SONICPLOTS: AN APPLICATION FOR ENHANCING DATA ACCESSIBILITY AND PATTERN RECOGNITION FOR VISUALLY IMPAIRED AND SIGHTED USERS

University *of* Exeter

Submitted by Luke Williams to the University of Exeter as a dissertation thesis for the degree of Master of Science in August 2024

I certify that all material in this thesis which is not my own work has been identified and that any material that has been previously submitted and approved for the award of degree by this or any other University has been acknowledged.

Signature: 

Contents

1 Abstract

In a world increasingly shaped by data, traditional methods of data visualisation often exclude visually impaired individuals, limiting their ability to engage with and interpret complex data. This thesis presents the development and evaluation of SonicPlots, an R-based application designed to convert numerical data into auditory signals. Built on the Shiny framework, SonicPlots enables users to explore and analyse data through sound, complementing traditional visualisations with auditory cues.

The project began with the creation of a basic prototype, version 1.0, which provided fundamental sonification capabilities but faced significant challenges related to limited features, performance issues, a confusing user interface, and no data modelling. User feedback collected through surveys and usability testing was instrumental in guiding the development of version 2.0, which introduced substantial improvements, including improved visualisation options, a redesigned UI, and new customisation features for sonification.

The methodology shown in this thesis involved iterative development, rigorous testing, and user-centred design principles, ensuring that SonicPlots evolved to meet the diverse needs of its users. The evaluation of SonicPlots, through both functional testing and user feedback, demonstrated significant improvements in performance and usability, with the average run time decreasing from **2.52** seconds to **0.98** seconds between versions. Additionally, the tool was well received for its enhanced accessibility features and the expanded range of data visualisation and modelling options, resulting in a rise in user satisfaction from **43%** to **87%**.

SonicPlots offers itself as a versatile and accessible tool for both visually impaired and sighted users. This thesis concludes with a discussion of future enhancements, including the integration of more advanced modelling techniques, improved audio controls, and further accessibility upgrades, presenting SonicPlots as a comprehensive tool for data sonification and analysis.

2 Introduction

In the realm of data analysis, visualisation has long been a dominant approach, allowing researchers and analysts to represent complex data sets through graphs, charts, and other visual means. However, traditional visualisations have limitations, particularly in their accessibility to visually impaired individuals. Moreover, even for sighted users, visual representations can sometimes fail to reveal intricate patterns or anomalies, especially when dealing with large or multidimensional data. This challenge has driven the exploration of alternative methods for data interpretation, with sonification emerging as a promising solution. Sonification, the process of converting data into sound, leverages the human auditory system's ability to detect subtle changes in pitch, rhythm, and volume, offering a unique and powerful way to explore and understand data.

This thesis presents SonicPlots, an application designed to enhance data accessibility and pattern recognition for both visually impaired and sighted users through data sonification. Developed using the R programming language, SonicPlots is built on the Shiny framework, integrating a range of features that allow users to upload data sets, customise sonification parameters, and interact with the data in real

time. The application supports various types of data visualisation, including time series plots and scatter plots, and incorporates advanced data modelling techniques, such as ARIMA model fitting, to offer a comprehensive tool for data analysis.

The development of SonicPlots was driven by the need to create a tool that is not only functional but also accessible and user-friendly. From its inception, the application has undergone multiple iterations, each informed by user feedback and rigorous testing. This iterative approach has allowed for the continuous refinement of the tool, ensuring that it meets the diverse needs of its users.

The following sections of this thesis will detail the different phases of SonicPlots' development, from the initial version 1.0 prototype to the current version, highlighting the challenges encountered and the improvements made along the way. The thesis will also explore the methodology used in data collection and processing, the technical aspects of the sonification process, and the software development techniques employed. Finally, the evaluation of the tool through user feedback and comparison with existing sonification solutions will be discussed, providing insights into its effectiveness and areas for future enhancement.

3 Literary Review

3.1 Significance

Data sonification is the process of converting data into non-speech audio signals to represent and interpret information. Using the human auditory system's capability to detect patterns and anomalies as the foundation, sonification transforms numerical data sets into meaningful auditory experiences. This technique involves mapping data variables to auditory parameters such as pitch, volume, and timbre, allowing listeners to interpret complex data through sound.

The significance of data sonification lies in its ability to make complex data more accessible, especially for visually impaired individuals. Traditional data visualisation methods, such as graphs and charts, rely heavily on visual perception, which can exclude those with visual impairments from effectively interpreting data. Sonification provides an alternative by engaging the auditory senses, allowing visually impaired users to perceive and analyse data through sound. This inclusivity promotes a broader understanding and use of data across diverse user groups.

Moreover, data sonification can reveal insights that might not be as apparent through visual means. The human ear is adept at distinguishing subtle variations in sound, which can make it easier to detect patterns and anomalies in data. For example, changes in data trends can be represented by shifts in pitch, while the intensity of data points can be conveyed through volume variations. This auditory representation can provide a different perspective on data, complementing visual analysis and enhancing overall data interpretation.

3.2 Historical Context

The concept of data sonification has evolved significantly over the years. Early work in the field focused on scientific and medical applications, where researchers sought ways to interpret complex data sets

using sound. One of the notable milestones in the development of sonification was the introduction of auditory display systems in the 1980s and 1990s, which paved the way for more sophisticated and user-friendly sonification tools.

Key figures in the field, such as Greg Kramer, played a crucial role in advancing the understanding and application of auditory displays. Kramer's work in the 1990s on auditory scene analysis provided foundational insights into how sound can effectively convey information (Kramer & Bregman, 1992). His research demonstrated that well-designed auditory cues could significantly enhance the usability of sonification systems, making them more intuitive and effective for users.

Over time, the field has expanded to include various domains such as environmental monitoring, financial data analysis, and accessibility tools for the visually impaired. The integration of advanced technologies, including machine learning and interactive interfaces, has further enriched the capabilities of data sonification, making it a valuable tool for contemporary data science and analytics.

Data sonification is a powerful technique that transforms data into sound, making it accessible and interpretable for a wider audience. Its historical development and the contributions of key researchers have established a solid foundation for its ongoing evolution and application in diverse fields.

3.3 Existing Solutions

Highcharts Sonification

Highcharts Sonification is an extension of the popular Highcharts visualisation library, enabling the conversion of chart data into sound. This tool integrates seamlessly with existing HighCharts visualisations, making it convenient for users familiar with the HighCharts interface. It supports various sound parameters and allows basic interaction with the auditory output. However, Highcharts Sonification is somewhat limited in its capability to handle large data sets and complex auditory mappings. The sound synthesis options are relatively basic, which may not be sufficient for detailed and nuanced data sonification. In addition, it lacks advanced features for real-time data interaction (Highcharts, 2021).

Sonification Sandbox

Sonification Sandbox is a tool developed to assist researchers and educators in creating sonified representations of data. It offers a range of functionalities, including the ability to customise sound mappings extensively. Users can adjust parameters such as pitch, tempo, and instrument type, providing a high degree of control over the auditory output. Despite its robust feature set, the Sonification Sandbox can be challenging for users without a background in sound design or data sonification. Its user interface, while powerful, is not as intuitive as some of the more basic tools, potentially creating a steep learning curve for new users (GVU Center at Georgia Tech. Sonification Soundbox).

Comparison of Tools

When comparing the strengths and weaknesses of these tools, several factors emerge. HighCharts Sonification is highly accessible and easy to use, making it ideal for beginners and those requiring quick and simple sonification. However, it falls short in terms of flexibility and advanced customisation. On the other hand, the Sonification Sandbox offers extensive customisation and advanced features but can be

more challenging to use without specialised knowledge.

A common limitation across these tools is the lack of integration with advanced data models, such as ARMA/ARIMA, Exponential Smoothing (ETS) and Seasonal Decomposition of Time Series (STL) which are crucial for modelling and predicting complex time series data. Additionally, user-friendliness remains an issue, particularly for more advanced tools like the Sonification Sandbox, which can intimidate users without a background in music or data science. Improving the balance between ease of use and advanced functionality is essential for the broader adoption of sonification tools.

There is a noticeable lack of graphing options and data models accompanying the sonification across various toolkits, which can aid in a more comprehensive understanding of the data. None of the listed toolkits are fully comprehensive and often lack one or more mentioned features. Furthermore, some toolkits are primarily focused on creating music for artistic purposes rather than analysing data, which is not suited to producing key insights into intricate data sets.

3.4 Theoretical Foundations

Auditory Scene Analysis (Kramer & Bregman, 1992)

Albert Bregman's work on Auditory Scene Analysis (ASA) provides a fundamental framework for understanding how humans perceive complex auditory environments. ASA describes the process by which the auditory system organises sound into perceptually meaningful elements, similar to how the visual system processes visual scenes. The principles of ASA are highly relevant to sonification as they help in designing auditory displays that are intuitive and easy to interpret. For instance, Bregman identified that sounds with similar spectral and temporal properties are grouped together, while those that differ are perceived as separate streams. This principle can be applied in sonification by ensuring that data points that should be perceived as related share common auditory attributes, such as pitch or timbre, in turn facilitating a clearer and more intuitive understanding of the data (Kramer & Bregman, 1992).

Geiger Counter (Geiger, 1913)

One of the earliest and most iconic examples of sonification is the Geiger counter, a device that detects and measures ionising radiation. Developed by Hans Geiger in the early 20th century, the Geiger counter converts radiation levels into auditory clicks, providing an intuitive way to monitor radiation intensity. Each click corresponds to a detected ionising event, with the frequency of clicks increasing as radiation levels rise. This auditory feedback allows users to instantly perceive changes in radiation levels without needing to inspect a meter visually. The Geiger counter's use of sound to convey critical information exemplifies the power of sonification in situations where rapid, real-time monitoring is essential. The device has been widely used in fields ranging from nuclear physics to environmental monitoring, demonstrating the enduring utility of auditory data representation in various scientific and safety contexts (Geiger, 1913; Walden / Strickland, 2012).

Earcons (Brewster et al., 1994)

The concept of earcons, introduced by Stephen Brewster and colleagues, refers to non-speech audio cues used to provide information in a structured and hierarchical manner. Earcons are particularly effective

for representing categorical data and improving the user interface of sonification systems. Brewster's research demonstrated that well-designed earcons could significantly improve the usability of auditory displays by making them more intuitive and reducing cognitive load. For example, different pitches and rhythms can represent different types of data, while variations in volume can indicate data intensity. The effectiveness of earcons lies in their ability to convey complex information quickly and efficiently, which is crucial in applications where rapid data interpretation is needed (Brewster et al., 1994).

3.5 Applications

Bellido-Tirado et al., 2010

Data sonification has been effectively used in scientific research to facilitate data exploration and analysis. One notable application is in the field of astronomy, where sonification helps researchers analyse large data sets, such as the data collected from telescopes. For example, the "Asteroseismology and Sonification of Stellar Data" project transforms the oscillations of stars into sound, allowing astronomers to "listen" to star quakes and gain insights into stellar structures and dynamics (Bellido-Tirado et al., 2010). This auditory approach provides an additional layer of data interpretation, complementing traditional visual analysis and helping researchers identify patterns that might be overlooked visually.

Fitch & Kramer, 1994

In the medical field, sonification aids in diagnostics and patient monitoring by converting vital signs and other medical data into auditory signals. For instance, continuous auditory monitoring of a patient's heart rate and oxygen levels can alert healthcare providers to critical changes in a patient's condition more rapidly than visual monitors alone. The "Sonification of Medical Data" project demonstrated that using auditory cues for real-time monitoring in intensive care units could enhance the detection of physiological changes and improve response times (Fitch & Kramer, 1994). This application highlights the potential of sonification to improve patient outcomes through more immediate and intuitive data interpretation.

