"Understanding Biomedical Timeseries-Data"

Practical Assignment 1

Informatics 1 for Biomedical Engineers

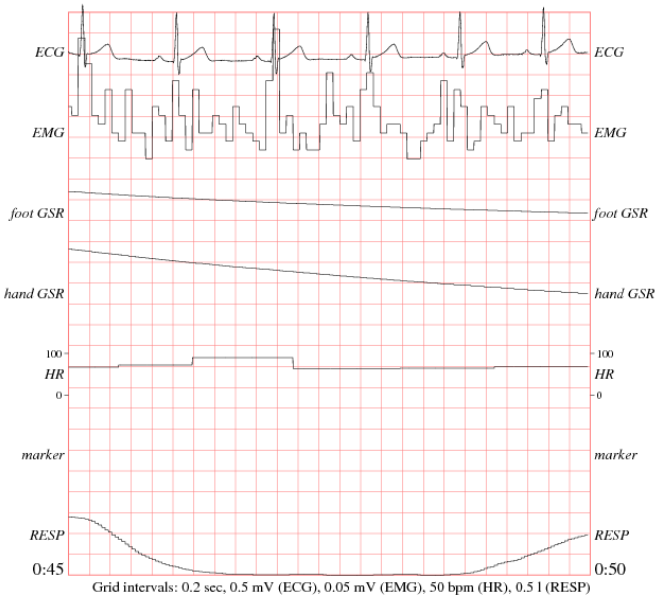
Technische Universität Graz

Abstract

Biomedial timeseries—such as ECG recordings—are usually precise high frequency series of sensor values, which requires a lot of memory. A number of file system have been developed to store them in a compact fashion. A common one is WFDB[1] which separates the data from their description.

In order to learn dealing with such data, this assignment will tackle the parsing of so-called header files in the wfdb format. They contain descriptions concerning which information is containied in the binary files and how they are encoded.

In this task you will learn how to work with command line parameters, how to handle file and folder structures, how to read files to extract information and how to created Python specific data structures such as lists, dictionaries and sets.



[1] `https://physionet.nlm.nih.gov/physiotools/wpg/wpg.pdf`

1. Tasks

1.1. Dataset (0pt)

In this lecture we will be working with real data. The *drivedb* dataset contains a collection of rectords of healthy people driving a pre-defined route. The initial goal of this dataset was to help automatic stress detection.

Download the dataset using one of the provided resources. Using a unix based system you can use the following command:

```
rsync -Cavz physionet.org::drivedb <target directory>
```

Alternatively use the provided archive and extract it to a folder. Note that it should be reachable via a relative path. Ideally as a subfolder in the same directory as your code.

Resources:

- `https://physionet.org/physiobank/database/drivedb/`
- Fallback: `https://physionet.nlm.nih.gov/pn3/drivedb/`
- Pre-prepared Archive: `https://www.dropbox.com/s/k3p69532r2a9vki/dataset.zip?dl=1`

1.2. Command Line Parameters (1pt)

Your program should accept an optional command line parameter. It defines the path to the `RECORDS` file, which contains information about the dataset.

Split the path (the command line param) into folder and file name. Should the param NOT be present, use `DATA/` as path to the dataset and set the filename to `RECORDS`. **Hint:** It may be helpful to actually use these names at the start.

Should the requested folder be missing or the records file not present, provide an output stating what went wrong and end your program with `exit()`.

Example call with parameter:

```
python assignment_1.py DATA/RECORDS
```

1.3. RECORDS File and File Checks (2pt)

The RECORDS file is plaintext (you can open it with every plaintext editor) and contains the name of every record in the dataset. Each row should contain a name. (**Note:** With some editors it may appear as if everything is mushed together in a single line. You can assume, the `readlines()` function splits them accordingly.) Extract all names and check of there is a header file (.hea) present. (**Hint:** Use `strip()` to remove dangling linebreaks and whitespaces automatically.)

Create an output for every entry in *RECORDS* with the following schema:
.hea is present:

```
Record: <name> ... OK

# Example:
Record: drive01 ... OK
```

.hea is **NOT** present:

```
Record: <name> ... NOT FOUND - skipping

# Example:
Record: drive01 ... NOT FOUND - skipping
```

For the remainder of this assignment only use actually found header files. .

1.4. Header Parsing(4pt)

Open all found header files and extract their information. Physionet[2][3] provides a very detailed instruction regarding how to handle these files. For this lecture the following short version suffices:

Typically informations are separated by whitespaces. The names in the `< >` always define the key for your dictionary.

