MSc. Data Science and Data Analytics

IRISH BABYNAMES PACKAGE

by

SURYENDU MANOJ

24251159

Supervisor: Dr. Catherine Hurley

A thesis submitted in fulfilment of the requirements for the degree of
Masters in Data Science and Data Analytics
2024–2025
to the Department of Mathematics & Statistics
Maynooth University

8th August 2025

Abstract

The Irishbabyname package gives an interactive and analytical framework to explore the baby name trend in Ireland. The data for this package is sourced from Central Statistic Office (CSO), Ireland. The package includes functions for visualizing the popularity of each name throughout the years 1964 to 2024, identifying phonetically similar names in girls and boys and visualising the trend of unisex names. The package also offers an interactive Shiny dashboard to facilitate interpretation and exploration of data trends. A vignette has also been included as part of the package to help the users on how to use the package.

Contents

1 Introduction

The Irish Babynames project addresses a growing need for transparent, reproducible tools to explore historical naming patterns in Ireland. Gaining knowledge of name conventions can help one better understand historical cultural dynamics, migratory histories, and societal trends. Because of the intricacy of raw data formats and the extensive preprocessing necessary, historical baby name data in Ireland is a rich but underutilized resource. This project presents the irishbabynames R package, a complete and user-friendly toolkit for examining and displaying Irish baby name patterns, in order to address this difficulty. Drawing on open data published by the Central Statistics Office (CSO), this work presents an R package, irishbabynames,that provides tools and interactive visualizations to explore baby name trends in Ireland.

My primary objective of the project is therefore to package the functionality into a coherent R package. Although CSO publishes raw occurrence counts, dealing with them directly requires substantial data wrangling. By packaging the cleaned dataset together with plotting functions and interactive components into a single installable library, we lower the barrier to entry and encourage new analyses of social change, migration, and naming fashion. This package aims to aggregate and clean official Irish baby name data sourced from the Central Statistics Office (CSO) via the csodata package (Horgan et al. 2024), the data for boys and girls were extracted seperately and combined together to form the base dataset for the irishbabyname package, enable users to easily filter, visualize, and analyze name popularity trends over time, provide interactive and engaging tools including built-in visualizations and a Shiny app dashboard that help users quickly uncover insights, such as shifts in naming patterns, gender based comparisons, and the popularity of specific names or groups of names, offer advanced features like similar name suggestions based on phonetic matching, and interactive exploration of unisex names to enrich user experience,serve as a valuable educational resource for data analysts, researchers, and the general public interested in Irish demographic patterns and cultural trends through baby naming data.

The irishbabynames package was inspired by Hadley Wickham's babynames package (Hadley Wickham 2021) and Mine Çetinkaya-Rundel's ukbabynames package (Çetinkaya-Rundel et al. 2022) for USA and UK respectively. Both of these packages have provided accessible, user friendly resources for researchers and the general public to analyze naming data from the United States and the United Kingdom, respectively. These packages have established standards for the organization, analysis, and visualization of naming data. The usefulness and popularity of providing cleansed and readily analyzable datasets have been proven by these packages. Although comparable raw data for Ireland is available through CSO, there hasn't been a comparable thorough and approachable package for Irish baby names. To make up this gap, this project introduces the irishbabynames R package, which not only allows for comparing data but also has inbuilt visualisation that allows users easily visualise the Irish baby names along with an interactive Shiny dasboard.

Recognizing that publicly available data from the Central Statistics Office (CSO) is complex and requires substantial processing, this package streamlines the analysis workflow by offering cleaned, well-structured datasets and integrated visualization tools. Data were collected separately for boys (VSA50) and girls (VSA60) via the csodata package, then combined into a cohesive dataset. With functionalities built using R's tidyverse framework (dplyr, ggplot2, and tidyr) (Hadley Wickham et al. 2023),(Hadley Wickham 2016),(Hadley Wickham, Vaughan, and Girlich 2024) and enhanced interactivity through plotly (Sievert 2020) and ggiraph (Gohel and

Skintzos 2025), the Irish Babynames package is designed to be accessible and engaging. The in-built visualistaions allows the users to visualise long term trends of indivdidual names from 1964 to 2024, discover phonetically similar names for girls and boys using the Metaphone algorithm, analyse unisex names by initial letter and year and interactively explore all the visualisations via a shiny dashboard, a vignette is also created as a part of the package for the users as a guide to understand how the package can be used. The created package can be currently installed along with vignette using github; remotes::install_github("SuryenduManoj/irishbabynames", build_vignettes = TRUE). It is hosted in the GitHub and not CRAN. All of the files related to the package are uploaded and accessible in the Github repository. All the scripts related to the functions mentioned in this report is listed in the 'thesis_readme/readme.md'. This folder is also accessible in my Github repository.

In this thesis, I describe about the data source, structure of the data,the method for converting raw data into a clean long form dataset. It also details the design of the package including core plotting fuctions that is; plot_trend(), plot_similar_girlsnames(), plot_similar_boysnames(), plot_unisex_names()) and the Shiny application structure and how the vignette is documented. The thesis also demonstrates key results of each plot such as the rise and fall of Gaelic versus Anglicized names, insights into gender convergence in naming, and case studies of names like "Aoife," "Sean," and "Ava", evaluate performance and usability, presenting timing benchmarks, dependency management, and documentation (vignettes, help pages). The thesis concludes with reflections on unanswered questions such as the drivers behind rapid name shifts and propose extensions, including integration with UK datasets, machine-learning approaches to phonetic clustering, and a public web deployment.By the end of the document, readers will have a better understanding of how the babynames package was built and what it can tell us about the Irish naming customs over the last 60 years.

2 Background

In Ireland, the Central Statistics Office (CSO) has continuously produced comprehensive yearly records of baby names, providing raw occurrence figures for both boys and girls separately. However, these datasets need a lot of processing and data wrangling, which limits the academic community's ability to use them for exploratory study. There hasn't been a comprehensive and easily accessible package with in-built interactive visualisation for Irish baby names, despite the fact that CSO provides equivalent raw data for Ireland.

2.1 Data

The dataset used in this package is derived from open data published by the Central Statistics Office (CSO) of Ireland, specifically through the 'csodata' R package (Horgan et al. 2024). The source tables, VSA50 (boys) and VSA60 (girls), contain annual occurrence counts of registered baby names from 1964 to 2024. This data gets updated anually, adding the data for the latest year. To maintain confidentiality, the CSO excludes names that appear less than three times in a given year.To build a comprehensive dataset, the two gender-specific tables were cleaned and standardized to share a common structure and were then combined into a single unified dataset. This dataset includes the name, year of registration, statistic(it mentions whether the value is the occurence or rank for that name for that year) and gender(male or female). This curated dataset forms the core of the irishbabynames package and serves as the foundation for all subsequent analysis and visualization tools developed within the project. As the package develops, there are few additional variables, such as name rank, overall and gender specific proportions, were computed according to the visualisation functions, however these variables are not a part of the main dataset. The resulting structure facilitates filtering by year, name, or gender, enables trend analysis over time, and supports the creation of interactive visualizations.

The below code loads irishbabynames package and displays the first 6 rows of the dataset with the variables Statistic, Name and all the years from 1964 to 2024 and Gender.

```r
library(irishbabynames)
head(irishbabynames[, 1:5]) #restricted to 5 column for a clean and readable preview
```

```
##                                              Statistic    Name 1964 1965 1966
## 1 Girls Names in Ireland with 3 or More Occurrences Rank  Aadhya   NA   NA   NA
## 2 Girls Names in Ireland with 3 or More Occurrences Rank   Aadya   NA   NA   NA
## 3 Girls Names in Ireland with 3 or More Occurrences Rank   Aaira   NA   NA   NA
## 4 Girls Names in Ireland with 3 or More Occurrences Rank  Aairah   NA   NA   NA
## 5 Girls Names in Ireland with 3 or More Occurrences Rank Aaliyah   NA   NA   NA
## 6 Girls Names in Ireland with 3 or More Occurrences Rank   Aanya   NA   NA   NA
```

2.2 Similar Packages

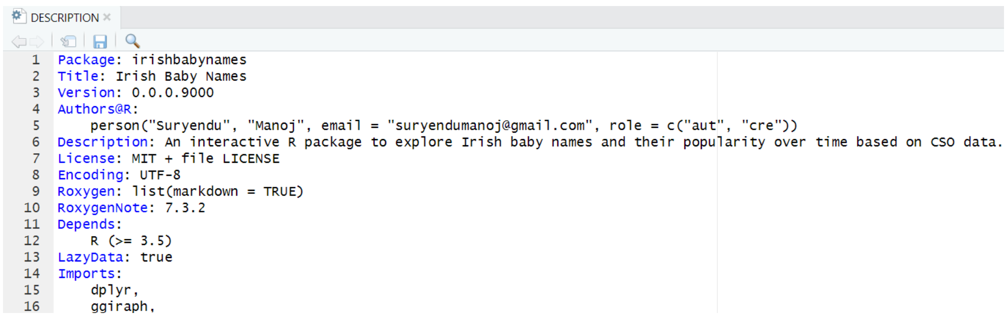
Popular R packages like "babynames" (Hadley Wickham 2021) for U.S. data and "ukbabynames" (Çetinkaya-Rundel et al. 2022) for UK data are examples of prior work in this area. These programs have established standards for the organization, analysis, and visualization of naming data. The usefulness and popularity of providing cleansed and readily analyzeable datasets have been proven by these packages. The babynames package provide Full baby name data provided by the SSA, this includes all names with at least 5 uses. The babynames (U.S.) and ukbabynames (UK) R packages both serve as valuable sources of historical baby name data for their respective countries. The ukbabynames package includes decade-by-decade popularity rankings of baby names from 1904 to 1994, as well as a thorough list of UK baby names that appeared more than three times annually between 1974 and 2020. While both the packages do not contain built-in visualizations, it has inspired a large number of public visualizations, teaching examples, and third-party Shiny apps. For developers creating new tools or packages 'babynames' and 'ukbabynames' packages serves as a richer source of inspiration for visual features, vignettes, and user engagement. There is an existing irish baby names package in R, 'iebabynames' (Stefan 2025), this package is a tidy data package containing Irish baby name data, designed for data exploration and analysis, not for built-in visualizations, it follows the design of similarly structured packages like babynames (U.S.) and ukbabynames (UK). The iebabynames (Ireland), babynames (U.S.), and ukbabynames (UK) packages all provide structured, tidy datasets, optimized for easy exploration and analysis in R. Despite the fact that none of these packages have visualization features built in, they enable users to filter the data, select subsets of interest (e.g., top-ranking names, gender-specific trends, or historical comparisons) and produce unique visualizations that are suitable for publication using external libraries (ggplot2, plotly, ggiraph), enabling users to flexibly illustrate trends, gender ratios, and name-popularity dynamics. Among these packages, babynames (U.S.) is highly utilized in tutorials and educational settings, demonstrating extensive visualization examples across data science communities. The irishbabynames R package is introduced in this project to fill this gap. Separate datasets for boys and girls acquired using the 'csodata' package are combined, cleaned, and aggregated before being presented in a unified and user-friendly format. Additionally, this project has sophisticated phonetic matching capabilities for name suggestions, dynamic visualisations, and an easy to use Shiny app dashboard, going beyond simple data providing. Together, these resources improve accessibility, encourage repeatable analysis, and allow for a more thorough examination of name trends, migration trends, and social change in the Irish setting. This effort greatly reduces the challenge for academics, analysts, educators, and the general public interested in examining Ireland's name patterns by incorporating new interactive components and expanding upon approaches from well-established packages.

3 Description of the package

3.1 Structure of the package

The structure of the irishbabynames R package follows the standard best practices for R package development, incorporating organized directories, modular code, documentation, and testing to ensure reproducibility, usability create a productive workflow by using best practices,reviewing materials like Hadley Wickham's R Packages (H. Wickham and Bryan 2023) . The devtools (Hadley Wickham et al. 2022), usethis (Hadley Wickham, Bryan, et al. 2024), and roxygen2 (Hadley Wickham, Danenberg, et al. 2024) ecosystems' development tools, which simplify processes like function generation, documentation, and version control, were used to build the package. The first step of package creation was naming the package, which had several formal requirements, then the package was created using create_package(), this is where the package lives in its source form (H. Wickham and Bryan 2023) . Each function in the package is documented using roxygen2 to generate help files and maintain clarity. DESCRIPTION and NAMESPACE are two crucial files that provide metadata about the package. The overall metadata of the package, such as the package name and which other packages it depends on is in the DESCRIPTION file. NAMESPACE file can also indicates which functions of the package makes available for others users and, which functions should be imported for from other packages. The usethis::create_package() adds a simple DESCRIPTION file, while the NAMESPACE file is generated automatically to handle exports and imports, ensuring functions are accessible and packages are properly loaded.

