Child Language Acquisition & 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(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)
- The languages and number of papers 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

Here we used the GitHub repository created for the Grambank release paper (i.e. grambank-analysed and not grambank). This is because there are 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 (keeping the dialect with the most complete data),

2) Making dummy variables from the six non-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.

The result of these commands is 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. 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. Here are decisions to note:

- 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
definitely endangered	threatened endangered	7 (Shifting)	Shifting
severely endangered	severely endangered	8a (Moribund)	Moribund
critically endangered	critically endangered	8b (Nearly extinct)	Nearly extinct

UNESCO	LEI (ElCat)	EGIDS	AES
extinct	dormant awakening	9 (Dormant) 9 (Reawakening) 9 (Second language only) 10 (Extinct)	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, because in all scales, this 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. I 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. Computerized matching requires 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
kiddgarcia_matching = read.csv("data/name_matching.csv")

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

# 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

Histograms

Hexbin graphs

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

1) Build a distance matrix between all languages

2) Run the multidimensional scaling algorithm

3) Plot and bin the results.

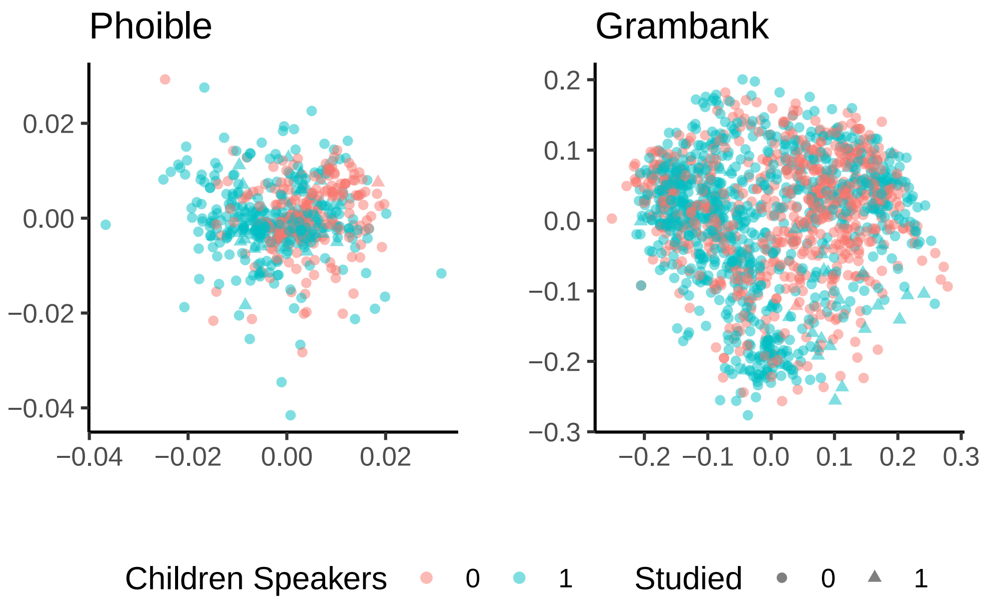
To plot the results, we are required to get custom ggplot code, which has been included in this repository. The custom code was created thanks to Allan Cameron via Stackoverflow.

We do this for the Grambank and Phoible dataset.

```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", "MDS.X", "MDS.Y", "count",
                       "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

  # 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)
```

