Supplementary material 2 - Method and code

Contents

This supplementary material refers to the following submission at *Humanities and Social Sciences Communications*. This supplementary material shows the code that was used for the main analyses included in the paper.

Title: Expansion by migration and diffusion by contact is a source to the global diversity of linguistic nominal categorization systems

Authors:

- Marc Allassonnière-Tang*, marc.allassonniere-tang@mnhn.fr (CNRS, MNHN, Université de Paris, France)
- Olof Lundgren, olof.lundgren@ling.lu.se (Lund University, Sweden)
- Maja Robbers, maja.robbers@lingfil.uu.se (Uppsala University, Sweden)
- Sandra Cronhamn, sandra.cronhamn@ling.lu.se (Lund University, Sweden)
- Filip Larsson, filip.larsson@ling.lu.se (Lund University, Sweden)
- One-Soon Her, hero@thu.edu.tw (Tunghai University, Taiwan)
- Harald Hammarström, harald.hammarstrom@lingfil.uu.se (Uppsala University, Sweden)
- Gerd Carling, gerd.carling@ling.lu.se (Lund University, Sweden

First, we read the basic packages.

```
# basic settings
options(stringsAsFactors = FALSE)
options("scipen"=999, "digits"=4)
# set seed for reproducibility
set.seed(123)
library(tidyverse)
library(readxl)
```

Data visualization

Read the data

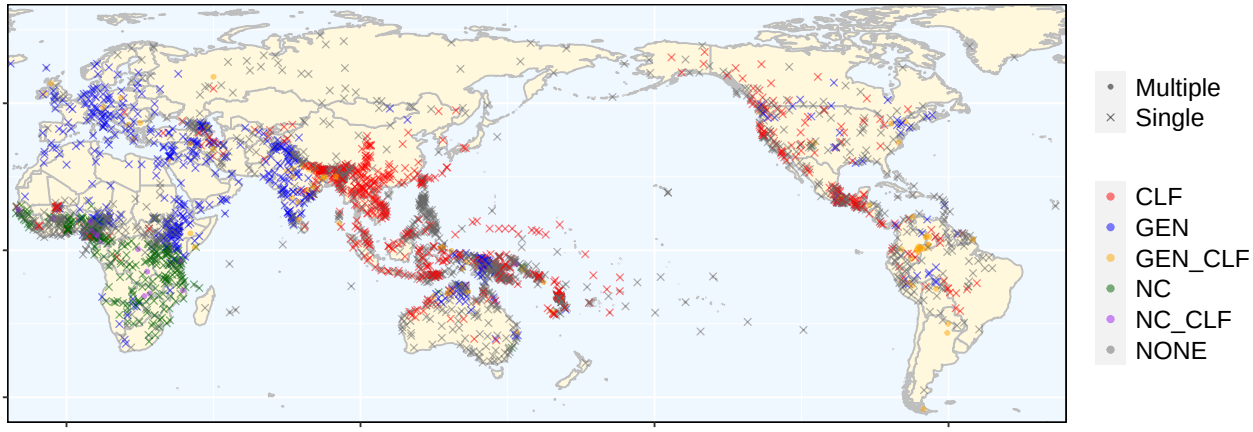
```
data <- read.csv("data_tidy/data_classification.csv") %>%
  select(-c(Source))
str(data)
```

```
## 'data.frame':    3077 obs. of  9 variables:
##  $ Glottocode     : chr  "aari1239" "abad1241" "abar1238" "abau1245" ...
##  $ Family_GLOT_Lv1: chr  "sout2845" "aust1307" "atla1278" "sepi1257" ...
##  $ Genus          : chr  "South Omotic" NA NA "Upper Sepik" ...
##  $ Area_GLOT      : chr  "Africa" "Papunesia" "Africa" "Papunesia" ...
##  $ Longitude      : num  36.6 147 10.2 141.3 42 ...
##  $ Latitude       : num  5.95 -9.03 6.58 -3.97 44.25 ...
##  $ Classifiers    : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ Gender         : logi  TRUE FALSE FALSE TRUE FALSE FALSE ...
##  $ NounClass      : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
```

Visualize the geographical distribution of classifier, gender, and noun class.

```
library(rnaturalearth, rnaturalearthdata)
world <- ne_countries(scale = "medium", returnclass = "sf")

# generate subset for plotting the map
data %>%
  filter(!is.na(Area_GLOT)) %>%
  # annotate one column for different systems
  mutate(System = case_when(NounClass == TRUE & Classifiers == TRUE & Gender == FALSE ~ "NC_CLF",
                            NounClass == TRUE & Classifiers == FALSE & Gender == FALSE ~ "NC",
                            NounClass == FALSE & Gender == TRUE & Classifiers == FALSE ~ "GEN",
                            NounClass == FALSE & Gender == FALSE & Classifiers == TRUE ~ "CLF",
                            NounClass == FALSE & Gender == TRUE & Classifiers == TRUE ~ "GEN_CLF",
                            NounClass == TRUE & Gender == TRUE & Classifiers == FALSE ~ "GEN_NC",
                            NounClass == TRUE & Gender == TRUE & Classifiers == TRUE ~ "GEN_NC_CLF",
                            NounClass == FALSE & Gender == FALSE & Classifiers == FALSE ~ "NONE"),
         Systems = case_when(System %in% c("NC","GEN","CLF","NONE") ~ "Single",
                             System %in% c("NC_CLF","GEN_CLF","GEN_NC","GEN_NC_CLF") ~ "Multiple")) %>%
  select(Area_GLOT, long = Longitude, lat = Latitude, System, Systems) %>%
  filter(!is.na(long)) -> l

# function to change the coordinates to pacific-centred for visualization
shiftlong<-function(long) {
    if (long<(-20)){
        return(360+long)
        } else {return(long)
            }
    }

# apply the function
l$long<-sapply(l[,"long"],shiftlong)
l$lat<-l[,"lat"]

# visualize the numbers
l$System %>% table() -> tb

# change the coordinates in world map
mapWorld <- map_data('world', wrap=c(-20,340), ylim=c(-60,100))
ggplot() +
  # set ocean color to blue
  theme(panel.background = element_rect(fill = "aliceblue"),
        # show panel border
        panel.border = element_rect(fill = NA)) +
```

```
  # can add xlim ylim below to zoom in on an area
  # e.g, for SMATTI xlim = c(60, 140), ylim = c(0,45)
  coord_equal(expand = FALSE) +
  geom_polygon(data = mapWorld, aes(x=long, y = lat, group = group) ,
               # set land color and coast color
               fill = "cornsilk", color = "grey") +
  # add language points
  geom_point(data = l, aes(x = long, y = lat,
                           shape = Systems,
                           color = System), alpha = 0.5) +
  theme(legend.position = "right",
        axis.title=element_blank(),
        legend.title = element_blank(),
        axis.text = element_blank(),
        legend.text=element_text(size=14)) +
  scale_shape_manual(values = c(20, 4)) +
  scale_color_manual(values=c("red","blue","orange","darkgreen","purple","dimgray","pink","brown"))
```

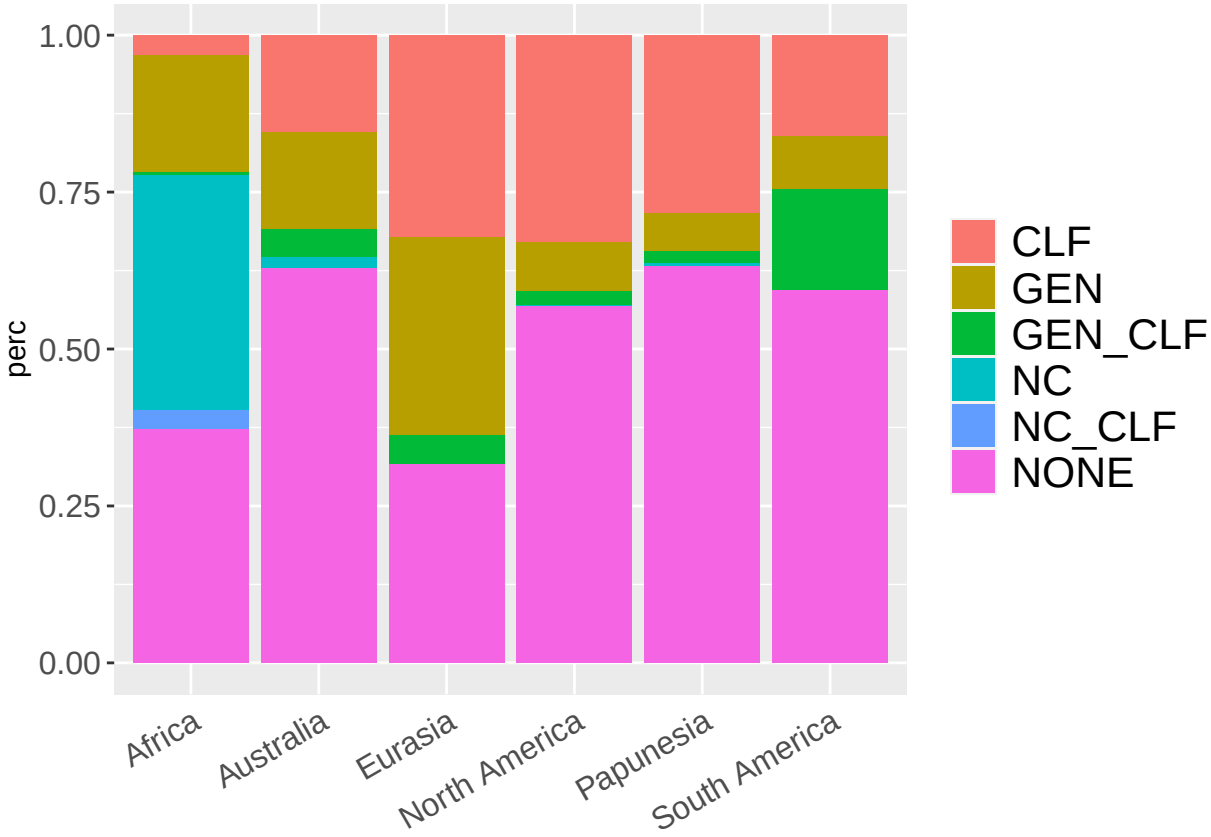



```
# save the map as png file
ggsave('Figure_1.png', width = 8.5, height = 3, dpi = 600)
```

An overview of the features by area is shown below. As expected, Eurasia has the most data points for gender and classifiers, since classifiers are mostly found in Asia and gender in Europe. Africa also has a lot of data points for both noun class and gender, which also matches with the literature, e.g., https://wals.info/chapter/31. Languages that have both gender and classifier systems are mostly found in South-America, which also matches with the linguistic literature. We can also see that most languages do not have either of the three systems.

```
# for area
l%>%
  select(System, Area_GLOT) %>%
  group_by(Area_GLOT,System) %>%
  summarise(count=n()) %>%
  mutate(perc=count/sum(count)) %>%
  ggplot(aes(x=Area_GLOT, y = perc, fill = System)) +
  geom_bar(stat = "identity") +
  theme(axis.title.x=element_blank(),
        legend.title = element_blank(),
        legend.text=element_text(size=16),
        axis.text = element_text(size=12),
        axis.ticks.x=element_blank()) +
```

```
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



2.1 Analysis of PNN and DNN

We add the different types of geographical and phylogenetic neighbors for each language. The method of delaunay neighbors (DNN) is used for geographical neighbors. For phylogenetic nearest neighbors (PNN), we use the count of nodes between languages as a representation of their phylogenetic distance (further developing the method from Mace and Jordan 2011 https://royalsocietypublishing.org/doi/full/10.1098/rstb.2010.0238).

Getting the PNNs

For the PNNs, we take all the languages that are found under the parent of a language and consider them as phylogenetic neighbors. To do so, we first read the updated data from the Glottolog website.

