QuickStart

Release:	1.0.0
Last updated:	June 25, 2015
Author:	Sergiy Yakovenko

Mac Win

BOXsci

query

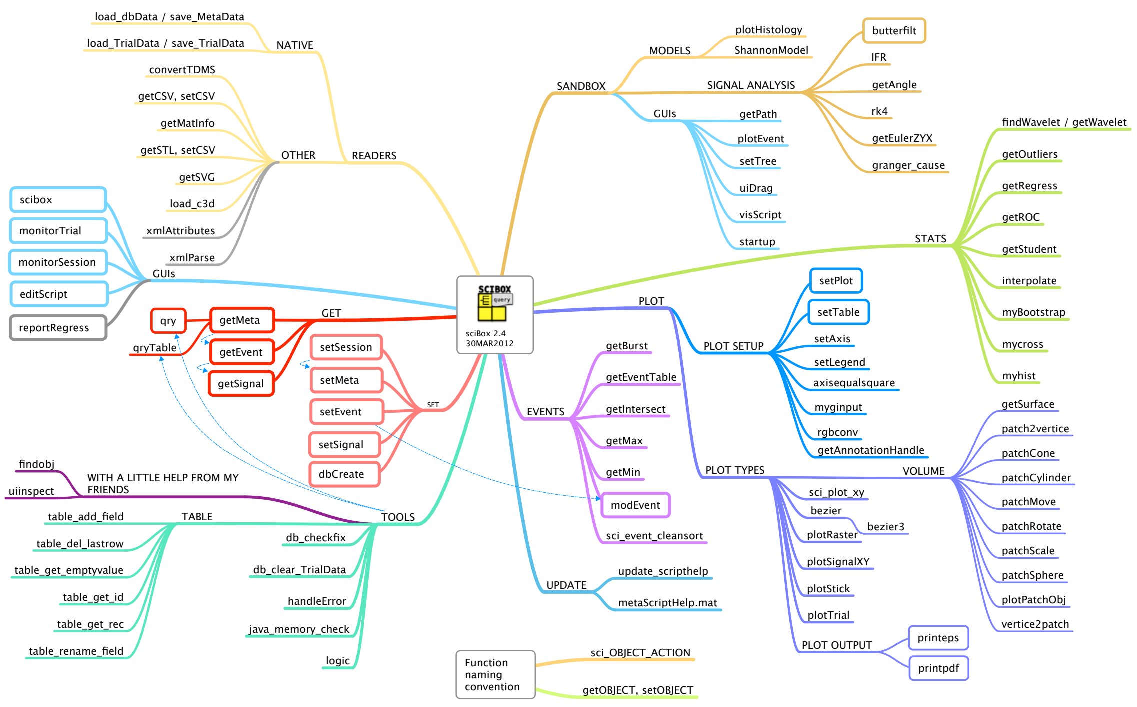
Web: https://sites.google.com/site/matBoxSite/

Table of Contents

1. Introduction

The **relational model** for database management is a database model based on first-order predicate logic, first formulated and proposed in 1969 by Edgar F. Codd. The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries. (source: http://en.wikipedia.org/wiki/Relational_model)

Figure. The complete layout of BOXsci functions where the backbone scripting functions are highlighted with box outlines.



Quick Q&A

Q: What is BOXsci for Matlab?

A: BOXsci is a relational database for analysis and storage of neurophysiological/biomechanical/biophysical data implemented *fully* in Matlab. Additional licenses for other toolboxes, external software or SQL skills are **not** required. All you need is a copy of basic Matlab and rudimentary knowledge about data types.

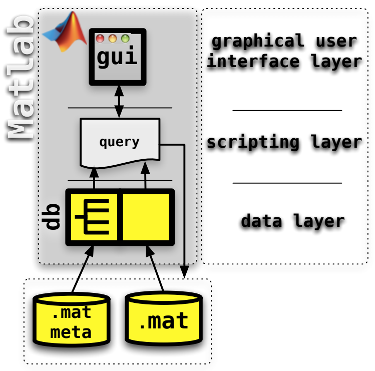


Figure. **Workflow schematic of BOXsci.** The application consists of several transparent levels of access. The relational database is represented by a global structure in workspace. It is dynamically manipulated to load and analyze data stored in MAT files.

Q: What are the features of BOXsci?
A: BOXsci simplifies access to complex scientific data sets. It allows three levels of data access (see Fig. 1.1):

1. The data layer is a low-level access to the database structure containing <*meta*> and <*signal*> tables with standard Matlab data inspectors, which allow independent from SciBox data manipulation for users with advanced programming skills. Any additional <*meta*> information that satisfies few organizational constraints can be appended directly to the database structure and becomes part of it.

2. The scripting layer allows data queries that can execute on-the-fly signal processing analysis that accepts any Matlab script, and plots intermediate and final results. The group of essential query scripts consists of only three functions **getMeta**, **getEvent**, **getSignal**. These functions reflect three basic steps in data analysis flow of standard analysis of neurophysiological data (see Examples of Data Analysis of two projects below: the studies of the motor cortex contribution to movement control in cats and humans from two laboratories in University of Montreal).

3. The graphical user interface (**GUI**) layer allows users without programming skills to access and analyze data in two typical views: a trial-by-trial view of multiple signals or a multiple sessions with multiple trials view of several signals. In these data views, events and signals can be manipulated manually with a drag-and-drop interface or with automatic event detection scripts, e.g. see example of the automatic burst detection during locomotion (#.#). Additional report generators enable users to perform typical analyses on the whole database, e.g. averages of signals, raster plots of neuron spiking, temporal and magnitude regressions.

Q: Is this toolbox specific to neurophysiological studies?
A: BOXsci is a generic tool for studies with multiple subjects, sessions, and trials. Its database architecture is flexible enough to accommodate any particular experimental needs with little overhead.

Q: What are BOXsci system requirements?
A: BOXsci is built and tested in Matlab. However, it can be used in conjunction with free Matlab runtime at the cost of flexibility with scripting and troubleshooting. Matlab version higher than 2008a should allow you to run scripting and graphical components. BOXsci runs on both Win and Mac computers.

