Child Language Acquisition Diversity & Language Endangerment

Sam Passmore & Evan Kidd

This document details the data, code, figures, and decisions used in Passmore & Kidd (2024).

Packages necessary for this report:

```r
library(dplyr)
library(tidyr)
library(ggplot2)
library(ggpubr)
library(patchwork)
library(forecast)

# To install rcldf:
# devtools::install_github("SimonGreenhill/rcldf", dependencies = TRUE)
library(rcldf)

## Parameters
font_size = 12
```

Data Sources

There are four key sources of data used in this project:

- Grambank (Skirgärd, et al. 2023)
- Phoible (Moran & McCloy, 2019)
- Paper descriptions from the top four Child Language Acquisition Journals (Evan & Garcia, 2022)
- Agglomerated Endangerment Scale (AES) (Hammarström et al., 2018)

Grambank, Phoible, and Glottolog are included as submodules as part of the repository, linking directly to their GitHub repositories. We include the data from Evan & Garcia (2022) as a csv file, which is included in this repository, but also available at this repository: https://osf.io/jmxnw

Grambank

Grambank contains the grammatical structure of 2,467 languages, from 215 language families, across 195 features ranging from word order to verbal tense, to nominal plurals (Skirgärd et al., 2023).

For the analysis, we used the GitHub repository created for the Grambank release paper (https://github.com/grambank/grambank-analysed), which contains some convenience scripts that we can use in the repository to make our analyses easier. In the Grambank release paper (Skirgard et al. 2023), there is a series of data curation steps that are used to create a dataset ready for analysis. The key steps are:

1) Limiting data to one dialect per language, leaving 2,430 languages and keeping the dialect with the most complete data.

2) Converting the six non-binary variables to binary variables, bringing the total to 201 features.

3) Using random forests to impute the missing data, which totals around 24% of the dataset.

4) Reducing the dataset to its most complete state (1,509 languages, and 114 features).

These steps are performed within the system command in the text below. More details can be found in the Grambank release paper, or within the Github repository.

These steps create a datafile where the rows are languages and the columns are Grambank features, which we will use for the multi-dimensional scaling analysis later.

```r
# Create the file if it doesn't exist (takes a minute or so)
gb_dir = "submodule/grambank-analysed/R_grambank"
if(!file.exists(
  paste0(gb_dir, "/output/GB_wide/GB_wide_imputed_binarized.tsv")
  )){
  system(
  "cd ./submodule/grambank-analysed/R_grambank/;
  git submodule update --init
  mkdir -p output/non_GB_datasets
    mkdir -p output/coverage_plots
    mkdir -p output/GB_wide
    Rscript make_glottolog-cldf_table.R
    Rscript unusualness/processing/assigning_AUTOTYP_areas.R
    Rscript make_wide.R
    Rscript make_wide_binarized.R
  Rscript impute_missing_values.R | tee impute_missing_values.log")
}

grambank = read.table(
  paste0(gb_dir, "/output/GB_wide/GB_wide_imputed_binarized.tsv"),
  sep = "\t",
  header = 1
)
```

Phoible

PHOIBLE is a repository of cross-linguistic phonological inventory data, which have been extracted from source documents and tertiary databases and compiled into a single searchable convenience sample. It contains the cross-linguistic phonological inventory of 2,186 languages, from 176 language families, and a total of 3,183 segment types. For a detailed description of Phoible, see Moran (2012). The data can be explored at the website: https://phoible.org/

The code below shows how we wrangle to Phoible data into the appropriate format for the multidimensional scaling analysis. Some languages have multiple entries, with varying phonological inventories. We reduce the data to one inventory per language, choosing inventories at random.

```
# phoible data
p_df = read.csv("submodule/phoible_cldf/cldf/values.csv")

# Some languages have been coded multiple times (doculets)
# We want to select one inventory per language,
# but we make no judgement on which inventory is better.
# I.e. Choice of doculet is random.
p_ss =  p_df %>%
  dplyr::group_by(Language_ID) %>%
  dplyr::filter(Inventory_ID == sample(unique(Inventory_ID), 1))

# The number of language ids matches the number of Inventory IDs
# (i.e. there is one language code per inventory code)
all(n_distinct(p_ss$Language_ID) == n_distinct(p_ss$Inventory_ID))
```

```
[1] TRUE
```

```
# Make the dataset wide, this will build a dataset where columns are
# phonemes and rows are languages.
phoible_wide = pivot_wider(p_ss,
                           id_cols = Language_ID,
                           values_from = Value,
                           names_from = Value)

# This changes the data to a 0 (phoneme is not used) 1 (phoneme is used)
phoible_df = apply(phoible_wide[, 2:ncol(phoible_wide)], 2, function(x)
  ifelse(is.na(x), 0, 1))
phoible = data.frame(Language_ID = phoible_wide$Language_ID, phoible_df)
```

Glottolog & the Agglomerated Endangerment Scale (AES)

The AES scale is part of the Glottolog datatset. Here, we download that dataset, extract the AES value, and attach it to the language metadata file from Glottolog.

The AES scale is an agglomeration of three different scales of languages endangerment: UNESCO Atlas of the World's languages in Danger; Ethnologue's Expanded Graded Intergenerational Disruption Scale (EGIDS; 20th edition); and The Catalogue of Endangered Languages Language Endangerment Index (ElCat)

For details on the construction of each scale, and the construction of the agglomorative scale we use here, see details in Hammarström et al. (2018).

Endangerment ratings are preferred in the following order, starting with ELCat:

ELCat > UNESCO > E20 > Glottolog > Unknown

The levels of endangerment in the AES scale are described in the table below:

Table 1: Mappings between the endangerment categories in the source databases and the Agglomerated Endangerment Scale (AES). (A recreation of Table 7 in Hammarström et al. (2018))

UNESCO	LEI (ElCat)	EGIDS	AES
safe	at risk	1 (National) 2 (Regional) 3 (Trade) 4 (Educational) 5 (Written) 6a (Vigorous)	**Not endangered**
vulnerable	vulnerable	6b (Threatened)	**Threatened**

UNESCO	LEI (ElCat)	EGIDS	AES
definitely endangered	threatened endangered	7 (Shifting)	**Shifting**
severely endangered	severely endangered	8a (Moribund)	**Moribund**
critically endangered	critically endangered	8b (Nearly extinct)	**Nearly extinct**
extinct	dormant awakening	9 (Dormant)	**Extinct**
		9 (Reawakening)	
		9 (Second language only)	
		10 (Extinct)	

We create an additional variable determining whether a language is expected to have children speakers based on their endangerment level. With respect to the AES descriptions, we describe this as 'shifting' or greater. In all scales, 'shifting' is the level where the description requires that no children are learning the language and therefore, are languages inaccessible to child language acquisition.

```r
glottolog =
  cldf("https://github.com/glottolog/glottolog-cldf/archive/refs/tags/v4.8.zip")

languages = glottolog$tables$LanguageTable
values = glottolog$tables$ValueTable
aes = values %>% filter(Parameter_ID == "aes")

languages = inner_join(aes, languages, by = c("Language_ID" = "ID")) %>%
  select(ID,
         Language_ID,
         Name,
         Value,
         Code_ID,
         Comment,
         Source,
         Glottocode) %>%
  mutate(Value = as.numeric(Value),
         nochildren_strict = ifelse(Value >= 3, 1, 0))

# Check all languages only have one code
n_distinct(languages$Language_ID) == nrow(languages)
```

```
[1] TRUE
```

Kidd & Garcia (2022)

The supplementary material of Kidd & Garcia have a list of all papers published in the top four child language acquisition journals - including metadata for the language spoken - available here https://osf.io/jmxnw. We have extracted a list of unique languages studied, and the number of papers written about that language into a csv file below. The names used to describe languages have been manually curated to match with the Glottolog dataset. An example of a curated match is ensure languages like *Tongan* in Kidd & Garcia, matches with *Tongan (Tonga Island)* in Glottolog. We require exact matches.

