

# BXH/XCEDE tools (1.8.16) user manual

Syam Gadde  
Duke-UNC Brain Imaging and Analysis Center (BIAC)  
Duke University  
Durham, NC

September 22, 2006

## Contents

<b>I</b>	<b>BXH/XCEDE Tools User Manual</b>	<b>1</b>
0.1	Introduction . . . . .	2
0.2	Overview . . . . .	3
0.2.1	Image Wrapping . . . . .	3
0.2.2	BXH/XCEDE QA tools . . . . .	4
0.2.3	XML events . . . . .	5
0.3	bxh_eventstats in-depth . . . . .	6
0.3.1	Using bxh_eventstats . . . . .	6
0.3.2	bxh_eventstats “event” query language . . . . .	9
<b>II</b>	<b>BXH/XCEDE Tools Reference</b>	<b>15</b>
0.4	afni2bxh . . . . .	16
0.5	analyze2bxh . . . . .	18
0.6	batch_showplay2xml . . . . .	20
0.7	bxh2analyze . . . . .	21
0.8	bxh2pgm . . . . .	23
0.9	bxh2ppm . . . . .	24
0.10	bxhabsorb . . . . .	26

0.11 bxh_brainmask . . . . .	30
0.12 bxh_correlate . . . . .	32
0.13 bxh_epochavg . . . . .	33
0.14 bxh_event2table . . . . .	36
0.15 bxh_eventmerge . . . . .	37
0.16 bxh_eventresp . . . . .	38
0.17 bxh_eventstats . . . . .	39
0.18 bxh_mean . . . . .	44
0.19 bxhreorient . . . . .	45
0.20 bxhselect . . . . .	46
0.21 bxhsetorient . . . . .	47
0.22 bxh_tfilter . . . . .	48
0.23 bxh_ttest . . . . .	49
0.24 dicom2bxh . . . . .	50
0.25 dumpheader . . . . .	52
0.26 eprime2xml . . . . .	53
0.27 eventstable2xml . . . . .	56
0.28 eventstats . . . . .	59
0.29 extractimagedata . . . . .	60
0.30 extractxyzdata . . . . .	61
0.31 ffile2bxh . . . . .	62
0.32 fmriqa_count . . . . .	63
0.33 fmriqa_generate.pl . . . . .	65
0.34 fmriqa_mean . . . . .	67
0.35 fmriqa_minmax . . . . .	68
0.36 fmriqa_oediff . . . . .	69
0.37 fmriqa_phantomqa . . . . .	70
0.38 fmriqa_phantomqa.pl . . . . .	71
0.39 fmriqa_spikiness . . . . .	72
0.40 fmriqa_stddev . . . . .	74
0.41 fmriqa_volmeasures . . . . .	75

0.42	iowa-signafive2bxh . . . . .	76
0.43	minc2bxh . . . . .	77
0.44	pfile2bxh . . . . .	79
0.45	presentation2xml . . . . .	81
0.46	printfrags . . . . .	84
0.47	showplay2xml . . . . .	85
0.48	signafive2bxh . . . . .	86
0.49	ximg2bxh . . . . .	88

### **III Example files 90**

.1	Example bxh_eventstats options file . . . . .	91
----	---	----

## **Part I**

# **BXH/XCEDE Tools User Manual**

## 0.1 Introduction

This is a suite of tools designed to read, write and manipulate XML descriptors using the BXH or XCEDE schemas. These descriptors are "wrappers" of the original, unadulterated image data files, and provide a standard interface to image data and other metadata extracted from the image headers. Also provided are tools to create and use XML "event" descriptors, for representing stimulus presentation and other time-based data.

An XML-based BXH or XCEDE image descriptor provides a consistent, portable, easily-parsable descriptor for one or more data files, which can consist of raw data or remain encapsulated in their native format (i.e. DICOM, SIGNA, Analyze, etc.). The BXH/XCEDE header is image-format-agnostic, and stores important header information (such as byte order, dimensions, image position, acquisition parameters, etc.) to allow BXH/XCEDE-enabled tools to read, write, display, and manipulate the stored data.