See also:
getMeta()
getEvent()
getSignal()

Q: What are the current problems?
A: The package is undergoing rapid development and improvement. The topics under development are:
- speeding up 70 ms/trial access speeds due to use of **load**('filename.mat') function (suggestions are welcome)
- implementation of "smart" vertical queries in the database, e.g. query constraints applied to several <*meta*> tables at different level of trial-session-subject hierarchy.

Matlab script

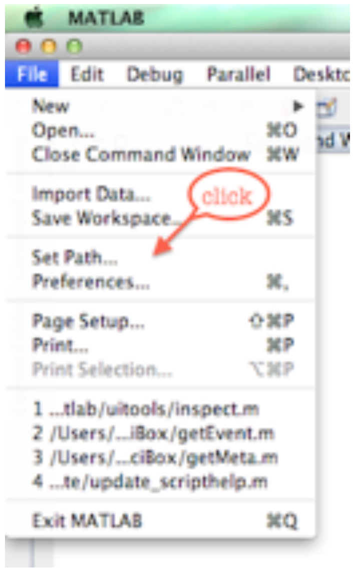
```
% load EMG table [24 signals @ 1kHz] for 10 trial
>> tic; nData = getSignal(idTrialList,idSignal,tPeriod); toc
Elapsed time is 0.703859 seconds.

% repeated call does not load data from hard drive
>> tic; nData = getSignal(idTrial,idSignal,tPeriod); toc
Elapsed time is 0.013308 seconds.
```

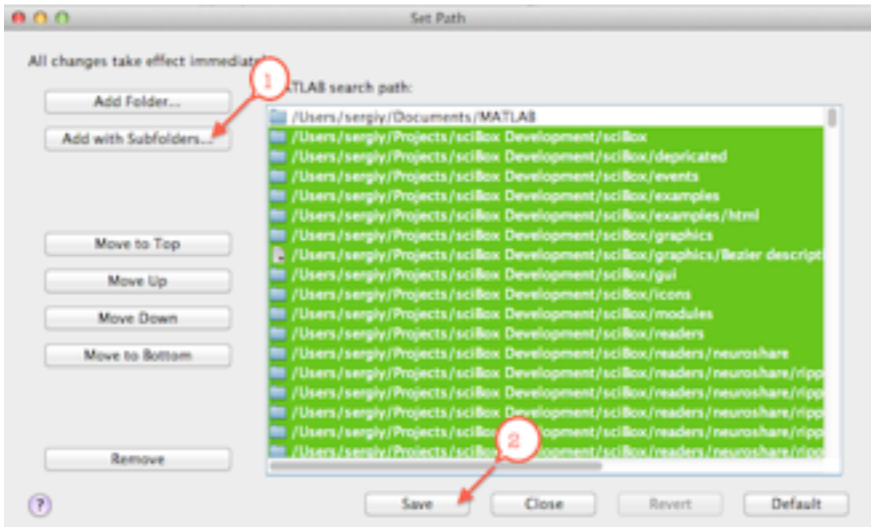
2. Installation Instructions & Requirements

TO INSTALL STANDALONE APP ON MAC/PC
1) copy APP/EXE to a folder on your computer;
2) install MCR framework indicated in the accompanying README wEXAMPLE.TXT
3) run APP/EXE

TO INSTALL M-SCRIPT MATLAB TOOLBOX ON MAC/PC
1) copy BOXsci to a folder on your computer;
2) navigate to the folder with BOXsci in Matlab and run install_boxsci.m
or
2) add & save path to BOXsci
in Matlab open path pop-up: File / Set Path …
Press button *Add with Subfolders…*
Select BOXsci folder
Press Save in the path pop-up.
Type in Matlab Command Window > boxsci

See also:
getSignal()





```
/Users/sergiy/Documents/MATLAB
/Users/sergiy/Projects/sciBox Development/sciBox
/Users/sergiy/Projects/sciBox Development/sciBox/depricated
/Users/sergiy/Projects/sciBox Development/sciBox/events
/Users/sergiy/Projects/sciBox Development/sciBox/examples
/Users/sergiy/Projects/sciBox Development/sciBox/examples/html
/Users/sergiy/Projects/sciBox Development/sciBox/graphics
/Users/sergiy/Projects/sciBox Development/sciBox/graphics/Bezier descript
/Users/sergiy/Projects/sciBox Development/sciBox/gui
/Users/sergiy/Projects/sciBox Development/sciBox/icons
/Users/sergiy/Projects/sciBox Development/sciBox/modules
/Users/sergiy/Projects/sciBox Development/sciBox/readers
/Users/sergiy/Projects/sciBox Development/sciBox/readers/neuroshare
/Users/sergiy/Projects/sciBox Development/sciBox/readers/neuroshare/ripp
/Users/sergiy/Projects/sciBox Development/sciBox/readers/neuroshare/ripp
/Users/sergiy/Projects/sciBox Development/sciBox/readers/neuroshare/ripp
/Users/sergiy/Projects/sciBo            opment/sciBox/readers/neuroshare/ripp
/Users/sergiy/Projects/sciBo            opment/sciBox/readers/neuroshare/ripp
```

Disclaimer: BOXsci toolbox for Matlab may require Matlab 2011b or later.

Some current features and limitations of BOXsci

FEATURES
- it is very stable
- it handles data of over 150 GB (with the possibility of access over network)
- BOXsci handles analog data with different sampling and synchronization, time-stamp data and 3D kinematic data
- interactive control and analysis of data that does not require Matlab knowledge
- Excel XLS/CSV integration
- user-friendly input functions

LIMITATIONS
- data size per trial in each table is limited to the available RAM (e.g. 4GB)

Additional features in development

- multidimensional data (images) and analysis (segmentation, etc)
- fast-queries with an optional add-on interface with MySQL databases
- simplified C3D importer
- interface with OpenSim, SimTK file formats

Send feedback, suggestions, and requests to sciBox.toolbox@gmail.com. Contact me if you need help to import your data.

If you are a novice Matlab user, take introductory description of Matlab environment and variables in workspace.

The required basic concepts:

- Matlab interface: workspace, editor, help
- *variable, function*
- variable type: *numeric, string, cell, and structure*
- programming flow: *for-loop, if-then logical statement*

Homework 1

Create a numeric array ***t*** from 0 to 2*π and plot ***y(t) = sin(t)*** so that positive values are *red* and negative values are *blue*. Use functions ***sin***, ***plot***.