```
# removing not used map data
rm(world, l)
library(ape)
library(phytools)
library(adephylo)
library(reshape2)
library(data.tree)
# this is data available from glottolog website (4.1)
glottolog_data <- read.table("data_raw/DataForCode/languoid.csv", header=TRUE, sep=",", quote='"') %>%
  mutate(iso = "none", wals = "none", population = 0) %>%
  select(name = id, father = parent_id, stock = family_id, glottocode = id,
         iso, wals, level, longitude, latitude, population)

#Read functions for extracting Glottolog trees from Glottolog data from the qlcData package
```

```
.selectDown <- function(families, data = glottolog_data) {
  g <- data
    getChildren <- function(x) { which(g$father == g[x,"name"]) }

    children <- sapply(families, function(x) {which(g$father == x)} )
    newChildren <- unlist(sapply(children, getChildren, simplify = F))

    while (length(newChildren) > 0) {
        children <- c(children, newChildren)
        newChildren <- unlist(sapply(newChildren, getChildren, simplify = F))
    }
    return(unique(children))
}

.selectUp <- function(names, type = "name", data = glottolog_data) {
  g <- data
    getParents <- function(x) { which(g$name== g[x,"father"]) }

    parents <- unique(sapply(names, function(x) {which(g[,type] == x)} ))
    newParents <- unique(unlist(sapply(parents, getParents, simplify = F)))

    while (length(newParents) > 0) {
        parents <- unique(c(parents, newParents))
        newParents <- unique(unlist(sapply(newParents, getParents, simplify = F)))
    }
    return(parents)
}


getTree <- function(up = NULL, kind = "glottocode", down = NULL, data = glottolog_data) {

  g <- data

    if (!is.null(up)) {
        avail <- up %in% unique(g[,kind])
        if (prod(avail) == 1) {
            parents <- .selectUp(up, type = kind)
        } else {
            stop(paste(up[!avail], collapse = ", "), " not available in Glottolog data")
        }
    } else {
        parents <- c()
    }

    if (!is.null(down)) {
        avail <- down %in% unique(g$father)
        availAsLeaf <- down %in% unique(g$name)
        if (prod(avail) == 1) {
            children <- unlist(sapply(down, .selectDown))
        } else if (prod(avail | availAsLeaf) == 1){
            if (sum(avail) > 0) {
                children <- unlist(sapply(down[avail], .selectDown))
            } else {
```

```
                children <- c()
            }
            leaves <- which(g$name %in% down[availAsLeaf & !avail])
            children <- c(children, leaves)
        } else {
            stop(paste(down[!avail], collapse = ", "), " not available as a father in Glottolog data")
        }
        if (anyDuplicated(children) > 0) {
          children <- unique(children)
          warning("some families are part of other families as specified in `down`.
                  Duplicates are removed")
        }
        # add root
        children <- unique(c(children, .selectUp(down)))
    } else {
        children <- c()
    }

  # combine up and down
    if (!is.null(up) & !is.null(down)) {
        all <- intersect(parents, children)
    } else {
        all <- c(parents, children)
    }

  result <- g[all,]

# remove nodes without branching
        f <- table(result$father)
    # remove empty names
    f <- f[-which(names(f) == "")]
        # always keep family that was given in input
        if (!is.null(down)) {f <- f[-which(names(f) == down)]}
        # always keep names given in input
        if (!is.null(up)) {
            ignoreNames <- result[which(result[,kind] %in% up), "name"]
            ignore <- which(names(f) %in% ignoreNames)
            if (length(ignore) != 0) {
            f <- f[-ignore]
            }
        }

        # for all others that occur just once, remove them from the tree
        for (i in names(f)[f==1]) {
            result[result$father == i, "father"] <- result[result$name == i, "father"]
            result <- result[result$name != i,]
        }

    result <- result[order(rownames(result)),]
    return(droplevels(result))
}
```

Then, we keep record all the languages that are found under the parent of a language. The end of the code generates the number of languages in the updated data to be sure that we have the same amount of languages

as in the original data.

```
# get the most updated trees from the Glottolog website
read.tree("data_raw/DataForCode/tree_glottolog_newick.txt") -> trees
# identify the families present in the tree
families <- data$Family_GLOT_Lv1 %>% unique()
# opening empty data frame
table <- NULL %>% as.data.frame()
for(i in 1:length(families)){
# extract the distance between languages in the dataset
data %>% filter(Family_GLOT_Lv1 == families[i]) -> tmp
# get the tree
getTree(down= families[i], up = tmp$Glottocode) -> tree
# the following function can also be used on the downloaded glottolog data
tree <- FromDataFrameNetwork(tree)
# remove the labels of the internal nodes
tree <- as.phylo.Node(tree)

# extract all the other languages under the parent of a language
df <- NULL %>% as.data.frame()
df <- rbind(df, rep(NA,600))
colnames(df) <- c("Glottocode",paste0("PNN_",1:599))
for(z in 1:length(tree$tip.label)){
  parent <- phangorn::Ancestors(tree,z,"parent")
  descendants <- phangorn::Descendants(tree,parent) %>% unlist()
  descendants <- descendants[which(descendants != z)]
  descendants <- tree$tip.label[descendants]
  descendants <- c(tree$tip.label[z], descendants)
df[1+z,] <- c(descendants, rep(NA,600-length(descendants)))
}

# save the output
df <- df[-1,]
table <- rbind(table, df)
}

# remove non-used data
rm(df, tmp, tree, trees)

not_all_na <- function(x) {!all(is.na(x))}
table %>% select_if(not_all_na) -> table

# merge the information of PNNs with the main data
data %>%
  merge(table, by = "Glottocode") -> data

# sanity check that the table has the languages in the dataset
table$Glottocode[table$Glottocode %in% data$Glottocode] %>% unique() %>% length()
```

Getting the DNNs

For geographical cohesion, we use delaunay neighbors. Basically, we build a network based on the geographical coordinates and extract the neighbors of each language based on the network. Again, we extract the updated Glottolog data and build the network.

```r
# this is data available from glottolog website (4.1) right now
glottolog_data <- read.table("data_raw/DataForCode/languoid.csv", header=TRUE, sep=",", quote='"')
#keep only spoken languages (and dialects) with geographic locations info
glottolog_data <- glottolog_data[!is.na(glottolog_data$latitude) &
                                   !is.na(glottolog_data$longitude) &
                                   glottolog_data$level != "family" &
                                   !grepl("sign", tolower(glottolog_data$name), fixed=TRUE) &
                                   glottolog_data$bookkeeping != "True",
                                 c("id", "iso639P3code", "family_id", "name", "level",
                                   "latitude", "longitude")]

# only keep the lanugages we have in our dataset
glottolog_data %>% filter(id %in% data$Glottocode) -> glottolog_data

# For entries that have precisely the same geographic coordinates as other entries, add 0.01 degrees
#(on the order of hundreds of meters/the scale of towns or villages):
same_coords <- duplicated(glottolog_data[, c("latitude", "longitude")]);
glottolog_data$latitude[ same_coords ] <- glottolog_data$latitude[ same_coords ] + 0.01
glottolog_data$longitude[ same_coords ] <- glottolog_data$longitude[ same_coords ] + 0.01

# extract the coordinates and set row names as ids
glottolog_coords <- as.matrix(glottolog_data[,c("longitude", "latitude")])
rownames(glottolog_coords) <- as.character(glottolog_data$id)
# extract all possible pairs between languages
glottolog_all_pairs <- expand.grid("p1"=1:nrow(glottolog_coords), "p2"=1:nrow(glottolog_coords))
glottolog_all_pairs <- glottolog_all_pairs[ glottolog_all_pairs$p1 < glottolog_all_pairs$p2, ]

# Delaunay triangulation taking into account hard-to-cross regions ####
# This is based on code from Cysouw, M., Dediu, D., & Moran, S. (2012).
#Comment on "Phonemic Diversity Supports a Serial Founder Effect Model of Language Expansion from Afric
# https://doi.org/10.1126/science.1208841,
# updated to run on R 3.6 by Marc Tang
# and massively changed to accomodate the much more dense Glottolog data by Dan Dediu

# Change longitudes to make everything Pacific-centered
glottolog_data$longitude_shifted <- ifelse( !is.na(glottolog_data$longitude) &
                                              glottolog_data$longitude < -20,
                                            360 + glottolog_data$longitude,
                                            glottolog_data$longitude )

# The barriers that nearest-neighbour cannot cross:
.generate_barrier <- function(p, step=0.5) # p is a vector of (x,y) coordinates, i.e p=c(x1,y1, x2,y2,
{
  if( length(p) < 2 || length(p) %%2 != 0 ) stop("p must be of the form c(x1,y1, x2,y2, ... xn,yn).");
  i <- 1; x <- y <- c();
  while( i <= length(p)-2 )
  {
    delta_xy <- max(abs(p[i] - p[i+2]), abs(p[i+1] - p[i+3]));
    n_points <- ceiling(delta_xy / step);
    t <- seq(0, 1, length.out=n_points);
    x <- c(x, t*p[i] + (1-t)*p[i+2]);
    y <- c(y, t*p[i+1] + (1-t)*p[i+3]);
    i <- i+2;
```

```
  }
  return (data.frame("longitude"=x, "latitude"=y));
}
geo_barriers <- rbind(.generate_barrier(c(-20,35.92, -5.65,35.92, -5.20,36.03, -1.32,36.29, 5.62,37.94,
                      .generate_barrier(c(28.21,41.96, 34.30,43.47, 41.25,41.61)), # Black sea
                      .generate_barrier(c(52.90,46.27, 49.06,44.80, 51.47,40.20, 51.57,36.87)), # Caspi
                      .generate_barrier(c(59.00,17.59, 59.00,-85.00)), # Indian ocean (Africa)
                      .generate_barrier(c(89.34,21.04, 89.34,-20.00)), # Indian ocean (India)
                      .generate_barrier(c(89.34,-20.00, 126.42,-12.44, 131.42,-9.79, 140.00,-10.18)),
                      .generate_barrier(c(149.11,-12.93, 157.93,-26.00, 157.93,-85.00)),
                      .generate_barrier(c(142.89,-40.35, 138.00,-40.35, 131.06,-34.11, 117.76,-37.68)),
                      .generate_barrier(c(124.55,29.58, 127.87,23.27, 134.33,30.00, 143.50,32.00, 161.3
                      .generate_barrier(c(-180.00,48.00, -145.00,48.00, -83.00,-22.00, -82.00,-85.00)),
                      .generate_barrier(c(-96.60,25.69, -80.59,24.17, -74.43,21.58, -68.72,20.29, -19.9
                      .generate_barrier(c(-168.99,68.03, -168.99,85.00)), # Chukchi sea
                      .generate_barrier(c(55.89,69.30, 46.80,71.78, 46.80,85.00)), # Arctic
                      .generate_barrier(c(3.50,70.00, 3.50,85.00)), # North Atlantic
                      .generate_barrier(c(10.70,-5.58, -15.00,-19.00)), # Africa (Gulf of Guinea)
                      .generate_barrier(c(-62.22,-45.25, -35.00,-45.25)), # Tierra del Fuego
                      .generate_barrier(c(-74.63,76.96, -57.93,67.74)), # Greenland
                      NULL
                      )
 # Pacific-centered
geo_barriers$longitude <- ifelse( !is.na(geo_barriers$longitude) & geo_barriers$longitude < -20, 360 +
                                    geo_barriers$longitude, geo_barriers$longitude )


# make a delauney triangulation of the languages, and remove the above specified boundaries:
library(deldir)
glottolog_delaunay <- deldir(glottolog_data$longitude_shifted, glottolog_data$latitude, # languages
                             # "boundaries" as dummy points
                             dpl=list(x=geo_barriers$longitude, y=geo_barriers$latitude))

# Remove the connections to the dummy points (the "boundaries"):
glottolog_delaunay_constrained <- glottolog_delaunay$delsgs
glottolog_delaunay_constrained <- glottolog_delaunay_constrained[ glottolog_delaunay_constrained$ind1 <=
                                                                  glottolog_delaunay_constrained$ind2

#keep only the actual glottolog points (but deal with duplication)
# Add the Glottolog id's of the languages:
glottolog_delaunay_constrained$glottocode1 <- as.character(glottolog_data$id)[ glottolog_delaunay$ind.or
glottolog_delaunay_constrained$glottocode2 <- as.character(glottolog_data$id)[ glottolog_delaunay$ind.or

# Plot this Delaunay network:
ggplot() +
  # landmasses as polygons
  geom_polygon(data=mapWorld, aes(x=long, y=lat, group=group), fill="grey") +
  geom_curve(data=glottolog_delaunay_constrained, aes(x=x1, y=y1, xend=x2, yend=y2),
             color="black", curvature=0, alpha=0.5) +
  # languages
  geom_point(data=glottolog_data, aes(x=longitude_shifted, y=latitude),
             shape=21, alpha=0.5, color="blue", fill="yellow", size=0.5) +
  # barriers
```

```
  geom_point(data=geo_barriers, aes(x=longitude, y=latitude), col="red", shape=20, size=1) +
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        legend.title = element_blank(),
        legend.text=element_blank())

# Convert to a network:
library(igraph)
igraph::graph_from_data_frame(vertices=data.frame("id"=glottolog_data$id,
                                                  "name"=glottolog_data$id,
                                                   # info about the vertices
                                                  glottolog_data[,c("latitude", "longitude","longitude_
                              d=glottolog_delaunay_constrained[,c("glottocode1","glottocode2")],
                              directed=FALSE) -> glottolog_delaunay_graph

# remove not used data
rm(geo_barriers, glottolog_all_pairs, glottolog_coords,
   glottolog_data, glottolog_delaunay, glottolog_delaunay_constrained, same_coords)
# sanity check
paste("If the following is 1, all nodes are connected: ",
      vertex_connectivity(glottolog_delaunay_graph), sep = "")
```

Based on the network, we can extract the nearest neighboors in the network and merge the Delaunay neighbor data with the data from phylogenetic neighbors.

Export the data if needed

```
write.csv(data.backup,
          "data_tidy/data.csv",
          row.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
```

Read the data if needed

```
data <- read.csv("data_tidy/data.csv") %>% as.data.frame()
str(data[,1:10])
```

```
## 'data.frame':    3077 obs. of  10 variables:
##  $ Glottocode     : chr  "aari1239" "abad1241" "abar1238" "abau1245" ...
##  $ Family_GLOT_Lv1: chr  "sout2845" "aust1307" "atla1278" "sepi1257" ...
##  $ Genus          : chr  "South Omotic" NA NA "Upper Sepik" ...
##  $ Area_GLOT      : chr  "Africa" "Papunesia" "Africa" "Papunesia" ...
##  $ Longitude      : num  36.6 147 10.2 141.3 42 ...
##  $ Latitude       : num  5.95 -9.03 6.58 -3.97 44.25 ...
##  $ Classifiers    : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ Gender         : logi  TRUE FALSE FALSE TRUE FALSE FALSE ...
##  $ NounClass      : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ PNN_1          : chr  "hame1242" "motu1246" "kosh1246" "awtu1239" ...
```

Comparing DNN and PNN

For each language that has a feature, look at the DNNs and PNNs. For each neighbor that has the same feature, the language gets +1. Then, for each language, take the ratio (sum(hasFeature)/amount of neighbors). Then, for each feature, look at the distribution of the ratio per language.