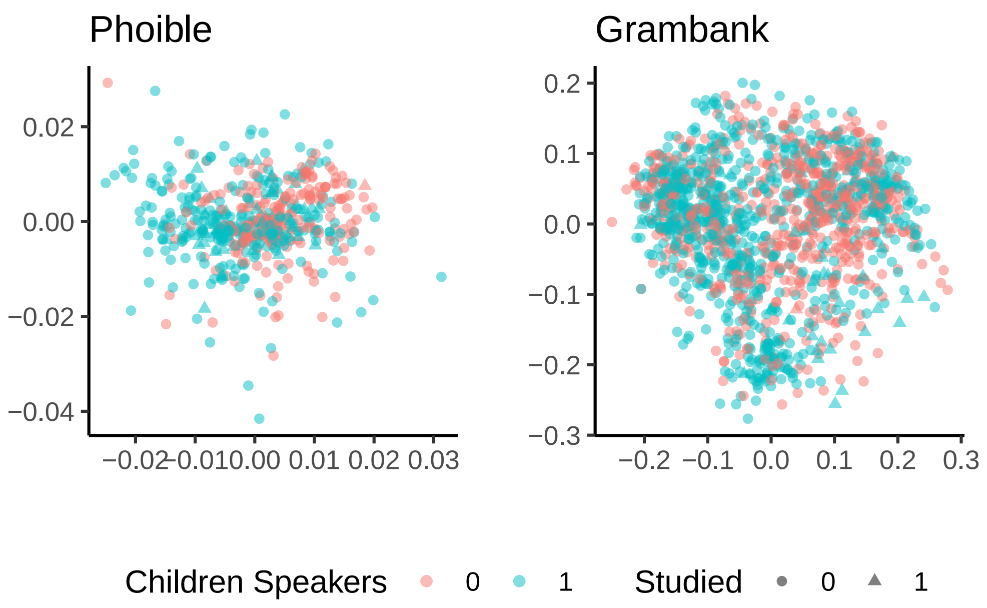
No-bin Plots

```
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')
```