Homework 2

Create a dataset ***x(t) = noise*** and ***y(t) = sin(t)*noise***. Create a function that returns elements of ***y(t)*** that are over a specified threshold, so that **z = *myfunc(y,thr);*** where ***z>thr***.
Use functions ***rand, sin, for-end, if-then-end, max***. Note that the problem can be solved without ***for-end*** loop.

Please, consider sending feedback about your experience to sergiy.yakovenko@gmail.com

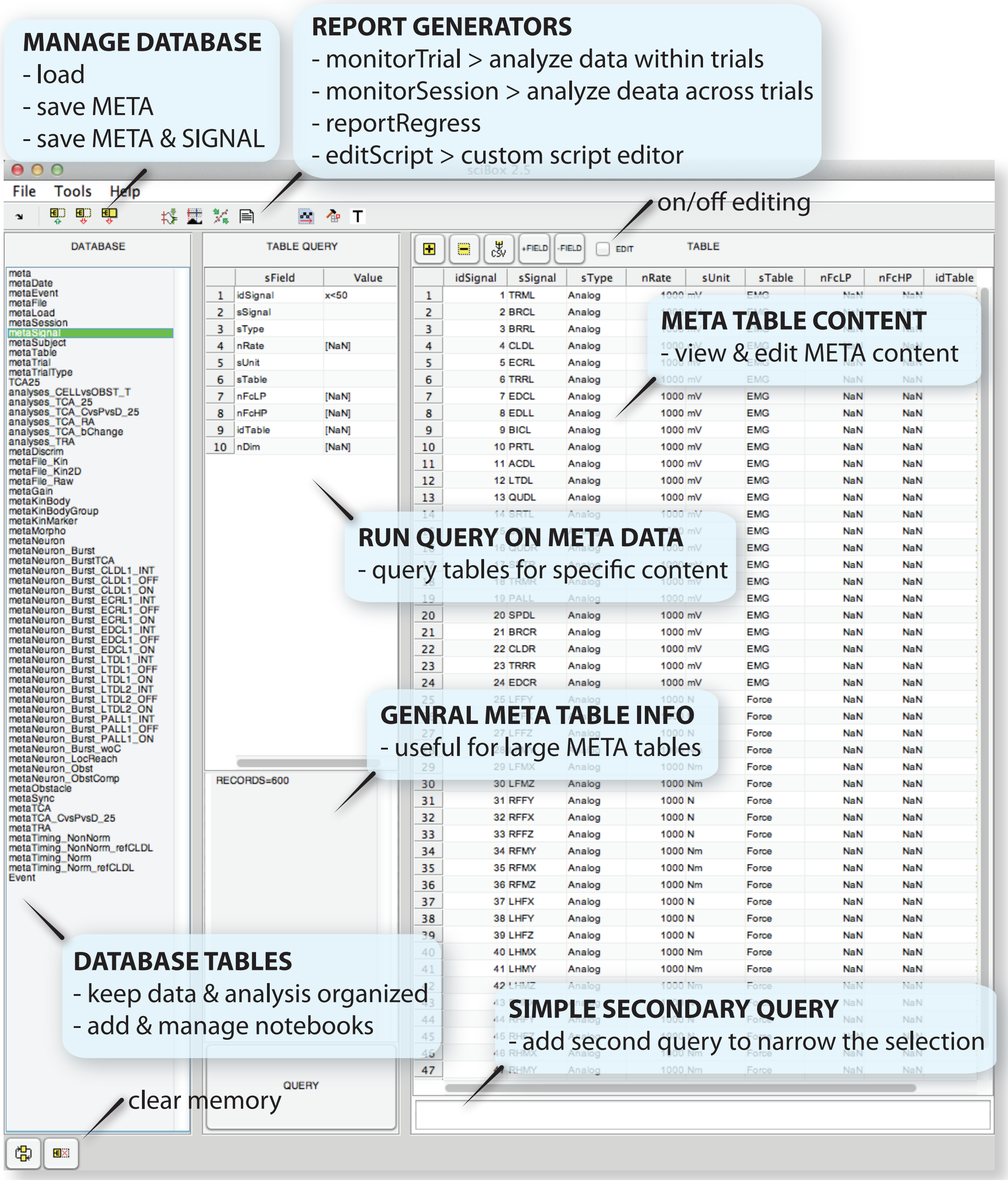
3. GUIs at a Glance • Drag & drop level of access

Graphical User Interface (GUI) consists of several types of report generators that enable users to view and analyze data within a single trial or across many trials, sessions or subjects. The interactions with data are intuitive and streamlined. Graphical objects displayed in reports link data in different reports for easy troubleshooting and visualization of multi-dimensional datasets.

BOXsci - boxsci

Access all meta data information and GUI tools from one window.

Figure 2.1. Main - **boxsci .** Access all information in ***dbData*** database structure.



DATABASE

meta
metaDate
metaEvent
metaFile
metaLoad
metaSession
metaSignal
metaSubject
metaTable
metaTrial
metaTrialType
TCA25
analyses_CELLvsOBST_T
analyses_TCA_25
analyses_TCA_CvsPvsD_25
analyses_TCA_RA
analyses_TCA_bChange
analyses_TRA
metaDiscrim
metaFile_Kin
metaFile_Kin2D
metaFile_Raw
metaGain
metaKinBody
metaKinBodyGroup
metaKinMarker
metaMorpho
metaNeuron
metaNeuron_Burst
metaNeuron_BurstTCA
metaNeuron_Burst_CLDL1_INT
metaNeuron_Burst_CLDL1_OFF
metaNeuron_Burst_CLDL1_ON
metaNeuron_Burst_ECRL1_INT
metaNeuron_Burst_ECRL1_OFF
metaNeuron_Burst_ECRL1_ON
metaNeuron_Burst_EDCL1_INT
metaNeuron_Burst_EDCL1_OFF
metaNeuron_Burst_EDCL1_ON
metaNeuron_Burst_LTDL1_INT
metaNeuron_Burst_LTDL1_OFF
metaNeuron_Burst_LTDL1_ON
metaNeuron_Burst_LTDL2_INT
metaNeuron_Burst_LTDL2_OFF
metaNeuron_Burst_LTDL2_ON
metaNeuron_Burst_PALL1_INT
metaNeuron_Burst_PALL1_OFF
metaNeuron_Burst_PALL1_ON
metaNeuron_Burst_woC
metaNeuron_LocReach
metaNeuron_Obst
metaNeuron_ObstComp
metaObstacle
metaSync
metaTCA
metaTCA_CvsPvsD_25
metaTRA
metaTiming_NonNorm
metaTiming_NonNorm_refCLDL
metaTiming_Norm
metaTiming_Norm_refCLDL
Event