See Figure 1 for the DESCRIPTION file

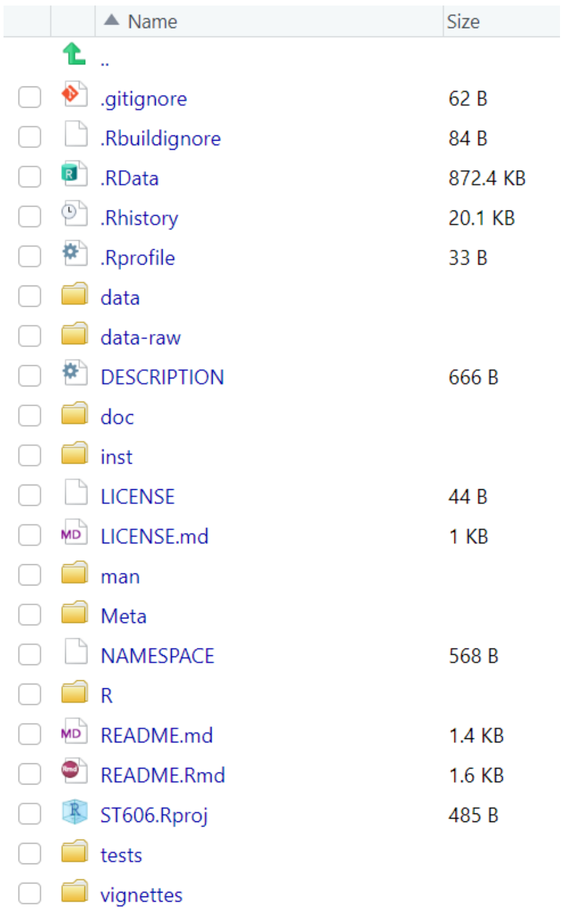


```
Package: irishbabynames
Title: Irish Baby Names
Version: 0.0.0.9000
Authors@R:
    person("Suryendu", "Manoj", email = "suryendumanoj@gmail.com", role = c("aut", "cre"))
Description: An interactive R package to explore Irish baby names and their popularity over time based on CSO data.
License: MIT + file LICENSE
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.2
Depends:
    R (>= 3.5)
LazyData: true
Imports:
    dplyr,
    ggiraph,
```

Figure 1: DESCRIPTION file of this package that describes the metadata of the package.

The raw baby name data was extracted using the csodata package, cleaned, and processed into a tidy format through a script stored in the data-raw/ directory. The processed dataset was then saved in the data/ folder in .rda format, making it easily accessible to users via data(irishbabynames). At the core, the R/ directory contains all user-defined functions, each saved in separate script files. These include plotting functions such as plot_trend(), plot_similar_boysnames(), plot_similar_girlsnames(), and plot_unisex_names(). The package also include shiny app dashboard, stored in the inst/shiny/irishbabynames_app/ directory. The app is launched via irishbabynames_dashboard(), a wrapper function in R/ directory. A vignette is also created in vignettes/ directory, as a document to guide the users on how to use this package.(See Figure 2 for the structure of the package.)

Additionally, the package uses unit testing with the testthat framework to ensure key functions work as expected. Testing is a crucial part of package development that ensures robustness



Name	Size
..	
.gitignore	62 B
.Rbuildignore	84 B
.RData	872.4 KB
.Rhistory	20.1 KB
.Rprofile	33 B
data	
data-raw	
DESCRIPTION	666 B
doc	
inst	
LICENSE	44 B
LICENSE.md	1 KB
man	
Meta	
NAMESPACE	568 B
R	
README.md	1.4 KB
README.Rmd	1.6 KB
ST606.Rproj	485 B
tests	
vignettes	

Figure 2: structure of the files created for the package in R directory.

and reliability by confirming that the code operates as intended. The usethis::use_testthat() creates a tests/testthat/ directory. As a function is defined in the package in the files under R/ directory, corresponding tests to .R files are added in tests/testthat/ (H. Wickham and Bryan 2023) . Vignettes and documentation were generated using roxygen2, and the package was built and installed devtools::install(). The irishbabynames package follows a standard structure which not only guarantees reproducibility and accessibility but also makes the package simple to expand and access by both technical and non-technical users.

3.2 Visualisation

The visualizations designed for the irishbabynames package are intended to support both broad exploration and focused analysis of naming patterns. These plots are interactive, built with plotly (Sievert 2020), allowing users to hover over data points for exact values and easily switch between names or genders. Additionally, a dedicated tab within the Shiny app facilitates exploration of unisex names, where users can filter by starting letter or exact match to see how usage is distributed across genders. The functions in the irishbabynames package provide clear and informative error messages. These messages help users when their inputs are invalid or not supported. For instance, if a name is entered that isn't in the dataset, the plotting functions will show an informative message to point out the issue instead of failing silently or producing no output. This strategy improves user experience by strengthening the package. This is especially important for users who are not familiar with the underlying data. By implementing these checks, we ensure that the functions fail in a controlled way. These visual tools are intended to be intuitive, dynamic, and responsive, providing a rich interface for exploring how naming practices in Ireland have evolved throughout time.

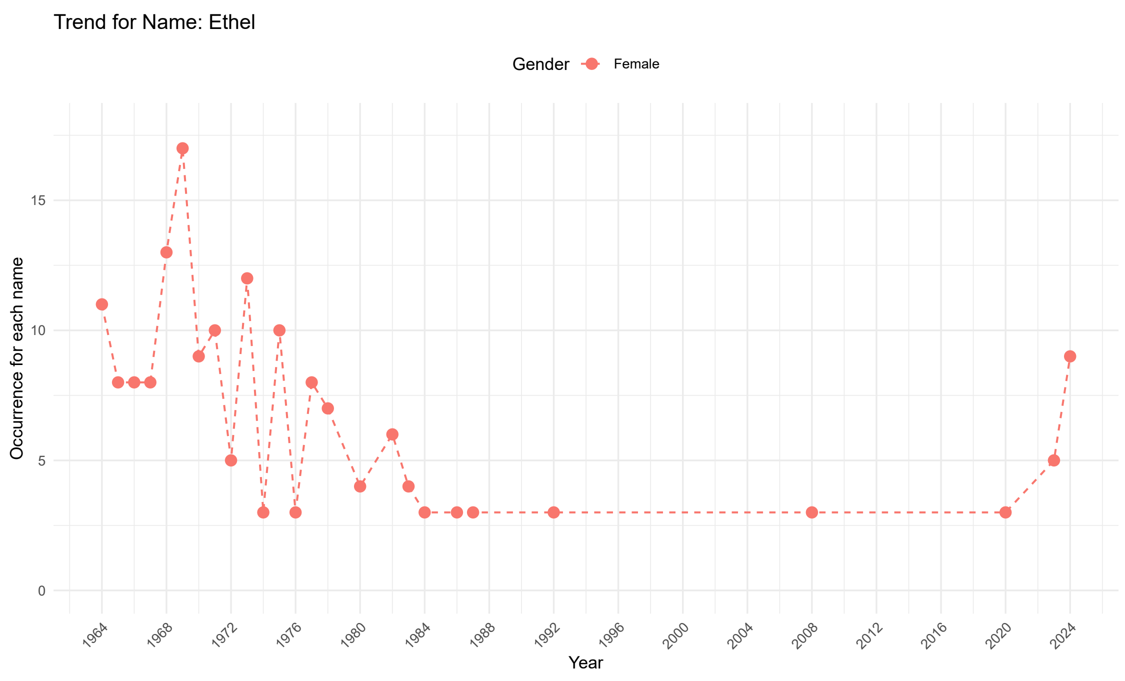
3.2.1 Trend Plot

The first visualization implemented in this package is a time series line plot that displays the popularity of an individual baby name throughout the year 1964 to 2024. This plot is essential to the package's performance since it provides users who want to discover more about the historical trends of a specific name in Ireland with a starting point. The plot is built using the plotly library (Sievert 2020), which provides interactive graphing capabilities in R. Unlike static ggplot2 plots (Hadley Wickham 2016), plotly allows users to hover over data points to see exact counts, zoom into specific time periods, and toggle between different names or genders dynamically. The fundamental function that powers this visualization is plot_trend(). This function takes only a single argument, which is the name to be visualized (e.g., "Alex"). The function begins by reshaping the irishbabynames dataset from wide to long format using pivot_longer(), transforming year columns into a single Year column with corresponding values under Value. By changing the Year and Value columns to numeric types and removing any missing or non-finite values, it guarantees numeric consistency. The data is then split into two key subsets: counts, which includes the actual number of name occurrences for both boys and girls, and ranks, which contains the rank of each name per year, renamed to Rank. These subsets are filtered based on the selected name_input, matched case-insensitively, and merged to include both the count and rank data for each relevant year.

The function uses tidyverse verbs like filter(), select(), and arrange() for preprocessing, and then passes the resulting data frame to plotly() to render the graph. The plot's aesthetic is simple and straightforward, with readability and usability in mind. Data points are connected using a dashed line, and markers are added to highlight each year's count. The hover tooltips dynamically show the year and the number of babies born with the selected name, which is particularly useful for identifying trends such as sudden spikes or long-term declines. Additionally, the function is reactive when integrated into the Shiny app, users can type in a name and instantly see its trend line update in the app without needing to reload or refresh.

The below code shows the output for the function of trend plot. It shows the trend of the name Ethel throughout year 1964 to 2024.

```r
plot_trend("Ethel") # plots the trend for the name, Ethel from 1964-2024
```



If the name that the user inputs is not found in the dataset, no plot is returned and instead of the plot, an appropriate error message is displayed mentioning the name is not found.

3.2.2 Plot for phonetically similar names

The next major visualization in the irishbabynames package focuses on exploring phonetically similar names using the Metaphone algorithm from the phonics package (Howard, II 2021) . I have created plot for phonetically similar names for boys and girls seperately. With this plot, users can enter a name and get a list of names that sound alike, even if they are written differently. A time series line graph is then used to plot these names and show how their popularity has changed over time. This function works particularly well for detecting name trends that follow phonetic fads, which are patterns in which many versions of a sound (e.g., “Shawn” vs. “Sean”) rise or fall together.

This visualization is built using the ggplot2 (Hadley Wickham 2016) and ggiraph (Gohel and Skintzos 2025) libraries. First, data is cleaned and reshaped using dplyr and tidyr. The Metaphone algorithm from the phonics package (Howard, II 2021) is then applied to convert both the input name and all names in the dataset into phonetic encodings. Names that share the same encoding as the input are treated as phonetically similar. The stringdist package (van der Loo 2014a) is optionally used to further filter names based on edit distance. This preprocessed dataset is plotted using geom_line_interactive() and geom_point_interactive() from the ggiraph package, providing tooltips that display name, year, occurrence for the names, and rank upon hovering. Additionally, the integration of both the plots for boys and girls in shiny dashboard lets users users choose the gender for the names they want to find the similarity for, this makes the the visualisations of this package easily accessible.

The R phonics package is made to offer a range of phonetic indexing algorithms that are currently in widespread and less-used applications. When a string is pronounced, the algorithms typically reduce it to a symbolic representation that approximates the sound produced. They serve as a stand-in for several string distance methods and can be used to match words and names. The general form of a phonetic spelling algorithm is to remove all of the non letter characters, so that numbers, spaces, hyphens, and other punctuation characters are removed from the subject string. Then, the string is transformed into a single case, typically upper case. These basic operations are so common that several of the implementations within this package share the same opening lines to preprocess a string.

Metaphone is a family of loosely related phonetic spelling algorithms created by Lawrence Philips. The original algorithm, usually just called Metaphone, is implemented in this package. Metaphone captures 16 core consonant sounds in multiple languages and represents them in the final phonetic spelling. In addition to source language flexibility, Metaphone is also adept at encoding ordinary words.

The Metaphone algorithm does not recognize accented characters. This causes names like "José" or "Seán" to be misinterpreted when processed directly. Since these characters are not changed to their unaccented versions, the algorithm may produce different phonetic representations for words that sound similar. To avoid these issues, it's important to standardize the names before using the Metaphone algorithm. This means removing accents and converting the characters to their ASCII equivalents. For instance, "José" would become "Jose." This process ensures accurate phonetic matching.

```
clean_names <- stri_trans_general(unique_names, "Latin-ASCII") # translates to unaccented cha
clean_names <- gsub("[^A-Za-z]", "", clean_names) # removes '-','', etc.
```

This code snippet starts by normalizing accented characters in unique_names. It transforms them into their nearest ASCII versions using stri_trans_general() with the "Latin-ASCII" transformation. This step ensures that names like "Émilie" or "Seán" are changed to "Emilie" and "Sean". Thus, these names become suitable for Metaphone algorithm. The next line improves the names further by removing any remaining non-alphabetic characters, including hyphens, apostrophes, or spaces, with gsub("[^A-Za-z]", " ", …). Together, these steps standardize the input. This ensures that the output consists of unaccented and purely alphabetic strings for further phonetic matching.