Bergson & Abramowitz, 2016

Environmental studies benefit from sonification by providing new ways to analyse and interpret complex environmental data. Climate scientists use sonification to study phenomena such as global temperature changes, pollution levels, and biodiversity metrics. For example, the "Sonification of Climate Data" project converted historical temperature data into sound, allowing listeners to perceive the dramatic increase in global temperatures over time (Bergson & Abramowitz, 2016). This auditory representation makes the data more accessible to the general public, raising awareness about climate change and its impacts.

Baier, Hermann, & Stephani, 2007

One successful project is "Listening to the Mind Listening," which transformed EEG data into musical compositions to help researchers and the public understand brain activity patterns. The project used various sonification techniques to map different brain wave frequencies to musical elements, creating an engaging and informative auditory experience (Baier, Hermann, & Stephani, 2007). The results of

this project demonstrated that sonification could effectively communicate complex scientific data to both experts and lay audiences.

Human auditory perception plays a critical role in the design of sonification systems. The psychological and cognitive processes involved in hearing and interpreting sounds must be considered to create effective sonification. Research has shown that humans are particularly good at detecting changes in pitch, rhythm, and volume, which can be used to represent dynamic data changes. Theories such as selective attention, which describes how listeners can focus on a particular sound source amidst competing sounds, are vital for designing sonification processes that highlight important data points while minimising background noise.

Additionally, cognitive load theory suggests that auditory displays should avoid overloading the listener with too much information at once. Instead, the sonification process should be designed to deliver information in manageable chunks, allowing users to process data efficiently. Understanding how users mentally map sounds to data points is also essential; for instance, rising pitch can intuitively represent increasing values, making data interpretation more natural.

3.6 Design and Evaluation

User-centred design (UCD) is critical in the development of effective sonification tools. By involving users throughout the design process, developers can ensure that the tool meets the actual needs and preferences of its intended audience. This approach not only enhances usability but also improves the overall effectiveness of the sonification system. UCD involves iterative cycles of design, testing, and refinement, incorporating user feedback at each stage. This iterative process helps identify and address potential usability issues early, ensuring that the final product is intuitive and user-friendly (Norman, 2013).

Evaluating the effectiveness of sonification tools requires a combination of qualitative and quantitative methods. User studies are a common approach, in which participants interact with the sonification tool and provide feedback on their experience. Usability testing focuses on how easily users can navigate the tool and understand the sonified data. Performance metrics, such as task completion time, error rates, and user satisfaction scores, provide objective measures of the tool's efficiency and effectiveness. Additionally, cognitive load assessments can be used to determine how much mental effort is required to use the tool, ensuring that it does not overwhelm users (Preece et al., 2015).

Designing intuitive and effective sonification systems involves adhering to several key guidelines. First, simplicity is crucial. The sonification output should not be too complex as this can confuse users and hinder data interpretation. Second, consistency in mapping data to sound parameters helps users develop an understanding of auditory cues. Third, flexibility is important; allowing users to customise the sonification mappings can cater to individual preferences and needs. Lastly, feedback mechanisms, such as real-time auditory feedback, can enhance user interaction and provide immediate insights into data changes (Walker & Nees, 2011).

User feedback plays a vital role in refining and improving the application's design. Regularly collecting and analysing user feedback helps identify areas of the tool that require improvement. This feedback

can highlight usability issues, suggest new features, and provide insight into how users interact with the tool. Incorporating user suggestions into subsequent design iterations ensures that the tool evolves to better meet user needs and expectations (Nielsen, 1994).

Designing and evaluating sonification systems presents several challenges. One major challenge is ensuring that the sonified data is intuitively understandable without extensive training. There is difficulty in balancing the amount of information conveyed through sound. Too much information can overwhelm users, while too little information can lead to misinterpretation (Hermann, Hunt, & Neuhoff, 2011).

To overcome these challenges, developers often employ strategies such as multimodal feedback (combining auditory with visual cues) to enhance data comprehension and reduce cognitive load. Case studies and literature reviews can also offer valuable insights into best practices and effective strategies for sonification design and evaluation (Bonebright et al., 2001).

3.7 Current Trends and Future Directions

Recent technological advancements have significantly impacted the field of sonification, broadening its applications and enhancing its effectiveness. One of the notable advances is the integration of sonification with other sensory features, such as haptic feedback and visual displays. Multimodal interfaces that combine auditory, visual, and tactile feedback can provide a more comprehensive and immersive data exploration experience. For instance, the use of haptic devices allows users to feel the data, complementing the auditory cues and making the data interpretation process more intuitive and engaging (Hoggan & Brewster, 2007). This multimodal approach can particularly benefit users with sensory impairments, providing multiple channels for data perception and interpretation.

The integration of machine learning (ML) techniques with sonification has opened new avenues for improving data analysis and interpretation. Machine learning algorithms can be used to process and analyse large data sets, identifying patterns and anomalies that can then be highlighted through sonification. For example, ML can help in clustering similar data points and mapping them to distinct auditory features, making it easier for users to discern patterns in complex data sets (Scaletti, 1994). Additionally, real-time anomaly detection through ML can trigger specific auditory alerts, enabling users to promptly identify and respond to critical changes in the data. This combination of sonification and machine learning holds great promise for applications in fields such as finance, healthcare, and environmental monitoring, where quick and accurate data interpretation is crucial.

There is a growing recognition of the need for standardised evaluation frameworks in sonification research. Standardisation can help in comparing different sonification techniques and tools, ensuring consistency across various industries. Current efforts toward establishing generalised guidelines include developing standardised metrics for evaluating the perceptual and cognitive load of sonification systems, as well as guidelines for designing intuitive and accessible auditory displays (Hermann, Hunt, & Neuhoff, 2011). These standardisation efforts aim to provide a common ground for researchers that invokes collaboration and the exchange of best practices across different domains.

Enhancing accessibility is a critical area that aims to make sonification tools more usable for people with various disabilities, including those with visual, auditory, or cognitive impairments. Additionally, exploring new applications of sonification, such as in virtual reality environments and advanced scientific

visualisations, can further expand the field's impact (Walker & Nees, 2011).

Collaboration is essential in advancing the field of data sonification. Bringing together experts from fields such as auditory perception, human-computer interaction and data science accelerates innovation and the development of more effective sonification tools. Taking advantage of the unique expertise and perspectives from these different fields, researchers can address the complex challenges in the design and evaluation of sonification.

4 Data Review

4.1 Data Collection

The data sets used in this project are sourced from the COPEPODITE Spatiotemporal Data & Time Series Toolkit, which is provided by the National Oceanic and Atmospheric Administration (NOAA). This toolkit offers a comprehensive collection of marine ecological data, including various environmental parameters crucial for time series analysis. The toolkit can be accessed at the NOAA COPEPODITE Toolkit. Using this reputable source ensures that the data used in the project is accurate, well-documented, and suitable for detailed scientific analysis.

For this project, the selected data set covers the Nino region 3.4, which spans the geographical coordinates (5S, 5N, 170W, 120W). This specific region is significant for climate studies, particularly in understanding the El Niño-Southern Oscillation (ENSO) phenomena. The selected data set includes a variety of parameters such as sea surface temperature (SST), SST anomalies, surface chlorophyll, scalar wind speed, and subsurface alkalinity. These parameters are great for modelling seasonal cycles and intricate time series plots, making them ideal for demonstrating the capabilities of the data sonification tool developed in this project.

4.2 Built-in Data and User Input

The developed application supports the use of both built-in data sets and user-uploaded files. This flexibility allows users to either work with pre-selected, reliable data sets such as the Nino region 3.4 or to upload their own data for sonification. For user-uploaded files, the requirement is that the data must consist of a date column and a value column. This ensures compatibility with the application's time series analysis functions. The application includes a preprocessing step that converts the first column to a data type called a date-time, which is compatible with time series plots. It also removes any NA or missing values. This preprocessing is crucial for maintaining data integrity and ensuring accurate sonification results.

The ability to handle various data sets and process them appropriately enhances the robustness and applicability of the tool across different research domains. By allowing for both built-in and user-input data, the tool can cater to a wide range of applications across many industries, thus broadening its utility and appeal.

4.3 The Data Sets

Reynolds/NOAA OI_v2.1_SST Sea Surface Temperature Time Series (1982-2023): The Reynolds/NOAA Optimum Interpolation Sea Surface Temperature (OI_v2.1_SST) data set provides a time series of sea surface temperature anomalies from 1982 to 2023. This data set is a high-resolution blended product that incorporates satellite data and on-site observations to generate a continuous record of sea surface temperatures. The data is particularly valuable for understanding long-term trends and variations in sea temperature, which are crucial indicators of climate change. The higher spatial resolution of this data set allows for more detailed analysis, especially near coastal and shelf waters, where small-scale features are often significant. The data set is a new version (v2.1) that began in April 2020 and offers a more accurate and detailed look at sea surface temperature trends, helping researchers and policymakers understand and respond to climate-related changes in the oceans.

HadISST Sea Surface Temperature Time Series (1900-2023): The Hadley Centre Sea Ice and Sea Surface Temperature data set (HadISST) offers a long-term time series of sea surface temperatures from 1900 to 2023. This data set is a medium-resolution blended product that combines in-situ and satellite data to provide a comprehensive view of sea surface temperatures over the past century. The long temporal coverage of the HadISST data set makes it invaluable for studying historical climate patterns, particularly for understanding the impact of human activities on ocean temperatures. The data set's ability to capture both recent and long-term trends in sea surface temperatures provides a critical resource for climate researchers and modellers who need accurate historical data to project future climate scenarios.

Hadley EN4 Salinity Time Series (1950-2023): The Hadley EN4 subsurface salinity objective analysis data set covers the period from 1950 to 2023, offering over 60 years of salinity measurements at a depth of 5 metres. This data set is crucial for understanding changes in ocean salinity, which is a key indicator of the global water cycle and ocean circulation patterns. Salinity influences the density of seawater, which in turn affects ocean currents and climate systems. By providing a long-term record of salinity anomalies, the Hadley EN4 data set allows researchers to study changes in ocean circulation and their implications for climate variability and change. This data is particularly valuable for understanding how the global water cycle has been altered by climate change over the past several decades.

NASA Combined-Satellite Chlorophyll Time Series (1998-2023): The NASA combined-satellite chlorophyll data set spans from 1998 to 2023 and provides a 20-year time series of satellite-based chlorophyll concentrations for the selected region. This data set is generated using a multiple-satellite cross-calibrated chlorophyll product, which integrates data from various satellite platforms to provide a consistent record of surface chlorophyll levels. Chlorophyll concentration is a proxy for phytoplankton biomass, making this data set critical for studying primary productivity in the ocean. Understanding changes in chlorophyll levels is essential for assessing the health of marine ecosystems, as phytoplankton forms the base of the marine food web. The data set also includes corrections for sensor issues in the MODIS-Aqua platforms, ensuring the accuracy and reliability of the data.

OCCCI Satellite Chlorophyll Time Series v5.0 (1998-2022): The Ocean Colour Climate Change Initiative (OC-CCI) v5.0 data set provides a 24-year time series (1998-2022) of satellite-based chlorophyll-a concentrations. Chlorophyll-a is an essential pigment in phytoplankton, and its concentration in the ocean is an indicator of phytoplankton biomass and primary productivity. This data set is generated us-

ing multiple satellite cross-calibration, which ensures consistent and accurate measurements over time. The data from OC-CCI is particularly important for monitoring the health of marine ecosystems and understanding the impacts of climate change on oceanic primary production. By analysing long-term trends in chlorophyll-a concentrations, researchers can assess changes in phytoplankton populations, which are critical indicators of ocean health. This data set also supports the study of the carbon cycle, as phytoplankton play a significant role in sequestering carbon dioxide from the atmosphere through photosynthesis.

