**Soundscape Data Analysis Guidelines**

(Edited version\_230626)

*Soundscapy is a python library for analysing and visualising soundscape assessments developed by:*

Mitchell, A., Aletta, F., & Kang, J. (2022). How to analyse and represent quantitative soundscape data. *JASA Express Letters, 2*, 37201. <https://doi.org/10.1121/10.0009794>

****

[**https://github.com/MitchellAcoustics/Soundscapy**](https://github.com/MitchellAcoustics/Soundscapy)

1. **Setting Up Visual Studio Code**
2. Install Python: Download from the official Python website and click “add to PATH” (https://www.python.org/downloads/)
3. Install Jupyter: Open a terminal or command prompt (Windows key + r) and run the following command to install Jupyter using pip, which is the Python package installer: “pip install jupyter. This command will install the Jupyter package on your system.
4. Install Visual Studio Code: Download VSCode and click “add to PATH” (https://code.visualstudio.com/download)
5. Install the Python & Jupyter Extensions in VSCode: Launch VS Code and go to the Extensions view (Ctrl+Shift+X or View > Extensions). Search for "Jupyter" and "Python" extensions and install them.

Now you can create and open Jupyter notebooks (.ipynb files) in VS Code using the Jupyter extension. The notebook interface will be displayed within the editor, and you can write code, execute cells, and view the results—all within the VS Code environment.

Jupyter Notebook supports multiple programming languages, including Python, R, Julia, and others. However, it is most commonly used with Python for data analysis, machine learning, and scientific computing.

The key feature of Jupyter Notebook is its ability to execute code in "cells" interactively. You can write and run code snippets in individual cells, which can be executed in any order. This allows you to experiment, test code, and visualize results in an iterative and interactive manner.

*Note: Users can use any IDE (Integrated Development Environment). However in our work we use VSCode because it is light and suitable for our purposes.*

1. **Cloning the repository in VSCode**
2. Install GIT (<https://git-scm.com/download/win>)
3. Open the terminal and navigate the directory to where you want to clone the repository. For example, cloning to a folder called “Soundscapy Repository” using:

*cd "C:\Users\8x9-45a\Desktop\Soundscapy Repository"*

1. Clone using the following command:

*git clone https://github.com/HussamAlBasha/Intern*

*Note: In this* [*Repository*](https://github.com/HussamAlBasha/Intern) *you can find updated Jupyter Notebooks to help you use Soundscapy for the mentioned above purposes.*

1. **Soundscapy Guidelines**
2. Install the “Soundscapy” library by**:** pip install Soundscapy

Soundscpy can be used for:

* 1. **Binaural Recording Analysis**
     1. Prepping the Results Dataframe

df = prep\_multiindex\_df(levels, incl\_metric=True)

* + 1. Load in a Binaural Recording

binaural\_wav = wav\_folder.joinpath("CT107.wav") #we can load and Binaural Recording

b = Binaural.from\_wav(binaural\_wav)

b.plot();

decibel = (levels[b.recording]["Left"], levels[b.recording]["Right"])

print(f"Calibration levels: {decibel}")

b.calibrate\_to(decibel, inplace=True)

print(f"Predefined levels: {levels[b.recording]}")

print(f"Calculated Levels: {b.pyacoustics\_metric('Leq', statistics=['avg'], as\_df=False)}")

* + 1. Calculating Acoustic Metrics(Python Acoustics, scikit-maad, MoSQITo)

**## 1) Python Acoustics**

stats = (5, 10, 50, 90, 95, 'min', 'max', 'kurt', 'skew')

b.pyacoustics\_metric("LAeq", stats, as\_df=True)

**## 2) scikit-maad**

b.maad\_metric("all\_spectral\_alpha\_indices", verbose=True)

b.maad\_metric("all\_temporal\_alpha\_indices", verbose=True)

**## 3) MoSQITo**

metric = "loudness\_zwtv"

stats = (5, 50, 'avg', 'max')

func\_args = { 'field\_type': 'free'}

b.mosqito\_metric(metric, statistics=stats, as\_df=True, parallel=True, verbose=True, func\_args=func\_args)

b.mosqito\_metric("sharpness\_din\_from\_loudness", stats, as\_df=True, parallel=True, verbose=True, func\_args=func\_args)

b.mosqito\_metric("roughness\_dw", stats, as\_df=True, verbose=True, func\_args=func\_args)

* + 1. Defining Analysis Settings

# Default Setting

analysis\_settings = AnalysisSettings.default()