Rather than processing each letter one by one, the Metaphone algorithm looks at groups of letters to find phonetic changes and irregularities in words. Initially, the algorithm uses diphthongs for its transformations. It removes all vowels from the encoded word and swaps MB with B if it appears at the end of the word. It also changes SCH into SK and CIA into X. This shows that the algorithm emphasizes the phonetic sounds of vowels alongside consonants rather than focusing on single vowel or consonant sounds (Koneru, Pulla, and Varol 2016). This algorithm improves the analysis of Irish baby names by providing better exploration, greater interactivity, and more linguistic detail in the visualizations..

```
phonics::metaphone(c("Mary","Maura","Marie","Mariam", "William")) #shows phonetic codes of th
```

```
## [1] "MR"  "MR"  "MR"  "MRM" "WLM"
```

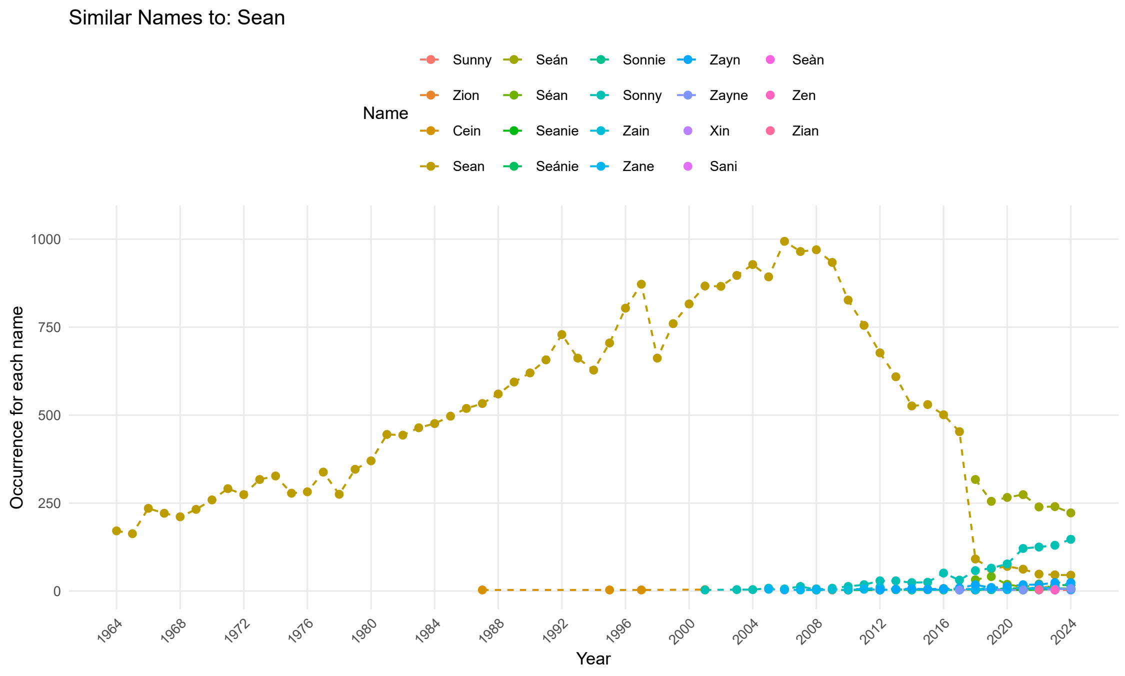
The above code snippet shows that 'Mary', 'Maura' and 'Marie' have same phonetical code, 'MR', even though they are spelled differently and how different they are from Mariam, with phonetical code 'MRM'. Thus, making 'Mary', 'Maura' and 'Marie' phonetically similar.

3.2.2.1 Plot for phonetically similar names for Boys The function, plot_similar_boysnames() behind this visualisation, accepts a maximum string distance as well as an input name. It first filters the dataset to include only boys' names with valid occurrence statistics. It reshapes the data into long format and computes the Metaphone code for every distinct name. It finds phonetically similar names by comparing the input name's Metaphone code to those in the collection.

```r
plot_similar_boysnames <- function(input_boys_name, max_distance = 1) {
  plot_similar_names(
    input_name = input_boys_name, #input name by the user
    gender_label = "Male",
    gender_stat = "Boys Names in Ireland with 3 or More Occurrences",
    gender_rank_stat = "Boys Names in Ireland with 3 or More Occurrences Rank"
  )
}
```

The function plot_similar_boysnames() acts as a wrapper for the main function plot_similar_names(). It creates similarity-focused plots for male baby names in Ireland. You can input a name and an optional max_distance parameter, which likely controls how closely names sound alike. The function assigns the gender label as "Male" and includes specific statistical labels for better visualization. It focuses on boys' names that appeared three times or more. This setup simplifies the plotting process for male names by reducing the need for repetitive manual adjustments. The final result product is a color-coded, interactive graphic in which each line indicates a name that is phonetically similar over time.

```r
plot_similar_boysnames("Sean") # plots phonetically similar names for Sean
```



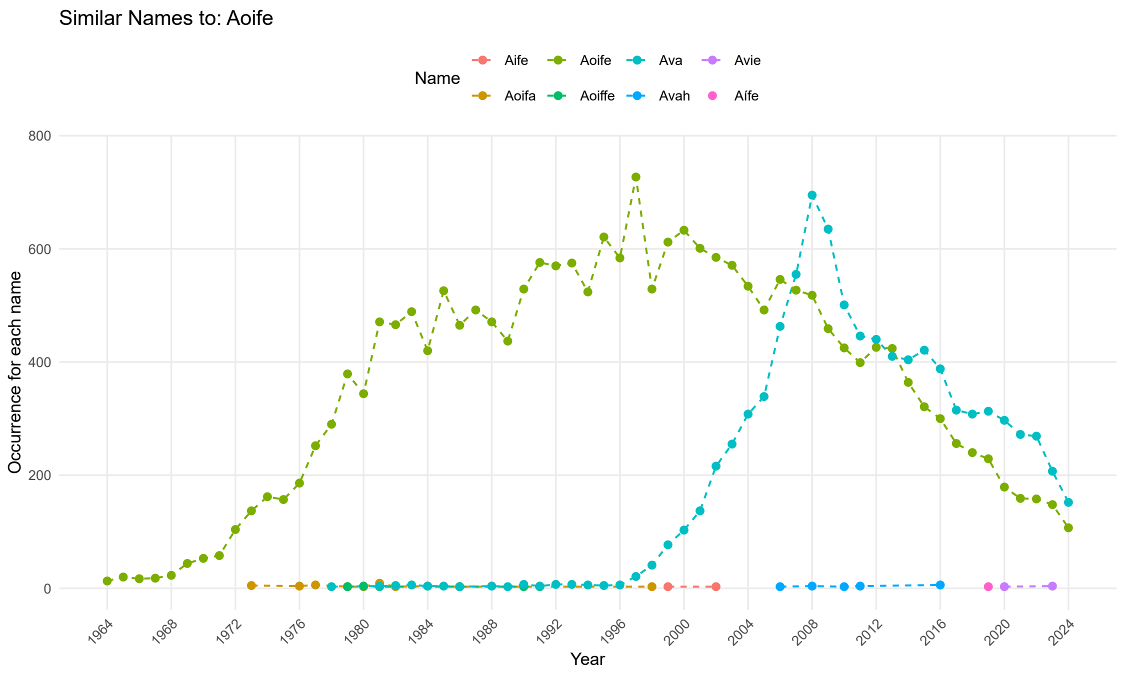
This plot shows trend of all the names phonetically similar to Sean from 1964 to 2024. Tooltips, allows to display comprehensive information when the user hovers over it. Tooltip shows the rank and occurrence of each name for that year. If no similar names to the input name are found, then an error message is shown mentioning no phonetically similar names found.

3.2.2.2 Plot for phonetically similar names for Girls

The plot_similar_girlsnames() function also acts as a wrapper function for main function plot_similar_names() and generates an interactive line plot that visualizes the historical trends of phonetically similar girls' names in Ireland. It similar to plot_similar_boysnames(), but for girls. Likewise it employs the Metaphone method to find names that, although having quite different spellings, sound alike when pronounced. The function first transforms the dataset into a long format with year-wise counts, filtering it to only include the names of girls with valid occurrence data. After that, it uses phonetic cleaning to eliminate non-alphabetic characters and uses the phonics package to translate names into Metaphone codes. It chooses a selection of names that sound similar by comparing the input name's Metaphone code to every other name in the dataset.

The plot is built using ggplot2 (Hadley Wickham 2016)and made interactive through ggiraph (Gohel and Skintzos 2025), allowing users to hover over points to see detailed tooltips. On hovering, users can see the name, year, rank and occurrence of the name.

```r
plot_similar_girlsnames("Aoife") # plots phonetically similar names for Aoife
```

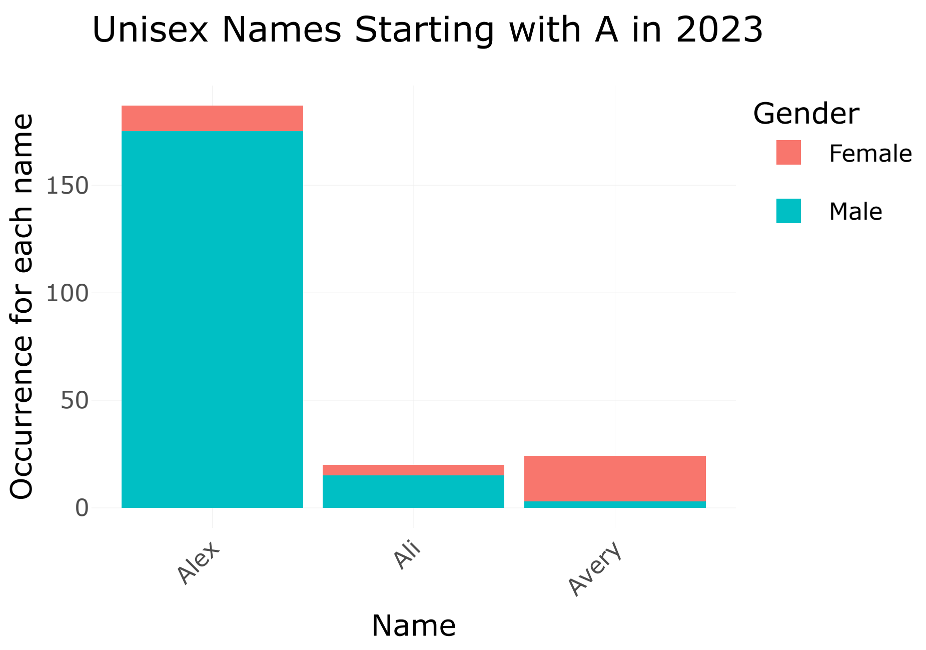


This plot shows trend of all the names that are phonetically similar to Aoife from 1964 to 2024. The plot also shows a similar error message to the plot of boys similar names when no phonetically similar names are found.

3.2.3 Plot for Unisex names

The function, plot_unisex_names(), generates an interactive bar chart that visualizes unisex baby names, names given to both boys and girls. This plot allows users to select names that start with a specific letter in a given year. This plot highlights gender-neutral naming patterns, a topic that is becoming more and more relevant in modern culture, which gives the Irish Babynames package a distinct purpose. The function first filters the dataset to include only occurrence statistics for male and female names, reshapes it into long format, and identifies names used for both genders in the selected year. It further limits the list to names starting with the specified letter, allowing users to explore subsets of unisex names based on personal or analytical interest.

```r
# plots unisex names for the names that starts with A for the year 2023
plot_unisex_names("A","2023")
```



The plot is a color-coded, interactive stacked bar chart, where each bar represents a unisex name split by gender for a specific year. On hovering, users are able to see the name, gender and exact occurrence value for that name. Since there can be years where there are no unisex names starting with the user specified letter, in that case, an error message is displayed, mentioning no unisex names starting with that letter is found in the specific year. This visualization is especially significant since it exposes naming customs that defy gender stereotypes and offers an adaptable, entertaining method to investigate a particular name category within Irish naming culture.

4 Shiny Dashboard

The Shiny app dashboard found in the irishbabynames package offers a responsive and intuitive interface for analyzing trends in Irish baby names, allowing users to do so without the need to write any code. The shiny dashboard is built using shiny package in R, a framework for building web applications and dashboards directly from R (Chang et al. 2025) . As a reactive framework, it can refresh any connected objects when the source changes (Kasprzak et al. 2021). The shiny dasboard for irishbabynames package allows users to filter, visualise and compare name data through drop-down menus, text inputs, and interactive plots.

4.1 Basic structure of Shiny application

To create an R/Shiny application, you need the R environment, which you can get from CRAN. While it's not required, many developers like to use RStudio, a popular Integrated Development Environment (IDE) for R, because it makes the development process much easier. Installing the Shiny package is necessary for building R/Shiny applications.