```
# reload the data to long format
data %>%
  select(Glottocode, Classifiers:DNN_14) %>%
  gather("Feature","Target",-c(Glottocode, PNN_1:DNN_14)) -> data.long
# make the list of features
feature.list <- data.long$Feature %>% unique()

# create empty table to store the result
table <- NULL %>% as.data.frame()

for(i in 1:length(feature.list)){
# select the feature
data.long %>%
  filter(Feature == feature.list[i]) %>%
  gather("Type","Neighbor",c(PNN_1:DNN_14)) %>%
    mutate(distance = as.numeric(str_replace_all(Type,".*_",""))) -> input.long
# create a table for matching
data.long %>%
  filter(Feature == feature.list[i]) -> input
# replace its neighbors by the value of the feature
input.long$Neighbor <- input$Target[match(input.long$Neighbor,input$Glottocode)]

# can set the distance to 1:501 if want to see the change according to the increase of distance
#for(z in 501:501){
# count the neighbor value for the feature for each language
input.long %>%
  filter(Target == TRUE) %>%
  filter(!is.na(Neighbor)) %>%
  select(-c(Feature, Target, distance)) %>%
  mutate(Type = str_replace_all(Type, "\\_.*","")) %>%
  group_by(Glottocode,Type, Neighbor) %>%
  summarise(perc = n()) %>% mutate(perc=perc/sum(perc)) %>%
  mutate(Neighbor = case_when(Neighbor == FALSE ~ 0,
                              Neighbor == TRUE ~ 1)) %>%
  mutate(perc = Neighbor*perc) %>%
  ungroup() %>% group_by(Glottocode, Type) %>%
  summarise(perc = sum(perc)) %>% ungroup() %>%
  mutate(Feature = feature.list[i]) %>%
  as.data.frame() -> tmp
table <- rbind(table, tmp)
#} # bracket if filter by distance
}

# extract the ratios
table %>%
  rename(Neighbor = Type, Value = perc) %>%
  select(-Glottocode) -> tmp
# remove not used data
rm(input, input.long)
```

Compare the distributions of PNNs for classifier and gender

```
wilcox.test(tmp %>% filter(Feature == "Classifiers" & Neighbor == "PNN") %>% pull(Value),
            tmp %>% filter(Feature == "Gender" & Neighbor == "PNN") %>% pull(Value),
```

```
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Classifiers" & Neighbor == "PNN") %>% pull(Value) and tmp %>% filt
## W = 195193, p-value <0.0000000000000002
## alternative hypothesis: true location shift is less than 0
```

Compare the distributions of PNNs for classifier and noun class

```
wilcox.test(tmp %>% filter(Feature == "Classifiers" & Neighbor == "PNN") %>% pull(Value),
            tmp %>% filter(Feature == "NounClass" & Neighbor == "PNN") %>% pull(Value),
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Classifiers" & Neighbor == "PNN") %>% pull(Value) and tmp %>% filt
## W = 79472, p-value <0.0000000000000002
## alternative hypothesis: true location shift is less than 0
```

Compare the distributions of PNNs for gender and noun class

```
wilcox.test(tmp %>% filter(Feature == "Gender" & Neighbor == "PNN") %>% pull(Value),
            tmp %>% filter(Feature == "NounClass" & Neighbor == "PNN") %>% pull(Value),
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Gender" & Neighbor == "PNN") %>% pull(Value) and tmp %>% filter(Fea
## W = 84686, p-value = 0.000001
## alternative hypothesis: true location shift is less than 0
```

Compare the distributions of DNNs for classifier and gender

```
wilcox.test(tmp %>% filter(Feature == "Classifiers" & Neighbor == "DNN") %>% pull(Value),
            tmp %>% filter(Feature == "Gender" & Neighbor == "DNN") %>% pull(Value),
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Classifiers" & Neighbor == "DNN") %>% pull(Value) and tmp %>% filt
## W = 207207, p-value = 0.00000000005
## alternative hypothesis: true location shift is less than 0
```

Compare the distributions of DNNs for classifier and noun class

```
wilcox.test(tmp %>% filter(Feature == "Classifiers" & Neighbor == "DNN") %>% pull(Value),
            tmp %>% filter(Feature == "NounClass" & Neighbor == "DNN") %>% pull(Value),
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Classifiers" & Neighbor == "DNN") %>% pull(Value) and tmp %>% filt
## W = 79545, p-value <0.0000000000000002
## alternative hypothesis: true location shift is less than 0
```

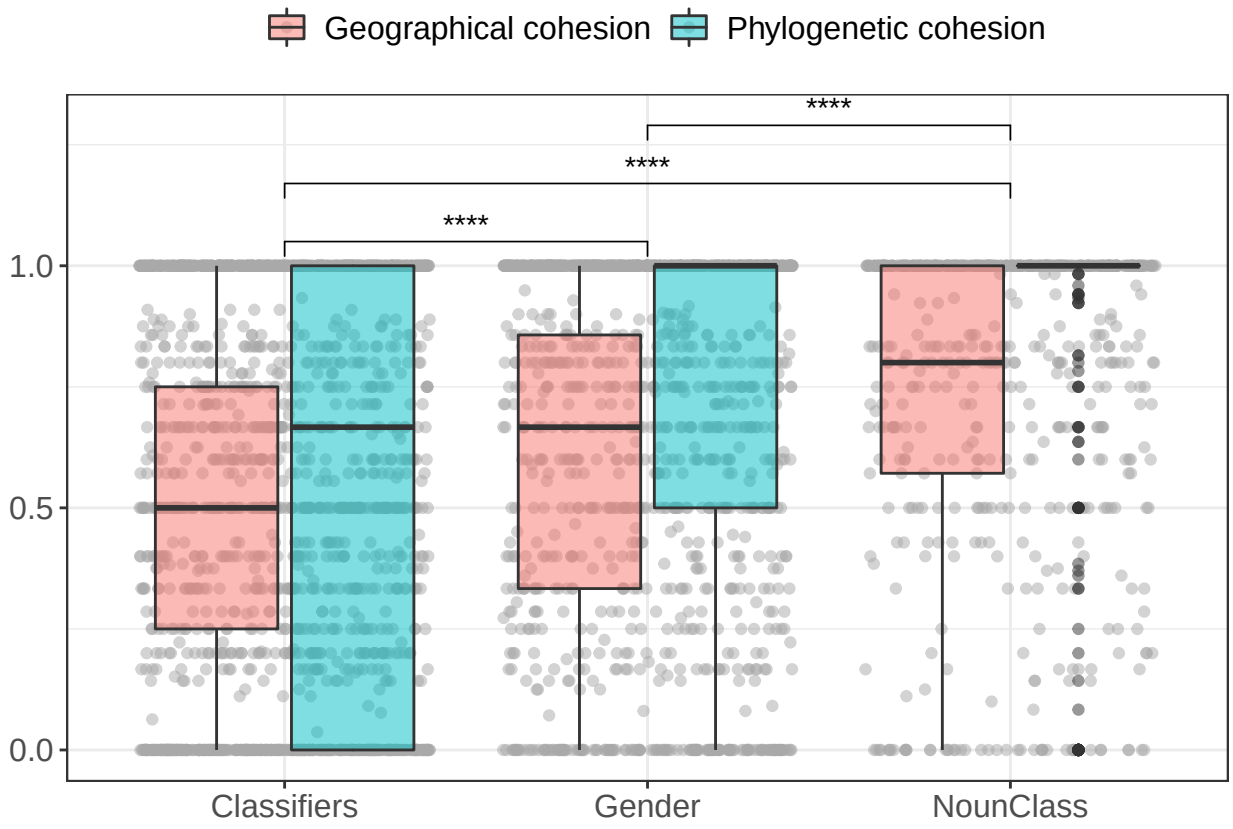
Compare the distributions of DNNs for gender and noun class

```
wilcox.test(tmp %>% filter(Feature == "Gender" & Neighbor == "DNN") %>% pull(Value),
            tmp %>% filter(Feature == "NounClass" & Neighbor == "DNN") %>% pull(Value),
            #paired = T,
            alternative = "less" )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tmp %>% filter(Feature == "Gender" & Neighbor == "DNN") %>% pull(Value) and tmp %>% filter(Fea
## W = 82972, p-value = 0.000005
## alternative hypothesis: true location shift is less than 0
```

Comparing the different neigbors across different systems

```
library(ggpubr)
my_comparisons <- list( c("Classifiers", "Gender"), c("Classifiers", "NounClass"), c("Gender", "NounClas
tmp %>%
  mutate(Neighbor = case_when(Neighbor == "DNN" ~ "Geographical cohesion",
                              Neighbor == "PNN" ~ "Phylogenetic cohesion")) %>%
  ggplot(aes(x = Feature, y = Value, fill = Neighbor)) +
  geom_jitter(alpha = 0.5, color = "darkgray") +
  geom_boxplot(alpha = 0.5) +
  #ggtitle("Latitude") +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        legend.position = "top",
        legend.title = element_blank(),
        axis.title.y = element_blank(),
        legend.text = element_text(size =12),
        axis.text = element_text(size = 12)) +
   stat_compare_means(comparisons = my_comparisons, label = "p.signif") #+
```




```
ggsave("Figure_2.1.png", width = 6, height = 4, dpi = 300)
```

We can also have a quick visual comparison of means and medians.

```
# get the mean
Mean <- tmp %>% group_by(Feature, Neighbor) %>% summarise(Value = mean(Value))
# get the median
Median <- tmp %>% group_by(Feature, Neighbor) %>% summarise(Value = median(Value))
Mean
```

```
## # A tibble: 6 x 3
## # Groups:   Feature [3]
##   Feature     Neighbor Value
##   <chr>       <chr>    <dbl>
## 1 Classifiers DNN      0.498
## 2 Classifiers PNN      0.566
## 3 Gender      DNN      0.605
## 4 Gender      PNN      0.755
## 5 NounClass   DNN      0.704
## 6 NounClass   PNN      0.859
```

```
Median
```

```
## # A tibble: 6 x 3
## # Groups:   Feature [3]
##   Feature     Neighbor Value
##   <chr>       <chr>    <dbl>
## 1 Classifiers DNN      0.5
## 2 Classifiers PNN      0.667
## 3 Gender      DNN      0.667
```

```
## 4 Gender      PNN      1
## 5 NounClass   DNN      0.8
## 6 NounClass   PNN      1
```

The overlap between DNN and PNN

Before starting the analysis, we can do some exploratory analysis to see if the neighbors make sense. First, we compare the output with a random baseline. In other words, if we randomize the location of the languages, what are the chances that GNNs will overlap with PNNs. First, we generate the random neighbors.

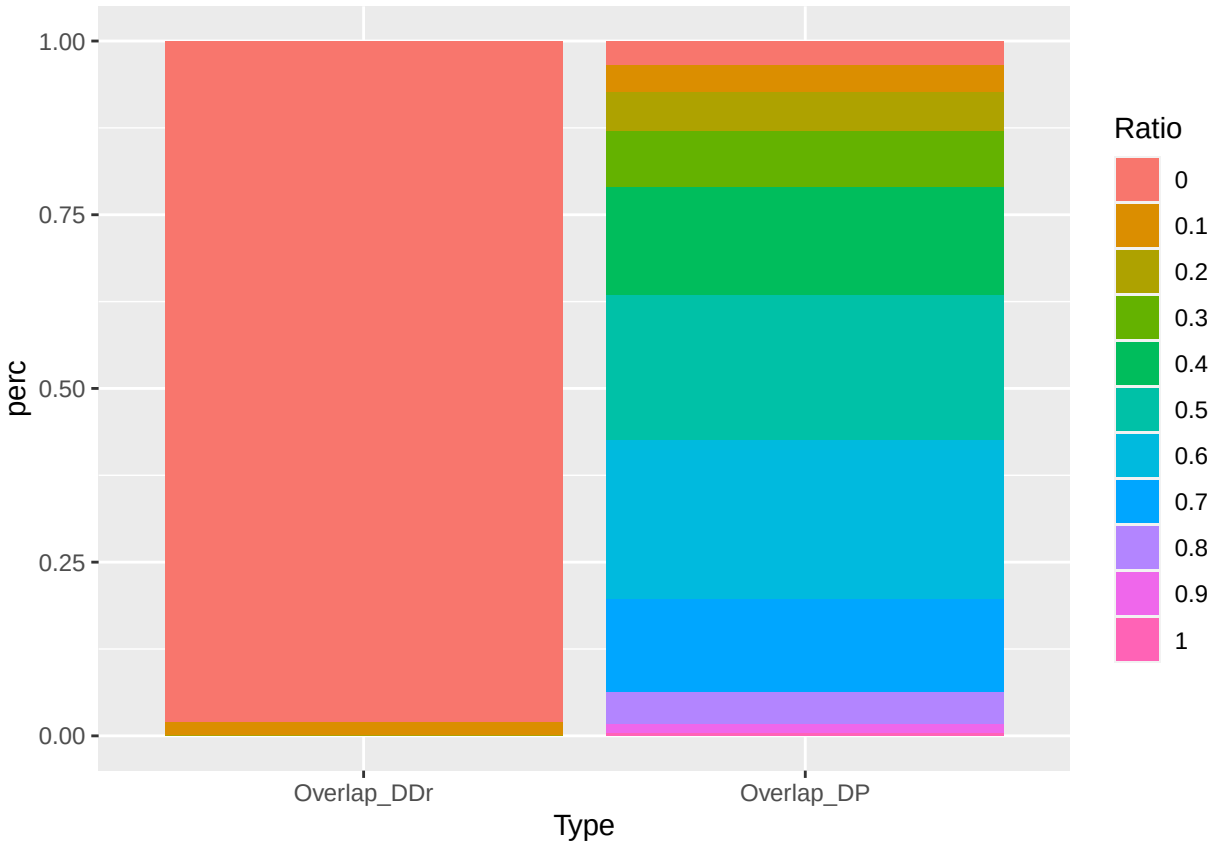
```
Window_size = 10
random.data <- data %>% select(-c(contains("DNN")))

for (x in 1:nrow(random.data)){
w <- sample(1:nrow(random.data),1)
random.data$Longitude[x] <- data$Longitude[w]
random.data$Latitude[x] <- data$Latitude[w]
}

library(geosphere)