This consistent data interface allows the creation of robust software tools, which will support any future data formats, as long as they can be wrapped with an XML header.

New developments include an XML-based extension to describe "event" data, defined as (*onset*, *duration*) pairs, associated with appropriate metadata. This is currently used for describing stimulus/response data from task presentation software and for storing image QA measures.

## 0.2 Overview

There are four main components to the tools. The following sections describe each of these components in detail.

### 0.2.1 Image Wrapping

The "Image Wrapping" component contains tools that read, write and manipulate BXH or XCEDE header files that "wrap" image files.

**0.2.1.1 BXH/XCEDE creation tools** These tools will create a BXH (or XCEDE, if the `-xcede` option is specified) XML file that "wraps" the image data in the various supported formats. The supported formats and programs are:

- Autodetect - **bxhabsorb**(p. 26)
- AFNI - **afni2bxh**(p. 16)
- Analyze7.5/SPM/NIfTI-1 - **analyze2bxh**(p. 18)
- DICOM - **dicom2bxh**(p. 50)
- MINC - **minc2bxh**(p. 77)
- GE P-file - **pfile2bxh**(p. 79)
- GE Signa 5.x - **signafive2bxh**(p. 86)
- GE XImg - **ximg2bxh**(p. 88)

Some of the important metadata in the image headers are extracted into the XML file using a standard naming scheme. In a typical installation, most of these tools are symbolically linked to the same executable – the behavior of the tool is switched based on the name of the link. This executable, `bxhabsorb`, attempts to autodetect the format of the input images, whereas the other tools assume a given input format.

**0.2.1.2 BXH/XCEDE conversion tools** These tools convert from BXH or XCEDE into other image formats:

- **bxh2analyze**(p. 21)
- **bxh2pgm**(p. 23)
- **bxh2ppm**(p. 24)

**0.2.1.3 BXH/XCEDE manipulation tools** These tools manipulate the BXH or XCEDE file in various ways.

- **bxhreorient**(p. 45)
- **bxhselect**(p. 46)
- **bxhsetorient**(p. 47)
- **dumpheader**(p. 52)
- **extractimagedata**(p. 60)
- **extractxyztdata**(p. 61)
- **printfrags**(p. 84)

## 0.2.2 BXH/XCEDE QA tools

These tools perform QA (quality assurance) calculations and produce images, graphs, and/or XML data as output. `fmriqa_phantomqa.pl` and `fmriqa_generate.pl` produce an HTML report with various QA measures. `fmriqa_phantomqa.pl` was designed for fMRI images of the BIRN stability phantom, and `fmriqa_generate.pl` has been used for human fMRI data. These two tools depend on various subsidiary tools (listed below) to perform various tasks, which can be used individually.

- **fmriqa\_phantomqa.pl**(p. 71)
- **fmriqa\_generate.pl**(p. 65)
- **fmriqa\_count**(p. 63)
- **fmriqa\_mean**(p. 67)
- **fmriqa\_minmax**(p. 68)
- **fmriqa\_oediff**(p. 69)
- **fmriqa\_phantomqa**(p. 70)
- **fmriqa\_spikiness**(p. 72)
- **fmriqa\_stddev**(p. 74)
- **fmriqa\_volmeasures**(p. 75)

### 0.2.3 XML events

**0.2.3.1 bxh\_eventstats** `bxh_eventstats` is an event-based epoch averaging tool. It collects event-synchronized "snippets" of the fMRI response, averages them, and optionally correlates them to a template hemodynamic response or to other averaged "snippets". The times of the chosen fragments are selected by querying XML events files for events matching given characteristics. `bxh_eventstats` uses various subsidiary tools that may also be used directly.