Creating an R/Shiny application is similar to developing interactive web applications in different ways. It involves building a graphical user interface for the client side and managing user requests that come from that side. A typical R/Shiny application is contained in a shiny R script named app.R, which is composed of two parts , ui and server part, which is placed in the root directory of the R/Shiny application. ui part is is used to define the graphical interfaces of an R/Shiny application. The appearance and layout of an R/Shiny application are configured in ui part. Moreover, different input widgets are established up in the UI to collect user inputs, such as filtering and text input. User inputs are collected through the UI and sent to the server for processing. The server calculates the results and then shows them as figures and plots in the places specified by the UI. The code for ui and server can also be separated into two R script for major R/Shinyapplications (Addokali and Elburase 2022) .

The following code snippet shows ui part of the shiny dashboard. The below code shows the ui side of Trend plot. Similar UI elements for the other plots that will be included in the shiny dashboard have also been created in the same .R file.

```
ui <- fluidPage(
 theme = shinytheme("flatly"), #a theme set for the dashboard that gives a modern look
 titlePanel(
   div(
     h1("Irish Baby Names Dashboard", style = "color:#2c3e50"),
     p("Explore trends and patterns in Irish baby names from 1964-2024.", style = "font-size
   )
 ),
 navbarPage("Irish Baby Names Dashboard",
# the first tab which displays the first plot, that is, Trend Plot
   tabPanel("Trend by Name",
             sidebarLayout(
               sidebarPanel(
                 wellPanel(
                   tags$hr(),
                   icon("baby"), strong(" Enter Name"),
                   textInput("trend_name", "Name:", value = "Alex"),
                   tags$hr()
                 )
               ),
               mainPanel(                # output displayed in the right side of the panel
                 tags$h3("Trend Over the Years"),
                 br(),
                 girafeOutput("trend_plot")
               )
             )))
)
```

The code below provided only includes the server component of plot_trend(), and similar server components for the other plots have also been developed within the same .R file.

Code snippet for server side of the shiny dashboard:

```
server <- function(input, output, session) { # collects user inputs
  output$trend_plot <- renderGirafe({
    plot_trend(input$trend_name) # plots the function and passes the text input by the user
  })
```

The dynamic nature of R/Shiny application development requires regular launches of the application to check its perfomance and layout. The app.R also contains a fucnction to call the shinyApp function. An R/Shiny application can be launched and displayed in the viewer panel or a new window of RStudio, or the default web browser of the system. It is recommended to launch an R/Shiny application in the web browser. The update in the design or functionality of an R/Shiny application, resulting from code changes, can be observed instantly by refreshing the web browser (Addokali and Elburase 2022).

4.2 The design of the shiny app

The inital stage in the development of an R/Shiny application is to design the UI in the ui side of app.R. The fluidPage() function provides a layout that adjusts to the size of the user's browser window, where as the navbarPage() function creates a multi-page layout that features a navigation bar. The R/Shiny application's interface is designed by arranging UI elements within the page layout. Typically, a single page is divided into several panels by using Panel() functions inside the Page() functions. Different functions like titlePanel(), tabPanel(), sidebarPanel(), and mainPanel() are combined into a unified set of UI elements within a single panel. The title panel sets up the title area of an R/Shiny application. The sidebar panel adds various widgets for collecting user inputs. The main panel serves as the main area for showing outputs created by server.R in the form of plots. The UI element identifiers defined on the UI side are used to access the input variable in the R environment, which contains the user inputs gathered by different UI elements. The application incorporates plotly and ggiraph graphics, enabling tooltips, zooming, and interactive exploration. Numerous activities, including reading user-uploaded text, modifying data, and producing displayed charts, are carried out based on user inputs and other server-defined variables. An R/Shiny application's standard workflow begins with the graphical interface's design and user input widget setups. Reactive functions are used on the server side to process user inputs received via the user interface. After then, the results are dynamically produced and shown on the client side, allowing for real-time user engagement and data exploration.

A call function, irishbabynames_dashboard(), was created to call the shinyApp in a seperate .R file. This function launches the Shiny dashboard included in the irishbabynames package in the app.R.

```r
irishbabynames_dashboard <- function() {
  app_dir <- system.file("shiny/irishbabynames_app", package = "irishbabynames")
  if (app_dir == "") {
    stop("Could not find app directory. Try re-installing the package.", call. = FALSE)
  }
  shiny::runApp(app_dir, display.mode = "normal")
}
```

This function directly points to the app.R. It does not contain the app code but loads the app when called. This function runs the app using runApp(), to which the directory of app.R is passed, calling the shiny application to display the dashboard.

The default theme of an R/Shiny application is a monotonous theme. So, a package called shinythemes package (Chang 2021) was developed enhance the visual style of an R/Shiny application by using the theme argument within the Page() function in the ui part of the .R script. One of the themes from shinythemes package that I have used is the 'flatly'. It is a pre-built bootstrap based UI theme, the flatly theme is a clean, modern, and professional looking design inspired by the Flat UI aesthetic. It uses a minimalist color palette with cool shades primarily blues and greys and flat buttons, panels, and input boxes.

See Figure 3 for the Shiny dashboard

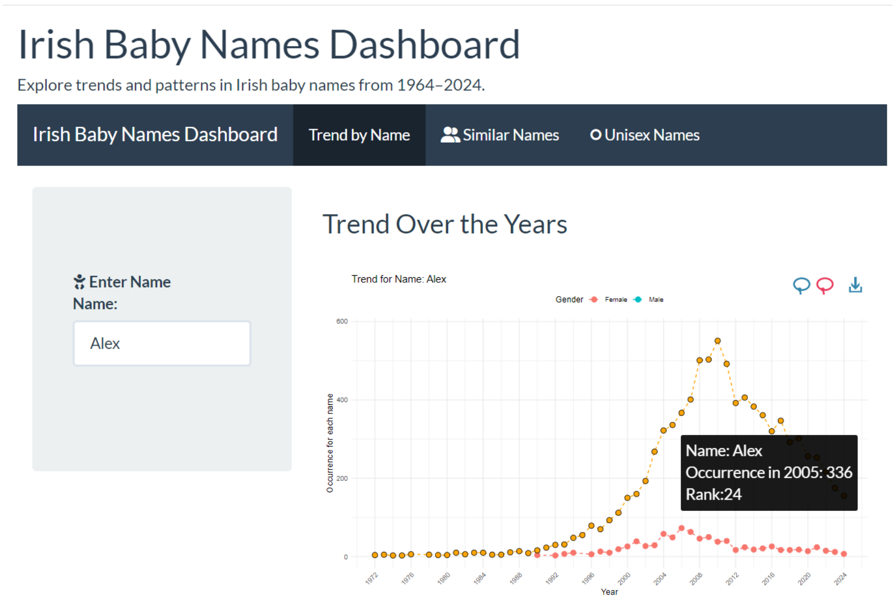


Figure 3: Shiny Dashboard created for this package with all plots intergrated

R/Shiny is a robust and user-friendly framework that allows R users to create interactive dashboards that encourage web browser data analysis and visualization. By including this dashboard, the application significantly lowers the barrier for non-technical users, allowing scholars, educators, and the general public to access sophisticated name analysis. It transforms the dataset into an exploration platform that enables users to investigate societal shifts throughout time, gender trends, and cultural naming patterns in an easy-to-use and captivating way.

5 Vignette

A vignette is a lengthy description about the package. The vignette for irishbabynames package act as a guide and showcase how the package can be used to explore baby names trend in Ireland. Vignettes illustrate plots, examples of plots that connect several functions. Written in R Markdown, vignettes support formatted text, embedded code chunks, and visual outputs, making them an excellent tool for tutorials and package overviews. These are particularly beneficial for new users who want to see how the package works in practice rather than just in theory. The vignette runs code from the package and allows to provide context and instructions for the created feature. The generated document could be a readable website or a PDF file. All the vignettes associated with the package can be seen using browseVignettes().

5.1 Basic structure

The vignette was created using usethis::use_vignette() function, this creates a vignettes/ directory. On creating a vignette, required dependencies are added to DESCRIPTION, that is, it adds knitr to the VignetteBuilder field and adds both knitr and rmarkdown to Suggests. A draft for the vignette is also created in the vignettes/ directory as .Rmd file. This draft document includes the key components of a R Markdown vignette to which contents can be added. rmarkdown::html_vignette is the output format, which ensures compatibility with CRAN standards while supporting interactivity. Once the vignette is drafted, code chunks are added. Code chunks were configured with eval=FALSE to avoid heavy computation or dependency issues during automated builds, while still allowing users to copy and run the code themselves.

Regularly render the complete vignette, this necessitates some planning because, in contrast to tests, a vignette is automatically rendered using the version of the package that is presently installed rather than the current source package because of the initial call to library(irishbabynames). Vignettes are built into the packages with install(build_vignettes = TRUE), then use browseVignettes() to view the vignette.

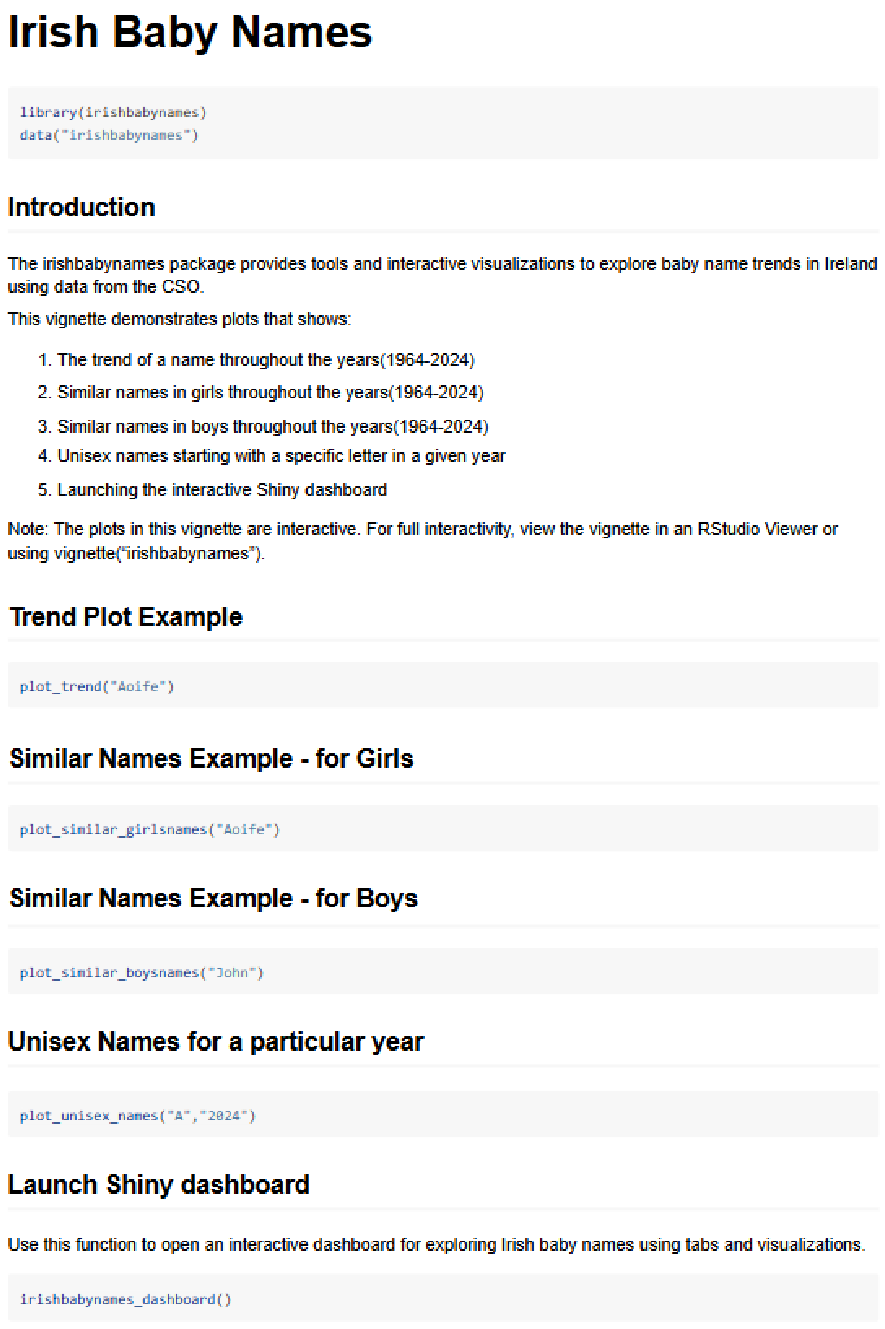
5.2 Vignette of irishbabynames package

The vignette in the Irishbabynames package acts as a guide detailing the main features of the package. It begins by loading irishbabynames library and preparing data. The user is then guided through a number of essential visualizations by the vignette, such as trend plots for individual names, plots of names that are phonetically similar (using Metaphone matching) for boys and girls, and unisex name distributions filtered by initial letter and year. These examples show how users can use user-friendly, interactive tools to draw insights from baby name data. Sample input is used to demonstrate each function, and the results are contextualized to underline their interpretive utility.

```r
vignette("irishbabynames") # code to load vignette of this package
```

Once the vignette is created, the vignette becomes accessible to users through the vignette(“irishbabynames”) command or browseVignettes(“irishbabynames”). It provides users with a separate instruction that they can read and follow within RStudio,it enhances the Shiny app and help files. All things considered, the vignette improves the package’s usability and accessibility by illustrating how to examine Irish baby name patterns in an organized, hands-on manner.