Some matches could not be made, which are listed in the table below. These are two signed languages that could not be linked to a specific Glottocode, and Greenlandic and Light Warlpiri, which do not have AES endangerment codes so could not be matched.

```r
# Raw data from Kidd & Garcia 2022
kiddgarcia = read.csv('data/journal_archive_data_2021.csv', sep = ";")
# Matched data
kiddgarcia_matching = read.csv("data/name_matching.csv")

languages = left_join(languages, kiddgarcia_matching,
                      by = c("Name" = "name_matching"))

# Languages that have been studied have a paper count
languages$studied = ifelse(is.na(languages$count), 0, 1)

# Languages that have had a phonological or morphosyntax study about them
kiddgarcia_topics = kiddgarcia %>%
  group_by(language) %>%
  mutate(phonology_bin = ifelse(topic == "Phonology", 1, 0),
         morphosyntax_bin = ifelse(topic == "Morphosyntax", 1, 0)) %>%
  summarise(phonology_sum = sum(phonology_bin),
            phonology_studied = phonology_sum > 0,
            morphosyntax_sum = sum(morphosyntax_bin),
            morphosyntax_studied = morphosyntax_sum > 0
            ) %>%
  left_join(kiddgarcia_matching, by = "language")

languages = left_join(languages, kiddgarcia_topics,
                      by = c("Name" = "name_matching",
                             "count" = "count",
                             "language" = "language"))
```

```r
# Are all languages matched?
sum(!is.na(languages$count)) == nrow(kiddgarcia_matching)
```

```
[1] FALSE
```

```r
# find un-matched languages
nm = !kiddgarcia_matching$name_matching %in% languages$Name
kiddgarcia_matching[nm,]
```

```
                language count          name_matching
29           Greenlandic     1      Greenlandic Inuit
33             Home sign     2              Home sign
79 Signing Exact English     4  Signing Exact English
97      Warlpiri (Light)     2         Light Warlpiri
```

Analysis & Figures

This section contains graphs used in the main text, as well as alternative depictions of the data.

Histogram

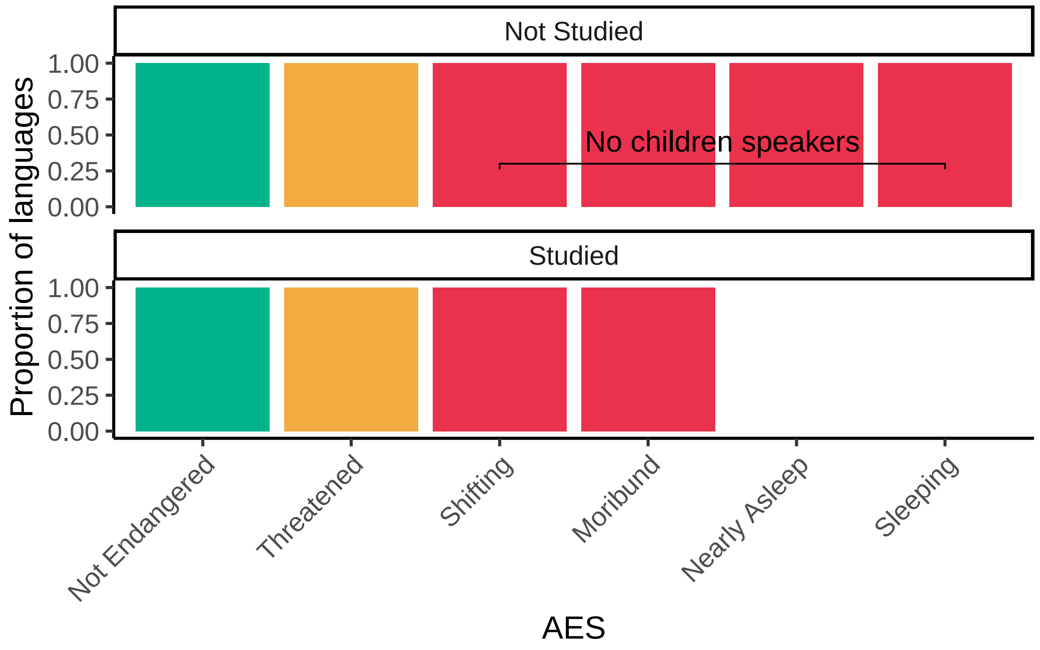
This histogram shows the proportion of languages studied by endangerment category, split by whether the languages have been studied or not, according to Kidd & Garcia (2022).

```r
plot_prop = languages %>%
  group_by(studied, Code_ID) %>%
  summarise(n = n(), .groups = 'keep') %>%
  mutate(freq = n / sum(n))

plot_prop$studied = ifelse(plot_prop$studied == 1, "Studied", "Not Studied")
plot_prop$AES = recode(plot_prop$Code_ID,
                       `aes-not_endangered` = "Not Endangered",
                       `aes-threatened` = "Threatened",
                       `aes-shifting` = "Shifting",
                       `aes-moribund` = "Moribund",
                       `aes-nearly_extinct` = "Nearly Asleep",
                       `aes-extinct` = "Sleeping"
```

```
                         )

plot_prop$AES = factor(plot_prop$AES,
                       levels = c("Not Endangered",
                                  "Threatened",
                                  "Shifting",
                                  "Moribund",
                                  "Nearly Asleep",
                                  "Sleeping"))

p_base = ggplot(plot_prop,
                aes(
                  y = freq,
                  x = AES,
                  fill = AES,
                  group = studied
                )) +
  geom_bar(stat = 'identity') +
  facet_wrap( ~ studied, nrow = 3) +
  ylab("Proportion of languages") +
  xlab("AES") +
  scale_fill_manual(values = c("#00b38a", "#f2ac42", rep("#ea324c", 4))) +
  theme_classic(base_size = font_size) +
  theme(legend.position = "none")

pp = p_base + geom_bracket(
  xmin = "Shifting",
  xmax = "Sleeping",
  y.position = 0.3,
  tip.length = 0.05,
  vjust = -0.5,
  label = "No children speakers",
  data = data.frame(studied = factor(
    "Not Studied", levels = c("Studied", "Not Studied")
  ),
  AES = "Sleeping")
)  +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggsave(plot = pp, filename = "figures/histogram.png", height = 5)
```

```
Saving 5.5 x 5 in image
```

```
pp
```



Hexbin graphs

To build the Hexbin projections of linguistic diversity, there are four key steps:

1) Match our dataset of studied languages (Kidd & Garcia 2022), to the Phoible and Grambank datasets

2) Build a distance matrix between all languages

3) Run the multidimensional scaling algorithm

4) Plot and bin the results.

