SpeechLab Neural Data Format (SNDF)

26-Mar-11

v2

Neural and other data is stored in up to 2 types of .mat files containing the following variables:

1) **CONTINUOUS DATA** (CNT; "_cnt" mask at the end of the file name)

Required variables:

SampValues
sampled values of the continuous signal
numeric array [T x N]
in mV for voltages unless specified in *DataUnits* variable

SampFreq
sampling frequency of *SampValues*
numeric scalar [1 x 1]
in Hz

ChLbl
recording channel labels: names or numbers
cell vector of strings [1 x N]

SubjectID
subject identifier
char vector [1 x Z]

Log
history of the data processing that led to the current data file
cell array [X x Y]
consists of rows of cells corresponding to consecutive processing steps. The first cell in each row is the name of the script used to process the data; the second cell contains the date and time when the script was run; all other cells in the row can include script parameters and additional information if desired. The first cell in the first row has the path and the name of the original raw data file

Optional variables:

SampTimes
onset timestamps of continuous data fragments
numeric vector [F x 1]
in ms unless specified in *TimeUnits* variable

scripts will assume that data within fragments is sampled at *SampFreq* and that all the fragments are the same lengths unless there is a *FragLengths* variable (see below). If not present it is assumed that *SampTimes*=0

FragLengths
the lengths of the continuous data fragments
integer vector [F x 1]
in samples
the sum of the vector elements should be equal to the number of samples in *SampValues*

DataUnits
unit of measurement for continuous data
char vector [1 x Z]
default: “mV”. Other useful values: “a/u” (for arbitrary units)

TimeUnits
unit of measurement for time
char vector [1 x Z]
default: “ms”. Other useful values: “a/u” (for arbitrary units)

Notes
any additional information added by people running scripts on the data set
cell array [X x Y]
each row is a new entry; the first cell in the row has identifier of the person making the note; the second cell contains the date and time the note was inserted; all the following cells in the row can have any information in any form

2) **DISCRETE DATA** (DSC; “_dsc” mask at the end of the file name)

Required variables:

EvtTimes
timestamps of detected segment times
numeric array [Emax x NEC]
in ms unless specified in *TimeUnits* variable
Each column is sorted in the ascending order. NaN for empty cells

EvtID
numeric identifiers of classified data segments corresponding to *EvtTimes*
integer array [Emax x NEC]
NaN for empty cells. Values have to be natural numbers (no negative values or zeros) otherwise it is impossible to link it to *EvtLbl*

EvtLbl
labels of classified data segments corresponding to *EvtID*
cell array [EIDmax x 1]
empty for empty cells

Log
history of the data processing that led to the current data file
cell array [X x Y]
consists of rows of cells corresponding to consecutive processing steps. The first cell in each row is the name of the script used to process the data; the second cell contains the date and time when the script was run; all other cells in the row can include script parameters and additional information if desired. The first cell in the first row has the path and the name of the original raw data file

Optional variables:

SegValues
sampled values of data segments extracted from the continuous data around *EvtTimes*
numeric array [Emax x T1 x N]
in mV for voltages unless specified in *DataUnits* variable
NaN for empty cells. Assumed to contain continuous segments of data unless there is *SegMask* variable which says otherwise (see below). Must be paired with *ChLbl* variable

ChLbl
recording channel labels: names or numbers - in *SegValues* array corresponding to *ChLbl* variable in continuous data files
cell vector [1 x N]

SegSampFreq
sampling frequency of *SegValues*
numeric scalar [1 x 1]
in Hz

SegMask
mask to determine how each segment is extracted from the continuous data
integer vector [1 x T1]
in samples
Each number corresponds to the location of the signal to be extracted in the *SampValues* array relative to the *EvtTimes* value. Timestamp of the value from *EvtTimes* itself is coded by 0. In general, the extracted segments do not have to be continuous. Note: this variable can code only for the simplest type of data segment extraction when the values are taken from the *SampValues* array explicitly without mixing (as happens with filtering, alignment, up- and down-sampling, etc)

LinkedCntData

continuous data file(s) that the events have been extracted from / are associated with
cell vector of strings [1 x Z]
by default scripts will assume that the linked continuous data files are in the same directory as the discrete data file

DataUnits
unit of measurement for continuous data
char vector [1 x Z]
default: “mV”. Other useful values: “a/u” (for arbitrary units)

TimeUnits
unit of measurement for time
char vector [1 x Z]
default: “ms”. Other useful values: “a/u” (for arbitrary units), “idx” (for indices corresponding to the *SampValues* variable)

