. Vol. 24. Elsevier.

Lê, Sébastien, Julie Josse, and François Husson. 2008. "FactoMineR: An R Package for Multivariate Analysis." *Journal of Statistical Software* 25: 1–18.

Moran, Steven, Daniel McCloy, and Richard Wright, eds. 2014. *PHOIBLE Online*. Munich: Max Planck Digital Library [http://phoible.org/].

Simons, Gary F., and Charles D. Fennig. 2018. "Ethnologue: Languages of the World, Twenty-First Edition." Dallas, Texas: SIL International. http://www.ethnologue.com.

Van Den Wollenberg, Arnold L. 1977. "Redundancy Analysis an Alternative for Canonical Correlation Analysis." *Psychometrika* 42 (2). Springer: 207–19.