See Figure 4 for the vignette



Irish Baby Names

```
library(irishbabynames)
data("irishbabynames")
```

Introduction

The irishbabynames package provides tools and interactive visualizations to explore baby name trends in Ireland using data from the CSO.

This vignette demonstrates plots that shows:

1. The trend of a name throughout the years(1964-2024)
2. Similar names in girls throughout the years(1964-2024)
3. Similar names in boys throughout the years(1964-2024)
4. Unisex names starting with a specific letter in a given year
5. Launching the interactive Shiny dashboard

Note: The plots in this vignette are interactive. For full interactivity, view the vignette in an RStudio Viewer or using vignette("irishbabynames").

Trend Plot Example

```
plot_trend("Aoife")
```

Similar Names Example - for Girls

```
plot_similar_girlsnames("Aoife")
```

Similar Names Example - for Boys

```
plot_similar_boysnames("John")
```

Unisex Names for a particular year

```
plot_unisex_names("A","2024")
```

Launch Shiny dashboard

Use this function to open an interactive dashboard for exploring Irish baby names using tabs and visualizations.

```
irishbabynames_dashboard()
```

Figure 4: Vignette for the irishbabynames package, this helps users how to use the functions of this package

This shows examples on how to use different functions of the irishbabynames package including how to call the pacakeg's dashboard.

6 Challenges and solutions

During the development of irishbabynames package, there were several challenges I encountered. The majority of challenges were related to following the standard structure of R package and using development tools like load_all(), test_that() from devtools package (Hadley Wickham et al. 2022). Initially, it was confusing to understand where each function, data and shiny app should be placed as they are created in separate folder and and how to test them efficiently within the package environment. However, I was able to get beyond these challenges and create a productive workflow by using best practices,reviewing materials like Hadley Wickham's R Packages (H. Wickham and Bryan 2023) , and testing iteratively with the devtools package.

Another challenge I encountered was finding similarity in names for both boys and girls and visualizing them. Initially, I experimented this visualisation with string distance metrics like Levenshtein distance from stringdist library (van der Loo 2014b) . Levenshtein algorithm estimates the distance between two texts, where this distance is calculated by how many modifications must be made to the first text in order to make it identical to the second text (Addokali and Elburase 2022). Levenshtein calculates the number of character changes needed to change a name, but it frequently yielded unusual findings for names that sound the same but have different spellings, hence it fails with names that sound the same but are spelled differently, resulting it in unreliable in capturing phonetic resemblance, which I think is more important when it comes to names. After exploring alternative methods, I chose the Metaphone algorithm because metaphone identifies words with similar pronunciation and is used to index the words based on their pronunciation (Awad and Lee 2016). For the goal of recommending phonetically related names, Metaphone proved to be more precise and effective, particularly when dealing with Irish names that had anglicized versions or non-standard spellings. Additionally, it worked well with the package's interactive elements, giving users a useful and significant name suggestion feature.

A limitation tied directly to the dataset was how small the occurrence values were for many names, this made it challenging to visualize unisex names. The CSO data only includes names with at least three occurrences per year, and unisex names are typically less common overall. Furthermore, when filtering the unisex names by both initial letter and single year the resulting data frequently contains names with little usage. This made it challenging to use a line chart to adequately represent the data because the values were too small to show any distinct patterns or significant trends. Many data points appeared compressed near the x-axis or failed to render visibly. To address this issue, the line chart was replaced with a bar chart, which provides a clearer representation of names with smaller values by allowing each bar to be distinctly visualized. In order to match the visualization strategy with the features of the underlying data, the bar chart made the gender based comparison of unisex names more visually accessible and made even low occurrence names stand out.

7 Evaluation

In contrast to the babynames (Hadley Wickham 2021) and ukbabynames (Çetinkaya-Rundel et al. 2022) packages, which offer static datasets with no interactivity, the irishbabynames package offers a dynamic interactive visualisation integrated with a Shiny dashboard and phonetic matching features to explore the naming trends in Ireland. This package offers interactive tooltips, user driven filtering and enhanced unisex names exploration which are few features not seen in babynames and ukbabynames package.

7.1 Visualisation results

The irishbabynames provides a collection of interactive visualisations that provide an informative framework to explore baby name trends in Ireland throughout the year 1964 to 2024. The primary plot created is the trend plot, which displays the user input name's popularity over time and enables users to spot long-term trends. This is complemented by the phonetic similarity visualization, which uses the Metaphone algorithm to group names that sound similarly, independent of spelling. This allows users to investigate linguistic and cultural naming variations, including anglicized forms and alternative spellings. The unisex names visualization displays names used by both boys and girls, filtered by year and first letter, to emphasize naming practices that defy conventional gender standards. Even with low occurrence numbers, this plot's clarity is guaranteed by the use of bar charts.

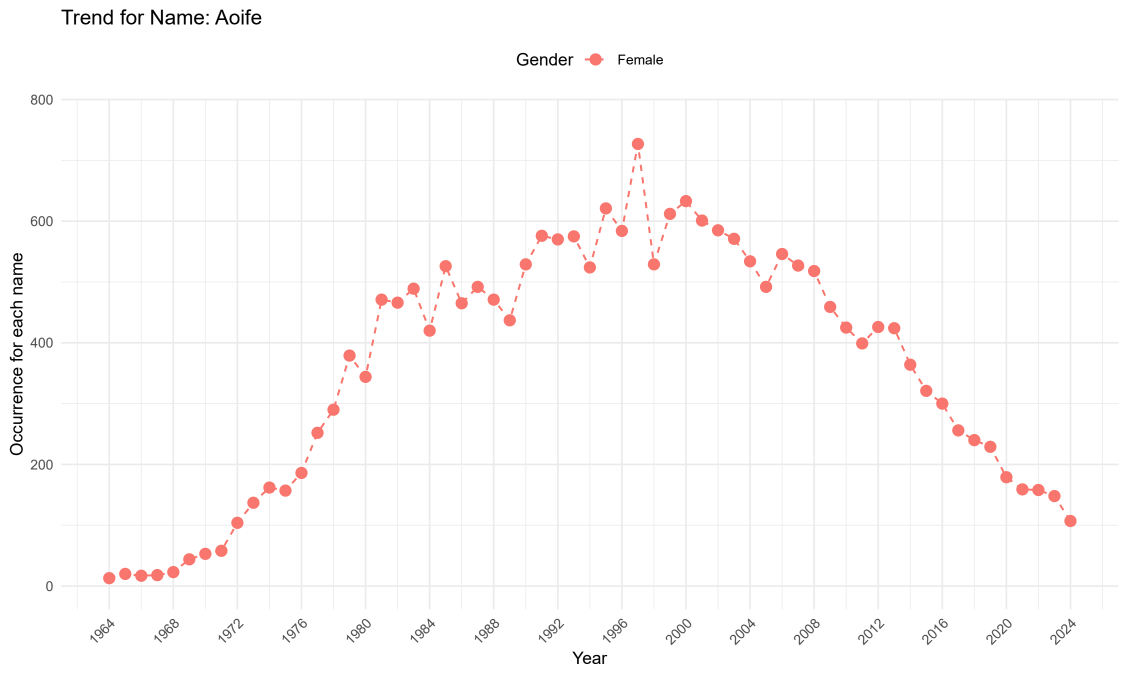
There is a comparable visualization tool known as NameVoyager, launched as part of the Baby Name Wizard project (Wattenberg 2005), which features a stream graph-style layout for examining the trends of baby names throughout the years. The NameVoyager is a web-based visualisation, that allows the user to see a set of stripes representing all the names in the database. Every name appears as a smooth, flowing band. The width shows its relative popularity. This allows for viewing and comparing multiple names at the same time. This design emphasizes visual storytelling and helps with recognizing patterns over the decades. The irishbabynames package, on the other hand, emphasizes a detailed and analytical approach, illustrating the trend of a chosen name through interactive dashed line graphs created with ggplot2 and ggiraph and bar charts using Plotly, which allow for rich, interactive features like zooming. Tootlip shows the details of each name when hovered and enhancing the overall user experience. These visualizations are integrated into an interactive shiny dashboard that offers easy access and user friendly interface to explore the names. The existing method emphasizes clarity and thoroughness for specific names.

7.1.1 Trend Plot

The trend plot in irishbabynames package provides insights into how individual names have evolved in popularity throughout the years 1964 to 2024. The fact that names like "Aoife," "Jack," and "Sean" have been used regularly for a long time shows how deeply ingrained they are in Irish culture and history. However, names like "Mary," which were common in the 1960s and 1970s, are obviously becoming less common over time, suggesting that naming preferences are shifting across generations. These historical tendencies align with well-known sociological phenomena, such as the rise in modern, globally-sounding names in more recent decades and the decline in names with religious implications. The visualization also helps detect peaks or spikes for a specific period of time that may be tied to cultural, social, or media influences. These variations would be difficult to detect through static tables, but the interactive trend plot makes them immediately visible and easier to explore. The interactive nature of Plotly enhances the analytical experience from a technology and usability standpoint. Hovering over specific points allows visitors to examine exact figures per year along with the rank of the name.

The below code shows the output for the function of trend plot.

```
plot_trend("Aoife") # plots the trend for the name, Aoife from 1964-2024
```



The plot titled "Trend for Name: Aoife" displays the popularity of the name Aoife (female) in Ireland from 1964 to 2024, measured as occurrences of each year. The graph reveals a clear rise, peak, and decline pattern, which can be interpreted in four distinct phases:Emergence and Rise (1964–1990s), Peak popularity, Gradual decline and cultural interpretation.

Less than 200 cases of Aoife were recorded in the early years. But in the middle of the 1970s, the name's popularity started to soar. Throughout the late 1980s and early 1990s, this upward tendency persisted, indicating a rising generational or cultural predilection for traditional Irish names at this time. With around 800 occurrences, the name Aoife peaked in popularity in the late 1990s and early 2000s, making it one of the most popular female names at the time. This peak might be an indication of a strong cultural acceptance of Irish heritage and identity in naming customs. Over the next 20 years, the name's usage steadily declines after reaching its peak. Aoife's popularity steadily declined, despite the fact that it was still fairly common for a while. This suggests that younger generations are choosing more contemporary names. The rate drops below 200 by 2024, becoming closer to what it was before the 1990s. When naming trends shift, they tend to resemble the typical life cycle of a well-known name: early obscurity, a spike in popularity brought on by social or cultural favor, and then a drop. The survival of Aoife's peak indicates its cultural significance and strong identity throughout Irish society, despite the fact that more modern trends are still altering naming customs. This just an instance for the name "Aoife", any names that the user inputs can be analysed in the similar way, which makes this plot an comprehensive tool analyse the naming trends in Ireland.

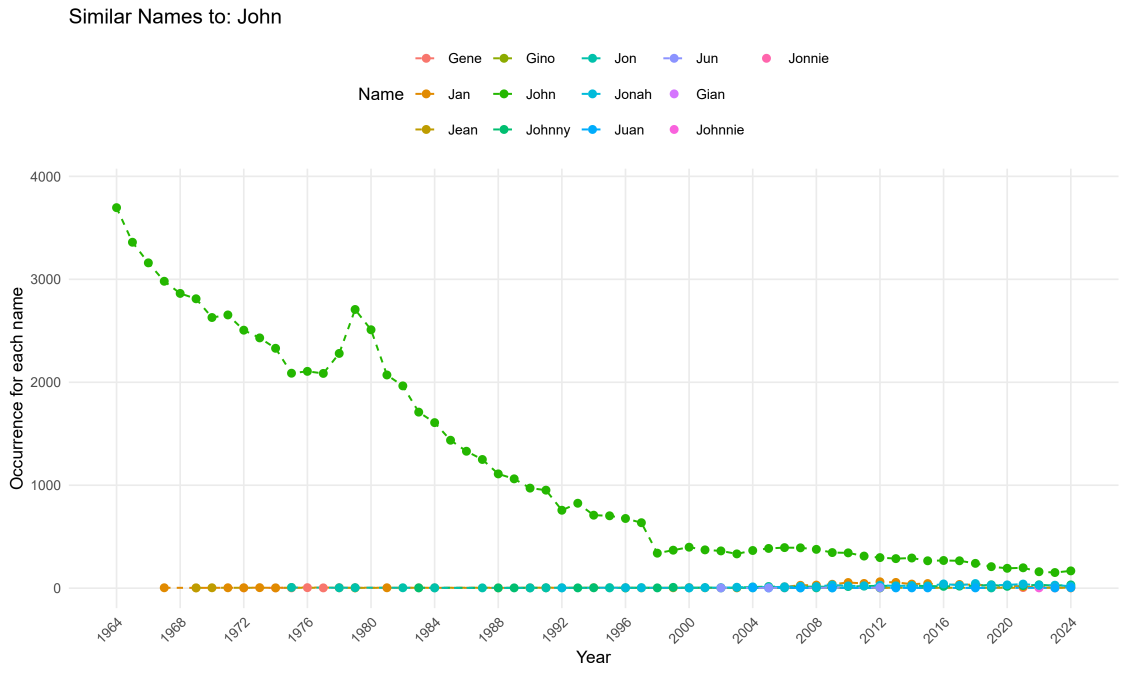
7.1.2 Phonetically similar plot for boys and girls

The plot of phonetically similar boys and girls names lets users investigate the historical trends in Ireland for names that sound similar, regardless of spelling. This plot is generated using Metaphone algorithm of phonics package (Howard, II 2021). Metaphone identifies the names with same metaphonic code as the user input name. For example, entering a name like “Cillian” may return similar sounding names such as “Killian” or “Kilian” or even “Cilian”. This method highlights how various spellings or variations of a name increase and decrease in tandem or divergence, allowing for a more thorough investigation of naming trends beyond precise matches.