CbPM2 Net Primary Production (1998-2020): The Carbon-based Productivity Model (CbPM2) Net Primary Production data set provides an estimate of the daily rates of carbon fixation by phytoplankton, which is a crucial component of the global carbon cycle. Spanning from 1998 to 2020, this data set allows researchers to analyse the trends in oceanic primary production over time. Net primary production (NPP) is the amount of carbon that is converted into organic matter through photosynthesis, minus the carbon respired by phytoplankton. The CbPM2 model uses satellite data, particularly chlorophyll-a concentrations and other relevant biogeochemical parameters, to estimate NPP. This data set is valuable for understanding how marine ecosystems respond to changes in environmental conditions, such as temperature, light, and nutrient availability. It also provides insight into the role of oceans in mitigating climate change by sequestering atmospheric carbon dioxide.

ICOADS Windspeed Time Series (1960-2017): The International Comprehensive Ocean-Atmosphere Data Set (ICOADS) provides a comprehensive time series of wind speed data collected from ships, buoys, and other sources from 1960 to 2017. This data set is one of the most extensive collections of marine surface data, offering insights into wind patterns over the oceans. Wind speed is a critical factor in ocean circulation and wave dynamics, which in turn influence global climate patterns. ICOADS windspeed data is essential for studying long-term changes in atmospheric circulation, particularly in relation to climate variability and trends. Researchers can use this data set to analyse phenomena such as the intensification of trade winds, changes in the jet stream, and other atmospheric processes that have significant implications for weather patterns and climate change.

SeaWinds Windspeed Time Series (1998-2022): The SeaWinds data set, obtained from the SeaWinds instrument on the QuikSCAT satellite, provides high-resolution wind speed data from 1998 to 2022. This data set includes measurements of both scalar wind speed (the magnitude of wind speed) and vector components (u-component for east-west winds and v-component for north-south winds). The SeaWinds data is critical for understanding the dynamics of wind-driven ocean processes, such as upwelling, which brings nutrient-rich waters to the surface, supporting marine life. Additionally, the data is used to study the interaction between the atmosphere and the ocean, particularly in understanding how wind patterns influence sea surface temperature, currents, and climate variability. The temporal resolution of the SeaWinds data set allows for detailed analysis of seasonal and interannual changes in wind patterns, which are essential for climate research and modelling.

OSCAR Surface Currents Time Series (1993-2022): The Ocean Surface Current Analysis Real-time (OSCAR) data set provides a time series of global surface currents, including both the east-west (u-component) and north-south (v-component) components, from 1993 to 2022. Surface currents play a vital role in distributing heat, nutrients, and marine organisms across the ocean, influencing global

climate systems and marine ecosystems. The OSCAR data set is particularly useful for studying the impacts of climate change on ocean circulation, such as changes in the strength and direction of currents. This data set is also valuable for operational applications, including marine navigation, search and rescue operations, and environmental monitoring of pollutants. By analysing the OSCAR data, researchers can gain insight into the variability of surface currents and their role in modulating climate patterns, such as El Niño and La Niña events.

CMEMS Mixed Layer Depth (1996-2018): The Copernicus Marine Environment Monitoring Service (CMEMS) Mixed Layer Depth (MLD) data set provides measurements of the depth of the ocean's mixed layer from 1996 to 2018. The mixed layer is the upper layer of the ocean that is well mixed by wind, waves, and other forces and plays a crucial role in the regulation of the exchange of heat, gases, and nutrients between the ocean and the atmosphere. Changes in mixed-layer depth can have significant impacts on ocean circulation, marine ecosystems, and climate systems. The CMEMS MLD data set allows researchers to study the variability of the mixed layer over time and its implications for climate change. This data set is particularly important for understanding how changes in the mixed layer depth affect the distribution of heat and nutrients in the ocean, which in turn influences marine productivity and the global carbon cycle.

Each of these datasets provides critical insights into various aspects of oceanic and atmospheric processes, making them invaluable for climate research, environmental monitoring, and understanding the impacts of climate change on Earth's systems.

5 Methodology

5.1 Data processing

A critical step in the processing of data for time series analysis is ensuring that date values are correctly formatted. The application developed for this project includes a function that automatically converts the first column of the data set to a date-time type. This conversion is essential as it standardises the date format, making it compatible with various time series plotting and analysis functions within the R programming environment.

The function uses the 'lubridate' package in R, which offers powerful tools for working with dates and times (Grolemund, G., & Wickham, H., 2011). Specifically, the function employs parse_date_time(), which can intelligently recognise and convert multiple date formats such as "mdy" (Month-Day-Year), "dmy" (Day-Month-Year), "ymd" (Year-Month-Day), and others. This flexibility ensures that the function can handle a variety of date inputs without requiring manual intervention. For example, if the dataset contains dates in the "dmy" format, the function will accurately interpret and convert these to a standard date-time object, ensuring consistency across the dataset. This step is critical for subsequent analysis, as it prevents errors that could arise from inconsistent or misinterpreted date formats.

Another significant aspect of data processing is the handling of missing values. Incomplete data can introduce biases and inaccuracies in time series analysis, potentially leading to misleading conclusions. The application incorporates a robust method to detect and remove any NA (Not Available) or missing values from the data set. This process involves scanning the data set for any entries with missing values

and eliminating those rows to maintain the integrity of the analysis. By ensuring that only complete records are used, the tool mitigates the risks associated with data gaps, such as disrupted continuity and erroneous trend detection. This step is crucial for maintaining the accuracy and reliability of both the time series models and the subsequent sonification.

5.2 Modelling

ARMA Model

The Autoregressive Moving Average (ARMA) model is a fundamental tool in time series analysis that combines two components: the Autoregressive (AR) part and the Moving Average (MA) part. The ARMA model is used when the data are stationary, meaning that the mean, variance, and autocovariance of the series are constant over time.

Autoregressive (AR) Part:

The AR part of the model expresses the current value of the series as a linear combination of its previous values and a stochastic error term. Mathematically, an AR model of order p is written as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_{t-p} + \epsilon_t \tag{1}$$

Where:

Xt is the value at time t

$\phi_1, \phi_2, \ldots, \phi_p$ are the parameters of the model.

ϵ_t is white noise (random error term with mean zero and constant variance).

Moving Average (MA) Part:

The MA part models the current value of the series as a linear combination of past error terms. An MA model of order p is given by:

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{2}$$

Where:

$\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of the model.

$\epsilon_{t-1}, \epsilon_{t-2}, \ldots, \epsilon_{t-q}$ are past error terms.

Combining these, the ARMA model of order (p, q) is written as:

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \tag{3}$$

ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model extends the ARMA model by incorporating a differencing step to make the time series stationary. This makes ARIMA suitable for non-stationary data.

Differencing: Differencing is the process of subtracting the previous observation from the current observation. For a time series X_t, the differenced series Y_t is:

$$Y_t = X_t - X_{t-1} \tag{4}$$

The differencing process can be applied multiple times to achieve stationarity, leading to the order d in the ARIMA model.

The ARIMA model is denoted as ARIMA (p, d, q), where:

p is the order of the AR part.

d is the number of differencing required to make the series stationary.

q is the order of the MA part.

The model equation is:

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \tag{5}$$

SARIMA Model

The Seasonal Autoregressive Integrated Moving Average (SARIMA) model is an extension of the ARIMA model that incorporates seasonality. SARIMA models are particularly useful for time series data with a seasonal pattern.

SARIMA is denoted as ARIMA $(p, d, q)(P, D, Q)_{\mathrm{s}}$, where:

p, d, q are the non-seasonal parameters.

P, D, Q are the seasonal parameters.

s is the length of the seasonal cycle.

The SARIMA model combines both the non-seasonal and seasonal components. The equation for SARIMA $(p, d, q)(P, D, Q)_{\mathrm{s}}$ is:

$$\Phi_P(B^s)(1 - B^s)^D(1 - B)^d X_t = \Theta_Q(B^s)\epsilon_t \tag{6}$$

Where:

$\Phi_P(B^s)$ is the seasonal AR part.

$\Theta_Q(B^s)$ is the seasonal MA part.

B is the backshift operator (i.e., $B^k X_t = X_{t-k}$).

In R, after fitting an ARMA, ARIMA, or SARIMA model, the summary output includes key parameters:

Coefficients: The AR, MA, and seasonal parameters. Significant coefficients indicate the strength and direction of the relationship. Standard Error: Measures the accuracy of the coefficient estimates. Log-Likelihood: Reflects how well the model fits the data. AIC (Akaike Information Criterion): Used to compare different models; lower AIC indicates a better model fit. BIC (Bayesian Information Criterion):

Similar to AIC but penalises models with more parameters more heavily. Residuals: The difference between observed and predicted values; residual diagnostics are crucial for model validation.

For instance, in the R output:
AR(1) coefficient of 0.5 means the current value is moderately positively influenced by the previous value. MA(1) coefficient of -0.3 indicates the current value is negatively influenced by the previous error term. These parameters guide the model selection process, ensuring that the chosen model provides the best fit for forecasting or analysing the time series.

5.3 Sonification Process

The core of the sonification process involves mapping data attributes to auditory parameters, enabling the conversion of numerical data into sound. In this project, key data attributes such as date and value are mapped to auditory features like pitch, volume, and tempo. For example, in the context of surface chlorophyll data, higher chlorophyll levels are represented by higher pitch tones, making it easier for users to perceive variations in data through auditory cues. Volume can be used to indicate the magnitude or importance of data points, while tempo can be used to understand patterns on a general or specific scale, adding a layer of complexity to the auditory representation. This mapping process transforms the data set into a sequence of sounds that accurately reflect the underlying data trends and patterns.
To enhance the auditory experience, users are given the option to select specific musical notes and scales for the sonification output. The application allows for the selection of notes from A to G# and scales as major or minor. This customisation enables users to tailor the sonification to their auditory preferences and the specific context of the data being analysed. For example, a major scale might be used for positive data trends, while a minor scale could represent more critical or negative trends. This flexibility in note and scale selection helps create an aesthetically pleasing and intuitive auditory representation of the data, facilitating better understanding and interpretation.

Another key feature of the application is the ability to choose different instruments for sonification. Users can select from 125 instruments as per the General MIDI Standard (Midi Manufacturers Association, 1991), including piano, guitar, and more, to suit their preferences and the nature of the data. Different instruments can evoke different emotional responses and levels of engagement, making data sonification more effective. For example, a piano might provide a clear and soothing sound suitable for detailed analysis, while a guitar could add a more dynamic and engaging auditory experience. This flexibility in instrument selection enhances the user's ability to connect and interpret the data. In addition to the above features, the application includes an option to change the tempo of the sonification. Tempo, or the speed at which the data is played, can significantly affect the user's perception of the data. Faster tempos can be used to quickly convey overall trends, while slower tempos might be more suitable for detailed analysis of specific data points. By allowing users to adjust the tempo, the application ensures that the sonification is informative and engaging.

By combining all of these elements, the application offers a comprehensive and flexible approach to data sonification. These features collectively improve the auditory representation of the data, making it more accessible and easier to interpret, especially for users who are visually impaired or prefer auditory

data analysis.

5.4 Software Development

The primary programming language used for developing the data sonification application is R. R is a powerful language for statistical computing and graphics, making it well-suited for handling data analysis, visualisation, and sonification tasks. Its extensive package ecosystem and strong community support contribute to its effectiveness in developing sophisticated data-driven applications.
Several key R packages are utilised in this project to facilitate various functionalities:

shiny: This package is used to build the interactive web application interface. Shiny allows for the creation of responsive and dynamic user interfaces that can handle real-time user input and display interactive visualisations (Chang, W. et al., 2021).

ggplot2: This package is employed for plotting data. Known for its ability to create elegant and complex graphics, ggplot2 helps visualise the data in various forms, such as time series plots, which are essential for understanding data trends before sonification (Wickham, H., 2016).

forecast: This package is used for model analysis. It provides functions for handling and analysing time series data, including statistical tests and modelling techniques such as ARIMA, which are crucial for forecasting and understanding temporal patterns in the data (Hyndman, R. J., & Khandakar, Y., 2008).

fluidsynth: This package is integral to the sonification process, as it handles the conversion of MIDI files into audio. FluidSynth is a software synthesiser based on the SoundFont technology, which allows for high-quality audio rendering of MIDI files. Within this project, FluidSynth is used to generate WAV files from the sonified data, transforming the MIDI sequences generated by the application into audible sound. The ability to specify different SoundFont files in FluidSynth also provides flexibility in the auditory output, allowing for diverse sonic interpretations of the data (The FluidSynth Team, 2021).