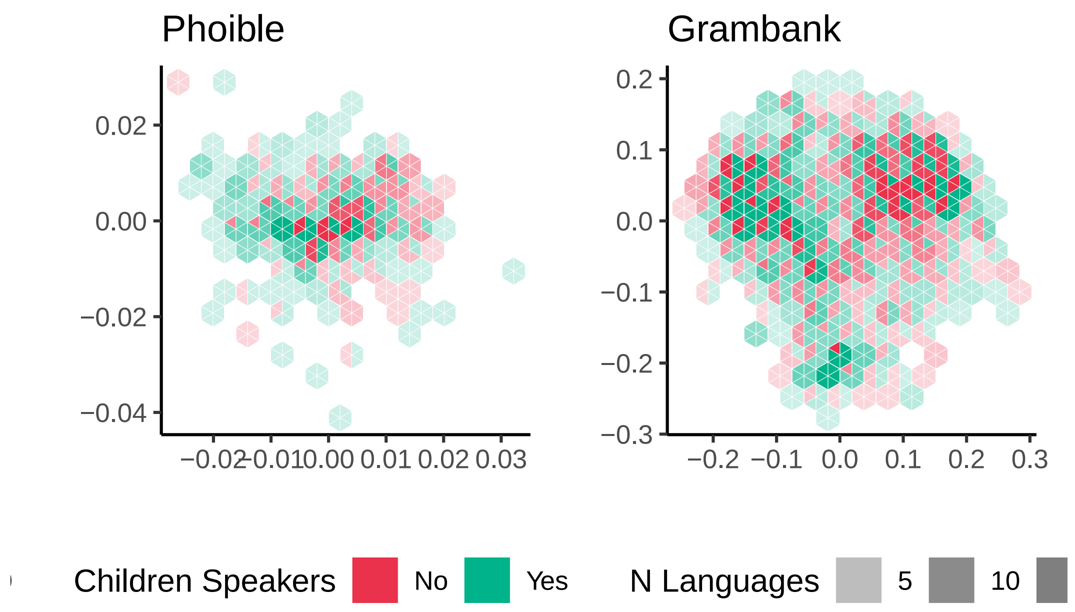
Phoible has two outlier languages. They are Dan (Family: Mande; Country: Côte d'Ivoire, Guinea, Liberia) and Khams Tibetan (Family: Sino-Tibetan; Countries: China [Tibet], India, Myanmar). The below plots remove these points to make the view of phonological diversity more clear. However, the distance of these languages from the majority of phonological diversity is worth investigating further.

```
plot_phoible = plot_phoible %>%
  dplyr::filter(!Name %in% c("Dan", "Khams Tibetan"))

p1.1_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("")

(p1.1_phoible | p1_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



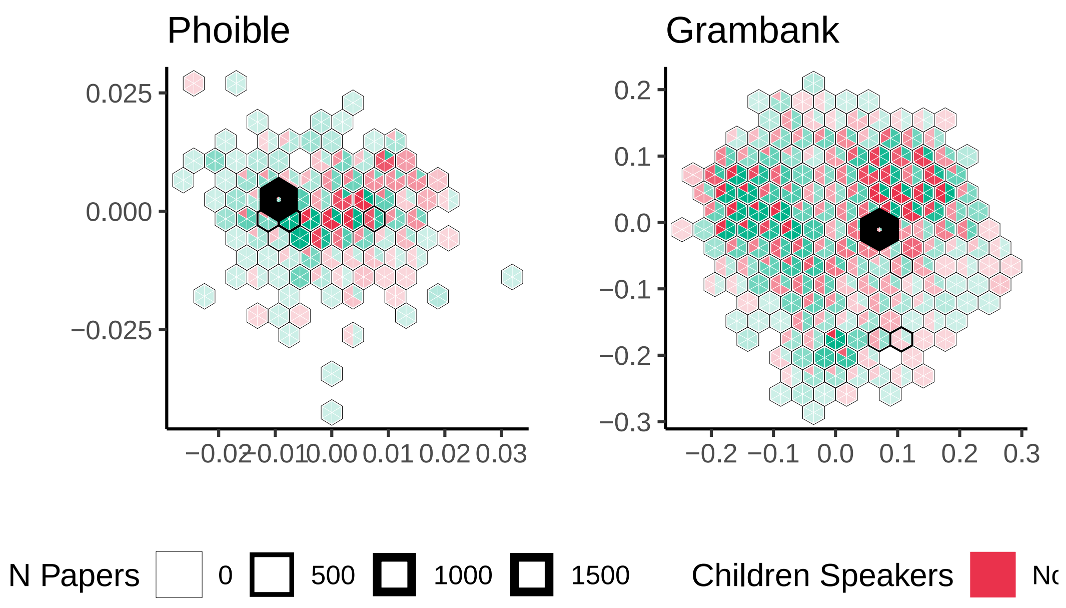
Hexbin Plots

```
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')
```



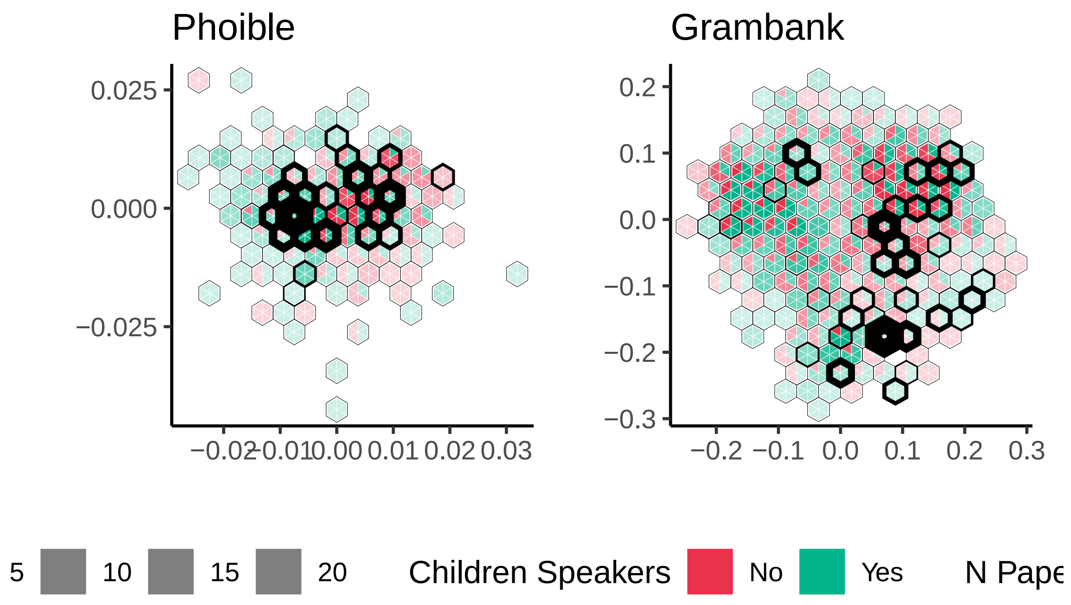
Hexbin & Studied Languages Plots