TABLE QUERY

	sField	Value
1	idSignal	x<50
2	sSignal	
3	sType	
4	nRate	[NaN]
5	sUnit	
6	sTable	
7	nFcLP	[NaN]
8	nFcHP	[NaN]
9	idTable	[NaN]
10	nDim	[NaN]

RECORDS=600

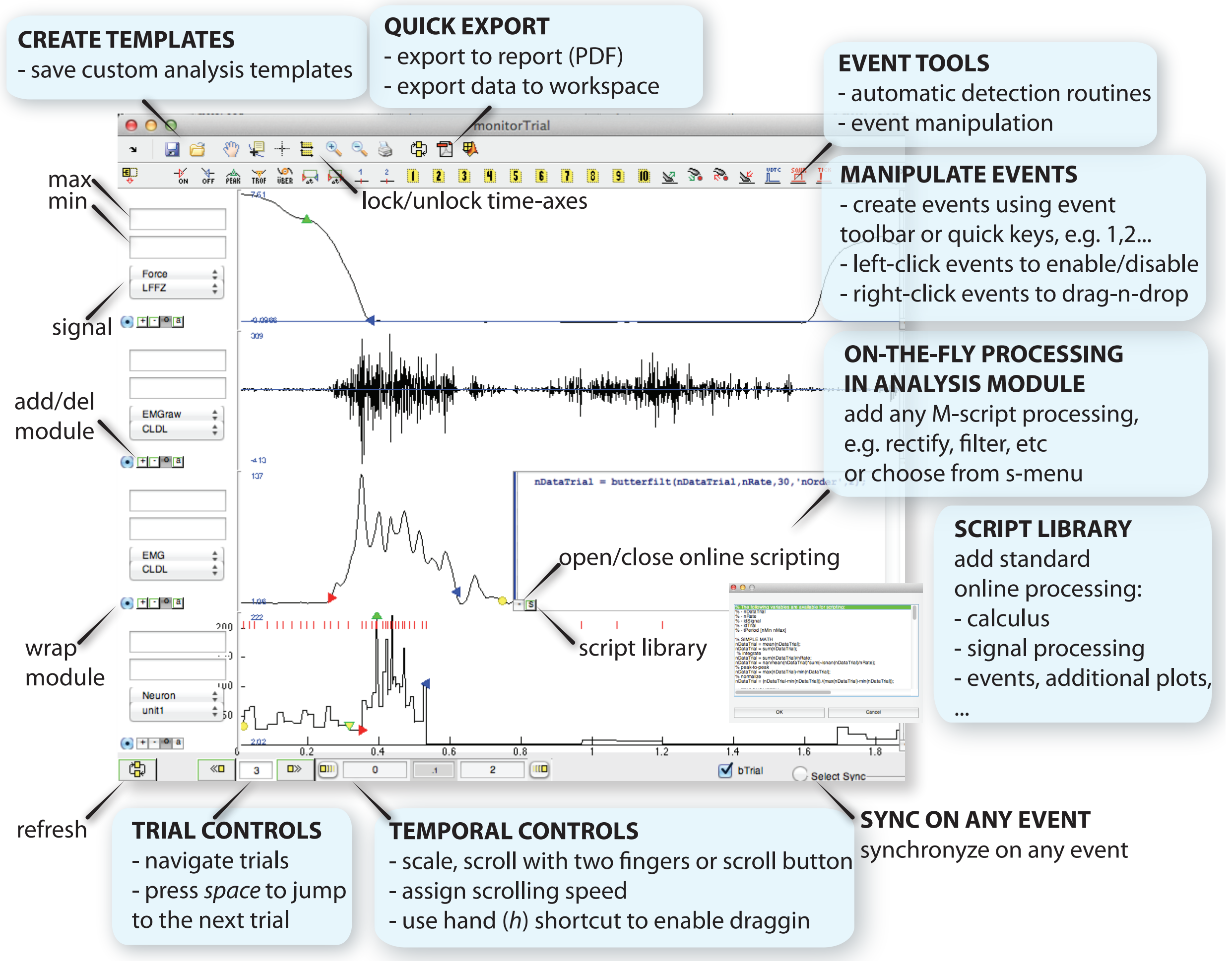
TABLE

	idSignal	sSignal	sType	nRate	sUnit	sTable	nFcLP	nFcHP	idTable
1	1	TRML	Analog	1000	mV	EMG	NaN	NaN	
2	2	BRCL	Analog	1000	mV	EMG	NaN	NaN	
3	3	BRRL	Analog	1000	mV	EMG	NaN	NaN	
4	4	CLDL	Analog	1000	mV	EMG	NaN	NaN	
5	5	ECRL	Analog	1000	mV	EMG	NaN	NaN	
6	6	TRRL	Analog	1000	mV	EMG	NaN	NaN	
7	7	EDCL	Analog	1000	mV	EMG	NaN	NaN	
8	8	EDLL	Analog	1000	mV	EMG	NaN	NaN	
9	9	BICL	Analog	1000	mV	EMG	NaN	NaN	
10	10	PRTL	Analog	1000	mV	EMG	NaN	NaN	
11	11	ACDL	Analog	1000	mV	EMG	NaN	NaN	
12	12	LTDL	Analog	1000	mV	EMG	NaN	NaN	
13	13	QUDL	Analog	1000	mV	EMG	NaN	NaN	
14	14	SRTL	Analog	1000	mV	EMG	NaN	NaN	
15	15	[illegible]	[illegible]	1000	mV	EMG	NaN	NaN	
16	16	QUDR	Analog	1000	mV	EMG	NaN	NaN	
17	17	SPTR	Analog	1000	mV	EMG	NaN	NaN	
18	18	TRMR	Analog	1000	mV	EMG	NaN	NaN	
19	19	PALL	Analog	1000	mV	EMG	NaN	NaN	
20	20	SPDL	Analog	1000	mV	EMG	NaN	NaN	
21	21	BRCR	Analog	1000	mV	EMG	NaN	NaN	
22	22	CLDR	Analog	1000	mV	EMG	NaN	NaN	
23	23	TRRR	Analog	1000	mV	EMG	NaN	NaN	
24	24	EDCR	Analog	1000	mV	EMG	NaN	NaN	
25	25	LFFY	Analog	1000	N	Force	NaN	NaN	
26	26	[illegible]	[illegible]	[illegible]	[illegible]	Force	NaN	NaN	
27	27	LFFZ	Analog	1000	N	Force	NaN	NaN	
28	28	[illegible]	[illegible]	[illegible]	[illegible]	Force	NaN	NaN	
29	29	LFMX	Analog	1000	Nm	Force	NaN	NaN	
30	30	LFMZ	Analog	1000	Nm	Force	NaN	NaN	
31	31	RFFY	Analog	1000	N	Force	NaN	NaN	
32	32	RFFX	Analog	1000	N	Force	NaN	NaN	
33	33	RFFZ	Analog	1000	N	Force	NaN	NaN	
34	34	RFMY	Analog	1000	Nm	Force	NaN	NaN	
35	35	RFMX	Analog	1000	Nm	Force	NaN	NaN	
36	36	RFMZ	Analog	1000	Nm	Force	NaN	NaN	
37	37	LHFX	Analog	1000	N	Force	NaN	NaN	
38	38	LHFY	Analog	1000	N	Force	NaN	NaN	
39	39	LHFZ	Analog	1000	N	Force	NaN	NaN	
40	40	LHMX	Analog	1000	Nm	Force	NaN	NaN	
41	41	LHMY	Analog	1000	Nm	Force	NaN	NaN	
42	42	LHMZ	Analog	1000	Nm	Force	NaN	NaN	