- `bxh_eventstats`(p. 3)
- `bxh_brainmask`(p. 30)
- `bxh_correlate`(p. 32)
- `bxh_epochavg`(p. 33)
- `bxh_mean`(p. 44)
- `bxh_tfilter`(p. 48)
- `bxh_ttest`(p. 49)

Various screeds on topics related to `bxh_eventstats` can be found in **`bxh_eventstats in-depth`**(p. 6).

**0.2.3.2 XML events file creation tools** These tools are used to create the XML events files used by `bxh_eventstats` and other tools. Event data can currently be extracted from text files generated by E-Prime, Presentation, CIGAL, and other software that generates tabular text data.

- `eprime2xml`(p. 53)
- `presentation2xml`(p. 81)
- `showplay2xml`(p. 85)
- `eventstable2xml`(p. 56)

**0.2.3.3 XML events file manipulation tools** These tools are used to manipulate XML events files:

- `bxh_event2table`(p. 36)
- `bxh_eventmerge`(p. 37)
- `bxh_eventresp`(p. 38)

## 0.3 bxh\_eventstats in-depth

### 0.3.1 Using bxh\_eventstats

This section is intended to serve as a starting point for using `bxh_eventstats`, a program that collects and averages fragments of the hemodynamic response in a 4-D time-series. These fragments, or *epochs*, are time-locked to *events* (stored in XML events files) selected by the user using an event *query*. The duration of the epoch and starting time relative to the event onset time can be specified by the user. Multiple queries can be specified. Epochs for events which occur between image acquisitions are cubic-spline interpolated. The results of a given query can be optionally correlated to a vector template, or to the results of another query.

The requirements of `bxh_eventstats` are:

- one or more 4-D input images in BXH or XCEDE format. The image data, in most cases, should have been TR-aligned, motion-corrected and normalized to each other for the output to make any sense.
- for each input image, one or more XML events files describing the events occurring during the acquisition of that data. These events files may currently be generated from stimulus presentation data (see **XML events file creation tools**(p. 5) ), or image QA programs (**BXH/XCEDE QA tools**(p. 4) ). The XML events files specify an onset, duration, and other characteristics for each event.

Usage (shown on multiple lines for readability):

```
bxh_eventstats [opts]
                PREFIX
                IMG1.bxh EVENTS1a.xml,EVENTS1b.xml
                IMG2.bxh EVENTS2a.xml,EVENTS2b.xml
                ...
```

The first non-option argument specifies the prefix for all output filenames. This prefix may contain directory names, and may be a full or relative pathname. So for example, the prefix “output/stats” will send all output to the subdirectory named “output”, with all output filenames starting with the prefix “stats”.

The next arguments must come in pairs. The first argument of each pair is an input 4-D image in BXH or XCEDE format. The second argument is a list of one or more XML events files associated with that image. If there are more than one XML events files, the file names must be separated by commas. Naturally, the filenames themselves are prohibited from having commas.

The behavior of `bxh_eventstats` is parameterized by many options. Options to `bxh_eventstats` can be specified in two ways. The first is directly on the command-line, using options of the style `-OPTION` or `-OPTION OPTIONARG`. The second, and most common way is to put the options in an options file and instruct



`bxh_eventstats` to read the options from this file using the `--optsfromfile` option. The option names in the option file omit the leading double dash (`-`). A documented list of all options is in the example options file **Example bxh\_eventstats options file**(p. 91)

An example usage is as follows (newlines added for readability):

```
bxh_eventstats --optsfromfile eventstats_opts.txt
run001.bxh eventsRun1.xml,qa_events_run001.xml
run002.bxh eventsRun2.xml,qa_events_run002.xml
run003.bxh eventsRun3.xml,qa_events_run003.xml
run004.bxh eventsRun4.xml,qa_events_run004.xml
```

The above example collects and averages fragments of the image files `run00*.bxh`. Each file has two XML events files associated with them, one from stimulus presentation software, and another from the output of a quality assurance (QA) program.