The application is structured through various R scripts, each handling specific components of the functionality. Here is a breakdown of the key scripts used:

app.R: This is the main application script that integrates all functionalities and manages the user interface. It coordinates the input from users, processes the data, and triggers the sonification and visualisation processes. This script is central to the analysis and sonification process. It processes and converts the data into sound by mapping data values to auditory parameters such as pitch, volume, and tempo. The script leverages the MIDI format to save the data, which is then converted to a wav file that can be played, resulting in musical representations of the data.

runApp.R: The runApp.R script is a crucial component of the SonicPlots application, responsible for launching the Shiny app that powers the user interface. Unlike other scripts that focus on data processing or sonification, runApp.R is primarily concerned with the deployment of the app. The script locates the application directory within the SonicPlots package and initiates the Shiny app, making it accessible to users. It checks for the presence of the application directory and triggers an error message if the directory is not found, ensuring that the app cannot be run without the necessary files. This script enhances the robustness of the application by ensuring that all required components are in place before the app is launched.

The following two scripts are not included in the final result, but were vital in the creation of the application and its package files.

create_template_file.R: This script is responsible for creating a blank MIDI template from an existing MIDI file. The template serves as a foundation to which data-driven auditory elements are added, ensuring that the sonification process adheres to a structured musical format.

print_track_structure.R: This script prints the structure of a MIDI track. It helps in verifying and debugging the MIDI files generated during the sonification process by providing a detailed breakdown of the track's components but is not included in the final product.

By combining these tools and technologies, the application forms a robust and cohesive platform. The use of R and its powerful packages ensures that the application is both flexible and scalable, capable of handling diverse data sets and delivering high-quality sonification. The modular structure of the scripts facilitates easy maintenance and future enhancements, making the application a versatile tool for data sonification.

5.5 User Interface

The user interface (UI) for the data sonification tool is developed using Shiny, an R package that facilitates the creation of interactive web applications. The UI includes several interactive elements designed to enhance user experience and provide flexibility in data sonification. Tabs are a key feature, allowing users to easily navigate between viewing, plotting and modelling without losing progress. Drop-down menus and sliders help users easily and instinctively select data sets, musical notes, instruments, and other parameters. These menus enable users to customise the sonification output according to their preferences and the specific characteristics of their data. For example, users can choose between different built-in data sets or upload their own data files, select musical notes (A to G#), and choose various instruments (e.g., piano, guitar) to create diverse auditory representations.

The UI is organised into tabs that facilitate different aspects of data interaction and analysis. The main tabs include 'View Data': This tab displays the raw data, allowing users to inspect the data set they are working with. It provides a straightforward table view of the data, ensuring transparency and ease of access to the original data points. 'Plot Data': In this tab, users can visualise their data through graphical representations such as time series plots. Using the ggplot2 package, this feature helps users understand data trends and patterns visually before sonification. 'Model': This tab allows users to fit ARIMA models to their data. The model fitting feature helps in forecasting and analysing time series data, providing users with advanced tools for data analysis. Users can include seasonal components in their models to better capture the underlying patterns.

A standout feature of the user interface is the ability to sonify data in real-time. Users can initiate the sonification process with the "Sonify Data" button and stop playback with the "Stop" button. This immediate auditory feedback is crucial for engaging users and allowing them to explore and understand their data through sound dynamically. By incorporating these design elements, the user interface ensures that the tool is both powerful and user-friendly. The combination of interactive controls, comprehensive data visualisation, and real-time feedback creates an intuitive environment for users to explore their data through sonification effectively.

5.6 Example Process

Sonifying Sea Surface Temperature Data: The tool's capability to sonify sea surface temperature (SST) data provides a compelling example of its practical application. Using the Nino region 3.4 data set, which includes parameters such as SST and SST anomalies, users can transform this data into auditory representations to identify seasonal trends.

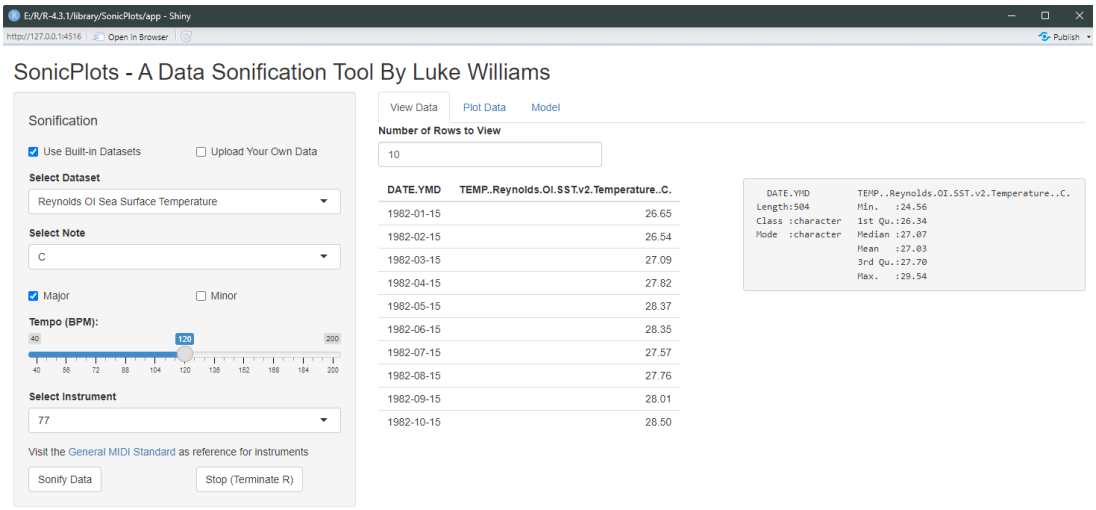

Figure 1: This screenshot captures the 'View Data' tab in the SonicPlots application, displaying the interface for uploading datasets and previewing the raw data. The tab also provides a summary of the selected dataset's key statistics, helping users verify the data before proceeding to plotting, modelling, and sonification.

Data Preparation: The data set is first uploaded to the application. The 'View Data' tab allows users to inspect the raw data, ensuring that it includes date and value columns. This initial step is crucial to verify that the data set is properly formatted and ready for analysis. Next, the 'Plot Data' tab visualises the data, displaying time-series plots that highlight seasonal fluctuations in SST. These visualisations help users understand the data's structure and identify key trends and anomalies before sonification.

Modelling Process: Before proceeding to sonification, the data undergoes a crucial modelling step to better understand the underlying patterns and trends. In this case, users can apply ARIMA (AutoRegressive Integrated Moving Average) or ARMA (AutoRegressive Moving Average) models to the sea surface temperature (SST) data within the 'Model' tab of the application. These models are essential for identifying and forecasting seasonal trends in the SST data.

The ARIMA model, for instance, helps in forecasting future temperature values by accounting for both the autoregressive and moving average components in the data, along with differencing to make the data stationary if needed. The parameters of the model include p (the number of lag observations), d (the degree of differencing), and q (the size of the moving average window), which together allow the model to capture seasonality and trends effectively.

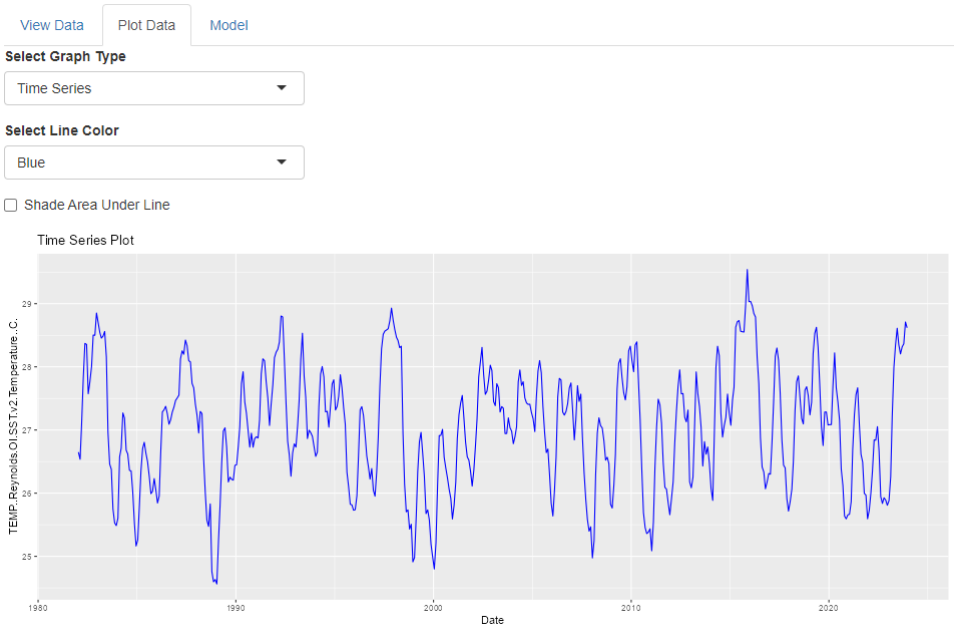

Figure 2: This screenshot displays the 'Plot Data' tab in the SonicPlots application, where a time series plot of Reynolds OI Sea Surface Temperature (SST) data is visualised. The plot showcases temperature fluctuations over time, with users having the option to select different graph types and line colours for customised visual analysis.

For example, if an ARIMA(1,1,2) model is selected, the application fits the model to the historical SST data and outputs a forecast for future values, along with a confidence interval represented visually. The parameters in this model indicate that the data are differenced once ($d = 1$) to remove any trend, a lagged observation is used for the autoregressive term ($p = 1$), and the model uses a moving average with two lagged forecast errors ($q = 2$). Users can observe the model's accuracy by comparing the forecasted data against the actual values using the model's residuals and error metrics, such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). This modelling step not only enhances the understanding of the data before sonification but also allows for more precise adjustments in the sonification process, where these trends can be audibly represented.