To plot the results, we are required to get custom ggplot code, which has been included in this repository. The custom code was created thanks to Allan Cameron via Stackoverflow.

We do this for the Grambank and Phoible dataset, and for a variety of settings. First, we plot the raw data from the MDS algorithm. Second we plot the data in bins, with no markings. The next plots highlight bins containing studied languages, first on a raw scale, then on a log scale. We also combine these to create hexbin plot found in the main text. Finally, we create the same plots comparing the spaces studied to unstudied.

At the end of this code block, we show a table containing the languages codes which are dropped from the datasets. Unmatched languages largely occur because Grambank or Phoible identifies a dialect, and Kidd & Garcia contain metadata on languages, or because no data exists in the datasets. In both datasets, dropped languages make up less than 1% of the total sample.

```r
# Read in custom Hextri code
source("hextri_grobs.R")

## Build the pipeline as a function
hextri_datacleaning = function(df, languages = languages){
  # Give data rownames to carry through the analysis
  rownames(df) = df$Language_ID

  # Make distance matrix
  distance_matrix = df %>%
    dplyr::select(-Language_ID) %>%
    cluster::daisy(metric = "gower", warnBin = FALSE)

  # Build Multi-dimensional scaling output
  mds_output = distance_matrix %>%
    cmdscale(eig=TRUE, k=2)

  # Wrangle into a dataframe
  mds_points = data.frame(mds_output$points)
  colnames(mds_points) = c("MDS.X", "MDS.Y")
  mds_points$Glottocode = rownames(mds_points)

  # Join Endangerment and Studied data
  plot_df = left_join(mds_points, languages, by = "Glottocode")
  plot_df = plot_df[, c(
    "Name",
    "Glottocode",
    "MDS.X",
    "MDS.Y",
    "count",
    "phonology_sum",
    "morphosyntax_sum",
    "nochildren_strict",
    "Code_ID",
    "Value"
  )]
  # If there is a missing value,
  # it means no papers were published in the journals reviewed
  plot_df$count[is.na(plot_df$count)] = 0
  plot_df$phonology_sum[is.na(plot_df$phonology_sum)] = 0
  plot_df$morphosyntax_sum[is.na(plot_df$morphosyntax_sum)] = 0

  # Ensure we don't have any missing data otherwise GGplot will complain
  plot_df = plot_df[complete.cases(plot_df),]

  # If a language has at leaast one paper, then it is a studied langauge
  plot_df$studied = ifelse(plot_df$count > 1, 1, 0)

  # Reverse code the binary variable so that a value of 1 indicates that a
  # language has children speakers
  plot_df$children = 1 - plot_df$nochildren

  # Return plot dataframe
  plot_df
}

# Run pipeline for each dataset
plot_phoible = hextri_datacleaning(phoible, languages = languages)
plot_grambank = hextri_datacleaning(grambank, languages = languages)

# unmatched data
phoible$Language_ID[!phoible$Language_ID %in% plot_phoible$Glottocode]
```

```
[1] "kali1299" "norw1259" "karo1306" "gund1246"
```

```r
grambank$Language_ID[!grambank$Language_ID %in% plot_grambank$Glottocode]
```

```
[1] "amam1246" "bala1316" "east2782" "khan1277" "situ1238" "sout3261" "zamb1245"
[8] "zani1235"
```

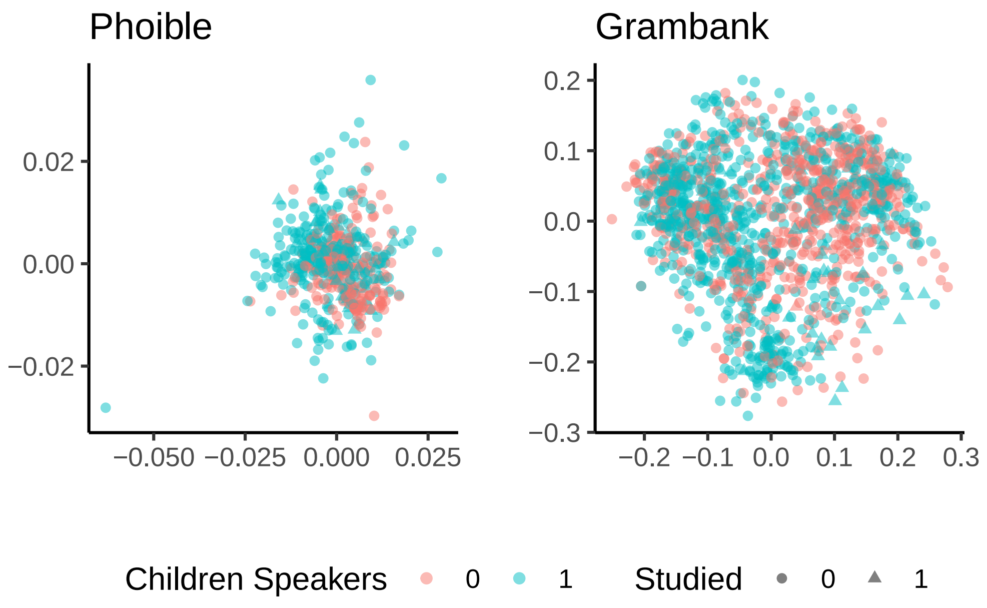
The dialect names for the language codes above are:

Phoible: Norwegian Bokmål; Karo; Kundjeyhmi

Grambank: Amam; Balade; Eastern Kikongo; Southern Tangkhul Naga; Situ; Southwestern Dargwa; Zamba; Zaniza Zapotec.