**Queries** Queries are used to select events whose onsets will be used to calculate the starting times of hemodynamic response fragments, or *epochs*. These epochs will be averaged together to create an *epoch average* and written as output images. Multiple epoch averages can be created using different queries. These fragment collections are also known as *bins*. In other analysis programs, you may be required to explicitly specify onset times by creating a timing file – here you specify timings by selecting or excluding particular events by matching some of their defining characteristics.

There are two event query languages supported by `bxh_eventstats`: “event” and “XPath”. The examples in this document will use only the “event” query language. For more info, see **Query syntax**(p.13) . For more info on XPath, see the XPath specification. (For XPath aficionados, note that any XPath queries are applied as predicates to each event element, as if it had been specified as `//events/event[QUERY]`)

An example event in an XML events file is shown below:

```
<event type="shape">
  <onset>60.4</onset>
  <duration>1.5</duration>
  <value name="color">red</value>
  <value name="shape">square</value>
  <value name="position">center</value>
</event>
```

Note that this event has onset and duration fields, as well as several “values”, indicating various characteristics of this event. These characteristics, as well as onset, duration, and other special fields can be matched in a query. For example, specifying the following query in the option file:

```
querylabel RedSquare
query "$type=='shape' & %color=='red' & %shape=='square' "
```

will create a bin called “RedSquare” that collects epochs time-locked to all events whose “color” and “shape” values are “red” and “square” respectively. This kind of query is called a “primary query”. Note: the ‘\$’ and ‘’ prefixes are meant to distinguish names of special fields from the names of “value” fields; if there is no conflict, then the prefixes can be omitted.

bxh\_eventstats also supports the filtering of these events based on other events that are occurring at the same time. Suppose audio stimuli are running independently but simultaneously with the visual stimuli. An example event:

```
<event type="audio">
  <onset>55</onset>
  <duration>10</duration>
  <value name="frequency">1000</value>
</event>
```

Suppose the red square visual stimuli are the primary events that interest you, but you would like to look at only those that occur simultaneously with events similar to the one above. Then you can use an additional *filter query*, which specifies an additional restriction on the events selected for the bin:

```
querylabel RedSquareTone
query "$type=='shape' & %color=='red' & %shape=='square' "
queryfilter "$type=='audio' & %frequency=1000"
```

If a filter query (specified by a queryfilter line) is specified for one bin in an option file, then a query filter must be specified for all bins. Empty filter queries of the form:

```
queryfilter ""
```

act as a pass-through filter, and are useful if you don’t need to filter anything, but are required to specify a queryfilter option for a given bin.

The third type of query, an *epoch exclusion query*, is used to exclude an event from a bin if any matched exclusionary event overlaps with the epoch surrounding that event. This is useful if you are using the events output of fmriqa\_generate.pl, which generates several QA metrics for each acquired image volume. An example QA event:

```
<event>
  <onset>3</onset>
  <duration>1.5</duration>
  <value name="volmean">244.938</value>
  <value name="cmassx">117.079</value>
  <value name="cmassy">133.368</value>
  <value name="cmassz">39.6865</value>
  <value name="volmean_z_grand">0.969522618695653</value>
  <value name="cmassx_z_grand">1.39452591264842</value>
  <value name="cmassy_z_grand">0.800837946534433</value>
  <value name="cmassz_z_grand">0.791582533551492</value>
```

```

<value name="volmean_z_indiv">0.969522618695653</value>
<value name="cmassx_z_indiv">1.39452591264842</value>
<value name="cmassy_z_indiv">0.800837946534433</value>
<value name="cmassz_z_indiv">0.791582533551492</value>
</event>

```

If you wanted to select “red square” events, but excluding those events whose epochs include volumes whose mean intensity is more than three standard deviations from the average, you could use:

```

querylabel RedSquare
query "$type=='shape' & %color=='red' & %shape=='square' "
queryepochexclude "volmean_z_indiv > 3"

```

Just like `queryfilter`, if you specify `queryepochexclude` for any bin, you must specify it for all bins. An empty `queryepochexclude` ("" ) acts like a pass-through filter, excluding nothing.