# keep only one row per language
random.data -> tmp
output <- NULL
output2 <- NULL
GNNs <- NULL
for(i in 1:nrow(tmp)){
  output <- distHaversine(tmp[i,c("Longitude","Latitude")],tmp[,c("Longitude","Latitude")])
  names(output) <- tmp$Glottocode
  output2 <- output %>% sort()
  # short note here, some languages in Phoible have the same location, e.g., ID 2552 and 2468
  # we use the following line to remove cases when target language = neighbor
  output2 <- output2[which(names(output2) != tmp$Glottocode[i])]
  GNNs <- rbind(GNNs,names(output2)[1:Window_size])
}

# add row/column names and change format
rownames(GNNs) <- tmp$Glottocode
colnames(GNNs) <- paste("DNN_",1:Window_size,sep = "")
GNNs <- as.data.frame(GNNs, stringsAsFactors = F) %>%
  mutate(Glottocode = tmp$Glottocode)
random.data <- merge(random.data, GNNs, by = "Glottocode", all.x = T)

# remove not used data
rm(output,output2, GNNs, tmp)
str(random.data %>% select(contains("DNN")))
```

```
## 'data.frame':    3077 obs. of  10 variables:
##  $ DNN_1 : chr  "xara1244" "koko1269" "selk1253" "beri1254" ...
##  $ DNN_2 : chr  "bong1289" "krio1252" "isuf1235" "ulwa1239" ...
##  $ DNN_3 : chr  "mala1544" "mana1295" "waga1260" "sian1257" ...
##  $ DNN_4 : chr  "ziaa1250" "coas1294" "limo1248" "tetu1246" ...
##  $ DNN_5 : chr  "naba1256" "ngas1240" "maon1241" "iris1253" ...
##  $ DNN_6 : chr  "tena1240" "sayu1241" "cent2312" "mang1399" ...
##  $ DNN_7 : chr  "nyor1246" "tswa1255" "want1252" "nyan1313" ...
```

```
##  $ DNN_8 : chr  "rund1242" "lako1247" "bagr1243" "hupa1239" ...
##  $ DNN_9 : chr  "waru1265" "yand1257" "pwoe1235" "katb1237" ...
##  $ DNN_10: chr  "cent2058" "rawa1264" "east2652" "binu1245" ...
```

Then, we compare the overlap between the different types of neighbors and the random neighbors

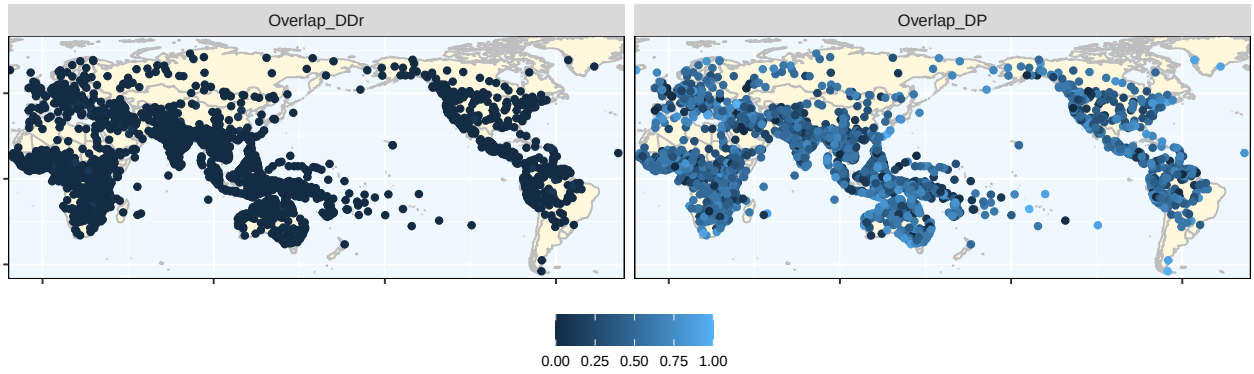
```
# for DNN and PNN
output <- NULL
for (i in 1:nrow(data)){
  output <- c(output, sum(c(data$DNN_1[i], data$DNN_2[i], data$DNN_3[i],
                            data$DNN_4[i], data$DNN_5[i],data$DNN_6[i],data$DNN_7[i],
                            data$DNN_8[i],data$DNN_9[i],data$DNN_10[i]) %in%
                            c(data$PNN_1[i], data$PNN_2[i], data$PNN_3[i],
                              data$PNN_4[i], data$PNN_5[i],data$PNN_6[i],data$PNN_7[i],
                              data$PNN_8[i],data$PNN_9[i],data$PNN_10[i]))/Window_size)
}
Overlap_DP <- output

# for DNN and randomDNN
output <- NULL
for (i in 1:nrow(data)){
  output <- c(output, sum(c(data$DNN_1[i], data$DNN_2[i], data$DNN_3[i],
                            data$DNN_4[i], data$DNN_5[i],data$DNN_6[i],data$DNN_7[i],
                            data$DNN_8[i],data$DNN_9[i],data$DNN_10[i]) %in%
                            c(random.data$DNN_1[i], random.data$DNN_2[i], random.data$DNN_3[i],
                              random.data$DNN_4[i], random.data$DNN_5[i], random.data$DNN_6[i],
                              random.data$DNN_7[i], random.data$DNN_8[i], random.data$DNN_9[i],
                              random.data$DNN_10[i]))/Window_size)
}
Overlap_DDr <- output
rm(output)

# visualize
cbind(Overlap_DP, Overlap_DDr) %>%
  as.data.frame() %>%
  gather("Type","Ratio") %>%
  mutate(Ratio = as.factor(Ratio)) %>%
  group_by(Type, Ratio) %>%
  summarise(count=n()) %>%
  mutate(perc=count/sum(count)) %>%
  ggplot(aes(x=Type, y = perc, fill = Ratio)) +
  geom_bar(stat = "identity")
```



We can try a different visualization and see if some areas have a high DNN-PNN overlap.

```
cbind(data %>% select(Longitude, Latitude),
      Overlap_DP, Overlap_DDr) %>%
  as.data.frame() %>%
  gather("Type","Ratio",-c(Longitude, Latitude)) -> mapdata

# function to change the coordinates to pacific-centred for visualization
shiftlong<-function(long) {
    if (long<(-20)){
        return(360+long)
        } else {return(long)
            }
    }

# remove NAs for coordinates
mapdata  %>%
  mutate(Longitude = as.numeric(Longitude)) %>%
  mutate(Latitude = as.numeric(Latitude)) %>%
  #mutate(Ratio = factor(Ratio)) %>%
  filter(!is.na(Longitude)) -> mapdata

# apply the function
mapdata$Longitude <- sapply(mapdata[,"Longitude"],shiftlong)

# change the coordinates in wolrd map
mapWorld <- map_data('world', wrap=c(-20,340), ylim=c(-60,100))
```

```
ggplot() +
  # set ocean color to blue
  theme(panel.background = element_rect(fill = "aliceblue"),
        # show panel border
        panel.border = element_rect(fill = NA)) +
  # can add xlim ylim below to zoom in on an area
  # e.g, for SMATTI xlim = c(60, 140), ylim = c(0,45)
  coord_equal(expand = FALSE) +
  geom_polygon(data = mapWorld, aes(x=long, y = lat, group = group) ,
               # set land color and coast color
               fill = "cornsilk", color = "grey") +
  # add language points
  geom_point(data = mapdata, aes(x = Longitude, y = Latitude, color = Ratio)) +
  facet_wrap(~Type) +
  theme(legend.position = "bottom",
        axis.title=element_blank(),
        legend.title = element_blank(),
        axis.text = element_blank(),
        legend.text=element_text(size=8))
```



If we want further details, the table below shows the overlap table between DNN and PNN for all languages

```
output <- NULL %>% as.data.frame()
for(x in 1:Window_size){
    tmp <- NULL
    tmp1 <- NULL
  for(z in 1:Window_size){
    sum(data %>% select(paste("DNN_",x, sep = "")) == data %>% select(paste("PNN_",z, sep = "")) ,
    na.rm = T) -> tmp
    tmp1 <- c(tmp1,tmp)
  }
  output <- rbind(output, tmp1)
}
rownames(output) <- c("DNN_1","DNN_2","DNN_3","DNN_4","DNN_5","DNN_6","DNN_7","DNN_8","DNN_9","DNN_10")
output %>% select(1:10) %>% rename(PNN_1 = 1, PNN_2 = 2, PNN_3 = 3, PNN_4 = 4, PNN_5 = 5,
                                   PNN_6 = 6, PNN_7 = 7, PNN_8 = 8, PNN_9 = 9, PNN_10 = 10) -> output
# remove not used data
rm(mapdata, random.data, tmp, tmp1)
output
```

```
##        PNN_1 PNN_2 PNN_3 PNN_4 PNN_5 PNN_6 PNN_7 PNN_8 PNN_9 PNN_10
## DNN_1    287   101    62    25    15    20    12     6     9      5
```

```
## DNN_2    305    98    48    27    21    16    12     6     8      3
## DNN_3    336    98    50    26    27     8     7    11     7      3
## DNN_4    292   121    52    23    16    12    12     5     6      4
## DNN_5    221    90    41    21    11     5     6    12     2      3
## DNN_6    125    62    24    15    13     6     6     6     8      6
## DNN_7     51    38    19    11     3     5     1     1     3      2
## DNN_8     20    12     7     0     2     0     1     0     1      1
## DNN_9      9     4     1     1     0     0     2     0     0      0
## DNN_10     3     0     0     0     0     0     0     0     0      0
```

We can also extract the mean overlap between the ten neighbors of DNNs and PNNs for all languages

```
## [1] "For all languages, the overlap between DNNs and PNNs is 0.488982775430614"
```

for five neighbors only

```
values <- NULL
tmp <- data
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i])))}
paste("For all languages, the overlap between DNNs and PNNs is ",
      sum(values)/(5*nrow(tmp)), sep = "")
```

```
## [1] "For all languages, the overlap between DNNs and PNNs is 0.196425089372766"
```

We can do the same analysis per feature, first for gender

```
output <- NULL %>% as.data.frame()
for(x in 1:Window_size){
    tmp <- NULL
    tmp1 <- NULL
  for(z in 1:Window_size){
    sum(data %>% filter(Gender == TRUE) %>% select(paste("DNN_",x, sep = "")) ==
          data %>% filter(Gender == TRUE) %>% select(paste("PNN_",z, sep = "")) ,
    na.rm = T) -> tmp
    tmp1 <- c(tmp1,tmp)
  }
  output <- rbind(output, tmp1)
}
rownames(output) <- c("DNN_1","DNN_2","DNN_3","DNN_4","DNN_5","DNN_6","DNN_7","DNN_8","DNN_9","DNN_10")
output %>% select(1:10) %>% rename(PNN_1 = 1, PNN_2 = 2, PNN_3 = 3, PNN_4 = 4, PNN_5 = 5,
                                   PNN_6 = 6, PNN_7 = 7, PNN_8 = 8, PNN_9 = 9, PNN_10 = 10) -> output
# remove not used data
rm(tmp,  tmp1)
# show the table
output
```

```
##        PNN_1 PNN_2 PNN_3 PNN_4 PNN_5 PNN_6 PNN_7 PNN_8 PNN_9 PNN_10
## DNN_1     52    17    14     7     4     3     3     2     2      1
## DNN_2     60    27    14     5     4     4     5     1     1      0
## DNN_3     63    17     8    10     6     2     2     5     2      0
## DNN_4     55    24     8     4     6     3     4     1     2      1
## DNN_5     34    11    13     1     5     1     2     3     0      1
## DNN_6     20    13     6     6     4     3     1     2     0      1
## DNN_7     12     8     0    10     0     1     0     0     2      2
## DNN_8      4     3     0     0     1     0     0     0     0      0
```

```
## DNN_9      1     2     0     0     0     0     1     0     0      0
## DNN_10     0     0     0     0     0     0     0     0     0      0
```

```
values <- NULL
tmp <- data %>% filter(Gender == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i],
                            tmp$DNN_6[i],tmp$DNN_7[i],tmp$DNN_8[i],tmp$DNN_9[i],tmp$DNN_10[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i],
                              tmp$PNN_6[i],tmp$PNN_7[i],tmp$PNN_8[i],tmp$PNN_9[i],tmp$PNN_10[i])))}
paste("For gender languages, the overlap between DNNs and PNNs is ",
      sum(values)/(Window_size*nrow(tmp)), sep = "")
```

```
## [1] "For gender languages, the overlap between DNNs and PNNs is 0.476813880126183"
```

for five neighbors only

```
values <- NULL
tmp <- data %>% filter(Gender == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i])))}
paste("For gender languages, the overlap between DNNs and PNNs is ",
      sum(values)/(5*nrow(tmp)), sep = "")
```

```
## [1] "For gender languages, the overlap between DNNs and PNNs is 0.190220820189274"
```

For classifiers