No-bin Plots

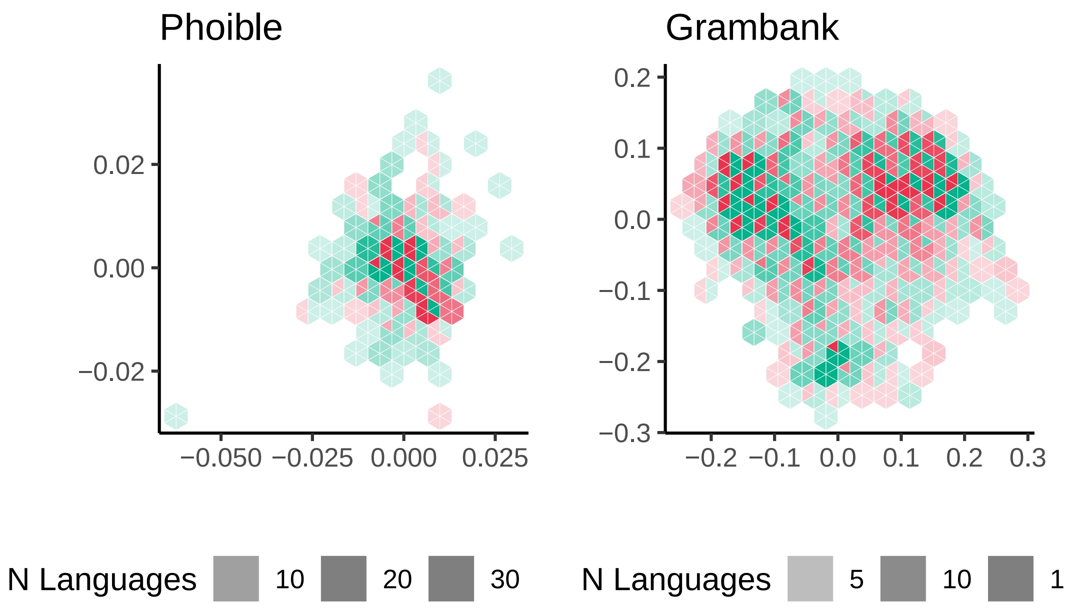
```
p1_phoible = ggplot(plot_phoible,
                    aes(MDS.X, MDS.Y, col = factor(children))) +
  geom_point(aes(shape = factor(studied)), alpha = 0.5) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(col=guide_legend(title="Children Speakers"),
         shape=guide_legend(title="Studied")) +
  ggtitle("Phoible") +
  xlab("") + ylab("")

p1_grambank = ggplot(plot_grambank,
                     aes(MDS.X, MDS.Y, col = factor(children))) +
  geom_point(aes(shape = factor(studied)), alpha = 0.5) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(col=guide_legend(title="Children Speakers"),
         shape=guide_legend(title="Studied")) +
  ggtitle("Grambank") +
  xlab("") + ylab("")

(p1_phoible | p1_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Hexbin Plots

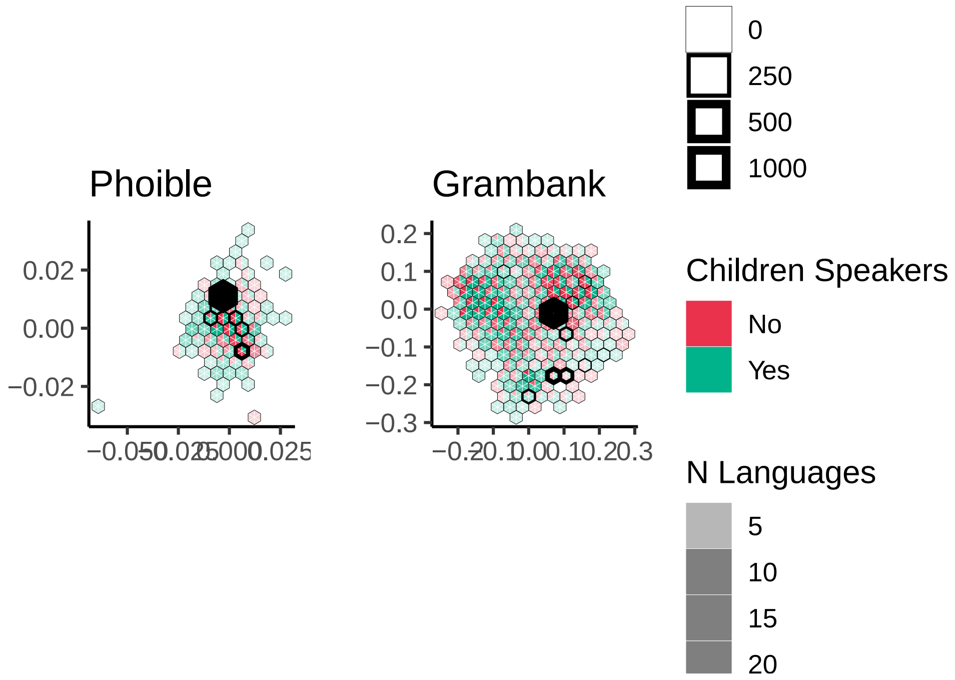
```
n_bins = 14

p2_phoible = ggplot(plot_phoible,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = n_bins) +
  geom_hex(fill = NA, color = "white", bins = n_bins, lwd = 0.01) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Phoible") +
  xlab("") + ylab("")
```

```
p2_grambank = ggplot(plot_grambank,
                     aes(MDS.X, MDS.Y, fill = factor(children),
                         col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.01, bins = n_bins) +
  geom_hex(fill = NA, color = "white", bins = n_bins, lwd = 0.01) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Grambank") +
  xlab("") + ylab("")

(p2_phoible | p2_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Hexbin & Studied Languages Plots

```
p3_phoible = ggplot(plot_phoible,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.05) +
  stat_summary_hex(aes(
    z = phonology_sum,
    linewidth = after_stat(value),
  ), fun = ~ sum((.x[.x > 0])), col = "black", fill = NA, bins = 15) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  # guides(fill=guide_legend(title="Children Speakers"),
  #        linewidth=guide_legend(title="N Papers"),
  #        alpha=guide_legend(title="N Languages")) +
  ggtitle("Phoible") +
  theme(legend.position = "none") +
  xlab("") + ylab("")

p3_grambank = ggplot(plot_grambank,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.05) +
  stat_summary_hex(aes(
    z = morphosyntax_sum,
    linewidth = after_stat(value),
  ), fun = ~ sum((.x[.x > 0])), col = "black", fill = NA, bins = 15) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3),
                  breaks = c(0, 250, 500, 1000)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Grambank") +
  xlab("") + ylab("")

p3 = (p3_phoible | p3_grambank)

p3
```