### 0.3.2 bxh\_eventstats “event” query language

The “event” query language used by `bxh_eventstats` is a relatively simple way to select events of interest from an XML events file. A query is typically applied to each event in turn and the event is returned if the query result is true. The following subsections describe the syntax and semantics of the query language, after which follow the technical specifications for the syntax.

**0.3.2.1 Query syntax and semantics** A query is composed of one or more “conditions” separated by logical operators (i.e. the symbols for “and” and “or”). Parentheses may be used to enforce a particular order of operations.

For the following examples, we will match against the following event taken from an XML events file:

```

<event type="shape">
  <onset>60.4</onset>
  <duration>1.5</duration>
  <value name="color">red</value>
  <value name="shape">square</value>
  <value name="position">center</value>
  <value name="response">5</value>
  <value name="correct_response">5</value>
</event>

```

**Simple parameter matching** The simplest query matches against parameters listed in the event, and consists of a “ ” followed by a simple string of characters:

```
%response
```

This query is true if the character string "response" matches the name of a value element in the event and the value stored in this element is non-zero. In this case the sample event above does have a value element with name "response", and the stored value is 5, which is non-zero, so this query would return true. Any value element can be queried in this manner, by specifying the name of the value element preceded by a " character.

There are various special parameters listed in the above element that do not store data in "value" elements. These are "magic", and are queried by preceding their name by a '\$' character. Current magic elements are '\$type', '\$units', '\$onset', '\$duration', '\$name', and '\$description'. The following query returns true if the event has an onset element (which it should) and it is non-zero:

```
$onset
```

The '\$' and " characters preceding the parameter names can in most cases be omitted if there is no ambiguity. The pre-defined magic parameter names listed above are assumed to start with a '\$' if it does not exist. If you need to explicitly match a value element whose name is the same as a pre-defined magic value, you will need to precede it with the " character. So, in the absence of an ambiguity, the following queries are equivalent to the ones above:

```
response  
onset
```

**Equalities and inequalities** A query may test whether a parameter matches a particular value. This is done with the equality operator "=". The following query returns true if the "response" value element exists and whether its stored value is equal to 5:

```
response == 5
```

To test whether the stored value of the "response" value element is **not** equal to 5, using the equality operator "!=":

```
response != 5
```

Queries can match parameters to character strings, not just numbers. Character strings are surrounded by single or double quotes (note that if an application requires your entire query to be surrounded by quotes, you should probably choose a different quote character for the quoted strings within the query itself). The following query tests whether the value stored in the "shape" value element is equal to "square":

```
shape == 'square'
```

Various inequality operators are also available. The following queries test whether the onset parameter is respectively "greater than", "less than", "greater than or equal to", or "less than or equal to" 42.5:

```
onset > 42.5
onset < 42.5
onset >= 42.5
onset <= 42.5
```

The parameter names and numeric and string literals can appear on either or both sides of an equality or inequality operator. The following queries are all valid:

```
5 == response           <i>(true)</i>
5 == 5                  <i>(true, always)</i>
5 == 4                  <i>(false, always)</i>
response == correct_response <i>(true)</i>
```

The results of evaluating the above queries on the sample event are shown above in parentheses. The latter query returns true if the "response" and "correct\_response" value elements exist and if their stored values are the same.

**Logical operators and parentheses** More complex queries can be created by joining two queries with a logical operator. If you choose any two valid queries, represented by QUERY1 and QUERY2, the following query is true if and only if both QUERY1 and QUERY2 are true:

```
QUERY1 & QUERY2
```

The following query is true if either or both of QUERY1 and QUERY2 are true:

```
QUERY1 | QUERY2
```

Parentheses are useful to add readability or to enforce an order of operations. The following query QUERY is true if and only if the query inside the parentheses is true:

```
( QUERY )
```

The '&' operator binds tighter than the '|' operator, so the following two queries are equivalent:

```
QUERY1 & QUERY2 | QUERY3 & QUERY4
( QUERY & QUERY2 ) | ( QUERY3 & QUERY4 )
```