```
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 = 0.5) +
  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)) +
  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") +
  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 = 0.5) +
  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)) +
  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_phoible | p3_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Hexbin & Studied Languages (Log Scale) Plots

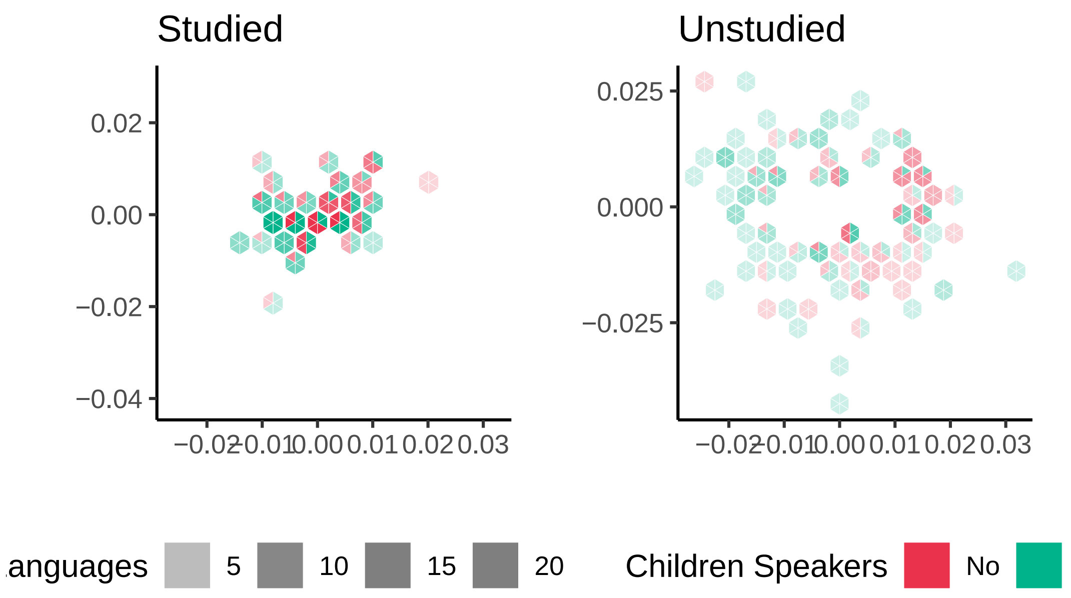
```
p4_phoible = ggplot(plot_phoible,
                    aes(MDS.X, MDS.Y, fill = factor(children),
                        col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = 15) +
  geom_hex(fill = NA, color = "white", bins = 15, lwd = 0.5) +
  stat_summary_hex(aes(
    z = count,
    linewidth = after_stat(value),
  ), fun = ~ sum(log(.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") +
  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 = 0.5) +
  stat_summary_hex(aes(
    z = count,
    linewidth = after_stat(value),
  ), fun = ~ sum(log(.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("Grambank") +
  xlab("") + ylab("")

(p4_phoible | p4_grambank) +
  plot_layout(guides = 'collect') &
  theme(legend.position = 'bottom')
```