First Row:

Record Name <record_name> name of the record (should match the filename without extension)

Number of signal <num_signals> Number of signals for this record (should match the number of remaining rows in the file)

Sampling Frequency <frequency The sampling frequency for each sensor. This is a float. Convert it accordingly using `.float()`

Anzahl der Frames <num_frame> Wie viele Werte gibt es pro Signal

Further Rows (Signal Rows):

File name <file_name> The name of the binary file which contains the sensor values. (Should be constant for every header file.)

Format <data_format> Format for the encoded data. Should this entry contain an `x` (z.B.: `16x2`) then this value is everything in front of the `x` (In the example `16`).

Samples per frame <samples_per_frame> 1 if undefined. Else the value after the `x`.

Unit <unit> Unit for the singal values. This should always be a string. Do not try to convert it to an integer!

[2] `https://www.physionet.org/physiotools/wag/header-5.htm`
[3] `https://physionet.nlm.nih.gov/physiotools/wag/header-5.htm`

ADC Precision <adc_resolution> Precision of the sensor (irrelevant for the remainder of this task).

ADC Calibration <adc_zero> 0 offset for a comparable analog sensor (irrelevant for the remainder of this task).

Initial Vlaue <initial_value> The first value of the signal.

Check Sum <checksum> Checksum of all signal values added up for integrity checks.

Block Size <block_size> Not relevant (Should always be 0)

Sensor Name The name of the signal or sensor. Can contain whitespaces and is everything after the block size field!

Create a dictionary for every header file. It should contain all read informations. A precise example can be found in Appendix A. Stick to the defined key names. The *signals* field should be a dictionary on its own for the various signals. Use the names (e.g.: "ecg", oder "foot gsr") as key.

Erstellen Sie ein Dictionary für jede Header Datei, welches die gelesenen Informationen Speichert. Ein genaues Beispiel finden Sie in Appendix A. Halten Sie sich an die definierten Key Namen. Das *signals* Feld soll selbst ein eigenes Dictionary für die einzelnen Signale sein. Verwenden Sie hier die Namen der Signale (z.B.: "ECG", oder "foot GSR" als Key.

You do not need to verify the contents of the file (e.g.: check if the signal value in the first row is correct).

Important: When reading a file Python will (if not told otherwise) always return Strings. Convert all numbers to integers or floats and convert all strings to lower case strings (`.lower()`) to avoid inconsistencies. Use these names as keys for your dictionaries.

Create a dictionary, which has the names of the records (e.g.: "drive01") as keys and the matching dictionary as value (the ones you previously created per header file).

Hint: These files are in a plaintext format. You can open them with any text editor.

1.5. Save as Pickle (1pt)

Use the `pickle.dump()` function[4] to store the previously assembled dictionary. The name should be *ass_1.p*.

Note: pickle saves in a binary format. Thus, open the file with the correct parameters (*"wb"*, 'w' for write and 'b' for binary)

1.6. Content Summary (2pt)

Summarize the extraxted informations based on the signals. List which signals you found, how many records support each signal and list any records missing a signal. Make sure your output matches the example.

Draw a box of # around this output and introduce your listing with the text`SIGNALS:`. Sort both the records and the sensors by name. The whitspaces are structured as follows: 1

[4] `https://docs.python.org/3.6/library/pickle.html#usage`

before SIGNALS, 3 in front of every signal name. The padding between the signal names and the number of records should be 3 based on the longest signal name (e.g.: *foot gsr* requires 3 and *ecg* needs 8). The padding of digits should always be 2 (require 2 spaces even if the number has only 1 digit).

Important: Do not hard code any signal or record name..

Solution: (Using the full dataset)

```
########################################################################
# SIGNALS:
#   ecg        18 record(s), missing:
#   emg        15 record(s), missing: drive02, drive03, drive04
#   foot gsr   18 record(s), missing:
#   hand gsr   16 record(s), missing: drive02, drive13
#   hr         16 record(s), missing: drive03, drive14
#   marker     16 record(s), missing: drive01, drive03
#   resp       18 record(s), missing:
########################################################################
```

Hinweis: Use format strings[5] to simplify the creation of the output. The 72 `#` can be printed using `print("#"*72)`.

2. Packages

The following Python packages are allowed or required in this assigmnent.

- collections (defaultdicts[6] could be helpful!)
- os
- pickle
- sys

3. Restrictions

- Do not add any bloat to your submission
- Use built-in functions where possible
- Do NOT load extra packages (no imports)
- Do NOT use any command line arguments!