Notes
any additional information added by people running scripts on the data set
cell array [X x Y]
each row is a new entry; the first cell in the row has identifier of the person making the note; the second cell contains the date and time the note was inserted; all the following cells in the row can have any information in any form

where:
T - number of time samples
N - number of recording channels
X, Y, Z - arbitrary array dimensions
F - number of data fragments
Emax - maximum number of isolated events on a single recording channel. Arrays containing events from several recording channels can include NaN values at the end to indicate no segments for specific channels
NEC - number of event channels. Either NEC=N and then there is a one-to-one correspondence between event and recording channels in *EvtTimes* and *SegValues* or NEC=1 and then it is assumed that the event timestamps and identifiers are the same across all recording channels
EIDmax - value of the maximum event identifier in *EvtID*
T1 - number of time samples in each data segment

Scripts read/create the data files, add/modify variables, perform various analyses on the data, plot/save figures, and record changes in the *Log* variable. Scripts pushed to the common repository should be written as functions, be well commented, and have a brief description of their function, input/output variables, and parameters in the header. Common types of scripts:
- reading data from an external data format into the MATLAB variable structure described above (creation

of CNT and/or DSC data files)
- generating artificial neural signal (creation of CNT and/or DSC data files)
- data filtering and pre-processing (modifying variables and specifically *SampValues*)
- spike detection (creation of a DSC data file)
- spike sorting (creation of an *EvtID* variable)
- data visualization and analysis (using existing data files to run analyses and plot figures)

Flexibility of the data and script format (additional capabilities that do not require making changes to the already existing data scripts):
1) can easily add more variables to the data files after different processing streams such as:
- parameters of the filters used on the data
- values for spike detection thresholds
- parameters used for spike alignment
- features used for classification (names and values)
2) can have as many of each data type files per session as desired (from zero to infinity). The files of the same data type can readily be combined into one (usually data needs to be the same size and be sampled at the same frequency for this) or broken into more files
3) can store virtually any data in this format. Here are some examples of how to store different types of data:
- continuous extracellular, ECoG, or EEG voltages: CNT file
- spike waveforms for extracellular recordings, ERPs for EEG, or trials for any type of data: DSC file
- trial markers, bad data markers, or external events markers: DSC file
- eye movements, audio waveforms with bird songs, EMG activity, breathing belt signal: CNT file
4) thanks to the modular script structure, can easily write short, readable, and powerful wrapper scripts in a matter of minutes with no required debugging, once the lower level scripts are coded
5) powerful compatibility: different users can readily use each others scripts; the same scripts can be run on various data types without extensive tuning of the code

Best practices:
- store data files from different sessions in separate folders
- start the name of each data file with the session name and end with “_cnt” or “_dsc” depending on the included data
- try to put as much relevant information in the data file name as possible
- break down data files larger than 1GB into smaller chunks (by channel and if necessary by time) and, on the other hand, combine small pieces of data into one bigger array if possible (by grouping channels)
- start with one of existing script templates to write your own one
- use “varargin” and “varargout” cell arrays for passing variables in and out of the scripts
- use input parser tools (ip) to parse input parameters. Especially useful and flexible is the “ip.addParamValue” method
- create a parallel version of the script (making explicit use of MATLAB parallel tools) or write a MEX file if the script is taking a long time to run

- make the script display its current state (using “disp” function) and the current time (using “datestr(now)”)
- put as much info about the script into its header as possible
- consider moving as many of the fixed parameters as possible into the script inputs (with set defaults) for greater flexibility
- update the *Log* variable at the end of each script
- when loading MAT files with data read only the variables the script will need and load everything into one structure *data*
- save all variables on disk as separate items within one file rather than as fields of one structure for more flexible access
- commit and pull scripts often, push scripts into the common repository only after decent advances and a round of debugging
- save plotted figures in 2 formats: FIG and PNG
- in all relevant scripts give an option of not saving new data files to disk and only outputting them through function output variables
- run a universal data check script to check compatibility and to save a quick summary of all your data files

Existing scripts:
- data_check_v2: checks data files for conformity with SNDFv2
- script_template_v2: basic template for creating SNDFv2 functions
- plx2mat_v2: converts Plexon (.plx) files into SNDFv2
- nlx2mat: converts Neuralynx data files into SNDFv2
- data_re_referencing_v2: re-reference continuous data against an internal or external signal
- spike_detect: detects and extracts spikes from continuous data files

Proposed scripts:
- data visualization scripts for continuous and discrete data
- de-noising scripts
- scripts generating artificial neural data

DISCUSSION:
- switch to object-oriented programming?
- think what other data variables to include and which ones should be required vs optional (for example, filters)