# Loading Customized Setting

ex\_settings = AnalysisSettings.from\_yaml(Path("example\_settings.yaml"))

# Reload the Setting after editing the setting file

ex\_settings = ex\_settings.reload()

* 1. **Statistical Analysis of the ISD Dataset**
     1. Loading the Data

import soundscapy

import pandas as pd

# Set display options

pd.set\_option('display.max\_columns', None)  # Show all columns

pd.set\_option('display.max\_rows', None)     # Show all rows

pd.set\_option('display.width', None)       # Set width to fit the entire DataFrame

df = pd.read\_excel("Data-Test.xlsx", sheet\_name="Tabelle1")

df

* + 1. Filtering the Data

df = pd.read\_excel("Data-Test.xlsx", sheet\_name="Tabelle1")

df.isd.filter\_location\_ids(['EustonTap']).head()

* + 1. Validating the Data

# 'df' will contain the validated DataFrame

# 'excl' will contain any excluded data or information returned by the method

df, excl = df.isd.validate\_dataset()

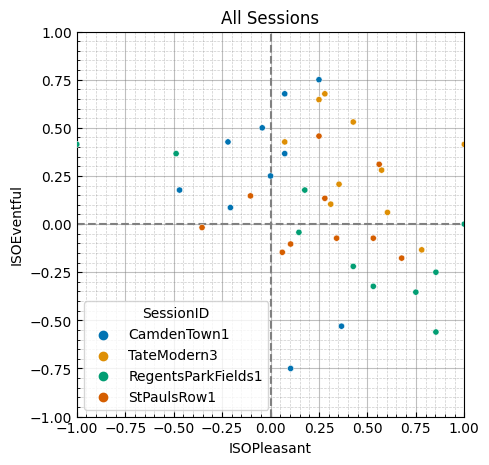
df

* + 1. Calculating the ISOPleasant and ISOEventful Coordinate Values

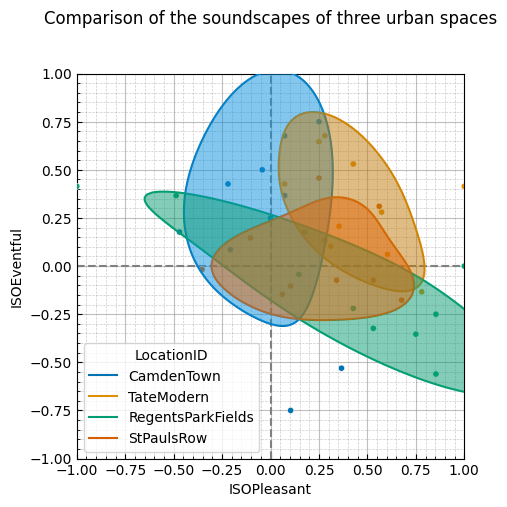
df = paqs.sspy.add\_paq\_coords()

df

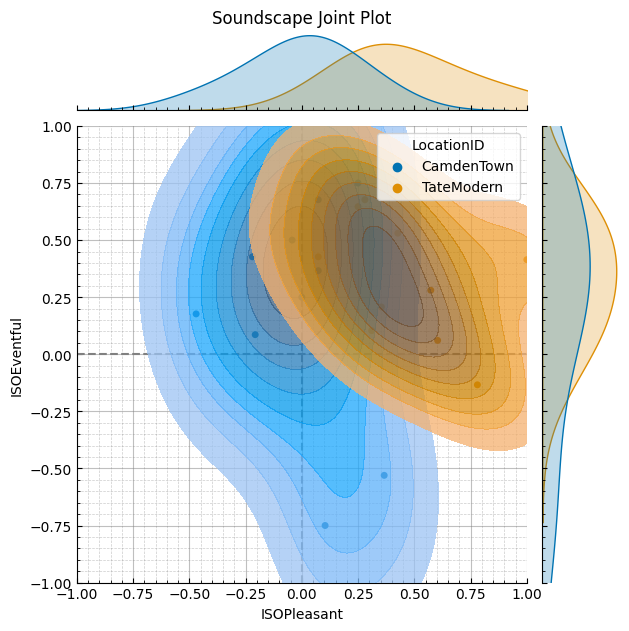
* 1. **Plotting**
     1. Plotting Functions for Visualising Circumplex Data
        1. Scatter Plots



* + - 1. Distribution (Density) Plots



* + - 1. Jointplot



* + 1. Plotting Functions for Visualising Likert Scale Data
       1. Radar/Spider Plot