**Shortcuts** Another way to test if a parameter matches a number or string is to specify it in parentheses after the parameter name. The following two queries are equivalent:

```
response (5)
response == 5
```

Here is an example using string literals:

```
shape('square')
shape == 'square'
```

You can also specify inequality operators. The following two queries are equivalent:

```
onset(>=10.5)
onset >= 10.5
```

A numeric range can be specified as two numbers separated by a hyphen ('-'). The query returns true if the parameter's value is in the (inclusive) range:

```
onset(10.5-64.2)
( onset >= 10.5 & onset <= 64.2 )
```

Any of the above types of shortcuts may be combined together in the parentheses, separated by commas. In this case, the query returns true if the parameter's values matches any one or more of the specified tests. The following two queries are equivalent:

```
response(5,6,8-10,<=3)
( response == 5 | response == 6 | ( response >= 8 & response <= 10 ) | response <= 3 )
```

**That's all** Arbitrarily complex queries can be constructed using the above rules. However, by its nature there are certain types of queries that may be impossible to specify using the "event" query language. If you are in need of something more complex, some tools support queries using the "XPath" query language. However, XPath is a much more complex language and to use it to its fullest potential you are required to thoroughly understand the structure of the XML events file.

**0.3.2.2 Technical specs** The following subsections are only likely to be of interest to developers.

**TOKENIZING** Tokenizing is greedy, largest match wins – i.e. '>=' does not start with a '>' token. In any of the rules below, using the first valid match will ensure proper tokenizing.

Arbitrary amounts of whitespace may separate tokens.

```
TOKEN      : :=  NUMTOKEN
              |   STRTOKEN
              |   PARAMTOKEN
              |   PAREN
              |   ", "
              |   "-"
```

```

| "&"
| "|"
| INEQ_OP
| EQ_OP

NUMTOKEN ::= DIGIT+ "." DIGIT*
| DIGIT+
| "." DIGIT+

STRTOKEN ::= "'" STRCHAR1* "'"
| '"' STRCHAR2* '"'

PARAMTOKEN ::= "$" PARAMSTART PARAMCHAR*
| "%" PARAMSTART PARAMCHAR*
| PARAMSTART PARAMCHAR*

PARAMSTART ::= "_" | LETTER

PARAMCHAR ::= "." | "_" | LETTER | DIGIT

STRCHAR1 ::= any ASCII character except single quote (')

STRCHAR2 ::= any ASCII character except double quote (")

PAREN ::= "(" | ")"

INEQ_OP ::= "<=" | ">=" | "<" | ">"

EQ_OP ::= "==" | "!="

```

**Query syntax** Allowable whitespace is not specified here for readability. All rules accept whitespace between contiguous components except rules that match single tokens, i.e. NUMTOKEN, STRTOKEN, and PARAMTOKEN.

```

QUERY ::= QUERY "|" QUERY
| AQUERY

AQUERY ::= AQUERY "&" AQUERY
| PCQUERY

PCQUERY ::= "(" QUERY ")"
| CONDITION

CONDITION ::= PARAM_NUM INEQ_OP PARAM_NUM
| PARAM_NUM_STR EQ_OP PARAM_NUM_STR
| PARAMTOKEN "(" TESTLIST ")"
| PARAMTOKEN

PARAM_NUM ::= PARAMTOKEN
| NUMTOKEN

PARAM_NUM_STR ::= PARAMTOKEN
| NUMTOKEN
| STRTOKEN