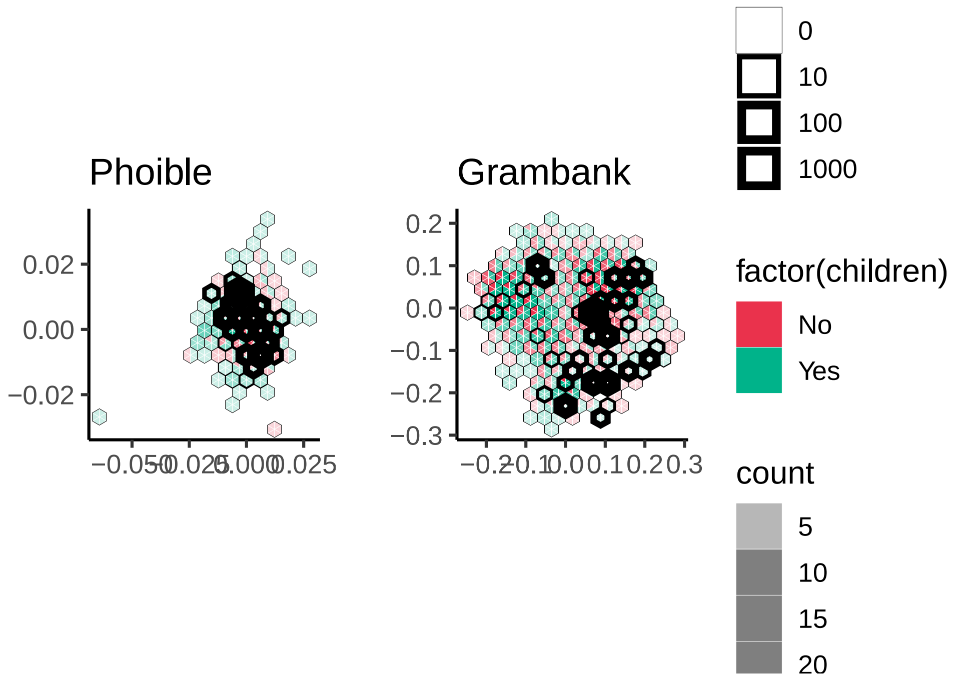


Next we extract the count data for each cell

Hexbin & Studied Languages (Log Scale) Plots

```
p4_phoible = ggplot(plot_phoible,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.05) +
  stat_summary_hex(aes(
    z = count,
    linewidth = after_stat(value),
  ), fun = ~ sum((.x[.x > 0])), col = "black", fill = NA, bins = 15) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3), trans = scales::pseudo_log_trans(base = 10)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  # guides(fill=guide_legend(title="Children Speakers"),
  #        linewidth=guide_legend(title="N Papers"),
  #        alpha=guide_legend(title="N Languages")) +
  theme(legend.position = "none") +
  ggtitle("Phoible") +
  xlab("") + ylab("")

p4_grambank = ggplot(plot_grambank,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.05) +
  stat_summary_hex(aes(
    z = count,
    linewidth = after_stat(value),
  ), fun = ~ sum(.x[.x > 0]), col = "black", fill = NA, bins = 15) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3),
                  trans = scales::pseudo_log_trans(base = 10),
                  breaks = c(0, 10, 100, 1000)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(values = c("#ea324c", "#00b38a"),
                    labels = c("No", "Yes")) +
  guides(linewidth=guide_legend(title="Log N Papers")) +
  ggtitle("Grambank") +
  xlab("") + ylab("")

p4 = (p4_phoible + p4_grambank)
```

```
p4
```



```
ggsave(plot = p4, filename = "figures/hexbinplots_logscale.png",
       height = 297 / 1.95,
       width = 210,
       units = "mm")

fig1 = p3 / p4 + plot_layout(guides = "collect") &
  theme(legend.direction = "vertical",
        legend.box = "vertical") &
  guides(fill = guide_legend(order = 1, title="Children Speakers"),
         alpha = guide_legend(order = 2, title="N Languages"),
         )

ggsave(plot = fig1, filename = "figures/hexbinplots_stacked.png",
       height = 297 / 1.95,
       width = 210,
       units = "mm")
```

Summary Statistics

Here, we extract the summary information from the plots, for discussion in the text.

```
### Count data ###
papercount_pb = layer_data(p3_phoible, 3)
papercount_gb = layer_data(p3_grambank, 3)

# count per cell
tail(sort(papercount_pb$value), 10)
```

```
[1]  15  28  29  36  59  61  63  65 104 446
```

```
tail(sort(papercount_gb$value), 10)
```

```
[1]   62   63   64   70   77  138  143  226  284 1010
```

```
# how many studied cells have less than 10 papers
pb.10 = sum(papercount_pb$value <= 10 & papercount_pb$value != 0)
gb.10 = sum(papercount_gb$value <= 10 & papercount_gb$value != 0)

pb.10p = pb.10 / sum(papercount_pb$value != 0)
gb.10p = gb.10 / sum(papercount_gb$value != 0)

cat("There are", pb.10, "cells in Phoible that have less than 10 papers (",
    scales::percent(pb.10p), ").",
    "and there are", gb.10,
    "cells in Grambank (", scales::percent(gb.10p), ")")
```

```
There are 11 cells in Phoible that have less than 10 papers ( 50% ). and there are 24 cells
```

```
# Proportion per cell
tail(sort(round(papercount_pb$value / sum(papercount_pb$value), 3)), 10)
```

```
[1] 0.016 0.029 0.030 0.038 0.062 0.064 0.066 0.068 0.109 0.469
```

```
tail(sort(round(papercount_gb$value / sum(papercount_gb$value), 3)), 10)
```

```
[1] 0.024 0.025 0.025 0.028 0.030 0.054 0.056 0.089 0.112 0.397
```

```r
# extract data from ggplot
pb_pdata.1 = layer_data(p3_phoible, 1)
gb_pdata.1 = layer_data(p4_grambank, 1)

pb_pdata.2 = layer_data(p3_phoible, 2)
gb_pdata.2 = layer_data(p4_grambank, 2)

pb_pdata.3 = layer_data(p3_phoible, 3)
gb_pdata.3 = layer_data(p4_grambank, 3)

# Hexbins counts
# number of bins
n_bins.pb = nrow(pb_pdata.2)
n_bins.gb = nrow(gb_pdata.2)

# number of studied bins
s_bins.pb = sum(pb_pdata.3$value > 0)
s_bins.gb = sum(gb_pdata.3$value > 0)

s_bins.pbp = s_bins.pb / nrow(pb_pdata.2)
s_bins.gbp = s_bins.gb / nrow(gb_pdata.2)

cat("There are", n_bins.pb, "in the Phoible plot.",
    s_bins.pb, "have been studied (", scales::percent(s_bins.pbp), ").")
```

```
There are 71 in the Phoible plot. 22 have been studied ( 31% ).
```

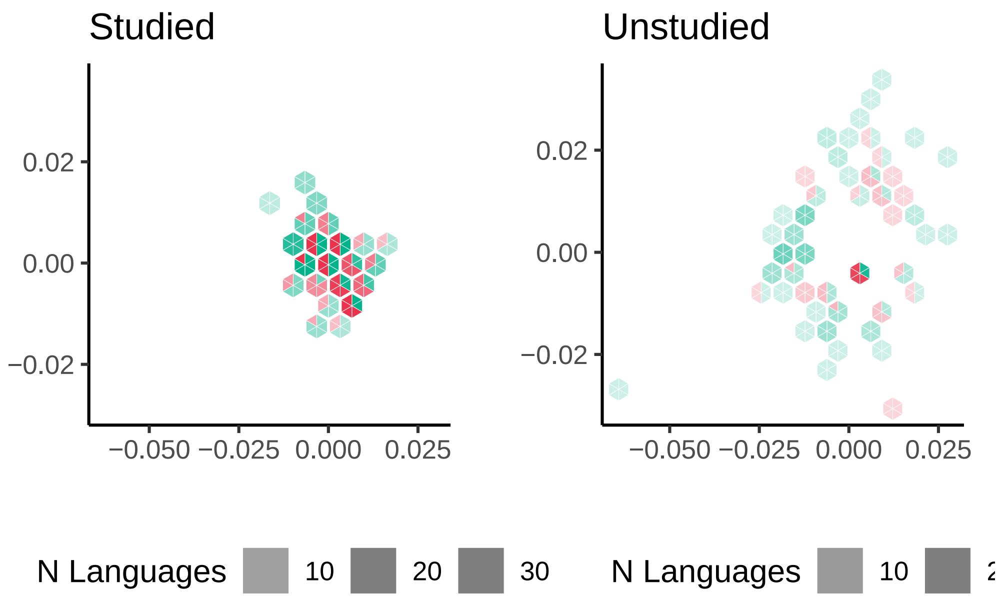
```r
cat("There are", n_bins.gb, "in the Grambank plot.",
    s_bins.gb, "have been studied (", scales::percent(s_bins.gbp), ").")
```

```
There are 185 in the Grambank plot. 52 have been studied ( 28% ).
```