43	43	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NaN	NaN	
44	44	RHFY	Analog	1000	N	Force	NaN	NaN	
45	45	RHFZ	Analog	1000	N	Force	NaN	NaN	
46	46	RHMX	Analog	1000	Nm	Force	NaN	NaN	
47	47	RHMY	Analog	1000	Nm	Force	NaN	NaN	

BOXsci - monitorTrial

Review and edit information across signals in single trials (see monitorSession for the different data cross-section view). Activate this GUI by clicking its icon in BOXsci Main toolbar or typing *monitorTrial* in the command line.

Figure 2.2. **monitorTrial.** This tool has streamlined interactive interface to navigate between (buttons or LEFT/RIGHT keyboard keys) and within trials (buttons or SHIFT-LEFT/RIGHT keyboard keys), to create new events (toolbar buttons or 1:9 keys, ALT-1:9 keys). Tables and signals can be added and the template of the tool can be saved for future analysis.

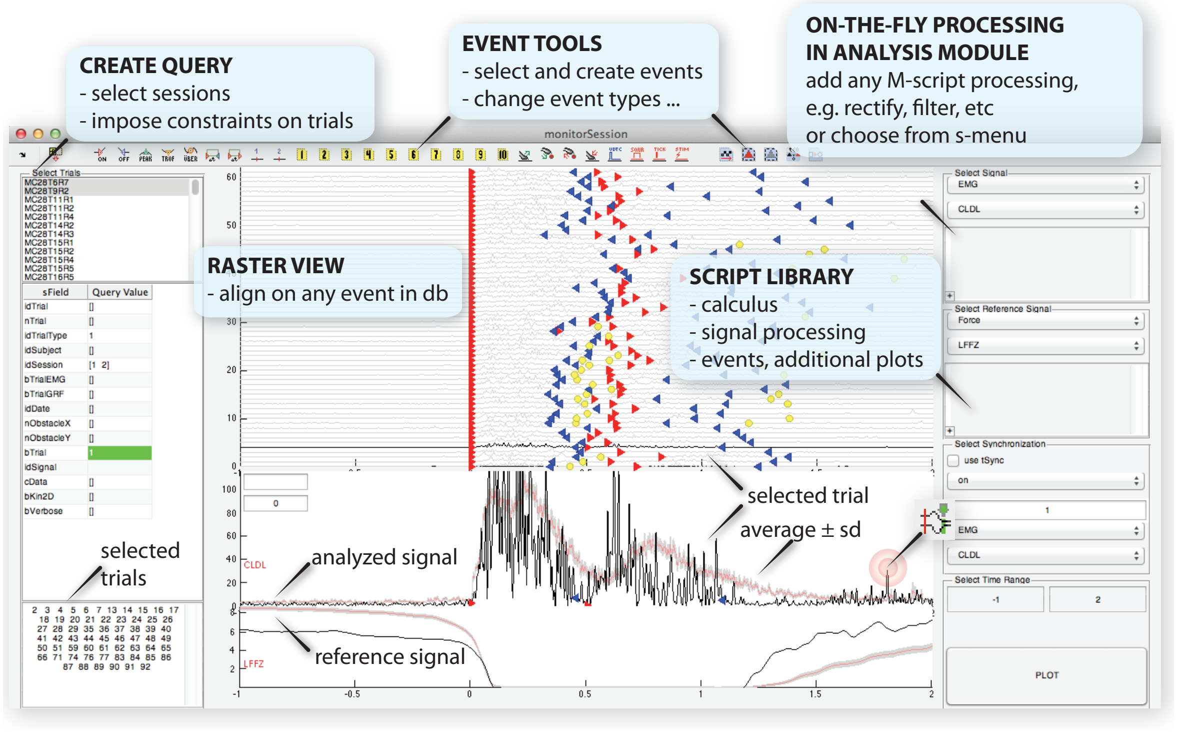

CREATE TEMPLATES
- save custom analysis templates
QUICK EXPORT
- export to report (PDF)
- export data to workspace
EVENT TOOLS
- automatic detection routines
- event manipulation
monitorTrial
max
min
signal
lock/unlock time-axes
MANIPULATE EVENTS
- create events using event toolbar or quick keys, e.g. 1,2...
- left-click events to enable/disable
- right-click events to drag-n-drop
Force
LFFZ
add/del module
EMGraw
CLDL
ON-THE-FLY PROCESSING IN ANALYSIS MODULE
add any M-script processing, e.g. rectify, filter, etc
or choose from s-menu
EMG
CLDL
open/close online scripting
SCRIPT LIBRARY
add standard online processing:
- calculus
- signal processing
- events, additional plots,
...
script library
wrap module
Neuron
unit1
OK
Cancel
refresh
bTrial
Select Sync
TRIAL CONTROLS
- navigate trials
- press *space* to jump to the next trial
TEMPORAL CONTROLS
- scale, scroll with two fingers or scroll button
- assign scrolling speed
- use hand (*h*) shortcut to enable draggin
SYNC ON ANY EVENT
synchronyze on any event

```
nDataTrial = butterfilt(nDataTrial,nRate,30,'nOrder'
```