```

```

TESTLIST  ::= VALTEST "," TESTLIST
           | VALTEST

VALTEST   ::= INEQ_OP NUMTOKEN
           | NUMTOKEN "-" NUMTOKEN
           | NUMTOKEN
           | STRTOKEN

NUMTOKEN  ::= DIGIT+ "." DIGIT*
           | DIGIT+
           | "." DIGIT+

STRTOKEN  ::= "'" STRCHAR_S+ "'"
           | '"' STRCHAR_D+ '"'

PARAMTOKEN ::= '$' PARAMSTART PARAMCHAR*
           | '%' PARAMSTART PARAMCHAR*
           |      PARAMSTART PARAMCHAR*

PARAMSTART ::= '_'
           | LETTER

PARAMCHAR  ::= '.'
           | '_'
           | LETTER
           | DIGIT

STRCHAR_S  ::= any ASCII character except single quote (')

STRCHAR_D  ::= any ASCII character except double quote (")

INEQ_OP    ::= "<="
           | ">="
           | "<"
           | ">"

EQ_OP      ::= "=="
           | "!="

```



## Part II

# BXH/XCEDE Tools Reference

The following pages contain the usage messages printed by each tool when given the `-help` command-line option.

## 0.4 afni2bxh

Usage:

```
afni2bxh [opts] afnifile.HEAD output.bxh
```

This program creates an XML wrapper for AFNI images.

```
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize x <size_t>
--hintsize x=<size_t>
--hintsize y <size_t>
--hintsize y=<size_t>
--hintsize z <size_t>
--hintsize z=<size_t>
--hintsize t <size_t>
--hintsize t=<size_t>
--hintorigin x <double>
--hintorigin x=<double>
--hintorigin y <double>
--hintorigin y=<double>
--hintorigin z <double>
--hintorigin z=<double>
--hintorigin t <double>
--hintorigin t=<double>
--hintspacing x <double>
--hintspacing x=<double>
--hintspacing y <double>
--hintspacing y=<double>
--hintspacing z <double>
--hintspacing z=<double>
--hintspacing t <double>
--hintspacing t=<double>
--hintgap x <double>
--hintgap x=<double>
--hintgap y <double>
--hintgap y=<double>
--hintgap z <double>
--hintgap z=<double>
--hintgap t <double>
--hintgap t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, size x and size y
    are assumed correct in most image headers, but they, as well as size z
    and size t options may be useful with image type 'pfile'. Origin and
    spacing hints will be used by most image types.
--forceorientation <str>
--forceorientation=<str>
    This option will force the labeled orientation of the image to match
    the given three letter orientation code. Each letter must come from
    the following groups in any order: R(ight) or L(ef); A(nterior) or
    P(osterior); S(uperior) or I(nferior). Only one letter from each group
    is allowed.
```

```
--xcede  
    Write XCEDE-style XML files.
```

## 0.5 analyze2bxh

Usage:

```
analyze2bxh [opts] [analyzefiles...] output.bxh
```

This program creates an XML wrapper for Analyze7.5/SPM/NIfTI-1 images.

```
--orientation <str>
--orientation=<str>
    Orientation of image, letters indication which way theX, Y, and Z
    dimensions (in that order) are pointing (e.g. LPS, IRP). Default is
    RAS (i.e. orientation used by SPM), or that specified in accompanying
    SPM .mat files. This option overrides all info in SPM .mat files
    and/or the Analyze 'orient' field (if --strictanalyze is specified).
    Equivalent to (and overrides) --forceorientation.
--strictanalyze
    Don't use SPM .mat files and use Analyze convention for orientation.
    The 'orient' field in the analyze header is parsed, and the default
    case (i.e. it is zero) means 'LAS'.
--avwbyteorder <str>
--avwbyteorder=<str>
    Specify byte order for AVW files (which don't store this info). This
    field should be 'l' for little-endian or 'b' for big-endian.
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize x <size_t>
--hintsize x=<size_t>
--hintsize y <size_t>
--hintsize y=<size_t>
--hintsize z <size_t>
--hintsize z=<size_t>
--hintsize t <size_t>
--hintsize t=<size_t>
--hintorigin x <double>
--hintorigin x=<double>
--hintorigin y <double>
--hintorigin y=<double>
--hintorigin z <double>
--hintorigin z=<double>
--hintorigin t <double>
--hintorigin t=<double>
--hintspacing x <double>
--hintspacing x=<double>
--hintspacing y <double>
--hintspacing y=<double>
--hintspacing z <double>
--hintspacing z=<double>
--hintspacing t <double>
--hintspacing t=<double>
--hintgap x <double>
--hintgap x=<double>
--hintgap y <double>
--hintgap y=<double>
--hintgap z <double>
```

```
--hintgapz=<double>
--hintgapt <double>
--hintgapt=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, sizex and sizey
    are assumed correct in most image headers, but they, as well as sizez
    and sizet options may be useful with image type 'pfile'. Origin and
    spacing hints will be used by most image types.
--forceorientation <str>
--forceorientation=<str>
    This option will force the labeled orientation of the image to match
    the given three letter orientation code. Each letter must come from
    the following groups in any order: R(ight) or L(ef); A(nterior) or
    P(osterior); S(uperior) or I(nferior). Only one letter from each group
    is allowed.
--xcede
    Write XCEDE-style XML files.
```