```
output <- NULL %>% as.data.frame()
for(x in 1:Window_size){
    tmp <- NULL
    tmp1 <- NULL
  for(z in 1:Window_size){
    sum(data %>% filter(Classifiers == TRUE) %>% select(paste("DNN_",x, sep = "")) ==
          data %>% filter(Classifiers == TRUE) %>% select(paste("PNN_",z, sep = "")) ,
    na.rm = T) -> tmp
    tmp1 <- c(tmp1,tmp)
  }
  output <- rbind(output, tmp1)
}
rownames(output) <- c("DNN_1","DNN_2","DNN_3","DNN_4","DNN_5","DNN_6","DNN_7","DNN_8","DNN_9","DNN_10")
output %>% select(1:10) %>% rename(PNN_1 = 1, PNN_2 = 2, PNN_3 = 3, PNN_4 = 4, PNN_5 = 5,
                                   PNN_6 = 6, PNN_7 = 7, PNN_8 = 8, PNN_9 = 9, PNN_10 = 10) -> output
# remove not used data
rm(tmp,  tmp1)
# show the table
output
```

```
##        PNN_1 PNN_2 PNN_3 PNN_4 PNN_5 PNN_6 PNN_7 PNN_8 PNN_9 PNN_10
## DNN_1     80    19    18     5     2     4     4     3     3      1
## DNN_2     81    27    16     6     6     5     2     1     2      3
## DNN_3     85    29    11     4    10     2     2     0     2      0
## DNN_4     79    31    15     6     1     3     4     2     1      0
## DNN_5     67    27    11     9     1     0     0     3     1      0
## DNN_6     30    19     8     2     3     1     3     2     3      2
## DNN_7      9    12     4     2     2     0     1     1     1      0
```

```
## DNN_8      2     3     2     0     1     0     0     0     1      1
## DNN_9      3     2     0     0     0     0     0     0     0      0
## DNN_10     1     0     0     0     0     0     0     0     0      0
```

```
values <- NULL
tmp <- data %>% filter(Classifiers == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i],
                            tmp$DNN_6[i],tmp$DNN_7[i],tmp$DNN_8[i],tmp$DNN_9[i],tmp$DNN_10[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i],
                              tmp$PNN_6[i],tmp$PNN_7[i],tmp$PNN_8[i],tmp$PNN_9[i],tmp$PNN_10[i])))}
paste("For classifier languages, the overlap between DNNs and PNNs is ",
      sum(values)/(Window_size*nrow(tmp)), sep = "")
```

```
## [1] "For classifier languages, the overlap between DNNs and PNNs is 0.498771498771499"
```

for five neighbors only

```
values <- NULL
tmp <- data %>% filter(Classifiers == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i])))}
paste("For classifier languages, the overlap between DNNs and PNNs is ",
      sum(values)/(5*nrow(tmp)), sep = "")
```

```
## [1] "For classifier languages, the overlap between DNNs and PNNs is 0.2"
```

For noun class

```
output <- NULL %>% as.data.frame()
for(x in 1:Window_size){
    tmp <- NULL
    tmp1 <- NULL
  for(z in 1:Window_size){
    sum(data %>% filter(NounClass == TRUE) %>% select(paste("DNN_",x, sep = "")) ==
          data %>% filter(NounClass == TRUE) %>% select(paste("PNN_",z, sep = "")) ,
    na.rm = T) -> tmp
    tmp1 <- c(tmp1,tmp)
  }
  output <- rbind(output, tmp1)
}
rownames(output) <- c("DNN_1","DNN_2","DNN_3","DNN_4","DNN_5","DNN_6","DNN_7","DNN_8","DNN_9","DNN_10")
output %>% select(1:10) %>% rename(PNN_1 = 1, PNN_2 = 2, PNN_3 = 3, PNN_4 = 4, PNN_5 = 5,
                                   PNN_6 = 6, PNN_7 = 7, PNN_8 = 8, PNN_9 = 9, PNN_10 = 10) -> output
# remove not used data
rm(tmp,  tmp1)
# show the table
output
```

```
##        PNN_1 PNN_2 PNN_3 PNN_4 PNN_5 PNN_6 PNN_7 PNN_8 PNN_9 PNN_10
## DNN_1     25    11     4     1     1     0     0     0     0      0
## DNN_2     38     4     1     1     2     1     0     1     1      0
## DNN_3     33    13     5     1     0     0     2     0     0      0
## DNN_4     29    17     6     3     2     0     1     1     0      1
## DNN_5     34     8     7     1     0     1     0     0     0      0
## DNN_6     16     8     1     1     2     0     0     1     0      1
```

```
## DNN_7      8     5     3     0     0     1     0     0     0     0
## DNN_8      1     2     1     0     0     0     0     0     0     1
## DNN_9      0     0     0     1     0     0     0     0     0     0
## DNN_10     0     0     0     0     0     0     0     0     0     0
```

```
values <- NULL
tmp <- data %>% filter(NounClass == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i],
                            tmp$DNN_6[i],tmp$DNN_7[i],tmp$DNN_8[i],tmp$DNN_9[i],tmp$DNN_10[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i],
                              tmp$PNN_6[i],tmp$PNN_7[i],tmp$PNN_8[i],tmp$PNN_9[i],tmp$PNN_10[i])))}
paste("For noun class languages, the overlap between DNNs and PNNs is ",
      sum(values)/(Window_size*nrow(tmp)), sep = "")
```

```
## [1] "For noun class languages, the overlap between DNNs and PNNs is 0.475394321766561"
```

for five neighbors only

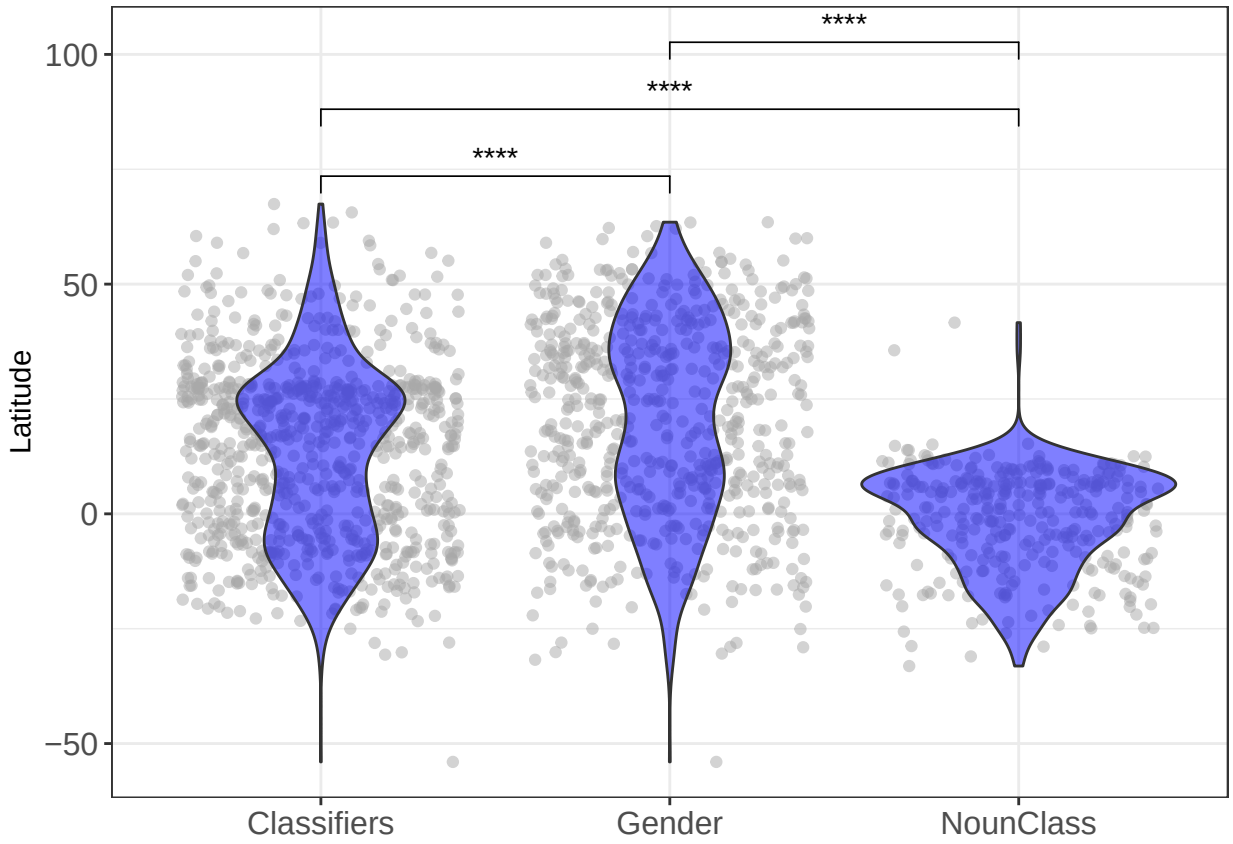
```
values <- NULL
tmp <- data %>% filter(NounClass == TRUE)
for(i in 1:nrow(tmp)){
  values <- c(values, sum(c(tmp$DNN_1[i],tmp$DNN_2[i],tmp$DNN_3[i],tmp$DNN_4[i],tmp$DNN_5[i]) %in%
                            c(tmp$PNN_1[i],tmp$PNN_2[i],tmp$PNN_3[i],tmp$PNN_4[i],tmp$PNN_5[i])))}
paste("For noun class languages, the overlap between DNNs and PNNs is ",
      sum(values)/(5*nrow(tmp)), sep = "")
```

```
## [1] "For noun class languages, the overlap between DNNs and PNNs is 0.182334384858044"
```

2.2 Testing longitude and latitude

Following the 'Continental Axis Theory', humans tend to migrate east and west rather than north and south to stay within similar climatic conditions for farming. This suggests that features that spread more by language expansion should have a wide range of latitude. To investigate this hypothesis, we have a look at the distribution of longitude and latitude

```
library(ggpubr)
my_comparisons <- list( c("Classifiers", "Gender"), c("Classifiers", "NounClass"), c("Gender", "NounClas
data %>%
  select(Glottocode, Longitude, Latitude, Classifiers:NounClass) %>%
  pivot_longer(names_to = "Feature", values_to = "Value", -c(Glottocode, Longitude, Latitude)) %>%
  filter(Value == TRUE) %>%
  ggplot(aes(x = Feature, y = Latitude)) +
  geom_jitter(alpha = 0.5, color = "darkgray") +
  geom_violin(alpha = 0.5, fill = "blue") +
  #ggtitle("Latitude") +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        axis.text = element_text(size = 12)) +
   stat_compare_means(comparisons = my_comparisons, label = "p.signif")
```




```
ggsave("Figure_2.2.png", width = 6, height = 4, dpi = 300)
```

Then, we run statistical tests to evaluate the variation between the distributions. First, for the comparison between classifiers and gender.

```
data %>%
  select(Glottocode, Longitude, Latitude, Classifiers:NounClass) %>%
  pivot_longer(names_to = "Feature", values_to = "Value", -c(Glottocode, Longitude, Latitude)) %>%
  as.data.frame() %>%
  filter(Value == TRUE) -> test

clf <- test$Latitude[which(test$Feature== "Classifiers") ]
gen <- test$Latitude[which(test$Feature== "Gender") ]
ncl <- test$Latitude[which(test$Feature== "NounClass") ]

wilcox.test(clf, gen)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  clf and gen
## W = 206354, p-value = 0.00000000006
## alternative hypothesis: true location shift is not equal to 0
```

Then, for the comparison between classifiers and noun class.

```
wilcox.test(clf, ncl)
```

```
##
```

```
##  Wilcoxon rank sum test with continuity correction
##
## data:  clf and ncl
## W = 187491, p-value <0.0000000000000002
## alternative hypothesis: true location shift is not equal to 0
```

Finally, for the comparison between gender and noun class.

```
wilcox.test(gen, ncl)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  gen and ncl
## W = 160002, p-value <0.0000000000000002
## alternative hypothesis: true location shift is not equal to 0
```

2.3 Comparing family density

The current report uses environmental data from Derungs et al (2018). In such a data, the languages are assigned variables based on the following method: the world is cut into 3267 grids. The environmental factors are extracted for each grid. Then, the closest center of a grid is used to assign languages to grids. See the following links for more details:

- https://royalsocietypublishing.org/doi/full/10.1098/rspb.2017.2851
- https://royalsocietypublishing.org/action/downloadSupplement?doi=10.1098%2Frspb.2017.2851&file=rspb20172851supp1.pdf

First, we read the environmental data

```
library(geosphere)
# get data from Derungs et 2018
load("data_raw/DataForCode/randPtsLangPoissEnv_3267.Rdata")
reg.spdf %>%
  as.data.frame() %>%
  select(c(id, lon, lat, dist_river, dist_ocean, elev, elevSD,
           n_warm_month_2ad, temp_mean_2ad, per_wetqu_2ad, per_cv_2ad,
           temp_warmqu_2ad, gras_2ad, crop_2ad, pop_2ad)) %>%
  # remove six cells with few NA values
  drop_na() -> env.data
rm(reg.spdf)

# duplicate master data
tmp <- data # data.backup
# reset master data
#data <- data.backup
# create empty vectors for storing the output
output <- NULL
output2 <- NULL
GNNs <- NULL
# for each language
for(i in 1:nrow(tmp)){
  # measure its distance with cell center points
  output <- distHaversine(tmp[i,c("Longitude","Latitude")],env.data[,c("lon","lat")])
  names(output) <- env.data$id
  # sort them and chooese the closest one
  output2 <- output %>% sort()
```

```
  GNNs <- rbind(GNNs,names(output2)[1])
}

# add row/column names and change format
rownames(GNNs) <- tmp$Glottocode
colnames(GNNs) <- c("GNN")

# adding the results in a new column of the main data file
data$id = GNNs
# calculate family density per grid
data %>% select(Family_GLOT_Lv1, id) %>% distinct() %>%
  group_by(id) %>% summarise(count = n()) %>%
  rename(id = 1, dens_fam = 2) -> dens.fam
# calculate lang density per grid
data %>% select(id) %>% table() %>% as.data.frame() %>%
  filter(Freq > 0) %>% rename(id = 1, dens_lang = 2) -> dens.lang
data.env <- merge(data, env.data, by = "id", all.x = T) %>%
  merge(dens.fam, by = "id", all.x = T) %>%
  merge(dens.lang, by = "id", all.x = T) %>%
  select(-c(id, lon, lat)) %>%
  select(Glottocode:Area_GLOT, Classifiers:NounClass, dens_lang, dens_fam ,dist_river:pop_2ad)
# remove not used data
rm(GNNs, tmp, output, output2, env.data, dens.lang, dens.fam)

#normalize the variables
range01 <- function(x){(x-min(x, na.rm = T))/(max(x, na.rm = T)-min(x, na.rm = T))}
data.env %>% mutate_at(vars(dens_lang:pop_2ad), range01) -> data.env

# Sanity check on numbers of NAs in the data
# colSums(is.na(data.env))
```

Export the environmental data if needed.

```
data.env %>% write.csv("data_tidy/data_env.csv", fileEncoding = "UTF-8", row.names = FALSE)
```

Read the environmental data if needed. We read all environmental factors here, but only family density is considered in the current subsection. The other environmental factors will be analyzed in subsection 2.5.