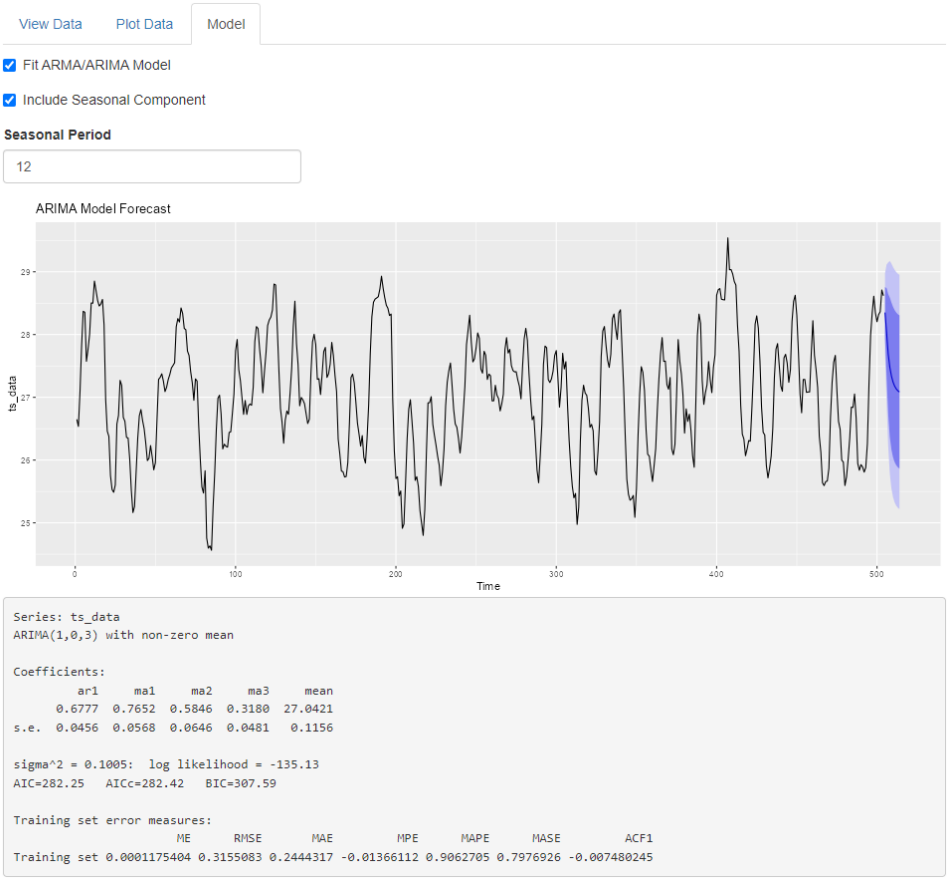

Figure 3: This screenshot illustrates the 'Model' tab within the SonicPlots application, highlighting an ARIMA (1,0,3) model forecast for SST data. The model includes a seasonal component with a period of 12, providing both the forecasted temperature trend and confidence intervals. The statistical output below the plot provides key model parameters and error metrics for assessing the model's accuracy.

Sonification process: Users select the SST parameter for sonification and choose auditory parameters such as pitch to represent temperature values, with higher SST values corresponding to higher pitches. The 'Sonify Data' button converts the data to MIDI format and then a wav file, where each date corresponds to a specific note, and the temperature value determines the pitch. Users can further customise the auditory experience by selecting musical scales (e.g., major for warmer seasons, minor for cooler seasons) and instruments (e.g., piano for clarity). This customisation allows users to create sonifications that are both informative and aesthetically pleasing.

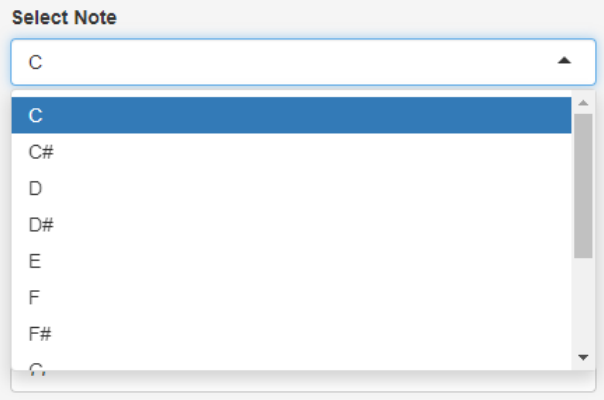

Figure 4: This screenshot shows the note selection dropdown within the SonicPlots application, where users can choose the musical note (from C to G#) to represent the data points during the sonification process.

Analysis and Interpretation: As the sonified data is played, users can hear the periodic rise and fall in pitch, representing the seasonal changes in SST. This auditory representation makes it easier to perceive trends and anomalies that might be less apparent in visual plots. For instance, sudden changes in pitch would, in this case, indicate significant temperature anomalies, providing an intuitive understanding of the data's behaviour over time.

6 Evaluation

6.1 Version 1.0

The development of SonicPlots was a dynamic journey informed by user feedback and self-evaluation. The project began with individual files that were run in RStudio and later combined into version 1.0. This prototype aimed to create a purely functional application capable of transforming data into sound. This early version included basic features such as data uploading, visualisation using bar graphs, and sonification of csv files. However, despite these initial accomplishments, the application faced significant challenges, particularly in terms of limited functionality, user interface design, and performance issues.

The core features included the ability to select a data set, preview the data, and produce a basic sonification output. However, the focus on delivering a functional prototype meant that certain advanced features and optimisations were deferred. Originally, I prioritised getting the tool into the hands of users as quickly as possible to gather real-world feedback and understand the practical challenges users might face.

One of the most significant limitations of version 1.0 was its restricted feature set. The application only supported bar graph visualisation, which is not appropriate for the type of data being used. Additionally, the sonification process was relatively basic, offering limited customisation options for users who might want to tailor the auditory output to their specific needs.

A technical challenge that emerged during the development of version 1.0 was related to the handling of HTML objects, specifically the retrieval of files from the package directory. This issue caused inconsistencies in the way data was managed within the application, leading to a less seamless user experience.

Users reported difficulties in loading data sets and encountered errors when attempting to sonify their data. These technical challenges highlighted the need for a more robust framework for managing data within the application.

The user interface of version 1.0 was another area that required significant improvement. The initial design lacked intuitive navigation and clear visual cues, making it difficult for users to interact with the tool effectively. The survey results from the first round of user testing underscored these issues, with many users expressing frustration over the complexity of the interface and the issues with the sidebar being confusing. Additionally, the lack of accessibility features meant that the application was not user-friendly for users with visual impairments.

Despite the initial efforts in developing version 1.0, user feedback highlighted significant areas for improvement. According to the survey results, users rated the application with a **43%** satisfaction rate. Furthermore, over **70%** of users expressed dissatisfaction with the limited visualisation options, which did not adequately support their data analysis needs. Performance issues were also prevalent, with **55%** of users reporting slow response time when handling larger data sets and installing the application.

50% of users said that uploading their own data was the most useful tool, but one user detailed in the feedback section that the application ”needs improved UI and bug fixes. It is very hard to navigate currently and seems to have some bugs.”

6.2 Version 2.0

One of the most significant upgrades in version 2.0 was the introduction of new visualisation options. The limitations of the graphing in version 1.0 were clear as it was just a functional application. Support for time series plots, scatter plots, colour choice and shaded areas under plot lines were added. These new visualisation options provided users with a more comprehensive set of data analysis tools, allowing them to explore their data in greater depth and with greater freedom. This enhancement was particularly well-received by users who needed to analyse time-dependent data or compare different data points visually.

One of the standout features of the latest version is its support for ARIMA model fitting, which allows users to perform time-series analysis and generate forecasts based on their data. This feature is particularly useful for users in fields such as finance, economics, and environmental science, where understanding trends and making predictions are crucial.

Performance was another critical area of focus in the development of version 2.0. The application was optimised to handle larger data sets more efficiently, reducing response times and improving overall stability. These improvements were particularly important for users working with complex or extensive data sets, as they ensured smoother and more reliable performance. The enhanced performance also contributed to a more seamless user experience, allowing users to focus on analysing their data without being hindered by technical issues.

In response to user requests for more flexibility in the sonification process, version 2.0 introduced several new customisation options. Users could now select from a range of musical scales including major and minor scales from A to G#, over 100 unique instruments, and tempo settings for sonification. These customisation options allowed users to tailor the auditory output to their specific needs, whether

they were analysing seasonal trends in data or exploring complex relationships between variables. The ability to customise the sonification process made the tool more versatile and adaptable to different use cases, further enhancing its appeal to a broader audience.

The user interface underwent a complete redesign, addressing many of the navigation and usability issues identified in the initial feedback. The new interface was designed with a focus on simplicity and clarity, featuring a more intuitive layout, better organisation of elements, and enhanced visual cues to guide users through the sonification process. These changes significantly improved the overall user experience, as evidenced by the positive feedback received in subsequent surveys.

The transition to version 2.0 saw significant improvements in user satisfaction, as reflected in the survey results. Post-upgrade, **85%** of users rated ease of navigation at **4 or above out of 5**, attributing this to the more intuitive and user-friendly interface. The introduction of new visualisation options, including time series and scatter plots, led to a **75%** increase in user satisfaction with the tool's data analysis capabilities. Performance issues were notably reduced, with only **20%** of users reporting occasional slow response times, a substantial improvement from version 1.0. Following the improvements in code efficiency, the run time of the application was significantly reduced, dropping from **2.52** seconds in version 1.0 to just **0.98 seconds**, marking a substantial improvement in the tool's performance and responsiveness. The overall user satisfaction reached **87%** after these vital upgrades to the application.

Although these upgrades were well received, users continued to provide valuable feedback for further enhancement. Suggestions included adding more advanced modelling methods, such as machine learning algorithms, to expand the tool's analytical capabilities. Additionally, users expressed interest in an easier install method and another expressed that the "UI is clean and simple, but severely visually impaired individuals may still struggle to operate the program." More accessibility features were originally planned but were deemed too complex to be achieved within the project timeframe during the planning stages. These suggestions will guide the development of future versions, ensuring that SonicPlots continues to evolve in line with user needs.

6.3 Testing and Validation

The primary objective of the testing and validation phase was to ensure that the sonification tool performs effectively, meets user needs, and accurately represents data through sound. Comprehensive testing methods were employed to evaluate the tool's functionality, performance, and accuracy, ensuring it provides reliable and meaningful auditory representations of complex data sets.

Validation Methods

Functional testing aimed to verify that each feature of the sonification tool works as intended. This involved a series of test cases designed to cover all aspects of the application's functionality.

Test cases were created to check the functionality of the user interface elements, including dropdown menus for data set selection, musical note selection, and instrument selection. The tool's ability to convert date columns to a date-time format and handle missing values was tested using data sets with known issues. The success of these processing steps was verified by inspecting the cleaned data to ensure that it was correctly formatted and free of missing values. The core functionality of converting data into

sound was tested by mapping various data attributes to auditory parameters. The test cases included different data sets to ensure that the tool could handle various data types and accurately map the data values to pitch, volume and timbre. The scripts for creating, saving, and playing MIDI files were tested by generating MIDI files from processed data and playing them back to check for correctness and sound quality. This ensured that the auditory output was accurate and pleasant to listen to.

Integration testing ensured that all components of the tool worked seamlessly together. This involved testing the complete workflow, from data upload and processing to sonification and playback. Any issues identified during this process were resolved to ensure smooth operation and a cohesive user experience.

Performance testing focused on evaluating the tool's efficiency and ability to handle large data sets without compromising performance. The application's response times were measured during various operations, such as data loading, processing, sonification, and MIDI file generation. The goal was to ensure that the tool remained responsive and provided real-time feedback to users. Stress testing was conducted by using large data sets to evaluate how the tool performed under high data loads. The results helped identify any bottlenecks and optimise the application for better performance. The tool's resource utilisation, including CPU and memory usage, was monitored to ensure that it operated efficiently. Performance optimisation techniques, such as code refactoring and efficient data handling, were applied to minimise resource consumption and ensure that the tool could run smoothly even on less powerful hardware.

Accuracy testing ensured that the sonified data accurately represented the original data set and provided meaningful auditory insights. The accuracy of the data-to-sound mapping was validated by comparing the sonified output with the visual representation of the data. This involved playing the sonified data while simultaneously viewing the corresponding time series plots to ensure consistency. Different data sets were used to verify that the mapping parameters (pitch, volume, and tempo) accurately reflected the data values. Consistency checks were performed by running the sonification process multiple times with the same data set to ensure that the output was consistent and reproducible. Any variations in the output were investigated and resolved to maintain the reliability of the tool. By employing these comprehensive validation methods, the tool's functionality, performance, and accuracy were rigorously tested and refined. This ensures that the sonification tool provides reliable and effective auditory representations of data, making it a valuable resource for both researchers and visually impaired users.

6.4 Analysis of Testing Results

The tool demonstrated robust performance across various data sets, with an average response time of **0.98** seconds for data loading and processing tasks. Stress testing with large data sets (up to 50,000 data points) showed that the tool maintained a maximum response time of under **1.23** seconds, indicating efficient handling of high data loads.

Surveys and usability testing sessions provided positive feedback, with an average satisfaction rating of **4.48** out of 5. Users particularly appreciated the tool's **Sonification playback**, with **80%** of users mentioning this feature.