This graph highlights the fact that while many names with identical phonetics exhibit parallel popularity trajectories, some diverge in size or time. For instance, a traditional Irish spelling like “Cillian” may have grown more significantly in recent years than an anglicized variation like “Killian.” These differences could be the consequence of cultural shifts, media influences, or regional preferences. Users may have a better understanding of how a name sounds, as well as how its spelling and presentation impact its progressive adoption in Irish culture, by seeing these variations. The function constructs an interactive ggplot2 object using ggiraph, where each line represents a phonetically similar name, and users can hover over data points to view tooltips showing the name, year, rank, and frequency.

The below code shows the output for the function of phonetically similar names for boys:

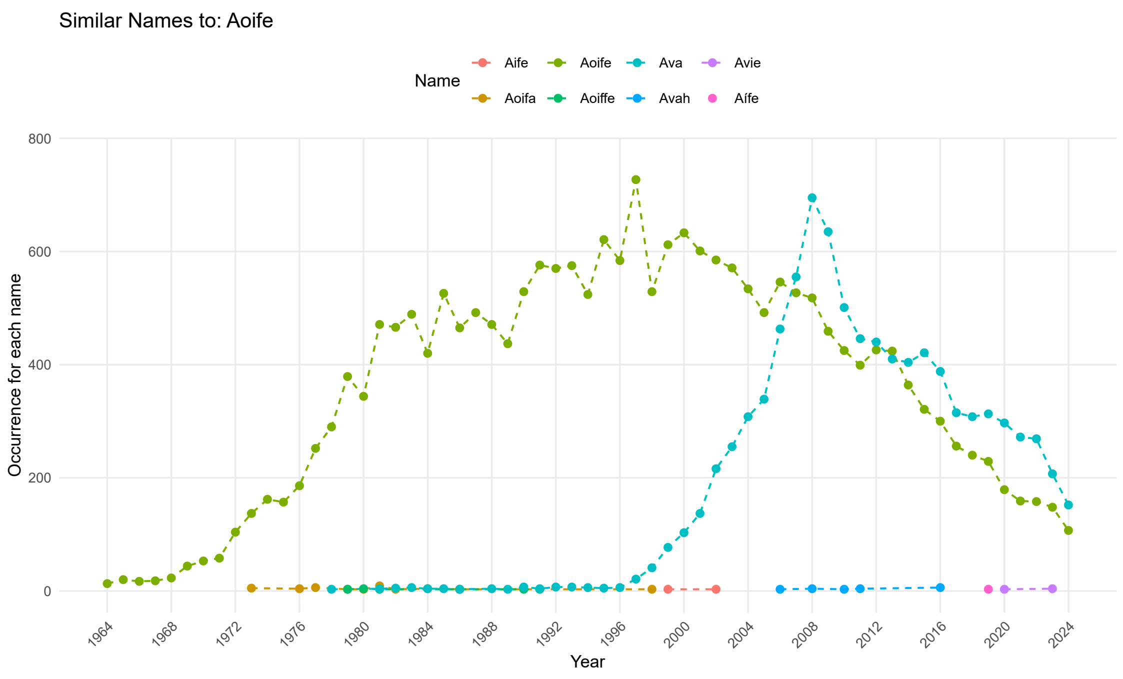
```
plot_similar_boysnames("John") #plots phonetically similar names to John from 1964-2024
```



This plot displays the popularity of phonetically similar names of “John”, measured as occurrences from 1964 to 2024. The green line makes it evident that John was by far the most prevalent of its phonetic variants during this time. John was one of the most common male names in the 1960s, with over 3,600 occurrences, making it one of the most popular names at the time. But throughout the years, its use gradually decreases, with only slight variations in the 1980s. It then continues to diminish into the 2000s and 2020s, leveling off below 500 occurrences. This pattern is indicative of a larger generational movement away from conventional names and toward more varied or contemporary choices. Despite their phonetic similarity to John, the majority of these names never surpass 40 occurrences, indicating that they never attained the same level of cultural significance in Ireland. Their foreign origin, specialized usage, or just a greater preference for the canonical form “John” could be the cause of this.

The below code shows the output for the function of phonetically similar names for girls :

```
plot_similar_girlsnames("Aoife") #plots phonetically similar names to Aoife from 1964-2024
```



This plot for phonetically similar names for girls, illustrates how names with identical sounds can have quite distinct paths to popularity. Ava reflects a contemporary rise, maybe influenced by international naming trends, whereas Aoife demonstrates traditional and consistent Irish usage. The image provides a nuanced perspective of Irish naming culture by revealing the phonetic variation and spelling preferences surrounding a core name. The choice to employ phonetic algorithms for more thorough, sound-based name exploration is further supported by the disparity in trends.

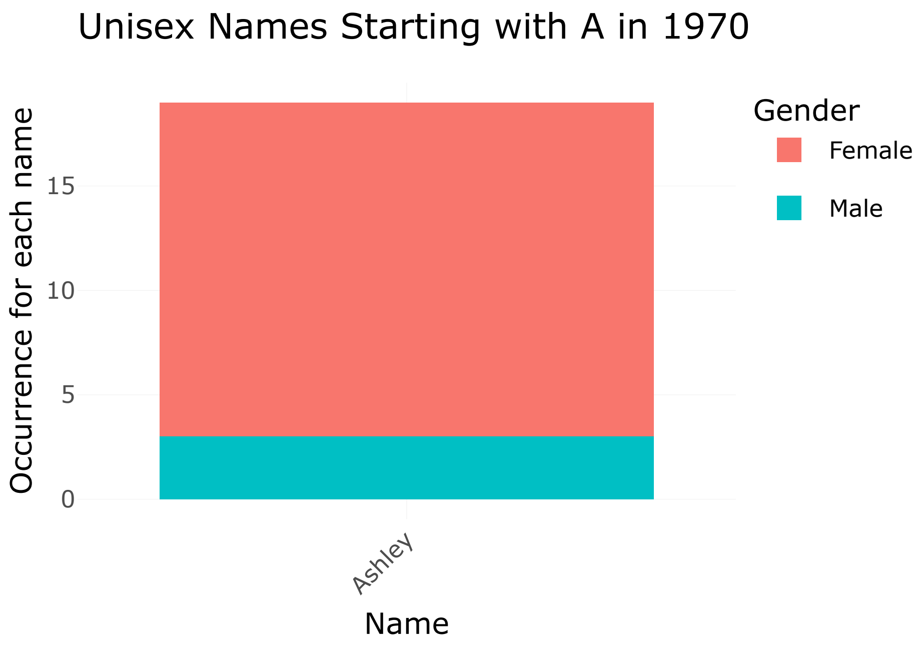
This plot can also be interpreted as a tool that also suggests the names to the user that sounds similar to the names user inputs. Using interactive visual tools (ggiraph) and the Metaphone algorithm, this visualization allows users to analyze phonetic naming clusters in detail. It supports the package’s goal of offering more than just raw name counts by allowing users to explore sound-based naming patterns that traditional exact-match analysis might miss.

7.1.3 Plot for Unisex names

The purpose of the unisex plot is to show names that begin with a specific letter and were given to both boys and girls in a chosen year. These names are displayed in the visualization as a stacked bar chart, where each bar stands for a name and colored segments shows the usage by gender for a particular year. The primary takeaway from the plot is that there aren’t many names that are actually gender-neutral. One gender uses most unisex names far more frequently than the other. A name like “Alex” might be used on both sides, for example, but it might be much more male oriented one year and more balanced or even female oriented the next. This suggests that a name's actual usage and cultural perception can be greatly influenced by naming trends or societal standards in a particular year, even if the name is theoretically deemed unisex.

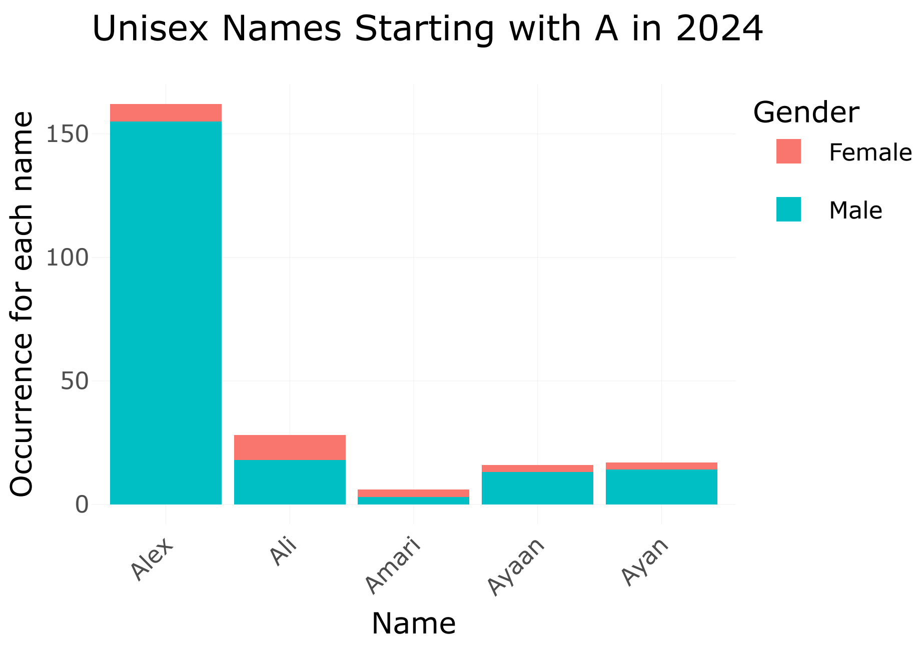
The below code shows the output for the function of unisex names for two different years:

```
plot_unisex_names("A","1970") #plots unisex names starting with A in 1970
```



The plot shows only one unisex name, Ashley, for the year 1970 starting with "A." Ashley was primarily used for girls, accounting for more than 75% of all instances, although it also occurs among boys rather frequently, which is consistent with its historical use as a unisex name in previous decades. The lower occurrence rate also highlights that fewer unisex names were in common use during 1970s, especially the names starting with "A."

```
plot_unisex_names("A","2024") #plots unisex names starting with A in 2024
```



This plot shows more than a few names names for the year 2024, starting with “A”. Alex is the most common name, occurring more than 100 times in the year 2024. Although there is a notable segment for girls, it is mainly used for guys, suggesting that it is unisex. All things considered, this plot parallels the current trend toward gender neutral or cross gender name adoption, particularly in light of the rise in popularity of multicultural or international names like Amari and Ayaan in recent years. Together, these plots shows the evolution of unisex naming across time. In 1970, there has only been a single unisex name , Ashley, and that too was dominated by one gender. By 2024, not only there is more unisex names in use , but their usage is more varied and gender-balanced.

Plotly’s interactivity and the stacking style let users hover over bars to view precise usage data for each gender. Overall, the unisex plot accomplishes its goals of providing a glimpse of gender fluidity in Irish naming traditions and exposing lesser-known, culturally intriguing names.

8 Conclusion and future work

The irishbabynames package significantly advances the study of naming patterns in Ireland by combining structured data, interactive visualizations, and phonetic analysis into an easy R toolkit. Using official data from the Central Statistics Office (CSO), the package collects, cleans, and standardizes infant name records spanning several decades. This makes it possible for users to analyze historical popularity, gender naming practices, and cultural preference shifts in a way that is both aesthetically pleasing and statistically significant.

Through a succession of well designed visualizations, the package provides users with a comprehensive grasp of naming patterns.The trend map highlights the historical popularity of specific names, helping to pinpoint cultural peaks and troughs. The phonetic similarity plots, which are rare in other national naming tools, provide a novel approach to exploring alternate spellings and sound-based name clusters through the use of the Metaphone algorithm. The visualization of unisex names sheds light on the progression of naming conventions in relation to gender neutrality, especially when examined across several decades. All of these visual components are flawlessly incorporated into a fully operational Shiny dashboard, allowing even non-technical users to enjoy a smooth and engaging experience.