Hexbin Plots: Studied vs Unstudied

Phoible

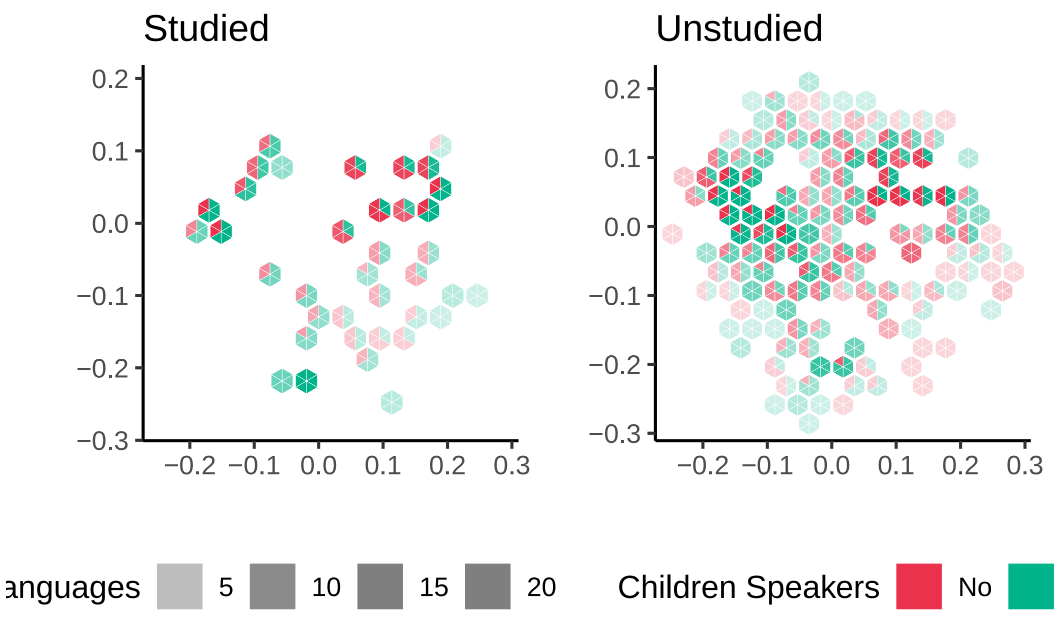
```
# Only show studied diversity
p5.1_phoible = ggplot(plot_phoible,
                     aes(MDS.X, MDS.Y, fill = factor(children),
                         col = factor(children))) +
  geom_hextri(color = "white", linewidth = 0.0, bins = n_bins) +
  stat_summary_hex(aes(
    z = count,
    fill = 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"),
```

```r
         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')
```



ARIMA models

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.