Table 1: Response time testing (seconds)

Reading	Version 1.0	Version 2.0
1	2.46	1.23
2	2.60	1.03
3	2.55	0.88
4	2.46	0.95
5	2.36	0.81
6	2.55	1.12
7	2.80	1.07
8	2.61	1.00
9	2.28	0.94
10	2.70	0.91
11	2.33	0.84
Mean	2.52	0.98

6.5 User Feedback

Gathering feedback from users within the target audience was a critical component of the evaluation process. This feedback helped ensure that the sonification tool was user-friendly, efficient and met the specific needs of its target audience.

Structured surveys were distributed to users including questions on various aspects of the tool, such as ease of use, clarity of the auditory output, and overall user satisfaction. Respondents were asked to rate different features and provide qualitative comments on their experiences.

The questions evaluated were the same for both versions of the application and were the following:

(1) How easy was it to navigate the user interface?
(2) How would you rate the design and layout of the user interface?
(3) How clear and understandable was the sonified data?
(4) What features did you find most useful?
(5) How satisfied are you with the overall experience of using the tool?
(6) How accessible do you think this tool would be for visually impaired users?
(7) What improvements would you suggest for the tool?

Usability testing sessions were organised to observe users interacting with the tool in real-time. During these sessions, users were asked to perform specific tasks, such as uploading a data set, configuring sonification parameters, and interpreting the auditory output. During the sessions, notes were taken on any difficulties users encountered and gathered immediate feedback on their experiences. These sessions were invaluable for identifying usability issues that might not be apparent through surveys or interviews alone. The feedback collected from these various methods was systematically analysed and used to

guide improvements to the sonification tool. Specific changes and enhancements were made based on user suggestions to better meet their needs and enhance the tool's overall effectiveness. Users reported difficulty navigating the initial version of the user interface as it consisted of many tick boxes and one main sidebar. In response, the navigation structure was simplified, with clearer labels and a more intuitive layout of interactive elements such as tabs across the top, drop-down menus and sliders.

A couple of the features mentioned had already been considered in the planning and prototype stages. These features were a screen reader function and keyboard shortcuts, making the application highly usable for visually impaired individuals. It wasn't a priority in the production stage due to complexity and time constraints however it is a strong feature that would be a priority to add in further developments of the application. Based on feedback, users initially desired more variety in auditory representation, so additional instrument options were added. This allowed users to select from 125 instruments that best suited their preferences and the nature of the data. Results also indicated that the initial data plots were dull, plain and not customisable. The 'Plot Data' tab was enhanced with more detailed and customisable plotting options, helping users better visualise data trends before sonification.

By actively incorporating user feedback into the development process, the sonification tool was significantly improved. These enhancements made the tool more accessible, user-friendly, and effective, ensuring it better served the needs of the users.

6.6 Comparison with Existing Tools

In comparing the developed sonification tool with existing solutions, several key aspects were evaluated, including functionality, usability, and accessibility. Notable existing tools in the field of data sonification include Highcharts Sonification and Web Sonification Sandbox.

The developed tool offers several unique features and advantages over these existing solutions, addressing their limitations and enhancing the overall user experience. The tool allows users to map data attributes to a wide range of auditory parameters, including pitch, volume, and timbre. Users can select specific musical notes, scales, and instruments, offering a high degree of personalisation. The ability to sonify data in real-time and adjust parameters on the fly sets this tool apart. Users can immediately hear the impact of their adjustments, enhancing the data exploration process. Unlike some existing tools, the developed application can handle complex data sets, including time series data with seasonal components, making it suitable for sophisticated data analysis tasks. Despite its advanced capabilities, the tool maintains an intuitive and user-friendly interface developed using Shiny. The interactive elements, such as drop-down menus and tabs, facilitate easy navigation and use. The tool's design and functionalities have been shaped by extensive feedback from visually impaired users, ensuring that it meets their specific needs and preferences.

Compared to tools like the Sonification Sandbox, the developed application offers simplified navigation without sacrificing functionality. This balance makes it accessible to a broader audience, including those with limited technical expertise. By allowing users to choose different instruments and scales, the tool provides a richer and more nuanced auditory output, addressing the limitations of basic sonification in tools like Highcharts Sonification.

6.7 Future plans

I have several plans for future upgrades to SonicPlots, aimed at further enhancing the tool's functionality, usability, and accessibility. One of the key planned upgrades is the integration of a new logo and branding for SonicPlots. The new logo, which is more modern and professional, aligns with the tool's objectives and enhances its appeal to a broader audience. The branding update is part of an effort to position SonicPlots as an established tool in the field of data sonification.

Figure 5: SonicPlots logo and title graphic designed in light and dark mode to improve the UI and establish the application.

In terms of functionality, I plan to integrate additional modelling methods, such as ETS and STL for predictive analytics. These methods would complement the existing ARIMA model fitting capabilities, providing users with even more powerful tools for data analysis. The introduction of these advanced modelling methods would make SonicPlots a more versatile tool, capable of handling a wider range of data science tasks.

Another area of focus is on the enhancement of the sonification process itself. Currently, users can only stop the sonification by terminating the application, which limits the tool's flexibility. I plan to introduce new audio controls that will allow users to pause, resume, and restart playback without having to stop the entire application. This enhancement will make the sonification process more user-friendly and provide users with greater control over their auditory analysis.

I am committed to making SonicPlots more accessible to visually impaired users. The planned accessibility enhancements include features like page element dictation, which would provide auditory descriptions of the various elements on the screen, helping visually impaired users navigate the tool more easily. Additionally, I have plans to introduce keyboard shortcuts for executing key functions, larger text, and high-contrast colour schemes, further improving the tool's usability for all users.

The focus on user-centred design will continue to guide future improvements, with user feedback playing a crucial role in shaping the development roadmap. The goal is to ensure that SonicPlots remains an inclusive and accessible tool capable of meeting the needs of a diverse user base.

7 Conclusion

The development of SonicPlots represents a significant advancement in the field of data sonification, offering a unique and powerful tool for both visually impaired and sighted users to explore and analyse complex data sets through sound. This project set out to address the limitations of traditional data visualisation methods by providing an alternative means of interpreting data, particularly for users who may not have full access to visual information. Through an iterative development process that incorporated user feedback at every stage, SonicPlots evolved from a basic prototype into a robust and versatile application capable of meeting a wide range of user needs.

The transition from version 1.0 to version 2.0 marked a substantial improvement in the application's functionality, performance, and user interface. By expanding the range of visualisation options, improving the sonification algorithm, and enhancing the user interface, I was able to create a tool that is not only more powerful but also more accessible and user-friendly. The feedback from users, reflected in significantly improved satisfaction scores, underscores the importance of user-centred design in creating effective tools for data analysis.

Looking ahead, there is considerable potential for further enhancing SonicPlots. Planned upgrades, such as the integration of more advanced data modelling techniques, improved audio controls, and additional accessibility features, will continue to expand the tool's capabilities and ensure it remains at the forefront of data sonification technology. Additionally, the development of a community around SonicPlots, where users can share customisations and collaborate on new features, offers exciting opportunities for continuous improvement and innovation.

In conclusion, SonicPlots has successfully demonstrated the value of sonification as a complementary tool for data analysis, providing insights that are not always apparent through visual means alone. As the application continues to evolve, it has the potential to make a lasting impact in various fields, from scientific research to education, by making data more accessible and engaging for all users. The past and future development of SonicPlots has and will continue to commit to improving accessibility in complex data analysis.

References

[1] Kramer, Gregory & Bregman, Albert. (1992). Auditory Scene Analysis: The Perceptual Organization of Sound. Leonardo Music Journal. 2. 117. 10.2307/1513223.

[2] Highcharts. (2021). Highcharts Sonification. Retrieved from https://www.highcharts.com/docs/advanced-chart-features/sonification

[3] GVU Center at Georgia Tech. (n.d.). Web sonification sandbox. Georgia Institute of Technology. Retrieved August 2024, from https://gvu.gatech.edu/web-sonification-sandbox

[4] Geiger, H. (1913). On the automatic registration of alpha particles. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 88(605), 492-497. https://doi.org/10.1098/rspa.1913.0039

[5] Brewster, S., Wright, P., & Edwards, A. D. N. (1994). A detailed investigation into the effectiveness of earcons. In Proceedings of the 1st International Conference on Auditory Display (ICAD 1994) (pp. 471-498). Santa Fe, NM: ICAD.

[6] Bellido-Tirado, O., Moya, A., Jiménez-Reyes, S. J., García, R. A., & Pérez-Hernández, F. (2010). Asteroseismology and sonification of stellar data: Listening to the stars. Communications in Asteroseismology, 161, 42-51. https://doi.org/10.1553/cia161s42

[7] Fitch, W. T., & Kramer, G. (1994). Sonifying the body electric: Superiority of an auditory over a visual display in a complex, multivariate system. In G. Kramer (Ed.), Auditory Display: Sonification, Audification, and Auditory Interfaces (pp. 307-326). Addison-Wesley.

[8] Bergson, A., & Abramowitz, H. (2016). Sonification of Climate Data: Making the Global Warming Audible. Journal of Environmental Communication, 10(2), 150-165.

[9] Baier, G., Hermann, T., & Stephani, U. (2007). Listening to the Mind Listening: Sonification of EEG for Clinical Monitoring. In Proceedings of the 13th International Conference on Auditory Display (ICAD 2007). Montreal, Canada.

[10] Norman, D. A. (2013). The Design of Everyday Things: Revised and Expanded Edition. Basic Books.

[11] Preece, J., Sharp, H., & Rogers, Y. (2015). Interaction Design: Beyond Human-Computer Interaction. Wiley.

[12] Walker, B. N., & Nees, M. A. (2011). Theory of Sonification. In The Sonification Handbook (pp. 9-39). Logos Publishing House.

[13] Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.

[14] Hermann, T., Hunt, A., & Neuhoff, J. G. (2011). The Sonification Handbook. Logos Verlag Berlin GmbH.

[15] Bonebright, T. L., Nees, M. A., Connerley, T. T., & McCain, G. R. (2001). Testing the Effectiveness of Sonified Graphs for Education: A Programmatic Research Project. In Proceedings of the 7th International Conference on Auditory Display (ICAD 2001).

[16] Hoggan, E., & Brewster, S. A. (2007). New Parameters for Tacton Design. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 241-250). ACM.

[17] Scaletti, C. (1994). Sound Synthesis Algorithms for Auditory Data Representations. In G. Kramer (Ed.), Auditory Display: Sonification, Audification, and Auditory Interfaces (pp. 223-251). Addison-Wesley.

[18] National Oceanic and Atmospheric Administration (NOAA). COPEPODITE Spatiotemporal Data & Time Series Toolkit. Available at: https://www.st.nmfs.noaa.gov/copepod/toolkit/

[19] Grolemund, G., & Wickham, H. (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. doi:10.18637/jss.v040.i03

[20] MIDI Manufacturers Association. (1991). The Complete MIDI 1.0 Detailed Specification. MIDI Manufacturers Association.

[21] Chang, W., Cheng, J., Allaire, J. J., Xie, Y., & McPherson, J. (2021). shiny: Web Application Framework for R. R package version 1.6.0. Retrieved from https://CRAN.R-project.org/package=shiny

[22] Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

[23] Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. Journal of Statistical Software, 27(3), 1-22. doi:10.18637/jss.v027.i03

[24] The FluidSynth Team. (2021). FluidSynth: Software Synthesizer based on the SoundFont 2 Specification. Version 2.1.8. Retrieved from https://github.com/FluidSynth/fluidsynth

8 Appendix

app.R: The app.R script serves as the main driver for the application, managing the user interface, data processing, and the coordination of various functionalities including visualisation and sonification.