```
data.env <- read.csv("data_tidy/data_env.csv")
str(data.env)
```

```
## 'data.frame':    3077 obs. of  21 variables:
##  $ Glottocode      : chr  "west2650" "oldk1249" "cent2314" "chon1284" ...
##  $ Family_GLOT_Lv1 : chr  "aust1307" "aust1305" "aust1305" "aust1305" ...
##  $ Genus           : chr  "Malayo-Sumbawan" NA NA NA ...
##  $ Area_GLOT       : chr  "Eurasia" "Eurasia" "Eurasia" "Eurasia" ...
##  $ Classifiers     : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Gender          : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ NounClass       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ dens_lang       : num  0.0357 0.0357 0.0357 0.0357 0 ...
##  $ dens_fam        : num  0.143 0.143 0.143 0.143 0 ...
##  $ dist_river      : num  0.0396 0.0396 0.0396 0.0396 0.0126 ...
##  $ dist_ocean      : num  0.258 0.258 0.258 0.258 0.187 ...
##  $ elev            : num  0.01065 0.01065 0.01065 0.01065 0.00482 ...
##  $ elevSD          : num  0.1437 0.1437 0.1437 0.1437 0.0177 ...
```

```
##  $ n_warm_month_2ad: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ temp_mean_2ad   : num  0.942 0.942 0.942 0.942 0.951 ...
##  $ per_wetqu_2ad   : num  0.209 0.209 0.209 0.209 0.287 ...
##  $ per_cv_2ad      : num  0.363 0.363 0.363 0.363 0.423 ...
##  $ temp_warmqu_2ad : num  0.837 0.837 0.837 0.837 0.849 ...
##  $ gras_2ad        : num  0.0963 0.0963 0.0963 0.0963 0.0224 ...
##  $ crop_2ad        : num  0.391 0.391 0.391 0.391 0.174 ...
##  $ pop_2ad         : num  0.0129 0.0129 0.0129 0.0129 0.0351 ...
```

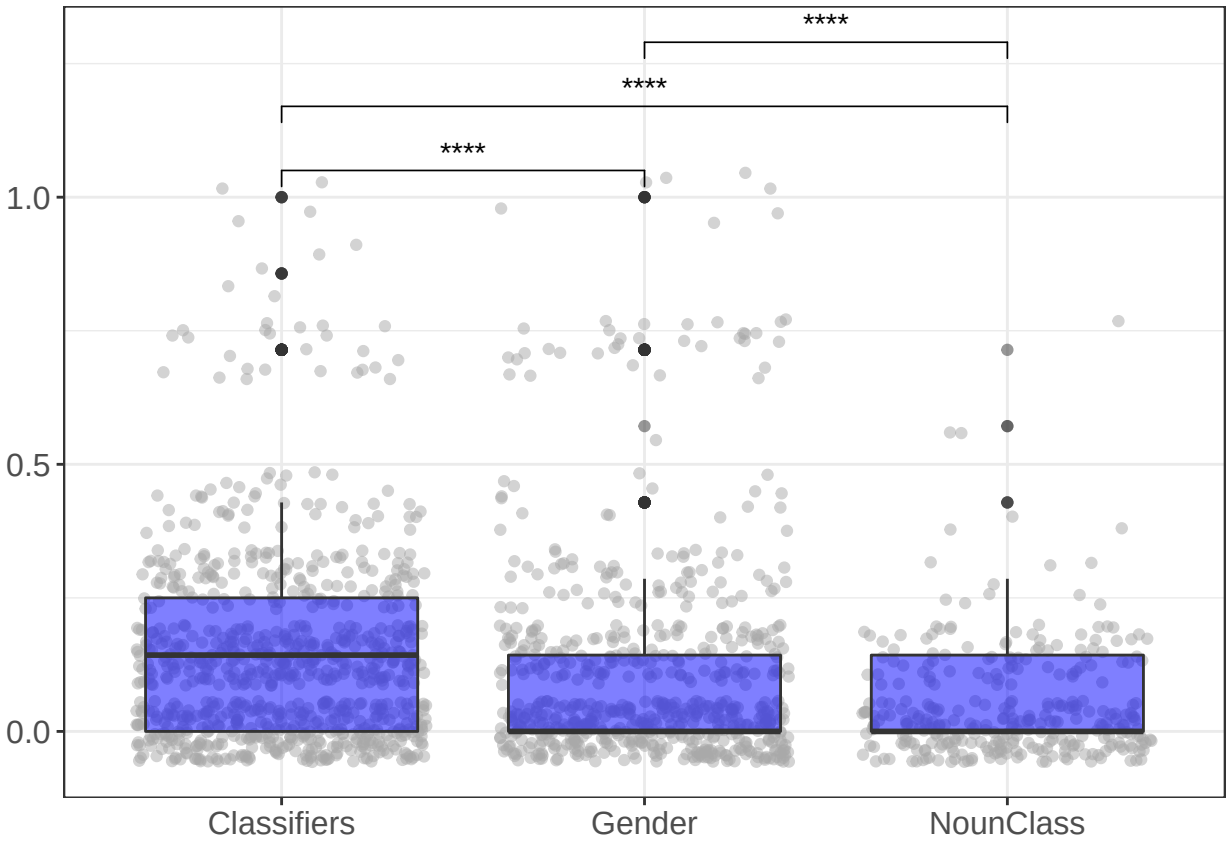
We test for family density. The hypothesis is: If classifiers spread more by feature diffusion and less by language expansion, while it is the opposite for gender and noun class, we expect that classifier languages will be found in areas with higher diversity of language families (i.e., higher language family density). First, we have a look at the mean and the median family density for each feature

```
data.env %>%
  select(Classifiers:NounClass, dens_fam) %>%
  pivot_longer(names_to = "Feature", values_to = "Value", -dens_fam) %>%
  filter(Value == TRUE) %>%
  group_by(Feature) %>% summarise(Mean = mean(dens_fam), Median = median(dens_fam), .groups = 'drop')
```

```
## # A tibble: 3 x 3
##   Feature       Mean Median
##   <chr>        <dbl>  <dbl>
## 1 Classifiers 0.145   0.143
## 2 Gender      0.121   0
## 3 NounClass   0.0532  0
```

We visualize the difference of distribution.

```
data.env %>%
  select(Classifiers:NounClass, dens_fam) %>%
  pivot_longer(names_to = "Feature", values_to = "Value", -dens_fam) %>%
  filter(Value == TRUE) %>%
  select(-Value) %>%
  ggplot(aes(x = Feature, y= dens_fam)) +
  geom_jitter(alpha = 0.5, color = "darkgray") +
  geom_boxplot(alpha = 0.5, fill = "blue") +
  #ggtitle("Latitude") +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        legend.position = "top",
        legend.title = element_blank(),
        axis.title.y = element_blank(),
        legend.text = element_text(size =12),
        axis.text = element_text(size = 12)) +
   stat_compare_means(comparisons = my_comparisons, label = "p.signif")
```



Then, we run statistical tests to evaluate the difference of family density across families with classifier, gender, and noun class. First, for the comparison between classifiers and gender.

```
data.env %>%
  select(Classifiers:NounClass, dens_fam) %>%
  pivot_longer(names_to = "Feature", values_to = "Value", -dens_fam) %>%
  filter(Value == TRUE) %>%
  select(-Value) %>%
  mutate(id = 1:nrow(.)) %>%
  pivot_wider(names_from = Feature, values_from = dens_fam) -> tmp
clf <- tmp$Classifiers[!is.na(tmp$Classifiers)]
gen <- tmp$Gender[!is.na(tmp$Gender)]
ncl <- tmp$NounClass[!is.na(tmp$NounClass)]

wilcox.test(clf, gen, alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  clf and gen
## W = 294410, p-value = 0.0000003
## alternative hypothesis: true location shift is greater than 0
```

Then, we compare the distributions of classifiers and noun class.

```
wilcox.test(clf, ncl, alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
```

```
##
## data:  clf and ncl
## W = 171006, p-value <0.0000000000000002
## alternative hypothesis: true location shift is greater than 0
```

Finally, we compare the distributions of gender and noun class.

```
wilcox.test(gen, ncl, alternative = "greater")
```

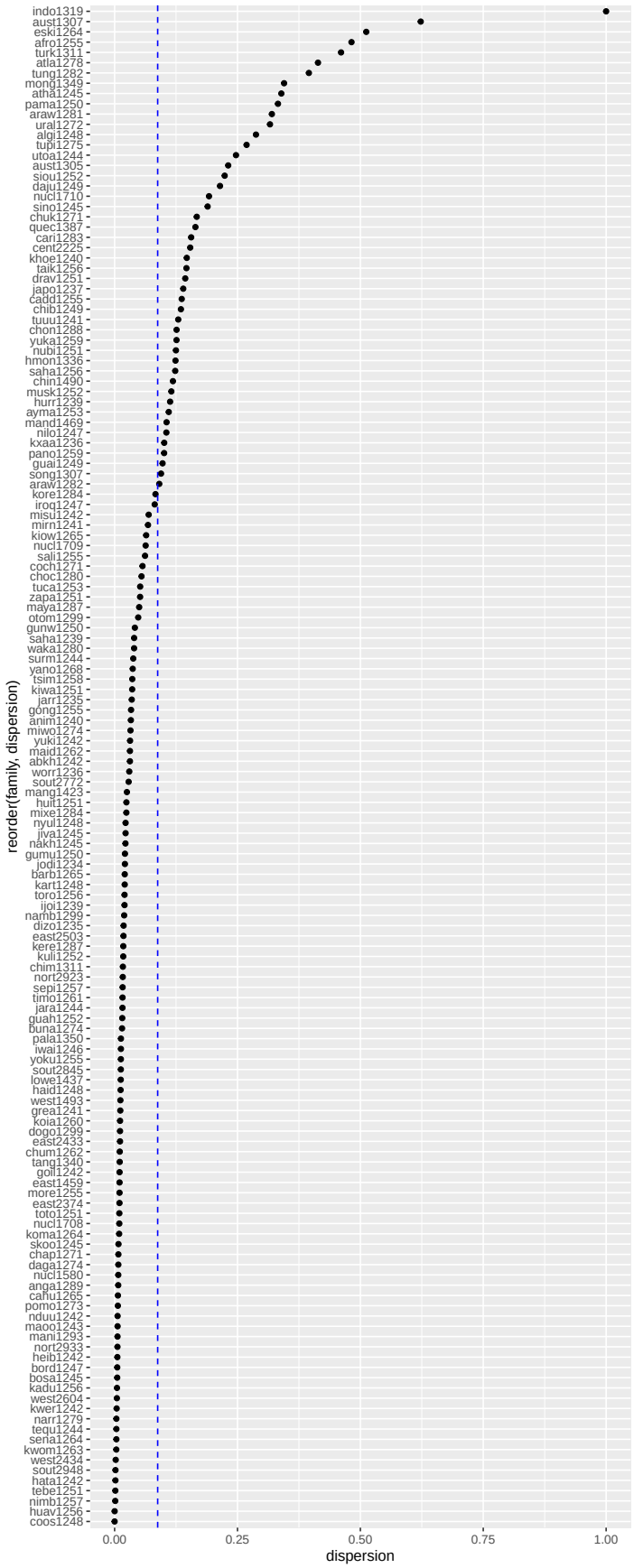
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  gen and ncl
## W = 117264, p-value = 0.0000006
## alternative hypothesis: true location shift is greater than 0
```

2.4 The dispersion of families

We visualize the geographical coverage of language families, i.e., the average dispersion per family, e.g., is the distance between languages larger for Indo-European compared to other families? The displayed value is normalized

```
df <- data %>% select(Glottocode, Family = Family_GLOT_Lv1, Longitude, Latitude)
geo.df <- data %>% select(Glottocode, Longitude, Latitude)
# extracting the list of families
families <- data %>% pull(Family_GLOT_Lv1) %>% unique

# create an empty table
table <- NULL %>% as.data.frame()
# for each family
for(i in 1:length(families)){
  # extract the Glottocode from the family pairs
  list.tmp <- df %>% filter(Family == families[i]) %>% pull(Glottocode)
  # generate all the possible combinations between the languages
  combn(list.tmp, 2) -> list.tmp
  cbind(list.tmp[1,],list.tmp[2,]) %>%
    as.data.frame() %>%
    rename(source = 1, target = 2) %>%
    merge(geo.df, by.x = "target", by.y = "Glottocode") %>%
    rename(Long.target = Longitude, Lat.target = Latitude) %>%
    merge(geo.df, by.x = "source", by.y = "Glottocode") %>%
    rename(Long.source = Longitude, Lat.source = Latitude) %>%
    rowwise() %>%
    mutate(dist = geosphere::distHaversine(c(Long.target, Lat.target),
                                           c(Long.source,Lat.source))) %>%
    ungroup() %>%
    as.data.frame() %>%
    mutate(family = families[i]) %>%
    select(source, target, family, dispersion = dist) -> list.tmp
  table <- rbind(table, list.tmp)
}

# extract information of gender
data %>%
  select(Family = Family_GLOT_Lv1, Gender) %>%
  mutate(System = as.numeric(Gender)) %>%
```

