Inspect\_EDF - README

## Short description

This folder provides a user-friendly way to work with an EEG database saved in the European Data Format (EDF). EDF is the most common format in clinical settings.

It will guide users from the installation of python (the programming language used to develop the tools of the folder), passing by tutorials to export EEG data in EDF format, to the use of the folder tools (jupyter notebooks or stand-alone python scripts returning reports) to inspect your database. *Ultimately, it will also allow you to extract EEG features such as sleep spindles, slow waves, spectral components etc.*

## Content

* Tutorials:
  + **1\_installing\_python**: to install python (via miniforge) and install the necessary packages to run scripts and notebooks
  + **2\_create\_EEG\_montage\_Profusion**: to create montage in the Profusion software that will greatly facilitate the export of EEG in EDF and reduce errors in exporting parameters
  + **3\_export\_EDF\_withmontage\_Profusion**: to export your EEG data as EDF with a pre-defined montage (from tuto 2)
  + **3bis\_export\_EDF\_manually\_Profusion**: to export your EEG data as EDF manually (not recommended)
* Jupyter notebooks:

Jupyter notebooks are tools in between tutorials and scripts that allow users to interact with code and the output. They are used here as step-by-step tutorials to understand the specificities of the EDF format and the parameters we need to be careful with (when exporting the EEG data)

* + **Inspect\_edf\_v1.ipynb**: Provide detailed explanation of the EDF format and its specificities. It will allow you to load your database and inspect each parameter step-by-step. The users will have to run each step (cell) one-by-one. It is recommended for the first use if the user is interested in checking the code.
  + **Inspect\_edf\_voila.ipynb**: Same content as inspect\_edf\_v1 but the code is hidden (so the notebook is more readable in itself). It is recommended for the first use if the user is not interested in the code.
* Python scripts:

Python scripts are tools that the user will just have to run in order to get reports and summary tables. They are recommended for users just interested in inspecting quickly their database (and that already understand them).

* + **Inspect\_edf\_perdataset.py**: Generates a report for the whole database, pointing to files that may have a problem in their parameters. It additionally creates summary tables (.tsv) to inspect the content of files if needed.
  + **Inspect\_edf\_perparticipant.py**: The report is generated at the participant level, so that each participant can be inspected separately (more readable than a table, but longer to go through.
  + **Check\_hypno\_config.py**: Allows to remap the label in the hypnograms in order to harmonize the dataset and ease the future analyses. For now, you need to manually insert the study folder and hypnogram suffix in a code editor (like spyder). It generates new and renamed hypnogram files (NIP\_Hypnogram\_remapped.txt) that contain harmonized sleep stage labels.

## Workflow

1. Install python and the necessary packages (if python is already, create a new environment with the .yaml in order to install the packages) with the 1st tutorial
2. Familiarize yourself with exporting your data in Profusion and then export your data with the 2nd and 3rd tutorials
3. Inspect your first database with inspect\_edf\_voila.ipynb to understand subtilities of the EDF format
4. Then use only the script inspect\_edf\_perdataset.py to get a quick report (and occasionally use notebooks for careful inspection)

## How to run notebooks and python scripts

Once python and packages are installed:

* open your terminal and activate your virtual environment (run `conda activate inspect\_edf`)
* Go to your folder tools (run `cd my\_folder\_path\tools`)
  + Run `jupyter notebook` to open a notebook (an html page will open and select your file)
  + Run `voila inspect\_edf\_voila.ipynb` to open the notebook with the hidden code
  + Run `python my\_script\_name` to run a python script