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

# 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
#### 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
```

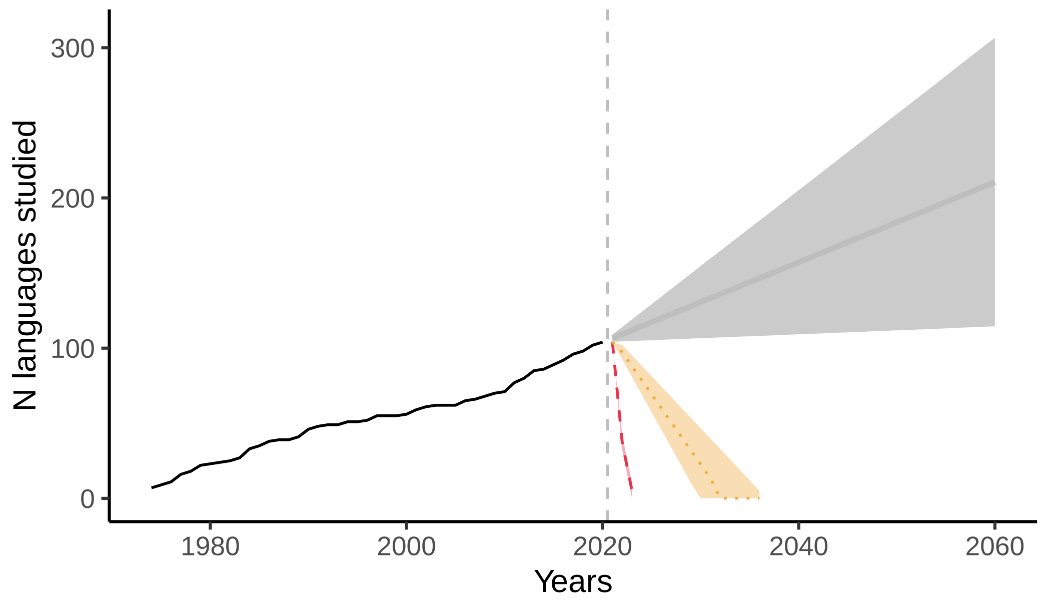
The ARIMA model shows that there is no significant relationship between the increase in the number of papers being published and the number of new languages being studied.

Plot ARIMA

```
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])
```

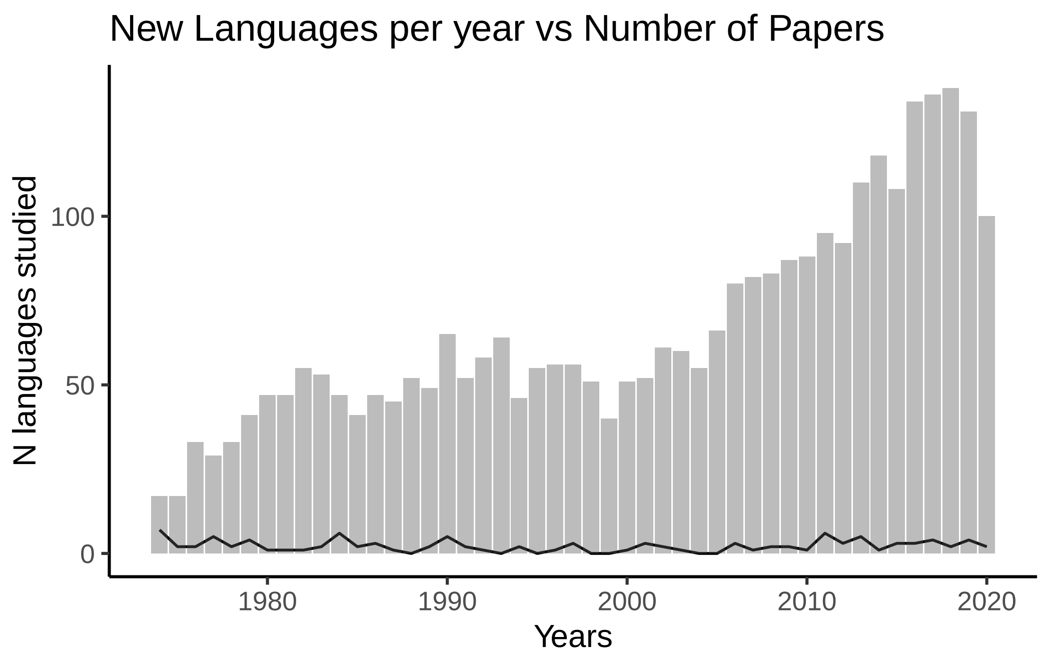
```r
)

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)

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 I 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 I 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

pp =
  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("") +
  ylim(c(0, 310))

pp
```

```
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()`).
```



Plot ARIMA & N Papers published

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

pp =
  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("N languages studied") +
  ggtitle("New Languages per year vs Number of Papers")


pp
```


New Languages per year vs Number of Papers
N languages studied
100
50
0
1980
1990
2000
2010
2020
Years