Hexbin Plots: Studied vs Unstudied

Phoible

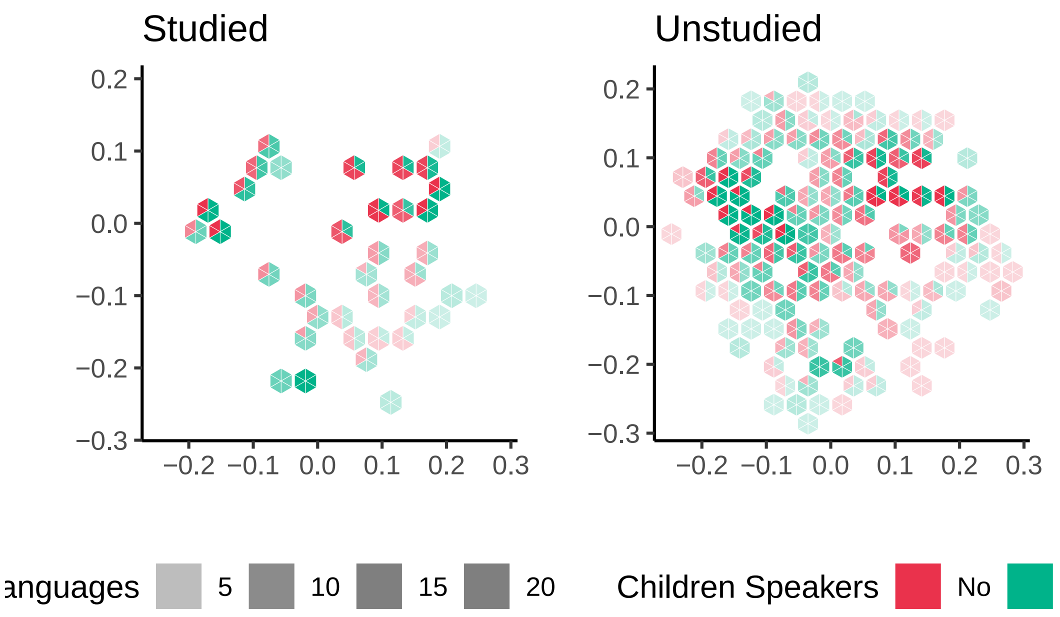
```r
# Only show studied diversity
p5.1_phoible = ggplot(plot_phoible,
                      aes(MDS.X, MDS.Y, fill = factor(children),
                          col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = n_bins) +
  stat_summary_hex(aes(
    z = count,
    fill = factor(after_stat(value) > 0)
  ), fun = ~ sum(log(.x[.x > 0])), col = NA, bins = n_bins) +
  geom_hex(fill = NA, color = "white", bins = n_bins, lwd = 0.5) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(
    values = c("#ea324c", "#00b38a", "white", "transparent"),
    labels = c("No", "Yes", "", "")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Studied") +
  xlab("") + ylab("")


## Only show studied undiversity
p5.2_phoible = ggplot(plot_phoible,
                      aes(MDS.X, MDS.Y, fill = factor(children),
                          col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  stat_summary_hex(aes(
    z = count,
    fill = factor(after_stat(value) > 0)
  ), fun = ~ sum(log(.x[.x > 0])), col = NA, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.5) +
  # # geom_point(aes(shape = factor(studied))) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(
    values = c("#ea324c", "#00b38a", "transparent", "white"),
    labels = c("No", "Yes", "", "")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Unstudied") +
  xlab("") + ylab("")

(p5.1_phoible + p5.2_phoible) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Grambank

```
# Only show studied diversity
p5.1_grambank = ggplot(plot_grambank,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = n_bins) +
  stat_summary_hex(aes(
    z = count,
    fill = factor(after_stat(value) > 0)
  ), fun = ~ sum(log(.x[.x > 0])), col = NA, bins = n_bins) +
  geom_hex(fill = NA, color = "white", bins = n_bins, lwd = 0.5) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(
    values = c("#ea324c", "#00b38a", "white", "transparent"),
    labels = c("No", "Yes", "", "")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Studied") +
  xlab("") + ylab("")


## Only show studied undiversity
p5.2_grambank = ggplot(plot_grambank,
                       aes(MDS.X, MDS.Y, fill = factor(children),
                           col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  stat_summary_hex(aes(
    z = count,
    fill = factor(after_stat(value) > 0)
  ), fun = ~ sum(log(.x[.x > 0])), col = NA, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.5) +
  # # geom_point(aes(shape = factor(studied))) +
  coord_equal() +
  theme_classic(base_size = font_size) +
  theme(aspect.ratio = 1, legend.position = "right") +
  scale_linewidth(range = c(0, 3)) +
  scale_alpha(range = c(0.2, 2)) +
  scale_fill_manual(
    values = c("#ea324c", "#00b38a", "transparent", "white"),
    labels = c("No", "Yes", "", "")) +
  guides(fill=guide_legend(title="Children Speakers"),
         linewidth=guide_legend(title="Log N Papers"),
         alpha=guide_legend(title="N Languages")) +
  ggtitle("Unstudied") +
  xlab("") + ylab("")
```

```
(p5.1_grambank + p5.2_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Language Endangerment & ARIMA models

First, we show how many languages are shifting between different EGIDS endangerment categories, based on the supplementary material from Bromham et al. (2022).

```
bromham_probabilities = readxl::read_xlsx("data/Bromham_suppdata.xlsx",
                                          sheet = 4,
                                          skip = 2)
colnames(bromham_probabilities)[1] = "ISO"

bromham_2060 = bromham_probabilities %>%
  select(ISO, contains("40"))

bromham_2060$EGIDS.value =
  apply(bromham_2060[, 2:ncol(bromham_2060)], 1, function(x)
    names(x)[x == max(x)])
bromham_2060$EGIDS.value = case_match(
  bromham_2060$EGIDS.value,
  "P[Y=1].40" ~ 1,
  "P[Y=2].40" ~ 2,
  "P[Y=5].40" ~ 5,
  "P[Y=7].40" ~ 7
)

# Current EGIDS score vs Predicted EGIDS score in 40 years time
bromham_nowvs40y = data.frame(
  now = bromham_probabilities$`1=1-6a, 2=6b, 3=7, 4=8a, 5=8b, 6=9, 7=10`,
  future = bromham_2060$EGIDS.value
)

# 3 represents EGIDS level 7 which is described as:
# The child-bearing generation knows the language well enough
# to use it among themselves but none are transmitting it to their children.
# I.e. no children are speaking the language
table(bromham_nowvs40y$now)
```

```

   1    2    3    4    5    6    7 
4058 1016  433  248  356  116  190 
```

```
table(bromham_nowvs40y$future)
```

```

   1    2    5    7 
4970  392  142  913 
```

```
table(bromham_nowvs40y$now >= 3)
```

```

FALSE  TRUE 
 5074  1343 
```

```
table(bromham_nowvs40y$future >= 3)
```

```
FALSE  TRUE
 5362  1055
```

```
sum(bromham_nowvs40y$now > bromham_nowvs40y$future)
```

```
[1] 1163
```

```
sum(bromham_nowvs40y$now < bromham_nowvs40y$future)
```

```
[1] 958
```

```
bromham_summary = bromham_2060 %>%
  group_by(EGIDS.value) %>%
  summarise(n = n()) %>%
  mutate(freq = n / sum(n))

bromham_summary = data.frame(studied = "2060",
                             EGIDS = bromham_summary$EGIDS.value,
                             n = bromham_summary$n,
                             freq = bromham_summary$freq)
```