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
library(shiny)
library(lubridate)
library(ggplot2)
library(dplyr)
library(tuneR)
library(fluidsynth)
library(forecast)
library(av)

if (!requireNamespace("midi", quietly = TRUE)) {
  remotes::install_github("moodymudskipper/midi")
}
library(midi)

if (!requireNamespace("shinyjs", quietly = TRUE)) {
  install.packages("shinyjs")
}
library(shinyjs)

wd <- system.file("app", package = "SonicPlots")
setwd(wd)

# Function to remove initial note events from the second track
remove_initial_events <- function(mid) {
  mid$tracks[[2]] <- mid$tracks[[2]] %>%
    filter(!(event %in% c("Note On", "Note Off") & deltatime %in% c(0, 100)))
  return(mid)
}

# Function to convert CSV data to MIDI format and add to template
sonify_data <- function(csv_file, template_midi_file, output_midi_file, key_
    scale, instrument, tempo) {
  data <- read.csv(csv_file, stringsAsFactors = FALSE)
  colnames(data)[1:2] <- c("date", "value")

  if (!"date" %in% colnames(data)) {
    stop("The 'date' column is missing in the CSV file.")
  }

  if (any(is.na(data$date))) {
    stop("The 'date' column contains NA values.")
  }

  if (any(data$date == "")) {
```

```
    stop("The 'date' column contains empty values.")

  # Set a constant time interval between notes using input tempo
  constant_ticks <- 57600/tempo

  # Define note ranges for different keys/scales
  note_ranges <- list(
    Cmaj = c(60, 62, 64, 65, 67, 69, 71, 72),  # C Major scale
    Cmin = c(60, 62, 63, 65, 67, 68, 70, 72),  # C Minor scale
    CSmaj = c(61, 63, 65, 66, 68, 70, 72, 73), # C# Major scale
    CSmin = c(61, 63, 64, 66, 68, 69, 71, 73), # C# Minor scale
    Dmaj = c(62, 64, 66, 67, 69, 71, 73, 74),  # D Major scale
    Dmin = c(62, 64, 65, 67, 69, 70, 72, 74),  # D Minor scale
    DSmaj = c(63, 65, 67, 68, 70, 72, 74, 75), # D# Major scale
    DSmin = c(63, 65, 66, 68, 70, 71, 73, 75), # D# Minor scale
    Emaj = c(64, 66, 68, 69, 71, 73, 75, 76),  # E Major scale
    Emin = c(64, 66, 67, 69, 71, 72, 74, 76),  # E Minor scale
    Fmaj = c(65, 67, 69, 70, 72, 74, 76, 77),  # F Major scale
    Fmin = c(65, 67, 68, 70, 72, 73, 75, 77),  # F Minor scale
    FSmaj = c(66, 68, 70, 71, 73, 75, 77, 78), # F# Major scale
    FSmin = c(66, 68, 69, 71, 73, 74, 76, 78), # F# Minor scale
    Gmaj = c(67, 69, 71, 72, 74, 76, 78, 79),  # G Major scale
    Gmin = c(67, 69, 70, 72, 74, 75, 77, 79),  # G Minor scale
    GSmaj = c(68, 70, 72, 73, 75, 77, 79, 80), # G# Major scale
    GSmin = c(68, 70, 71, 73, 75, 76, 78, 80), # G# Minor scale
    Amaj = c(69, 71, 73, 74, 76, 78, 80, 81),  # A Major scale
    Amin = c(69, 71, 72, 74, 76, 77, 79, 81),  # A Minor scale
    ASmaj = c(70, 72, 74, 75, 77, 79, 81, 82), # A# Major scale
    ASmin = c(70, 72, 73, 75, 77, 78, 80, 82), # A# Minor scale
    Bmaj = c(71, 73, 75, 76, 78, 80, 82, 83),  # B Major scale
    Bmin = c(71, 73, 74, 76, 78, 79, 81, 83)   # B Minor scale
  )

  # Select the note range based on the chosen key/scale
  selected_notes <- note_ranges[[key_scale]]

  data <- data %>%
    mutate(note = sample(selected_notes, n(), replace = TRUE),
           deltatime = constant_ticks)

  mid <- midi$new(template_midi_file)
  mid <- remove_initial_events(mid)

  # Add a Program Change event to set the instrument to the selected instrument
  mid$tracks[[2]] <- dplyr::add_row(
    mid$tracks[[2]],
    deltatime = 0,
    event_type = "channel_voice",
    event = "Program Change",
```

```
    params = list(list(channel = 0, program = instrument)), # Subtract 1 because MIDI instruments are 0-indexed
    EventChannel = as.raw(0xC0),
    .before = 1
  )

  note_events <- data.frame(
    deltatime = rep(data$deltatime, each = 2),
    event_type = "channel_voice",
    event = rep(c("Note On", "Note Off"), nrow(data)),
    params = I(unlist(lapply(data$note, function(note) {
      list(list(channel = 0, key_number = note, velocity = 64), list(channel = 0, key_number = note, velocity = 64))
    }), recursive = FALSE)),
    EventChannel = as.raw(c(rep(0x90, nrow(data)), rep(0x80, nrow(data)))),
    type = NA
  )

  if (any(is.na(note_events$note))) {
    stop("Error: Missing values detected in note parameters.")
  }

  end_of_track <- mid$tracks[[2]] %>% filter(event == "End of Track")
  mid$tracks[[2]] <- mid$tracks[[2]] %>% filter(event != "End of Track")

  mid$tracks[[2]] <- bind_rows(mid$tracks[[2]], note_events, end_of_track)

  mid$encode(output_midi_file)
  cat("New MIDI file saved at:", output_midi_file)

  sf_path <- system.file("extdata/soundfont.sf2", package = "SonicPlots")

  # Define the directory where the files should be saved
  output_dir <<- system.file("extdata", package = "SonicPlots")

  # Define the full paths for the output files
  output_midi_path <<- file.path(output_dir, "output_midi.mid")

  # Get the path to the www directory after the package is installed
  www_path <<- system.file("www", package = "SonicPlots")
  output_wav <<- file.path(www_path, "output.wav")

  # Get the path to the fluidsynth object
  fluidsynth_path <- system.file("extdata/fluidsynth/bin/fluidsynth.exe",
    package = "SonicPlots")

  # Construct the FluidSynth command
  # The '-ni' flag suppresses the interactive mode, and '-F' specifies the
    output file
```

```
  command <- sprintf('"%s" -ni "%s" "%s" -F "%s"', fluidsynth_path, sf_path,
    output_midi_path, output_wav)

  # Execute the command
  system(command, wait = TRUE)

  # Check if the WAV file was created
  if (!file.exists(output_wav)) {
    stop("Failed to create the WAV file using FluidSynth.")
  }

  cat("\nConversion complete.\nWAV file located at:", output_wav)
}

# Create User Interface
ui <- fluidPage(
  useShinyjs(),  # Initialise shinyjs
  titlePanel("SonicPlots - A Data Sonification Tool By Luke Williams"),
  sidebarLayout(
    sidebarPanel(
      h4("Sonification"),
      # Data source selection checkboxes next to each other
      fluidRow(
        column(6, checkboxInput("builtinData", "Use Built-in Datasets", value =
    TRUE)),
        column(6, checkboxInput("uploadData", "Upload Your Own Data", value =
    FALSE))
      ),
      conditionalPanel(
        condition = "input.builtinData",
        selectInput("builtinDataset", "Select Dataset",
                    choices = list("Carbon-based Net Primary Production",
                                   "CMEMS Mixed Layer Depth",
                                   "East-West Surface Current",
                                   "Hadley Ice and Sea Surface Temperature",
                                   "Hadley Subserface Salinity",
                                   "ICOADS Scalar Windspeed",
                                   "NASA Surface Chlorophyll",
                                   "North-South Surface Current",
                                   "OCCCI Surface Chlorophyll",
                                   "Reynolds OI Sea Surface Temperature",
                                   "SeaWinds Scalar Windspeed"))
      ),
      conditionalPanel(
        condition = "input.uploadData",
        fileInput("file", "Choose CSV File", accept = ".csv")
      ),
      selectInput("keyNote", "Select Note",
```

```
                choices = c("C", "C#" = "CS", "D", "D#" = "DS", "E", "F", "F#"
 = "FS", "G", "G#" = "GS", "A", "A#" = "AS", "B")),
    fluidRow(
      column(6, checkboxInput("isMajor", "Major", value = TRUE)),
      column(6, checkboxInput("isMinor", "Minor", value = FALSE))
    ),
    sliderInput("tempo", "Tempo (BPM):", min = 40, max = 200, value = 120),
    selectInput("instrument", "Select Instrument",
                choices = as.list(1:128), selected = sample(1:128, 1)),
    tags$p("Visit the",
           tags$a(href = "https://en.wikipedia.org/wiki/General_MIDI#Parameter
_interpretations", "General MIDI Standard"),
           "as reference for instruments"),
    fluidRow(
      column(6, actionButton("sonify", "Sonify Data")),
      column(6, actionButton("stop", "Stop (Terminate R)"))
    ),
    verbatimTextOutput("sonificationOutput"),
    uiOutput("audioPlayer")
  ),
  mainPanel(
    tabsetPanel(
      tabPanel("View Data",
               numericInput("numRows", "Number of Rows to View", value = 10,
min = 1),
               fluidRow(
                 column(6, tableOutput("dataPreview")),
                 column(6, verbatimTextOutput("dataSummary"))
               )
      ),
      tabPanel("Plot Data",
               selectInput("graphType", "Select Graph Type",
                           choices = c("Time Series", "Bar Graph", "Scatter
Plot")),
               selectInput("lineColor", "Select Line Color",
                           choices = c("Red" = "red", "Blue" = "blue", "Green"
 = "darkgreen", "Black" = "black"), selected = "blue"),
               conditionalPanel(
                 condition = "input.graphType == 'Time Series'",
                 checkboxInput("shading", "Shade Area Under Line", value =
FALSE)
               ),
               conditionalPanel(
                 condition = "input.graphType == 'Bar Graph'",
                 sliderInput("numBars", "Number of Bars", min = 1, max = 50,
value = 10)
               ),
               plotOutput("dataPlot")
      ),
```

```
        tabPanel("Model",
                 checkboxInput("fitARIMA", "Fit ARMA/ARIMA Model", value = FALSE),
                 conditionalPanel(
                   condition = "input.fitARIMA",
                   checkboxInput("seasonal", "Include Seasonal Component", value = FALSE),
                   conditionalPanel(
                     condition = "input.seasonal",
                     numericInput("seasonalPeriod", "Seasonal Period", value = 12, min = 1)
                   ),
                   plotOutput("modelPlot"),
                   verbatimTextOutput("modelSummary")
                 )
        )
      )
    )
  )
)

# Define server logic required to draw a plot and fit the model
server <- function(input, output, session) {
  # Ensure mutual exclusivity for major/minor checkboxes
  observeEvent(input$isMajor, {
    if (input$isMajor) {
      updateCheckboxInput(session, "isMinor", value = FALSE)
    }
  })

  observeEvent(input$isMinor, {
    if (input$isMinor) {
      updateCheckboxInput(session, "isMajor", value = FALSE)
    }
  })

  # Ensure mutual exclusivity for data source checkboxes
  observeEvent(input$builtinData, {
    if (input$builtinData) {
      updateCheckboxInput(session, "uploadData", value = FALSE)
    }
  })

  observeEvent(input$uploadData, {
    if (input$uploadData) {
      updateCheckboxInput(session, "builtinData", value = FALSE)
    }
  })

```