BOXsci - monitorSession

Review and edit information across sessions for the signal of interest. Activate this GUI by clicking its icon in BOXsci Main toolbar or typing *monitorSession* in the command line.

Figure 2.3. **monitorSession.** This tool has streamlined interactive interface to navigate whole sessions. It allows fast event annotation.



Select Trials
MC28T6R7
MC28T9R2
MC28T11R1
MC28T11R2
MC28T11R4
MC28T14R2
MC28T14R3
MC28T15R1
MC28T15R2
MC28T15R4
MC28T15R5
MC28T16R5

sField	Query Value
idTrial	[]
nTrial	[]
idTrialType	1
idSubject	[]
idSession	[1 2]
bTrialEMG	[]
bTrialGRF	[]
idDate	[]
nObstacleX	[]
nObstacleY	[]
bTrial	1
idSignal	
cData	[]
bKin2D	[]
bVerbose	[]

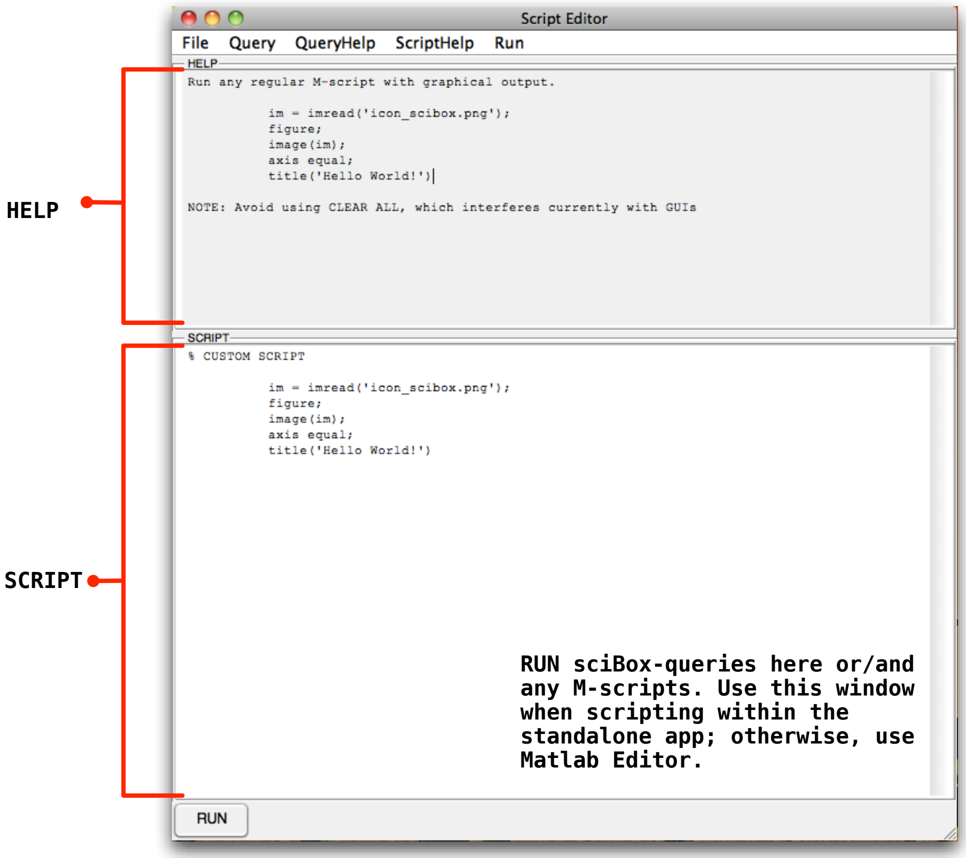
```
2 3 4 5 6 7 13 14 15 16 17
18 19 20 21 22 23 24 25 26
27 28 29 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49
50 51 59 60 61 62 63 64 65
66 71 74 76 77 83 84 85 86
87 88 89 90 91 92
```

Select Signal: EMG; CLDL
Select Reference Signal: Force; LFFZ
Select Synchronization: ☐ use tSync; on; 1; EMG; CLDL
Select Time Range: -1; 2

BOXsci - editScript

Write and run BOXsci scripts and analysis. This tool will be enhanced in the future releases. One of the ideas is to add interactive scripting and graphical workflows. Activate this GUI by clicking its icon in BOXsci Main toolbar or typing *editScript* in the command line.

Figure 2.4. **editScript.** This tool allows the execution of custom scripts in compiled version of standalone package.



```
Run any regular M-script with graphical output.

          im = imread('icon_scibox.png');
          figure;
          image(im);
          axis equal;
          title('Hello World!')

NOTE: Avoid using CLEAR ALL, which interferes currently with GUIs
```

```
% CUSTOM SCRIPT

          im = imread('icon_scibox.png');
          figure;
          image(im);
          axis equal;
          title('Hello World!')
```

RUN sciBox-queries here or/and any M-scripts. Use this window when scripting within the standalone app; otherwise, use Matlab Editor.

BOXsci - reportRegress

Create custom report of averages and regressions using this tool. Activate this GUI by clicking its icon in BOXsci Main toolbar or typing *reportRegress* in the command line.