The project has potential for further development despite its advantages. Currently, the package can be installed via github using: remotes::install_github("SuryenduManoj/irishbabynames", build_vignettes = TRUE). I would like to release it to CRAN, the Comprehensive R Archive Network, in the future. The package's visibility will improve, its long-term availability will be guaranteed, and contributions from the larger R community will be encouraged by posting it on CRAN. In order to comply with CRAN regulations, the package will go through extra testing before to release. These tests will include making sure it is cross-platform compatible, improving documentation, passing the R CMD check without any warnings or notes, and improving test coverage. By taking this step, the package would become a widely used instrument for demographic and cultural research rather than a stand-alone endeavor.

The dataset currently consists of names with 3 or more instances in the relevant year, this is due to confidentiality reasons. An additional feature to consider is adding a per million births metric, which will help to better understand name frequencies over time. This normalization of occurrences to a per million scale takes into account different birth rates across years and improves the clarity. However, the omission of 1,2 frequencies leads to an underestimation of actual total births. Using official birth data from the Central Statistics Office (CSO), the csodata::cso_get_data("VSQ01")(Horgan et al. 2024) dataset contains the total number of births per year, which allows to scale the data to per million births. This improvement could be added as an extra dataset and included in the plotting functions in a later release of the package.

References

Addokali, Belaid, and Ebrahem Elburase. 2022. "Using Levenshtein Distance Algorithm to Increase Database Search Efficiency and Accuracy" 10 (October): 1–7.

Awad, Abir, and Brian Lee. 2016. "A Metaphone Based Chaotic Searchable Encryption Algorithm for Border Management." In *ICETE 2016 - Proceedings of the 13th International Joint Conference on e-Business and Telecommunications*, edited by Christian Callegari, Marten van Sinderen, Enrique Cabello, Pierangela Samarati, Pascal Lorenz, Mohammad S. Obaidat, and Panagiotis Sarigiannidis, 397–402. ICETE 2016 - Proceedings of the 13th International Joint Conference on e-Business and Telecommunications. Portugal: SciTePress Digital Library. https://doi.org/10.5220/0005953503970402.

Çetinkaya-Rundel, Mine, Thomas J. Leeper, Nicholas Goguen-Compagnoni, and Sara Lemus. 2022. *Ukbabynames: UK Baby Names Data*. https://CRAN.R-project.org/package=ukbabynames.

Chang, Winston. 2021. *Shinythemes: Themes for Shiny*. https://CRAN.R-project.org/package=shinythemes.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2025. *Shiny: Web Application Framework for r*. https://CRAN.R-project.org/package=shiny.

Gohel, David, and Panagiotis Skintzos. 2025. *Ggiraph: Make 'Ggplot2' Graphics Interactive*. https://CRAN.R-project.org/package=ggiraph.

Horgan, Eoin, Conor Crowley, Vytas Vaiciulis, Mervyn O'Luing, and James O'Rourke. 2024. *Csodata: Download Data from the CSO 'PxStat' API*. https://CRAN.R-project.org/package=csodata.

Howard, II, James P. 2021. *phonics: Phonetic Spelling Algorithms in r*. https://doi.org/10.5281/zenodo.1041982.

Kasprzak, Peter, Lachlan Mitchell, Olena Kravchuk, and Andy Timmins. 2021. "Six Years of Shiny in Research - Collaborative Development of Web Tools in R." *The R Journal* 12 (2): 155–62. https://doi.org/10.32614/RJ-2021-004.

Koneru, Keerthi, Venkata Pulla, and Cihan Varol. 2016. "Performance Evaluation of Phonetic Matching Algorithms on English Words and Street Names - Comparison and Correlation." In, 57–64. https://doi.org/10.5220/0005926300570064.

Sievert, Carson. 2020. *Interactive Web-Based Data Visualization with r, Plotly, and Shiny*. Chapman; Hall/CRC. https://plotly-r.com.

Stefan, Müller. 2025. *Iebabynames: Ireland Baby Names, 1964-2024*. https://github.com/stefan-mueller/iebabynames.

van der Loo, M. P. J. 2014b. "The Stringdist Package for Approximate String Matching." *The R Journal* 6: 111–22. https://CRAN.R-project.org/package=stringdist.

———. 2014a. "The Stringdist Package for Approximate String Matching." *The R Journal* 6: 111–22. https://CRAN.R-project.org/package=stringdist.

Wattenberg, Martin. 2005."Baby Names, Visualization, and Social Data Analysis ." In *Information Visualization, IEEE Symposium on*, 1. Los Alamitos, CA, USA: IEEE Computer Society. https://doi.org/10.1109/INFOVIS.2005.7.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2021. *Babynames: US Baby Names 1880-2017*. https://CRAN.R-project.org/package=babynames.

Wickham, Hadley, Jennifer Bryan, Malcolm Barrett, and Andy Teucher. 2024. *Usethis: Automate Package and Project Setup.* https://CRAN.R-project.org/package=usethis.

Wickham, Hadley, Peter Danenberg, Gábor Csárdi, and Manuel Eugster. 2024. *Roxygen2: In-Line Documentation for r.* https://CRAN.R-project.org/package=roxygen2.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation.* https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, Jim Hester, Winston Chang, and Jennifer Bryan. 2022. *Devtools: Tools to Make Developing r Packages Easier.* https://CRAN.R-project.org/package=devtools.

Wickham, Hadley, Davis Vaughan, and Maximilian Girlich. 2024. *Tidyr: Tidy Messy Data.* https://CRAN.R-project.org/package=tidyr.

Wickham, H., and J. Bryan. 2023. *R Packages.* O'Reilly Media. https://books.google.ie/books?id=lTHFEAAAQBAJ.

Appendix

A.1 Package functions

These are the functions used int the package:

A.1.1 plot_trend(name_input)

Plots the popularity of an individual name for 1964 to 2024.

```r
plot_trend <- function(name_input) {
  occurences <- irishbabynames %>%
    pivot_longer(
      cols = matches("^[0-9]{4}$"),
      names_to = "Year",
      values_to = "Value"
    ) %>%
    mutate(
      Year = as.numeric(Year),
      Value = as.numeric(Value)
    ) %>%
    filter(!is.na(Value), is.finite(Value))

  counts <- occurences %>%
    filter(Statistic %in% c(
      "Girls Names in Ireland with 3 or More Occurrences",
      "Boys Names in Ireland with 3 or More Occurrences"
    ))
  ranks <- occurences %>%
    filter(Statistic %in% c(
      "Girls Names in Ireland with 3 or More Occurrences Rank",
      "Boys Names in Ireland with 3 or More Occurrences Rank"
    )) %>%
    rename(Rank = Value) %>%
    select(Name, Gender, Year, Rank)

  data_names <- counts %>%
    filter(tolower(Name) == tolower(name_input)) %>%
    left_join(ranks, by = c("Name", "Gender", "Year"))

  if (nrow(data_names) == 0) {
    return(girafe(ggobj = ggplot() + labs(title = paste("Name", name_input, "not found."))))
}
  data_names <- data_names %>%
    mutate(tooltip = paste0(
      "Name: ", Name,"<br>Occurrence in ",Year,": ",Value,
      "<br>Rank:", ifelse(is.na(Rank), "Not Ranked", Rank)
    ))
  p <- ggplot(data_names, aes(x = Year, y = Value, color = Gender)) +
    geom_line_interactive(aes(data_id = Gender, tooltip = tooltip, group = Gender), size = .6
    geom_point_interactive(aes(data_id = Gender, tooltip = tooltip), size = 3) +
    scale_x_continuous(
      breaks = seq(min(data_names$Year, na.rm = TRUE), 2024,4) # every 4 years, include 2024
    ) +
    scale_y_continuous(limits = c(0, max(data_names$Value, na.rm = TRUE)*1.05)) +
    labs(
      title = paste("Trend for Name:", name_input),
      x = "Year",
      y = "Occurrence for each name",
      color = "Gender"
    ) +
    theme_minimal() +
    theme(
      legend.position = "top",
      axis.text.x = element_text(angle = 45, hjust = 1)
    )

  girafe(ggobj = p, width_svg = 10, height_svg = 6)
}
```

A.1.2 plot_similar_names(input_name, gender_label, gender_stat, gender_rank_stat)

Function for phonetically similar boys' names and girls names act as a wrapper function for plot_similar_names.

```
plot_similar_names <- function(input_name, gender_label, gender_stat, gender_rank_stat) {
  # Filter for gender
  filtered <- irishbabynames %>%
    filter(Statistic == gender_stat)
  # Reshape and clean
  names_long <- filtered %>%
    pivot_longer(
      cols = matches("^[0-9]{4}$"),
      names_to = "Year",
      values_to = "Value"
    ) %>%
    mutate(
      Year = as.numeric(Year),
      Value = as.numeric(Value),
      Gender = gender_label
    ) %>%
    select(Name, Year, Value, Gender) %>%
```

```r
  filter(!is.na(Value), is.finite(Value))
# Get ranks
ranks <- irishbabynames %>%
  filter(Statistic %in% c(
    "Boys Names in Ireland with 3 or More Occurrences Rank",
    "Girls Names in Ireland with 3 or More Occurrences Rank"
  )) %>%
  pivot_longer(
    cols = matches("^[0-9]{4}$"),
    names_to = "Year",
    values_to = "Rank"
  ) %>%
  mutate(Year = as.numeric(Year)) %>%
  select(Name, Gender, Year, Rank)
# Clean and apply Metaphone
unique_names <- names_long %>% distinct(Name) %>% pull(Name)
clean_names <- stri_trans_general(unique_names, "Latin-ASCII")
clean_names <- gsub("[^A-Za-z]", "", clean_names)
clean_input <- stri_trans_general(input_name, "Latin-ASCII")
clean_input <- gsub("[^A-Za-z]", "", clean_input)
name_codes <- metaphone(tolower(clean_names))
input_code <- metaphone(tolower(clean_input))
similar_names <- unique_names[name_codes == input_code]
if (length(similar_names) == 0) {
  stop("No phonetically similar names found.")
}
# Merge and prepare
df <- names_long %>%
  filter(Name %in% similar_names) %>%
  left_join(ranks, by = c("Name", "Gender", "Year"))

if (nrow(df) == 0) {
  stop("No similar names with enough data to plot.")
}
# Tooltip
df <- df %>%
  mutate(tooltip = paste0(
    "Name: ", Name,
    "<br>Year: ", Year,
    "<br>Occurrence: ", Value,
    "<br>Rank: ", ifelse(is.na(Rank), "Not Ranked", Rank)
  )) %>%
  arrange(Name, Year)
df_lines <- df %>%
  group_by(Name) %>%
  filter(n() > 1) %>%
    ungroup()
  # Plot
  p <- ggplot() +
    geom_line_interactive(
      data = df_lines,
      aes(x = Year, y = Value, color = Name, tooltip = tooltip, data_id = Name, group = Name)
      size = 0.6, linetype = "dashed"
    ) +
    geom_point_interactive(
      data = df,
      aes(x = Year, y = Value, color = Name, tooltip = tooltip, data_id = Name),
      size = 2
    ) +
    scale_x_continuous(
      breaks = seq(min(df$Year, na.rm = TRUE), 2024, 4) # every 4 years, include 2024
    ) +
    scale_y_continuous(
      limits = c(0, max(df$Value, na.rm = TRUE) * 1.05) # buffer for readability
    ) +
    labs(
      title = paste("Similar Names to:", input_name),
      x = "Year",
      y = "Occurrence for each name",
      color = "Name"
    ) +
    theme_minimal() +
    theme(
      legend.position = "top",
      panel.grid.minor = element_blank(),
      axis.text.x = element_text(angle = 45, hjust = 1)
    )

  girafe(ggobj = p, width_svg = 10, height_svg = 6)
}
```

A.1.3 plot_unisex_names(start, year)

Visualises unisex names that start with a user given letter for a specific year

```
plot_unisex_names <- function(start, year) {
  # Step 1: Filter only valid gender stats
  unisex_data <- irishbabynames %>%
    filter(Statistic %in% c(
      "Boys Names in Ireland with 3 or More Occurrences",
      "Girls Names in Ireland with 3 or More Occurrences"
    ))
  # Step 2: Reshape the data
  long_data <- unisex_data %>%
    pivot_longer(cols = matches("^[0-9]{4}$"), names_to = "Year", values_to = "Value") %>%
    mutate(
      Year = as.numeric(Year),
      Value = as.numeric(Value)
    ) %>%
    filter(Year == year, !is.na(Value), is.finite(Value))
  # Step 3: Identify names used by both genders in that year
  names_with_both <- long_data %>%
    group_by(Name) %>%
    summarise(genders_present = n_distinct(Gender), .groups = "drop") %>%
    filter(genders_present == 2) %>%
    pull(Name)
  # Step 4: Filter names starting with the letter & having both genders
  filtered <- long_data %>%
    filter(Name %in% names_with_both,
           startsWith(tolower(Name), tolower(start)))

  if (nrow(filtered) == 0) {
    return(ggplotly(ggplot() +
                      labs(title = paste("No unisex names starting with", start, "in", year))
  }
  # Step 4: Plot
  p <- ggplot(filtered, aes(x = Name, y = Value, fill = Gender,
                             text = paste0("Name: ", Name,
                                           "<br>Gender: ", Gender,
                                           "<br>Occurrences: ", Value)
  )) +
    geom_col(position = "stack") +
    labs(
      title = paste("Unisex Names Starting with", toupper(start), "in", year),
      x = "Name", y = "Occurrence for each name", fill = "Gender"
    ) +
    scale_y_continuous(labels = scales::label_comma()) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  ggplotly(p, tooltip = "text")
}
```