To show how many new languages are being studied, and how that might be projected into the future, we use a simple ARIMA model of the number of new langauges studied each year, predicted by the number of papers published per year. We calculate how many more papers are being published per year.

```
# What languages are studied in what year
years = split(kiddgarcia$language, f = kiddgarcia$year)

# Determine which years new languages were studied
by_year = data.frame(year = names(years))
by_year$total_languages = NA

# How many languages were studied in year 1
# which languages
studied_languages = unique(years[[1]])
# number of unique languages
by_year$total_languages[1] = n_distinct(studied_languages)
```

```r
for(i in 2:length(years)){
  # what languages were studied this year
  new_year = unique(years[[i]])

  # How many of those are new languages
  new_total = by_year$total_languages[i - 1] +
    sum(!new_year %in% studied_languages)
  by_year$total_languages[i] = new_total

  # add the new languages to the studied languages pile
  new_languages = new_year[!new_year %in% studied_languages]
  studied_languages = c(studied_languages, new_languages)
}

# Calculate the number of new languages studied within each year
by_year$new_langs = c(7, diff(by_year$total_languages))

# calculate the number of papers studied in each year
papers_byyear = kiddgarcia %>%
  distinct(title, issue, volume, year) %>%
  group_by(year) %>%
  summarise(n_articles = n()) %>%
  mutate(year = as.character(year))

by_year = left_join(by_year, papers_byyear, by = "year")

# check final value is right
by_year$total_languages[nrow(by_year)] == n_distinct(kiddgarcia$language)
```

```
[1] TRUE
```

```r
## Paper summary over time
# Papers per year between 1980 and 2000
by_year %>% filter(year >= 1980 &
                     year <= 2000) %>%
  pull(n_articles) %>%
  summary()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  40.00   47.00   51.00   51.29   55.00   65.00 
```

```r
# Papers per year between 2001 and 2020
by_year %>% filter(year >= 2001 &
                     year <= 2021) %>%
  pull(n_articles) %>%
  summary()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   52.0    76.5    90.0    93.8   112.0   138.0 
```

```r
summary(by_year$n_articles - lag(by_year$n_articles))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
-31.000  -3.000   1.500   1.804   7.750  26.000       1 
```

```r
#### Model ####
ts_data = ts(by_year, start = c(1974, 1), frequency = 1)

ARIMAfit = forecast::tslm(new_langs ~ n_articles, data = ts_data)
summary(ARIMAfit)
```

```

Call:
forecast::tslm(formula = new_langs ~ n_articles, data = ts_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.2096 -1.1463 -0.1181  0.7763  5.1352 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept) 1.745211   0.607632   2.872   0.0062 **
n_articles  0.007037   0.008299   0.848   0.4010   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.749 on 45 degrees of freedom
Multiple R-squared:  0.01572,	Adjusted R-squared:  -0.00615 
F-statistic: 0.7188 on 1 and 45 DF,  p-value: 0.401
```

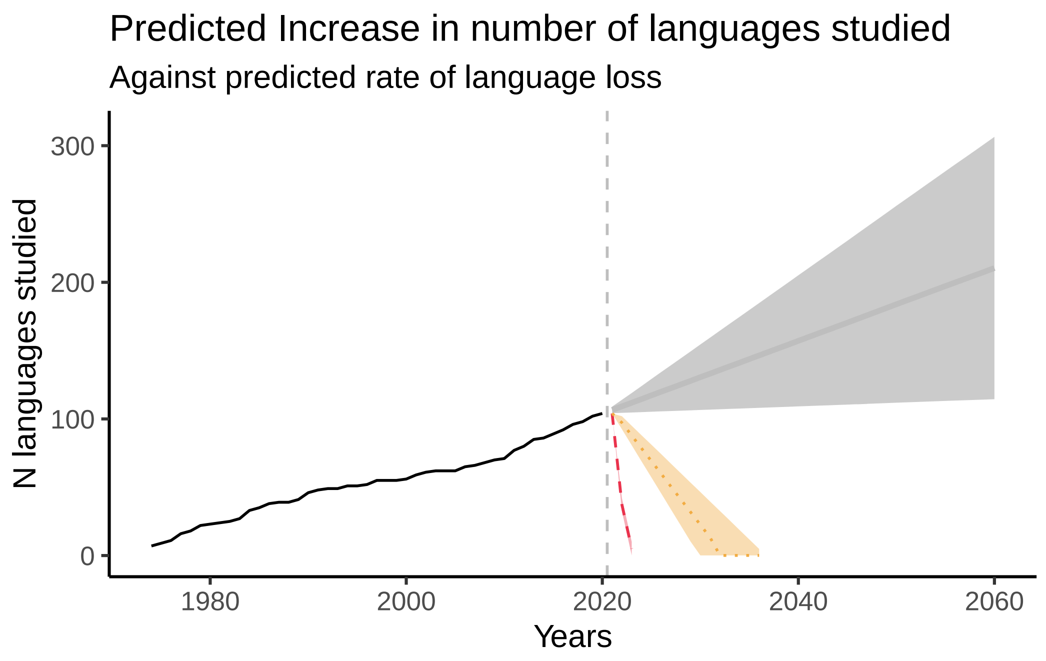
The ARIMA model shows that there is no significant relationship between the increase in the number of papers being published and the number of new languages being studied.