The description is needed…

4. Scripting cascade with getMeta, getEvent, getSignal • Scripting-level access (data retrieval)

All database data can be access with only three functions that are part of the processing cascade: getMeta, getEvent, and getSignal. Any analysis flow is easy to describe as a sequence of meta to signal information retrieval and on the fly data manipulation, e.g. averaging or any statistical measures. It is typical to start with defining i) trials of interest with getMeta, ii) period of interest with getEvent, and iii) analog data from a signal of interest with getSignal. Note that any regular Matlab script can be embedded in the signal query to speed-up analysis.

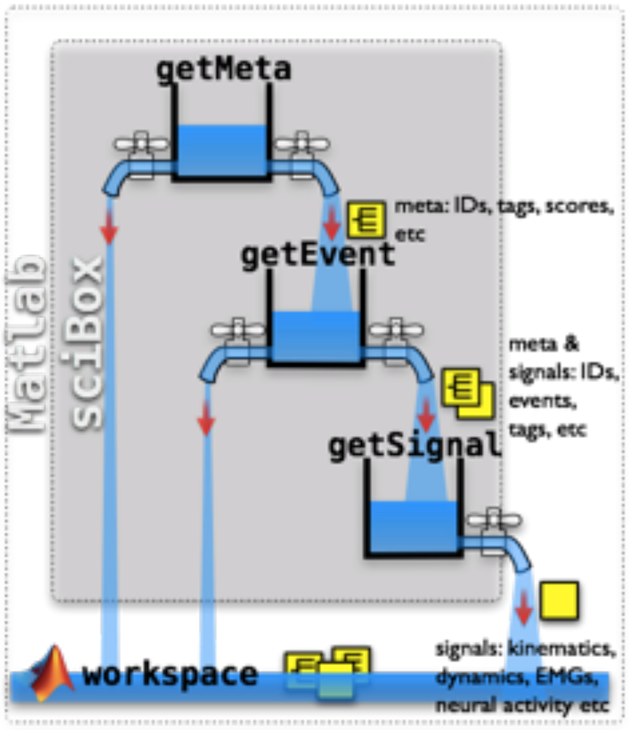

getMeta
meta: IDs, tags, scores, etc
getEvent
meta & signals: IDs, events, tags, etc
getSignal
signals: kinematics, dynamics, EMGs, neural activity etc
workspace
Matlab
sciBox

Figure. **Scripting cascade.** Three main functions of BOXsci to query and analyze signals.

getMeta()

Access meta data that satisfies query constraints. This function has a standard form:

result = **getMeta**(*whatTable*, *Query with Constraints*, *whatField*)

```
EXAMPLE 1
Table.name     = {'Peter','Jane','Spock','Kate'};
Table.id       = 1:4;
Table.bLogical = [false false true false];
% find the name of logical subject
getMeta(Table,qry('bLogical',true),'name')
>> ans = 'Spock'

EXAMPLE 2
getMeta('metaTrial',qry('bTrial',1,'idTrial',1:10))
```

is equivalent to a query:

```
bTrial==1 AND (idTrial==1 OR idTrial==2 OR ... idTrial==10)
```

which returns IDs of trials from 1 to 10 with bTrial field ==1

```
LOGICAL OPERATORS ARE SUPPORTED FOR NUMERICAL FIELDS
         use 'x' to denote a field variable e.g.
         ...'nObstacleX','x~=0 & x<10',...
MASK *string* IS SUPPORTED FOR CELL-STRING FIELDS:
         ...'sName','*Dr.Who*',...
```

NOTE if query is empty then return all requested values in table for a single field or a structure of with fields defined in **sFieldOutput** (see: help getMeta)

```
EXAMPLE 3
sNeuron = getMeta('metaNeuron', ...
   qry('sNeuron','MC29T14R1U2'),'sNeuron')

returns 'MC29T14R1U2'
```

getEvent()

```
Retrieve events.

EXAMPLE
% PLOT A HISTOGRAM OF TRRL EVENTS
% get ID of trials that satisfy constraints

idTrialList = getMeta('metaTrial',qry(...
   'idSession',1:13,'idTrialType',1,'bTrial',1));
sEventType = 'trough';
nEvent = 1;
% get events in trials
tTime = getEvent(idTrialList,qry('sTable','EMG','sSignal','TRRL'),...
sEventType,nEvent);
% find outliers using IQ and plot data
bTime = getOutliers(tTime,'bPlot',1);
```

getSignal()

This function simplifies the process of importing and updating of analog signals in the database. It creates new data tables and signals automatically cross-linking all essential information in metaTable, metaSignal, metaTrial tables. Note that a complimentary function setSession helps to create metaSubject, metaSession entries and link them to metaDate table.

```
nData = getData(idTrialList, Signal, tPeriod,... 'bPlot',1,'bPlotSD',1).
```

NOTE: @checkOkMemoryDump relies on a subjective amount of memory allocated for Matlab (default: 500MB).

Scripting details can be found online sciBox Toolbox http://sites.google.com/site/matBoxSite/

5. Examples

Example. Data access in the study of motor cortex contribution to movement control in cats.

The example M-script below demonstrates how to extract meta, discrete and analog data and how to generate a report. First, find ids of trials and signals of interest.

```
idTrial = getMeta('metaTrial', qry('idTrialType',1,'idSession',1));
idCELL = getMeta('metaSignal', qry('sTable','Neuron','sSignal','unit1'));
```

Next, query database for timestamp events of spikes synchronized on the onset of cleidobrachialis (CLDL), elbow flexor.

```
tCLDL = getEvent(idTrial, qry('sTable','EMG','sSignal','CLDL'), 'on', 1);
tCELL = getEvent(idTrial, idCELL, 'spike1', []);
tCELL = tCELL - ones(size(tCELL,1),1)*tCLDL;
% sync on tCLDL
```

Then, generate a report with an average profile of cell activity and the corresponding raster plot.

```
tLim = [-1 2]; % period of interest spans 1 sec before CLDL onset and 2 sec after the
event
tPeriod = [tCLDL'+tLim(1), tCLDL'+tLim(2)];
figure;
hPlot(1) = subplot(2,1,1);
hPlot(2) = subplot(2,1,2);
data.CELL = getSignal(idTrial,idCELL,tPeriod,...
     'tLim', tLim, 'bPlot', 1, 'hPlot',hPlot(1))
plotRaster(tCELL, tLim, hPlot(2), 'tLim', tLim); % add raster plot
```