A.1.4 irishbabynames_dashboard()

Launches the dashboard for the irishbabynames package

```
ui <- fluidPage(
  theme = shinytheme("flatly"), #theme for polished modern look
  titlePanel(
    div(
      h1("Irish Baby Names Dashboard", style = "color:#2c3e50"), #Main heading
      p("Explore trends and patterns in Irish baby names from 1964-2024.", style = "font-size
    )
  ),
  navbarPage("Irish Baby Names Dashboard",  #navigation tab
    tabPanel("Trend by Name",  #first tab
             sidebarLayout(
               sidebarPanel(
                 wellPanel(
                   tags$hr(),
                   icon("baby"), strong(" Enter Name"),
                   textInput("trend_name", "Name:", value = "Alex"),
                   tags$hr()
                 )
               ),
               mainPanel(        # area where lot is visualised
                 tags$h3("Trend Over the Years"),
                 br(),
                 girafeOutput("trend_plot")
               )
             )
    ),
    tabPanel(title = tagList(icon("user-friends"), "Similar Names"),
             sidebarLayout(
               sidebarPanel(
                 tags$h4("Find Similar Names"),
                 textInput("name", "Enter a Name:", value = ""),
                 pickerInput("gender", "Gender:", choices = c("Girls", "Boys"), multiple = FA
                 helpText("Matches based on phonetic similarity.")
               ),
               mainPanel(
                 tags$h4("Phonetically Similar Names"),
                 girafeOutput("similar_plot")
               )
             )
    ),
    tabPanel(title = tagList(icon("genderless"), "Unisex Names"),
            sidebarLayout(
              sidebarPanel(
                tags$h4("Explore Unisex Names"),
                textInput("unisex_start", "Enter Starting Letter:", value = ""),
                selectInput("year", "Select Year:", choices = 1964:2024, selected = 2024),
```

```r
                helpText("Filters unisex names starting with the chosen letter.")
              ),
              mainPanel(
                tags$h4("Unisex Names Distribution"),
                plotlyOutput("bar_plot")
              )
            )
    )
  ),
  tags$hr(),
  tags$footer(
    tags$p(
      "© 2025 IrishBabyNames Package | Built with Shiny",
      style = "text-align:center; color: #777; padding: 10px;"
    )
  )
)
server <- function(input, output, session) {  #collects user inputs
  output$trend_plot <- renderGirafe({
    plot_trend(input$trend_name)  #plots the function
  })
  output$similar_plot <- renderGirafe({
    req(input$name)

    if (input$gender == "Girls") {
      plot_similar_girlsnames(
        input_girls_name  = input$name,
        max_distance = 1
      )
    } else {
      plot_similar_boysnames(
        input_boys_name = input$name,
        max_distance = 1
      )
    }
  })
  output$bar_plot <- renderPlotly({
    req(input$unisex_start)
    plot_unisex_names(input$unisex_start, as.numeric(input$year))
  })
}
shinyApp(ui, server)  #starts the app connecting ui and server
```

A.2 Shiny dashboard

First tab displays trend plot where the user can enter a name on the left side and plot is displayed on the right side

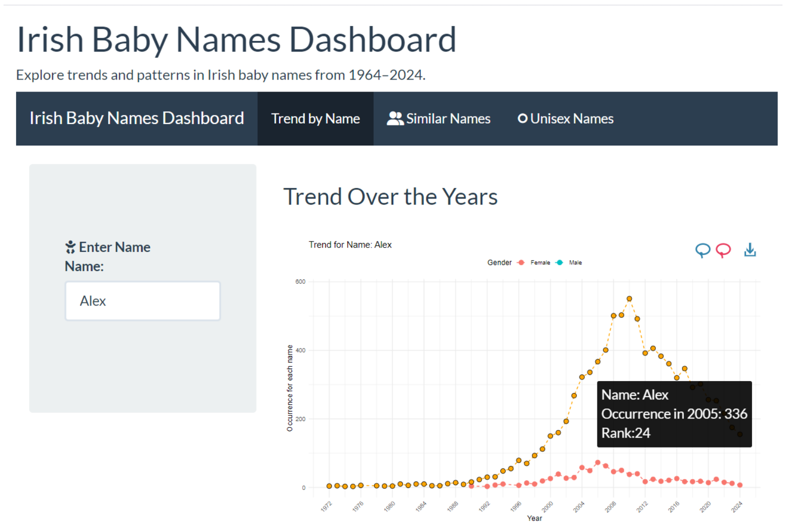


Figure 5: trend plot in shiny dashboard

Second tab displays phonetically similar plot where the user can filter boys/girls and enter a name on the left side and plot is displayed on the right side

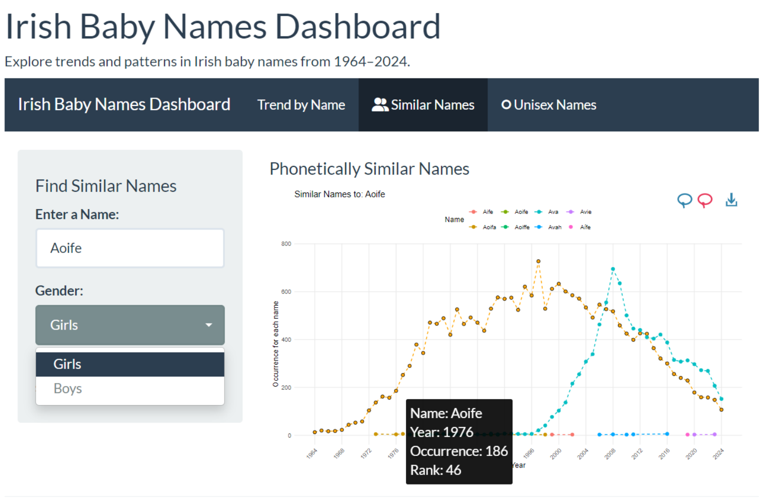


Figure 6: phonetically similar plot in shiny dashboard, with the option to filter girls/boys

Third tab displays unisex names plot where the user can enter a name and choose a year from the drop-down menu on the left side and plot is displayed on the right side

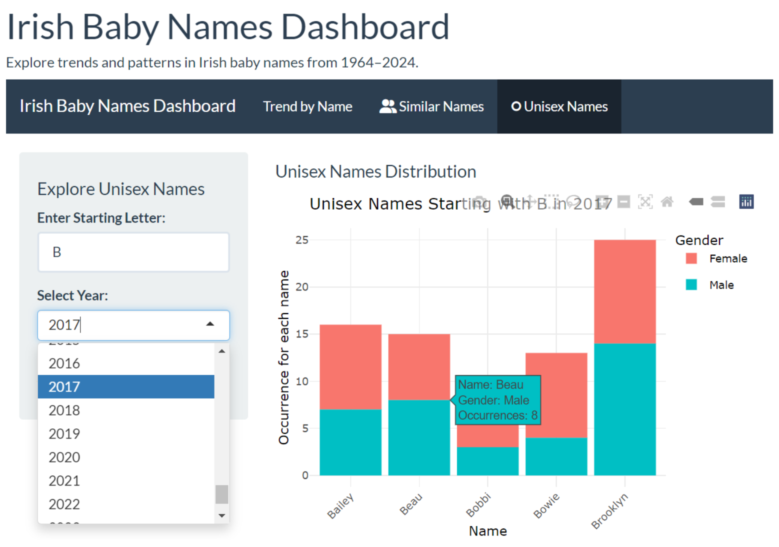


Figure 7: Unisex names plot in shiny dashboard with the option to select a year from the drop-down list

A.3 Package Documentation

This documentation is generated written in roxygen2 format. The document produced is used to generate a help page for the package, calling ?irishbabynames produces this documentation

```
#' Irish Baby Names
#'
#' A dataset of baby names in Ireland from the Central Statistics Office (CSO),
#' including both boys' and girls' names with occurrences per year.
#'
#' @docType data
#' @name irishbabynames
#' @format A data frame with columns:
#' \describe{
#'   \item{Name}{First name}
#'   \item{Gender}{Gender ("Male" or "Female")}
#'   \item{Statistic}{Type of statistic (e.g., counts or ranks)}
#'   \item{1964}{Number of occurrences and rank in 1964}
#'   \item{1965}{Number of occurrences and rank in 1965}
#'   \item{1966}{Number of occurrences and rank in 1966}
#'   \item{1967}{Number of occurrences and rank in 1967}
#'   \item{1968}{Number of occurrences and rank in 1968}
#'   \item{1969}{Number of occurrences and rank in 1969}
#'   \item{1970}{Number of occurrences and rank in 1970}
#'   \item{1971}{Number of occurrences and rank in 1971}
#'   \item{1972}{Number of occurrences and rank in 1972}
#'   \item{1973}{Number of occurrences and rank in 1973}
#'   \item{1974}{Number of occurrences and rank in 1974}
#'   \item{1975}{Number of occurrences and rank in 1975}
#'   \item{1976}{Number of occurrences and rank in 1976}
#'    \item{1977}{Number of occurrences and rank in 1977}
#'    \item{1978}{Number of occurrences and rank in 1978}
#'    \item{1979}{Number of occurrences and rank in 1979}
#'    \item{1980}{Number of occurrences and rank in 1980}
#'    \item{1981}{Number of occurrences and rank in 1981}
#'    \item{1982}{Number of occurrences and rank in 1982}
#'    \item{1983}{Number of occurrences and rank in 1983}
#'    \item{1984}{Number of occurrences and rank in 1984}
#'    \item{1985}{Number of occurrences and rank in 1985}
#'    \item{1986}{Number of occurrences and rank in 1986}
#'    \item{1987}{Number of occurrences and rank in 1987}
#'    \item{1988}{Number of occurrences and rank in 1988}
#'    \item{1989}{Number of occurrences and rank in 1989}
#'    \item{1990}{Number of occurrences and rank in 1990}
#'    \item{1991}{Number of occurrences and rank in 1991}
#'    \item{1992}{Number of occurrences and rank in 1992}
#'    \item{1993}{Number of occurrences and rank in 1993}
#'    \item{1994}{Number of occurrences and rank in 1994}
#'    \item{1995}{Number of occurrences and rank in 1995}
#'    \item{1996}{Number of occurrences and rank in 1996}
#'    \item{1997}{Number of occurrences and rank in 1997}
#'    \item{1998}{Number of occurrences and rank in 1998}
#'    \item{1999}{Number of occurrences and rank in 1999}
#'    \item{2000}{Number of occurrences and rank in 2000}
#'    \item{2001}{Number of occurrences and rank in 2001}
#'    \item{2002}{Number of occurrences and rank in 2002}
#'    \item{2003}{Number of occurrences and rank in 2003}
#'    \item{2004}{Number of occurrences and rank in 2004}
#'    \item{2005}{Number of occurrences and rank in 2005}
#'    \item{2006}{Number of occurrences and rank in 2006}
#'    \item{2007}{Number of occurrences and rank in 2007}
#'    \item{2008}{Number of occurrences and rank in 2008}
#'    \item{2009}{Number of occurrences and rank in 2009}
#'    \item{2010}{Number of occurrences and rank in 2010}
#'    \item{2011}{Number of occurrences and rank in 2011}
#'    \item{2012}{Number of occurrences and rank in 2012}
#'    \item{2013}{Number of occurrences and rank in 2013}
#'    \item{2014}{Number of occurrences and rank in 2014}
#'    \item{2015}{Number of occurrences and rank in 2015}
#'    \item{2016}{Number of occurrences and rank in 2016}
#'    \item{2017}{Number of occurrences and rank in 2017}
#'    \item{2018}{Number of occurrences and rank in 2018}
#'    \item{2019}{Number of occurrences and rank in 2019}
#'    \item{2020}{Number of occurrences and rank in 2020}
#'    \item{2021}{Number of occurrences and rank in 2021}
#'    \item{2022}{Number of occurrences and rank in 2022}
#'   \item{2023}{Number of occurrences and rank in 2023}
#'   \item{2024}{Number of occurrences and rank in 2024}
#' }
#' @source \url{https://www.cso.ie/en/}
"irishbabynames"
```