[5] `https://docs.python.org/3.6/library/stdtypes.html#str.format`

[6] `https://docs.python.org/3/library/collections.html#collections.defaultdict`

4. File Headers

All Python source files have to contain a comment header with the following information:

- Author: – Your name
- MatNr: – Your matriculate number
- Description: – Purpose of the file
- Comments: – Any comments as to why if you deviate from the task or restrictions

Please copy and paste the example and change its contents to your data. (Depending on your PDF-viewer you may have to fix the whitespaces)!

Example Header:

```
##############################################################################
# Author:      Patrick Kasper
# MatNr:       0730294
# Description: The main file. Assignment only has 1 file...
# Comments:    This is the example comment. I just made it a bit
#              longer so it spans across multiple lines.
##############################################################################
```

5. Coding Standard

For this lecture and the practicals we use PEP 8 [7] as a coding standard guideline. Please note that whilst we do not strictly enforce all these rules we will not tolerate messy or unreadable code. Please concentrate in particular on the following aspects.

Language Programming is done in English. This is to ensure someone else at the other end of the world can read your code. Please make sure all sources you submit are thus written in English. This covers both variable names and comments!

Spaces, not tabs. Indent your files with 4 spaces instead of tabs. PEP 8 forces this in Python 3 (whilst allowing more freedom in Python 2). Most editors have an option to indent with 4 spaces when you press the tab button!

Descriptive names. Use descriptive names for variables where possible! Whilst a simple i can be enough for a simple single loop code can become very messy really fast. If a variable has no purpose at all, use a single underscore (_) for its name.

Max 72 characters. Do not have lines longer than 72 characters. Whilst PEP 8 allows 79 in some cases we ask you to use the lower cap of 72 characters per line. Please note that this includes indentation.

[7] https://www.python.org/dev/peps/pep-0008/

6. Automated Tests

Your file will be executed with the following command:

```
>python assignment_1.py <path_to_records_file>
```

Please make sure your submission follows all the restrictions defined in this document. Your program will have a limited runtime of 2 minutes. If your submission exceeds this threshold, then it will be considered non-executable. Please note that this is a very generous amount of time for any of the tasks in this course.
If your submission fails any of the automated tests, we will look into your code to ensure the reason was not on our end and to evaluate which parts of your code are correct and awards points accordingly.

7. Abgabe

7.1. Deadline

14. November 2017 um 23:59:59

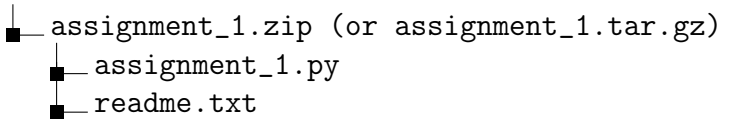
Any submission handed in too late will be ignored with the obvious exception of emergencies. In case the submission system is globally unreachable at the deadline it will be pushed back for 24 hours. If for whatever reason you are unable to submit your work, contact your tutor *PRIOR* to the deadline.

7.2. Uploading your submission

Assignment submissions in this course will always be archives. You are allowed to use .zip or .tar.gz formats.
In addition to your Python source files, please also pack a *readme.txt*. The file is mandatory but its contents are optional. In this file please write down how much time you spent on the assignment and where you ran into issues. Your feedback allows for immediate adjustments to the lecture and tutor sessions.
Upload your work to the Palme website. Before you do please double-check the following aspects:

- File and folder structure (see below)
- Comment header in every source file
- Coding Standard

7.3. Your Submission File



Please do **NOT** submit the dataset. The path will automatically be linked to your submission during tested!

Appendix A. Dictionary Example for `drive01`

Example Solution:

```
>>>parsed_records['drive01'] # parsed_records is just an example name
{
 'num_samples': 61499,
 'num_signals': 6,
 'record_name': 'drive01',
 'frequency': 15.5,
 'signals': {
  'ecg': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': 9084,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': -42,
   'samples_per_frame': 32,
   'unit': '1000'
  },
   'emg': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': 11155,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': 68,
   'samples_per_frame': 128,
   'unit': '10000'
  },
  'foot_gsr': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': -24751,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': 2503,
   'samples_per_frame': 2,
   'unit': '1000'
  },
  'hand_gsr': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': 20466,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': 11149,
   'samples_per_frame': 2,
   'unit': '1000'
  },
  'hr': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': 18582,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': 84,
   'samples_per_frame': 1,
   'unit': '1/bpm'
  },
  'resp': {
   'adc_resolution': 16,
   'adc_zero': 0,
   'block_size': 0,
   'checksum': -19336,
   'data_format': 16,
   'file_name': 'drive01.dat',
   'initial_value': 5474,
   'samples_per_frame': 1,
   'unit': '500'
  }
 }
}
```

Note: The keys for the signals (`signals`) contain whitespaces and **not** underscores (_)