Plot ARIMA

```r
future_forcasts =
  forecast(ARIMAfit,
           newdata = data.frame(
             year = 2021:2060,
             n_articles = rnorm(40, mean = 130, sd = 5)
           ))

plot_arima = data.frame(
  year = as.numeric(by_year$year),
  newlanguages_year = by_year$new_langs,
  newpapers_year = by_year$n_articles
)

plot_arima$cumsum_languages = cumsum(plot_arima$newlanguages_year)

plot_arima_pred = data.frame(
  year = c(future_forcasts$newdata$year),
  newlanguages_year = c(future_forcasts$mean),
  newlanguages_yearL = c(future_forcasts$lower[,1]),
  newlanguages_yearU = c(future_forcasts$upper[,1])
)

base_2020 = tail(plot_arima$cumsum_languages, 1)

plot_arima_pred$cumsum_languages =
  cumsum(plot_arima_pred$newlanguages_year) + base_2020
plot_arima_pred$cumsum_languagesL =
  cumsum(plot_arima_pred$newlanguages_yearL) + base_2020
plot_arima_pred$cumsum_languagesU =
  cumsum(plot_arima_pred$newlanguages_yearU) + base_2020

## Accounting for expected langauges loss
# 72 languages a year shifting to no children speakers
cumulativeloss_children = cumsum(rep(72, nrow(plot_arima_pred) - 1))
starting_point = tail(plot_arima$cumsum_languages, 1)
plot_arima_pred$cumsum_languages_nochildren =
  c(starting_point,
    plot_arima_pred$cumsum_languages[2:nrow(plot_arima_pred)] -
      cumulativeloss_children)
```

```r
plot_arima_pred$cumsum_languages_nochildrenU =
  c(starting_point,
    plot_arima_pred$cumsum_languagesU[2:nrow(plot_arima_pred)] -
      cumulativeloss_children)

plot_arima_pred$cumsum_languages_nochildrenL =
  c(starting_point,
    plot_arima_pred$cumsum_languagesL[2:nrow(plot_arima_pred)] -
      cumulativeloss_children)

# ggplot wont draw any error lines if they fall outside the plotting area
# So we alter the last values to the edge of the plotting zone
plot_arima_pred$cumsum_languages_nochildren[3] = 5
plot_arima_pred$cumsum_languages_nochildrenL[3] = 0.1
plot_arima_pred$cumsum_languages_nochildrenU[3] = 10

# 12 languages a year becoming endangered
cumulativeloss_endangered = cumsum(rep(12, nrow(plot_arima_pred) - 1))

plot_arima_pred$cumsum_languages_endangered =
  c(starting_point,
    plot_arima_pred$cumsum_languages[2:nrow(plot_arima_pred)] -
      cumulativeloss_endangered)

plot_arima_pred$cumsum_languages_endangeredU = c(starting_point,
    plot_arima_pred$cumsum_languagesU[2:nrow(plot_arima_pred)] -
      cumulativeloss_endangered)

plot_arima_pred$cumsum_languages_endangeredL =
  c(starting_point,
    plot_arima_pred$cumsum_languagesL[2:nrow(plot_arima_pred)] -
      cumulativeloss_endangered)

# ggplot wont draw any error lines if they fall outside the plotting area
# So we alter the last values to the edge of the plotting zone

plot_arima_pred$cumsum_languages_endangeredL[10:16] = 0.1
plot_arima_pred$cumsum_languages_endangered[12:16] = 0.1

# No NAs to avoid GGplot messages
plot_arima_pred[plot_arima_pred < 0] = NA
```

```r
pp.1 =
  ggplot(data = plot_arima,
         aes(x = year, y = cumsum_languages, group = 1)) +
  geom_line() +
  # Predicted increase
  geom_smooth(data = plot_arima_pred,
              aes(y = cumsum_languages, x = year,
                  ymin = cumsum_languagesL, ymax = cumsum_languagesU),
              col = "grey", stat = "identity", alpha = 0.5) +
  theme_classic(base_size = font_size) +
  # Children spoken languages losees
  geom_line(data = plot_arima_pred,
            aes(x = year, y = cumsum_languages_nochildren),
            col = "#ea324c", lty = "dashed") +
  geom_smooth(data = plot_arima_pred,
              aes(x = year, y = cumsum_languages_nochildren,
                  ymin = cumsum_languages_nochildrenL,
                  ymax = cumsum_languages_nochildrenU),
              fill = "#ea324c", stat = "identity", linetype=0) +
  # Spoken Language plot_arima_pred
  geom_line(data = plot_arima_pred,
            aes(x = year, y = cumsum_languages_endangered),
            col = "#f2ac42", lty = "dotted") +
  geom_smooth(data = plot_arima_pred,
              aes(x = year, y = cumsum_languages_endangered,
                  ymin = cumsum_languages_endangeredL,
                  ymax =cumsum_languages_endangeredU),
              fill = "#f2ac42", stat = "identity", linetype=0) +
  xlab("Years") +
  ylab("N languages studied") +
  geom_vline(aes(xintercept = 2020.5), col = "grey", lty = "dashed") +
  ggtitle("Predicted Increase in number of languages studied",
          subtitle = "Against predicted rate of language loss") +
  ylim(c(0, 310))

pp.1
```



Plot ARIMA & N Papers published

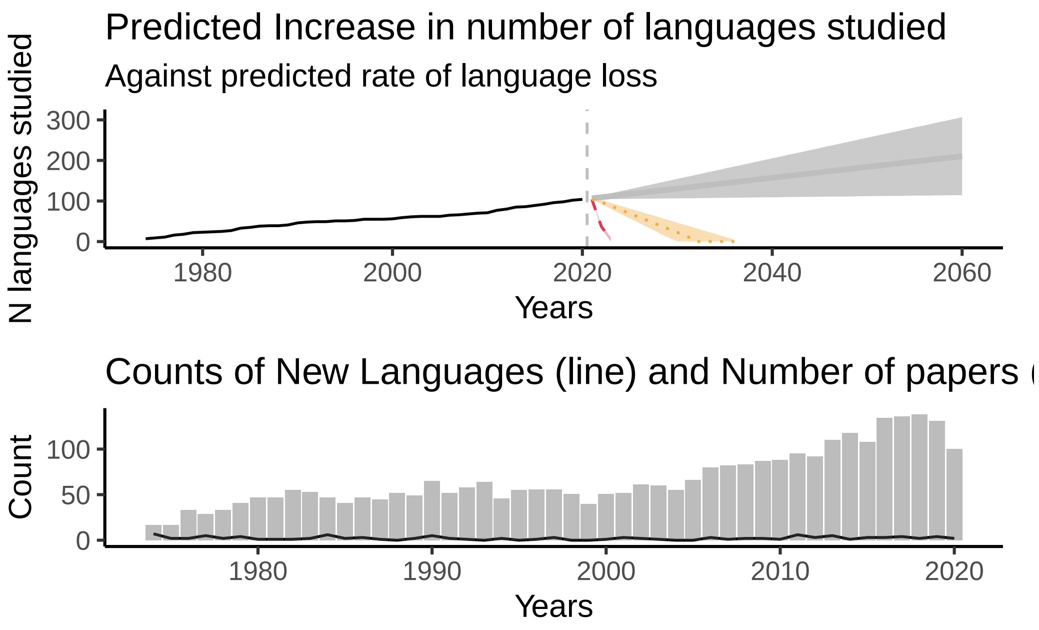
```
plot_arima$cumsum_papers = cumsum(plot_arima$newpapers_year)

pp.2 =
  ggplot(data = plot_arima,
         aes(x = year, y = newlanguages_year, group = 1)) +
  geom_line() +
  # Number of papers per year
  geom_col(aes(x = year, y = newpapers_year), alpha = 0.4) +
  theme_classic(base_size = font_size) +
  xlab("Years") +
  ylab("Count") +
  ggtitle("Counts of New Languages (line) and Number of papers (bars)")


stacked = pp.1 / pp.2
stacked
```

```
Warning: Removed 37 rows containing missing values (`geom_line()`).
```

```
Warning: Removed 37 rows containing missing values (`geom_smooth()`).
```

```
Warning: Removed 24 rows containing missing values (`geom_line()`).
```

```
Warning: Removed 24 rows containing missing values (`geom_smooth()`).
```



```
ggsave(plot = stacked, filename = "figures/arima_forecasts.png")
```

```
Warning: Removed 37 rows containing missing values (`geom_line()`).
```

```
Warning: Removed 37 rows containing missing values (`geom_smooth()`).
```

```
Warning: Removed 24 rows containing missing values (`geom_line()`).
```

```
Warning: Removed 24 rows containing missing values (`geom_smooth()`).
```

Additional Statistics

```r
library(magrittr)
```

```

Attaching package: 'magrittr'

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

    extract
```

```r
library(rvest)

# define the page to load
tt =
  read_html("https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers")
    # list all tables on the page
    html_nodes(css = "table")

ethnologue = tt[[1]] %>%
    # convert to a table
    html_table(fill = T)

most_spoken = sapply(strsplit(ethnologue$Language, '\\s*[()]'), `[`, 1)

studied_languages = unique(kiddgarcia$language)
most_spoken[!most_spoken %in% studied_languages]
```

```
 [1] "Mandarin Chinese"        "Modern Standard Arabic" "Standard German"
 [4] "Nigerian Pidgin"         "Egyptian Arabic"        "Yue Chinese"
 [7] "Vietnamese"              "Wu Chinese"             "Tagalog[b]"
[10] "Iranian Persian"         "Hausa"                  "Javanese"
[13] "Western Punjabi"         "Gujarati"               "Amharic"
[16] "Bhojpuri"                "Eastern Punjabi"        "Min Nan Chinese"
[19] "Jin Chinese"             "Levantine Arabic"       "Yoruba"
```