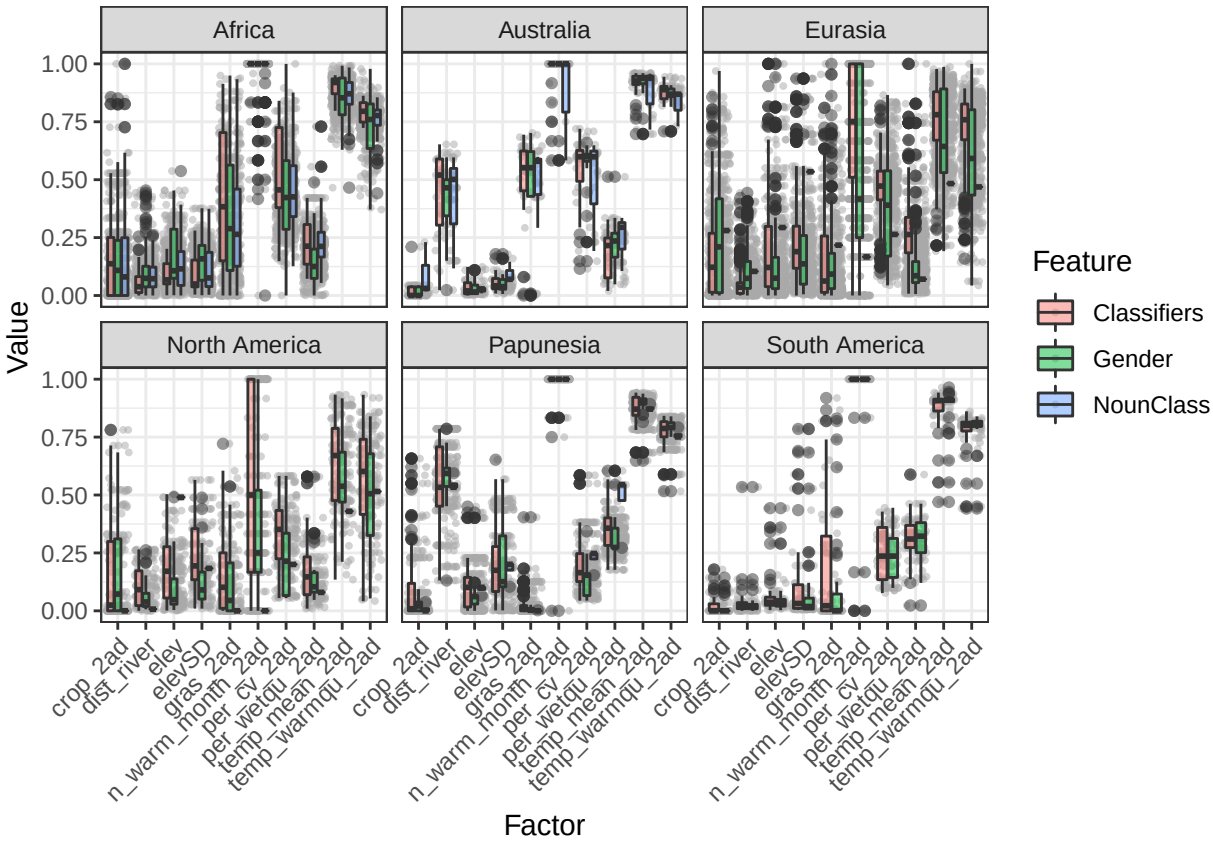
```
  group_by(Family) %>%
  mutate(ratio = sum(System),
         total = n()) %>%
  select(-c(System, Gender)) %>% distinct() %>%
  mutate(gender = ratio/total) %>%
  select(-c(ratio,total)) -> df.gender

# extract information of classifier
data %>%
  select(Family = Family_GLOT_Lv1, Classifiers) %>%
  mutate(System = as.numeric(Classifiers)) %>%
  group_by(Family) %>%
  mutate(ratio = sum(System),
         total = n()) %>%
  select(-c(System, Classifiers)) %>% distinct() %>%
  mutate(classifier = ratio/total) %>%
  select(-c(ratio,total)) -> df.classifier

# extract information of noun class
data %>%
  select(Family = Family_GLOT_Lv1, NounClass) %>%
  mutate(System = as.numeric(NounClass)) %>%
  group_by(Family) %>%
  mutate(ratio = sum(System),
         total = n()) %>%
  select(-c(System, NounClass)) %>% distinct() %>%
  mutate(nounclass = ratio/total) %>%
  select(-c(ratio,total)) -> df.nounclass

#normalize the variables
range01 <- function(x){(x-min(x, na.rm = T))/(max(x, na.rm = T)-min(x, na.rm = T))}

# normalize the dispersion per family
table %>%
  mutate(dispersion = as.numeric(dispersion)) %>%
  group_by(family) %>%
  mutate(family.count = length(family)) %>%
  mutate(dispersion.sum = sum(dispersion),
         dispersion = dispersion.sum/family.count) %>%
  ungroup() %>%
  mutate(dispersion = range01(dispersion)) %>%
  select(family, dispersion) %>%
  distinct() %>%
  arrange(desc(dispersion)) %>%
  ggplot(aes(x = reorder(family, dispersion), y = dispersion)) +
  geom_point() +
  coord_flip() +
  geom_hline(yintercept = 0.08758, color = "blue", linetype = "dashed")
```


reorder(family, dispersion)
indo1319
aust1307
eski1264
afro1255
turk1311
atla1278
tung1282
mong1349
atha1245
pama1250
araw1281
ural1272
algi1248
tupi1275
utoa1244
aust1305
siou1252
daju1249
nucl1710
sino1245
chuk1271
quec1387
cari1283
ceni2225
khoe1240
taik1256
drav1251
japo1237
cadd1255
chib1249
tuuu1241
chon1288
yuka1259
nubi1251
hmon1336
saha1256
chin1490
musk1252
hurr1239
ayma1253
mand1469
nilo1247
kxaa1236
pano1259
guai1249
song1307
araw1282
kore1284
iroq1247
misu1242
mirn1241
kiow1265
nucl1709
sali1255
coch1271
choc1280
tuca1253
zapa1251
maya1287
otom1299
gunw1250
saha1239
waka1280
surm1244
yano1268
tsim1258
kiwa1251
jarr1235
gong1255
anim1240
miwo1274
yuki1242
maid1262
abkh1242
worr1236
sout2772
mang1423
huil1251
mixe1284
nyul1248
jiva1245
nakh1245
gumu1250
jodi1234
barb1265
kart1248
toro1256
jjoi1239
namb1299
dizo1235
east2503
kere1287
kuli1252
chim1311
nori2923
sepi1257
timo1261
jara1244
guah1252
buna1274
pala1350
iwai1246
yoku1255
sout2845
lowe1437
haid1248
west1493
grea1241
koia1260
dogo1299
east2433
chum1262
tang1340
goil1242
east1459
more1255
east2374
toto1251
nucl1708
koma1264
skoo1245
chap1271
daga1274
nucl1580
anga1289
cahu1265
pomo1273
nduu1242
maoo1243
mani1293
nori2933
heib1242
bord1247
bosa1245
kadu1256
west2604
kwer1242
narr1279
tequ1244
sena1264
kwom1263
west2434
sout2948
hata1242
tebe1251
nimb1257
huav1256
coos1248
0.00
0.25
0.50
0.75
1.00
dispersion

2.5 Assessing environmental factors

First, we visualize the difference of environmental factors across languages with classifier, gender, and noun class in different Glottoareas. The source of the environmental factors is explained in 2.3.

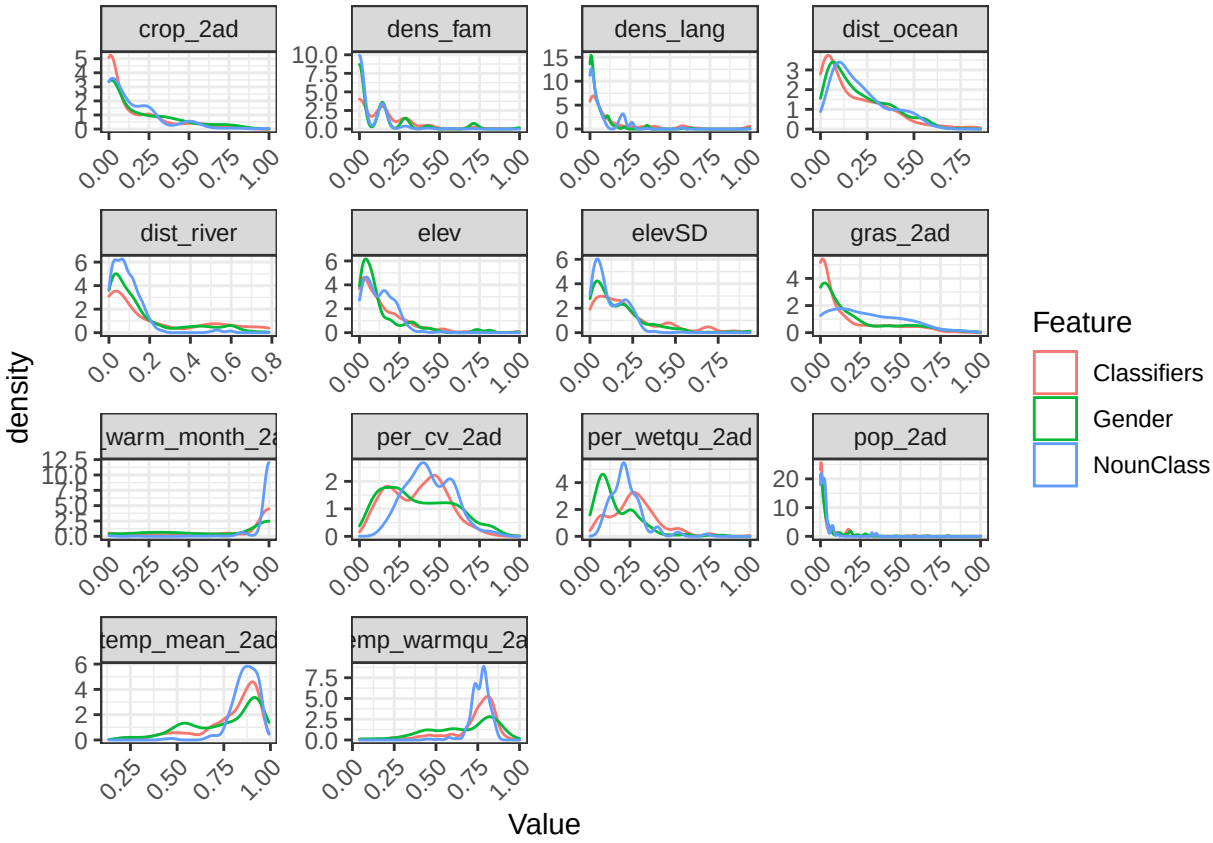
```
data.env %>%
  select(-c(Glottocode, Family_GLOT_Lv1, Genus, dist_ocean, dens_fam, dens_lang, pop_2ad)) %>%
  pivot_longer(names_to = "Factor", values_to = "Value", -c(Classifiers, Gender, NounClass, Area_GLOT))
  pivot_longer(names_to = "Feature", values_to = "Feature.Value", -c(Factor, Value, Area_GLOT)) %>%
  filter(Feature.Value == TRUE) %>%
  ggplot(aes(x =Factor, y = Value, fill = Feature)) +
  geom_jitter(alpha = 0.5, size = 0.7, color = "darkgray") +
  geom_boxplot(alpha = 0.5) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  facet_wrap(~Area_GLOT)
```



We can also visualize the variation of environmental factors according to their distribution.

```
data.env %>%
  select(-c(Glottocode:Area_GLOT)) %>%
  pivot_longer(names_to = "Factor", values_to = "Value", -c(Classifiers, Gender, NounClass)) %>%
  pivot_longer(names_to = "Feature", values_to = "Feature.Value", -c(Factor, Value)) %>%
  #filter(Feature != "NounClass") %>%
  filter(Feature.Value == TRUE) %>%
  ggplot(aes(x = Value, color = Feature)) +
  geom_density( alpha = 0.5) +
  facet_wrap(~Factor, scales = "free") +
  theme_bw() +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))#+
```




```
  #facet_wrap(~Feature)
```