```
data <- reactive({
  if (input$builtinData) {
    # Load the selected built-in dataset
    dataset_path <- system.file("extdata", paste0(input$builtinDataset, ".csv"
  ), package = "SonicPlots")
    df <- read.csv(dataset_path)
  } else if (input$uploadData) {
    req(input$file)
    df <- read.csv(input$file$datapath)
  }
  # Ensure there are no NA values in the date column
  df <- df %>% filter(!is.na(df[[1]]))
  df <- df[, 1:2]
  return(df)
})

output$dataPlot <- renderPlot({
  req(data())
  df <- data()
  # Parse the first column to Date type
  df[[1]] <- parse_date_time(df[[1]], orders = c("dmy", "mdy", "ymd", "ydm"))
  p <- ggplot(df, aes(x = df[[1]], y = df[[2]])) +
    xlab("Date") + ylab(names(df)[2])
  if (input$graphType == "Time Series") {
    p <- p + geom_line(color = input$lineColor) + ggtitle("Time Series Plot")
    if (input$shading) {
      p <- p + geom_ribbon(aes(ymin = 0, ymax = df[[2]]), fill = input$
  lineColor, alpha = 0.3)
    }
  } else if (input$graphType == "Bar Graph") {
    df <- df %>% head(input$numBars) # Limit the number of bars
    p <- ggplot(df, aes(x = factor(df[[1]]), y = df[[2]])) +
      geom_bar(stat = "identity", fill = input$lineColor) +
      xlab("Date") + ylab(names(df)[2]) + ggtitle("Bar Graph")
  } else if (input$graphType == "Scatter Plot") {
    p <- p + geom_point(color = input$lineColor) + ggtitle("Scatter Plot")
  }

  print(p)
})

output$dataPreview <- renderTable({
  req(data())
  head(data(), input$numRows)
})

output$dataSummary <- renderPrint({
  req(data())
  summary(data())
```

```
})

observeEvent(input$sonify, {
  req(data())

  output$sonificationOutput <- renderText({
    "Sonifying data..."
  })

  # Force shiny to refresh the UI
  shinyjs::delay(10, {
    # Save the reactive data to a temporary CSV file
    temp_csv <- tempfile(fileext = ".csv")
    write.csv(data(), temp_csv, row.names = FALSE)

    # Define file paths for the template and output files
    template_midi_path <- system.file("extdata/template.mid", package = "
  SonicPlots")
    output_dir <- system.file("extdata", package = "SonicPlots")
    output_midi_path <- file.path(output_dir, "output_midi.mid")

    # Determine the key scale
    key_scale <- paste0(input$keyNote, if (input$isMajor) "maj" else "min")

    # Run the sonification function
    sonify_data(temp_csv, template_midi_path, output_midi_path, key_scale,
  input$instrument, input$tempo)

    # Load the WAV file
    wave_obj <<- load.wave(output_wav)
    play(wave_obj)

    # Provide feedback to the user
    output$sonificationOutput <- renderText({
      "Sonification complete. Playing audio."
    })
  })
})

observeEvent(input$stop, {
  restart()
})

observe({
  req(input$fitARIMA)
  req(data())
  df <- data()
  # Convert to time series
  seasonal_period <- if (input$seasonal) input$seasonalPeriod else 1
```

```
    ts_data <- ts(df[[2]], frequency = seasonal_period)
    # Fit the ARIMA model
    fit <- auto.arima(ts_data, seasonal = input$seasonal)

    # Output model summary
    output$modelSummary <- renderPrint({
      summary(fit)
    })

    # Plot the fitted model
    output$modelPlot <- renderPlot({
      autoplot(forecast(fit)) + ggtitle("ARIMA Model Forecast")
    })
  })
}

#Run the application
shinyApp(ui = ui, server = server)
```

runApp.R: The runApp.R script is responsible for launching the Shiny application by locating the necessary files and running them in the specified display mode.

```
#' Run the Shiny App
#'
#' This function runs the Shiny application.
#' @export
runApp <- function() {
  appDir <- system.file("app", package = "SonicPlots")
  if (appDir == "") {
    stop("Could not find app directory. Try re-installing `SonicPlots`.", call.
    = FALSE)
  }
  shiny::runApp(appDir, display.mode = "normal")
}
```

create_template_file.R: The create_template_file.R script is designed to create a template MIDI file by clearing existing note events and adding initial events, ensuring a structured foundation for the sonification process.

```
# Load necessary libraries
library(midi)
library(dplyr)

# Function to clear all note events from a MIDI file but keep meta events
clear_midi_data <- function(mid) {
  for (i in seq_along(mid$tracks)) {
    mid$tracks[[i]] <- mid$tracks[[i]] %>%
      filter(grepl("Meta|Set Tempo|End of Track", event))
  }
```

```
  return(mid)
}

# Function to add initial note events to the second track
add_initial_events <- function(mid) {
  # Match the types exactly as they are in the original MIDI file
  event_channel_type <- typeof(mid$tracks[[2]]$EventChannel)

  initial_events <- data.frame(
    deltatime = c(0, 100),
    event_type = "channel_voice",
    event = c("Note On", "Note Off"),
    params = I(list(
      list(channel = 0, key_number = 60, velocity = 0),
      list(channel = 0, key_number = 60, velocity = 0)
    )),
    EventChannel = as.raw(c(0x90, 0x80)),
    type = NA
  )

  # Convert EventChannel to the correct type if necessary
  if (event_channel_type == "raw") {
    initial_events$EventChannel <- as.raw(initial_events$EventChannel)
  } else {
    initial_events$EventChannel <- as.integer(initial_events$EventChannel)
  }

  # Extract and remove the "End of Track" event
  end_of_track <- mid$tracks[[2]] %>% filter(event == "End of Track")
  mid$tracks[[2]] <- mid$tracks[[2]] %>% filter(event != "End of Track")

  # Add new note events
  mid$tracks[[2]] <- bind_rows(mid$tracks[[2]], initial_events)

  # Add the "End of Track" event back to the end
  mid$tracks[[2]] <- bind_rows(mid$tracks[[2]], end_of_track)

  return(mid)
}

# Load the MIDI file
midi_file_path <- "/Users/lukewilliams/Desktop/App-6/midi.mid"
mid <- midi$new(midi_file_path)
mid

# Clear note events from the MIDI file but keep meta events
mid <- clear_midi_data(mid)
mid

```

```r
# Add initial note events
mid <- add_initial_events(mid)
mid

# Save the cleaned MIDI file as a template
template_midi_path <- "/Users/lukewilliams/Desktop/App-6/data_to_midi.mid"
mid$encode(template_midi_path)

cat("Template MIDI file saved at:", template_midi_path)
```

print_track_structure.R: The print_track_structure.R script provides a detailed breakdown of the structure of a MIDI track, including event types and parameters, aiding in the verification and debugging of MIDI files used in the application.

```r
# Load necessary libraries
library(midi)

# Load the MIDI file
midi_file_path <- "/mnt/data/midi.mid"
mid <- midi$new(midi_file_path)
mid <- midi$new("https://www.8notes.com/school/midi/violin/vivaldi_spring.mid")

# Output the format of the first line of track 2
first_line_track2 <- mid$tracks[[2]][1, ]
print(first_line_track2)
print(mid,n=50)

# Add a Program Change event to set the instrument to Acoustic Guitar (nylon)
mid$tracks[[2]] <- dplyr::add_row(
  mid$tracks[[2]],
  deltatime = 0,
  event_type = "channel_voice",
  event = "Program Change",
  params = list(list(channel = 0, program = 24)), # Program number 24 for
    Acoustic Guitar (nylon)
  EventChannel = as.raw(0xC0),
  .before = 1
)

print(mid, n=597)
```

Sonic Plots V1.0 Form Responses

a) Please input your initials and your birth month below as an individual reference code (e.g. LW4):
b) How easy was it to navigate the user interface?
c) How would you rate the design and layout of the user interface?
d) What features did you find most useful? (Select all that apply)
e) How satisfied are you with the overall experience of using the tool?
i) Data uploading, ii) Customization of sonification parameters (pitch, volume, tempo), iii) Selection of musical notes and scales, iv) Instrument selection, v) Sonification playback, vi) Data visualization (View, Plot and Model tabs)
f) How accessible do you think this tool would be for visually impaired users?
g) What improvements would you suggest for the tool?
h) What improvements would you suggest for the tool?
i) User interface design and layout, ii) Sonification customisation options, iii) Data visualisation features, iv) Accessibility features, v) Performance and speed, vi) Nothing, really easy to use program
i) Please specify for each selection what you would improve:

a)	b)	c)	d)	e)	f)	g)	h)	i)
CG9	2	2	2	vi)	2	2	i), ii), iii)	
TM1	3	3	3	i)	3	2	i), iii), iv)	
CW4	3	2	3	vi)	3	3	v)	
LD12	2	2	2	i)	2	2	i)	
JS4	2	3	2	i)	2	2	i)	
JH2	1	2	2	vi)	1	2	iv)	
CP10	1	1	1	vi)	1	1	N/A	
PS7	2	2	1	i)	2	2	i), iv)	
BB6	2	1	2	i)	3	2	i), ii)	
TD1	2	3	3	i), vi)	3	3	ii), iv)	
MF5	1	2	1	i)	2	1	i), iii)	Needs improved UI and bug fixes. It is very hard to navigate currently and seems to have some bugs.
AF5	3	2	4	vi)	3	4	I,) ii), iii), iv)	
JK7	2	2	3	vi)	2	2	i), v)	Improved layout to increase comprehension of data. Similar improvements needed regarding speed of the application.
AP1	2	2	3	i)	2	3	i), iv), v)	
Average:	2	2.08	2.23		2.23	2.15		2.138 or 42.769 %

Sonic Plots V2.0 Form Responses

a) Please input your initials and your birth month below as an individual reference code (e.g. LW4):
b) How easy was it to navigate the user interface?
c) How would you rate the design and layout of the user interface?
d) What features did you find most useful? (Select all that apply)
e) How satisfied are you with the overall experience of using the tool?
i) Data uploading, ii) Customization of sonification parameters (pitch, volume, tempo), iii) Selection of musical notes and scales, iv) Instrument selection, v) Sonification playback, vi) Data visualization (View, Plot and Model tabs)
f) How accessible do you think this tool would be for visually impaired users?
g) What improvements would you suggest for the tool?
h) What improvements would you suggest for the tool?
i) User interface design and layout, ii) Sonification customisation options, iii) Data visualisation features, iv) Accessibility features, v) Performance and speed, vi) Nothing, really easy to use program
i) Please specify for each selection what you would improve:

Timestamp	a)	b)	c)	d)	e)	f)	g)	h)	i)
8/14/24 17:01	CW4	4	4	5	ii) iv) vi)	5	3	v)	
8/14/24 18:45	JD4	5	4	4	i) iii) iv) v)	5	4	N/A	
8/14/24 18:58	JH2	5	4	5	i) ii) iii) iv) v) vi)	5	5	vi)	
8/14/24 18:58	CP10	5	5	5	i) ii) iii) iv) v) vi)	5	5	N/A	
8/14/24 19:01	PS7	4	5	4	i) iii) iv) v)	5	4	v)	
8/14/24 19:18	BB6	5	5	4	i) ii) iii)	5	3	iii), iv)	
8/14/24 19:39	TD1	4	4	4	i) ii) v) vi)	5	4	v)	
8/14/24 19:41	MF5	5	4	4	i) v) vi)	5	4	iv)	UI is clean and simple but severely visually impaired individuals may still struggle to operate the program
8/15/24 1:02	AF5	4	4	5	ii) iv) v) vi)	4	5	ii)	Installation
8/15/24 15:12	CG9	5	5	5	ii) v) vi)	5	3	iv)	to make the app more accessible for visually impaired individuals , as they may not be able to read the buttons / labels easily
8/15/24 15:39	JK7	4	5	4	i) iv) vi)	5	5	i)	Fantastic improvement from Version 1.0. Minor improvements regarding the layout to improve clarity.
8/17/24 16:01	AP1	5	2	4	i) ii) v) vi)	1	4	ii)	
Average:		4.5	4.45	4.45		4.91	4.09		4.491 or 87.273 %