The result is now plotted in the report figure and the data for each trial is available in Matlab workspace.
data.CELL: [3001x49 single]

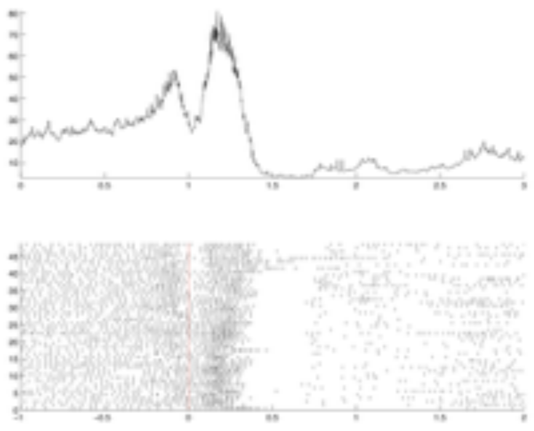


Fig. Graphical output of data query can be used for generating automatic reports.

Example. Data access in the study of motor cortex contribution to movement control in humans.

The example M-script below demonstrates query of EMG responses evoked by transcranial magnetic stimulation (TMS) in a human subject. Note that id signal 23 corresponds to

```
idSignal = getMeta('metaSignal',qry(...
  'sTable','EMG','sSignal','triceps lateral'));
idTrial  = getMeta('metaTrial',qry(...
  'idSubject',14,'bTrial',1,...
  'idTrialType',4,'idSignal',idSignal));
tTMS     = getEvent(idTrial,qry(...
  'sTable','Kinraw','sSignal','Right_HandY'),...
  'TMS',1);
tLim     = [-0.005 0.06];
tPeriod  = [tTMS'+tLim(1),tTMS'+tLim(2)];
nData    = getSignal(idTrial,idSignal,tPeriod,...
  'bSuperimposed',1,'bPlot',1,...
  'cData',[0,0,1],'tLim',tLim);
```

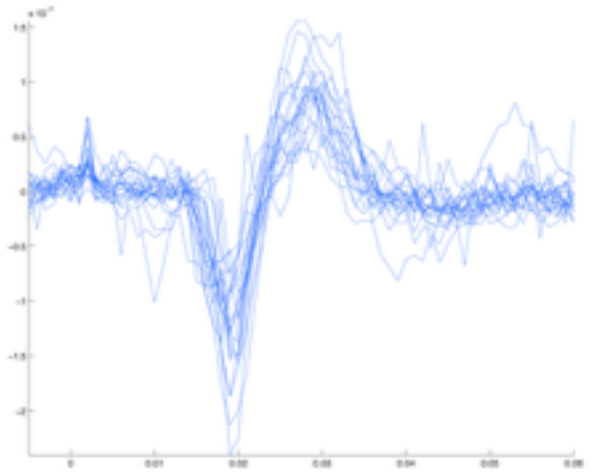


Fig. Graphical output of data query showing 22 superimposed trials with TMS-evoked responses at about 14ms after stimulus (time 0). The units of x-axis are in sec.

6. Data import • Data editing/appending

The dataset import may be the most challenging task that the user has to perform. Every experimental dataset may be different between different studies or even sessions in the same study. BOXsci has two levels of tools to help the process, the spreadsheet import and the script-based import.

Importing using spreadsheets

See Documentation: BOXsci - General Importer

Importing using custom scripts

The functions *dbCreate*, *setMeta*, *setSession*, *setEvent*, and *setSignal* simplify and streamline import and update of data in the database. The process of the database creation and import can be simplified to several lines of code.

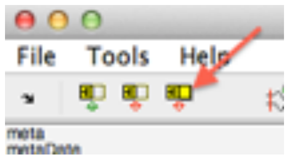
See the following example.

Example. Create and import dummy data using scripts

```
% EXAMPLE - DATA IMPORT
% 1) Create empty dbData
% 2) Insert META INFORMATION new subject, new session
% 3) Import analog data
% 4) Launch BOXsci
%
% NOTE: To save data set bSave property of setData to TRUE/1
% e.g. setData(... 'bSave',1)
%
% See Also: setSession, setMeta, setData, setEvent

clear global dbData
global dbData
dbData      = dbCreate; % create empty database structure

% 1) CREATE DATA (or import instead)
nRate       = 1000;     % sampling rate
tSim        = 10;
timeDummy   = single(1/nRate:1/nRate:tSim)'; % TIME
dataDummy1  = 10*single(cos(timeDummy*2*pi)>.5) + randn(size(timeDummy)); % EMG1
dataDummy2  = 10*single(sin(timeDummy*2*pi)>.5) + randn(size(timeDummy)); % EMG2
figure
plot(timeDummy,dataDummy1,'r',timeDummy,dataDummy2,'b')

% 2) CREATE SUBJECT & SESSION
idSession   =
setSession('sSubject','Subject1','sPrefix','S1','sSession','TEST1');
% 3) ADD SIGNAL
setSignal(timeDummy,qry('sTable','Misc','sSignal','Time'),1,...
   'nRate',nRate,...
   'sUnit','sec',...
   'idSession',idSession,...
   'idTrialType',3,...
   'tSync',0);
setSignal(dataDummy1,qry('sTable','EMG','sSignal','Agonist'),1,...
   'nRate',nRate,...
   'sUnit','sec',...
   'idSession',idSession,...
   'idTrialType',3,...
   'tSync',0);
% ADD SIGNAL
setSignal(dataDummy2,qry('sTable','EMG','sSignal','Antagonist'),1,...
   'nRate',nRate,...
   'sUnit','sec',...
   'idSession',idSession,...
   'idTrialType',3,...
   'tSync',0);

% 4) LAUNCH GUI
BOXsci
```

Any events can now be attached to this signal using *setEvent*, and any other information related to this trial should be added using setMeta.

It is recommended to save incrementally imported data before the database memory usage exceeds 1.5GB. This level is set to trigger automatic memory dump that clears all *Loaded* analog signals in memory. The data can be saved to disk by setting *bSave* property of *setSignal* to TRUE (see details in help setSignal) or by clicking on the toolbar button *Save Meta & Signal Data*.


File
Tools
Help
meta
metaData