We can also run a PCA (Principal Component Analysis) to visualize the interaction of the environemental factors in the data.

```
con_data<-as.matrix(data.env[,(which(colnames(data.env) == "dens_lang"):(which(colnames(data.env) == "p

# if you want to visualize the variance captured by the components
#pc <- princomp(con_data)
#plot(pc, type='l')
#summary(pc) # chose until components explains >85% variance
# if need to extract specific components
# pc <- prcomp(con_data)
#comps <- data.frame(pc$x[,1:2])

library(ggfortify)
autoplot(prcomp(con_data),
  data = data.env %>% mutate(System = case_when(NounClass == TRUE & Classifiers == TRUE & Gender == FALS
                             NounClass == TRUE & Classifiers == FALSE & Gender == FALSE ~ "NC",
                             NounClass == FALSE & Gender == TRUE & Classifiers == FALSE ~ "GEN",
                             NounClass == FALSE & Gender == FALSE & Classifiers == TRUE ~ "CLF",
                             NounClass == FALSE & Gender == TRUE & Classifiers == TRUE ~ "GEN_CLF",
                             NounClass == TRUE & Gender == TRUE & Classifiers == FALSE ~ "GEN_NC",
                             NounClass == TRUE & Gender == TRUE & Classifiers == TRUE ~ "GEN_NC_CLF",
                             NounClass == FALSE & Gender == FALSE & Classifiers == FALSE ~ "NONE")),
  colour = 'System',
```

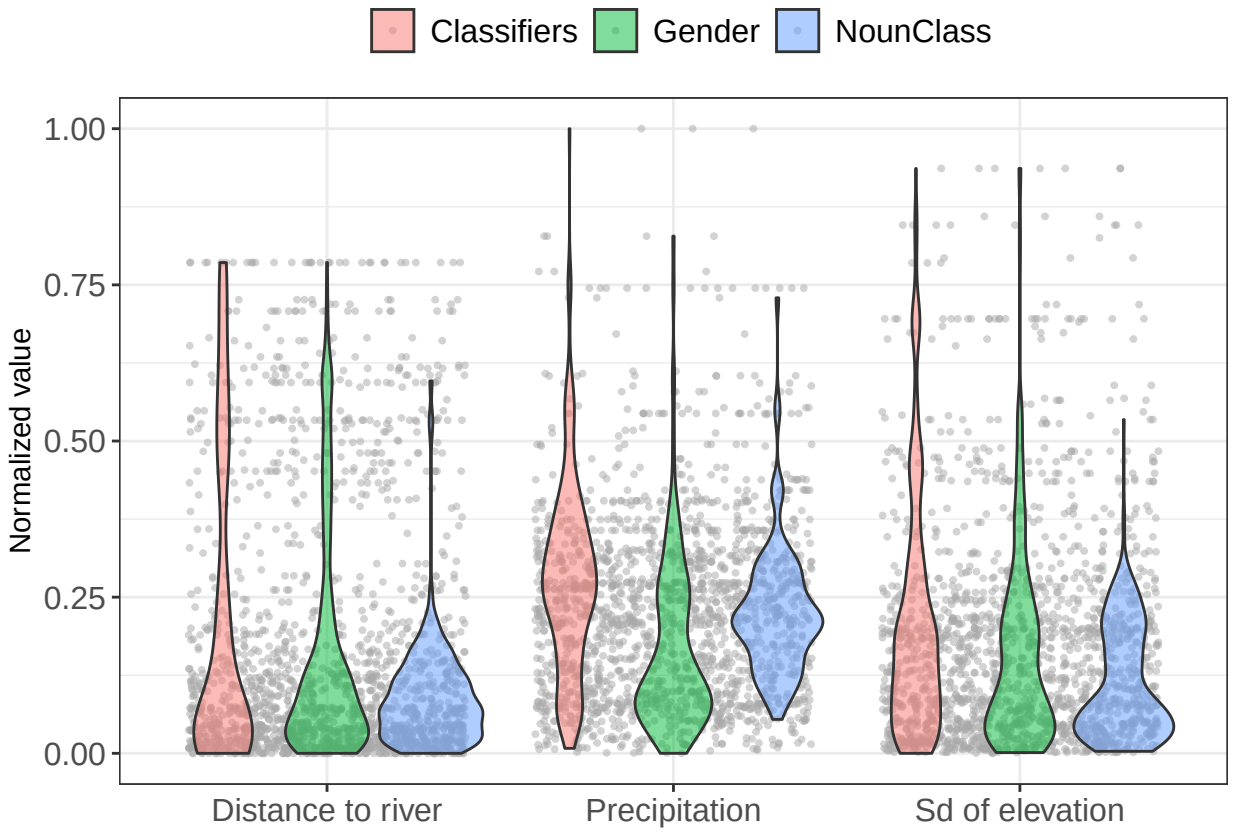
```
  #label = TRUE,
  #label.size = 3,
  #shape = FALSE,
  loadings = TRUE,
  loadings.colour = c('blue'),
  loadings.label = TRUE,
  loadings.label.colour = "blue",
  loadings.label.size = 5) +
  labs(title ="Visualization of principal component analysis") +
  theme_bw() +
  scale_color_manual(values=c("red","darkblue","orange","darkgreen","orange","lightgray"))
```

Comparing selected factors

We visualize the difference of distribution between classifier, gender, and noun class languages for the selected factors:

- dist_river = distance to the nearest large rive
- elevSD = standard deviation of elevation within a given search radius
- per_wetqu_2ad = precipitation in the wettest quarter

```
data.env %>%
  filter(Classifiers == TRUE) %>%
  select(Area_GLOT, dens_lang:pop_2ad) %>%
  mutate(Feature = "Classifiers") -> tmp.clf
data.env %>%
  filter(Gender == TRUE) %>%
  select(Area_GLOT, dens_lang:pop_2ad) %>%
  mutate(Feature = "Gender") -> tmp.gen
data.env %>%
  filter(NounClass == TRUE) %>%
  select(Area_GLOT, dens_lang:pop_2ad) %>%
  mutate(Feature = "NounClass") -> tmp.nc
tmp.env <- rbind(tmp.clf, tmp.gen, tmp.nc)
tmp.env %>%
  select(Feature, Area_GLOT, dist_river, per_wetqu_2ad, elevSD) %>%
  rename(`Distance to river` = dist_river, `Sd of elevation` = elevSD, `Precipitation` = per_wetqu_2ad)
  pivot_longer(names_to = "Factor", values_to = "Value", -c(Feature, Area_GLOT)) %>%
  ggplot(aes(x =Factor, y = Value, fill = Feature)) +
  geom_jitter(alpha = 0.5, size = 0.7, color = "darkgray") +
  geom_violin(alpha = 0.5) +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        legend.position = "top",
        legend.title = element_blank(),
        legend.text = element_text(size =12),
        axis.text = element_text(size = 12)) +
  ylab("Normalized value") #+
```

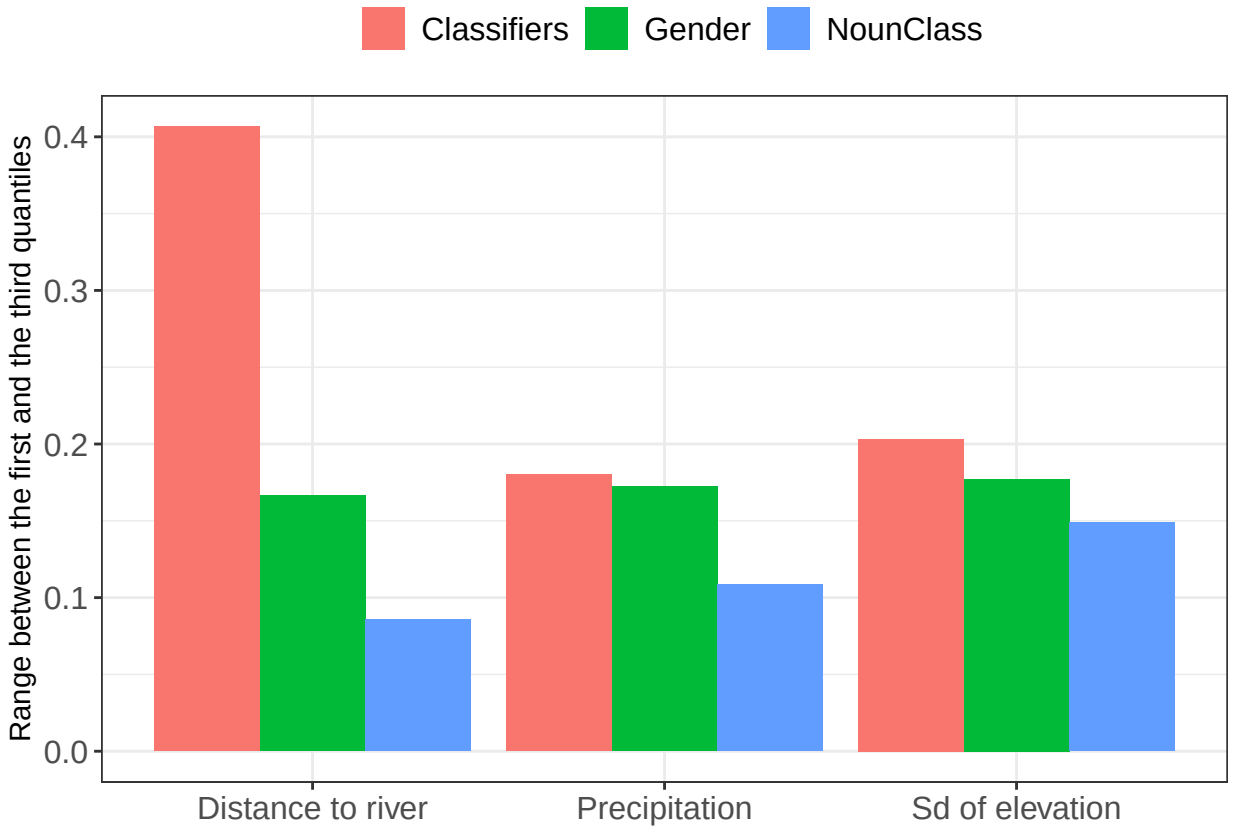



```
  #facet_wrap(~Area_GLOT, scales = "free")
ggsave("Figure_3.png", width = 6, height = 4, dpi = 300)
```

We compare the quantiles for each factor across languages with classifier, gender, and noun class.

```
output <- NULL
#factors <- colnames(tmp.env %>% select(dist_river:crop_2ad))
factors <- c("dist_river", "per_wetqu_2ad", "elevSD")
features <- c("Classifiers", "Gender", "NounClass")
for(i in 1:length(factors)){
  for(z in 1:length(features)){
  obj <- c(factors[i], features[z], summary(tmp.env %>%
                                              filter(Feature == features[z]) %>%
                                                pull(factors[i]))[c(2,5)])
  output <- rbind(output, obj)
  }}
output %>%
  as.data.frame() %>%
  rename(Factor = 1, Feature = 2, FirstQ = 3, ThirdQ = 4) %>%
  mutate(FirstQ = as.numeric(FirstQ), ThirdQ = as.numeric(ThirdQ),
         Range = ThirdQ - FirstQ,
         Factor = case_when(Factor == "dist_river" ~ "Distance to river",
                            Factor == "elevSD" ~ "Sd of elevation",
                            Factor == "per_wetqu_2ad" ~ "Precipitation")) %>%
  ggplot(aes(x = Factor, y = Range, fill = Feature)) +
  geom_bar(stat = "identity", position = "dodge") +
  ylab("Range between the first and the third quantiles") +
  theme_bw() +
```

```
  theme(axis.title.x = element_blank(),
        legend.position = "top",
        legend.title = element_blank(),
        legend.text = element_text(size =12),
        axis.text = element_text(size = 12))
```



We run levene tests (http://www.sthda.com/english/wiki/compare-multiple-sample-variances-in-r) to assess if the variance is similar between environmental factors across languages with classifier, gender, and noun class. First, we assess the factor of distance to river.

```
car::leveneTest(dist_river ~ Feature, data = tmp.env)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value              Pr(>F)
## group    2    68.5 <0.0000000000000002 ***
##       1762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then, we assess the factor of precipitation.

```
car::leveneTest(per_wetqu_2ad ~ Feature, data = tmp.env)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value        Pr(>F)
## group    2    21.4 0.00000000063 ***
##       1762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, we assess the factor of elevation.

```
car::leveneTest(elevSD ~ Feature, data = tmp.env)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value             Pr(>F)
## group    2    32.9 0.0000000000000094 ***
##       1762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# optionally fligner-killeen tests could also be considered
#fligner.test(dist_river ~ Feature, data = tmp.env)
```

The same analysis is performed with conover tests (https://cran.r-project.org/web/packages/conover.test/conover.test.pdf), to ensure the robustness of the analysis.

```
conover.test::conover.test(tmp.env$dist_river, tmp.env$Feature, kw=TRUE,
                           method="bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 10.5529, df = 2, p-value = 0.01
##
##
##                             Comparison of x by group
##                                   (Bonferroni)
## Col Mean-|
## Row Mean |   Classifi     Gender
## ---------+----------------------
##   Gender |   0.646688
##          |     0.7769
##          |
## NounClas |   3.215877   2.597113
##          |    0.0020*    0.0142*
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```

```
conover.test::conover.test(tmp.env$per_wetqu_2ad, tmp.env$Feature, kw=TRUE,
                           method="bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 264.4591, df = 2, p-value = 0
##
##
##                             Comparison of x by group
##                                   (Bonferroni)
## Col Mean-|
## Row Mean |   Classifi     Gender
## ---------+----------------------
##   Gender |   17.60989
##          |    0.0000*
##          |
```

```
## NounClas |   5.420899  -8.342979
##          |    0.0000*    0.0000*
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```

```
conover.test::conover.test(tmp.env$elevSD, tmp.env$Feature, kw=TRUE,
                           method="bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 63.9034, df = 2, p-value = 0
##
##
##                             Comparison of x by group
##                                   (Bonferroni)
## Col Mean-|
## Row Mean |   Classifi     Gender
## ---------+----------------------
##   Gender |   4.929913
##          |    0.0000*
##          |
## NounClas |   7.791757   3.702880
##          |    0.0000*    0.0003*
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```