## **0.6 batch\_showplay2xml**

Usage:

```
batch_showplay2xml pdigmfiles...
```

This program runs showplay2xml on each pdigm file given as an argument.  
If the input file name is called pdigm1, then the output file will  
be called events-pdigm1.xml.

## 0.7 bxh2analyze

Usage:

```
bxh2analyze [opts] input.bxh outputprefix
```

This program creates Analyze 7.5, SPM, or NIfTI-1 images from BXH- or XCEDE-wrapped images.

```
--version
    Print version string and exit.
--overwrite
    Overwrite files if they exist.
-b
    This option suppresses the writing of a BXH header for every Analyze
    header and image file.
-s
    This option suppresses the writing of an SPM .mat file for every
    Analyze header and image file.
-i
    This option suppresses the writing of the image (.img) files.
-h
    This option suppresses the writing of the Analyze header (.hdr) files.
-B
    If writing BXH headers, instead of writing one BXH header, this option
    forces the writing of several BXH headers, one per Analyze header and
    image file.
-v
    This option suppresses the splitting a time series into separate
    volumes. If not specified, then each image file will contain the data
    for one volume. If specified, then each time series will be output as
    one large file.
--niftihdr
    Generate NIfTI-1 format header (default is to attempt to generate a
    maximally compatible header).
--spmhdr
    Assume header will only be read by SPM, using SPM-specific values when
    possible (default is to attempt to generate a maximally compatible
    header).
--analyzehdr
    Generate "pure" Analyze 7.5 header, whatever that means. (default is
    to attempt to generate a maximally compatible header).
--shiftforsign
    If the input signed/unsigned pixel type is not supported by Analyze (or
    SPM if --spmhdr), attempt to fit to the corresponding supported
    unsigned/signed type by shifting pixel values to fit range of new type.
    Otherwise no shifting is performed. WARNING: this will destroy
    quantitative measures.
--analyzetypes
    Force the use of only Analyze 7.5 pixel types, even if we are writing
    SPM or NIfTI headers.
--nosform
    For NIfTI headers, do not write orientation information into the sform
    fields. Default is to write both qform and sform.
--spatialunits <str>
--spatialunits=<str>
    Force spatial units to a given value. Must be 'm', 'mm', or 'um'.
    Output metadata will not be converted if this does not correctly
```

---

```
    represent the input metadata.
--temporalunits <str>
--temporalunits=<str>
    Force temporal units to a given value. Must be 's', 'ms', or 'us'.
    Output metadata will not be converted if this does not correctly
    represent the input metadata.
```



## 0.8 bxh2pgm

Usage:

```
bxh2pgm input.bxh output.pgm
```

This program converts images wrapped with a BXH or XCEDE header into PGM format. 3-D or higher dimensionality images are represented as a sequence of 2-D images.

```
--version
    Print version string and exit.
--colorbar <str>
--colorbar=<str>
    Write a horizontal colormap 'bar' to this PGM file.
--colorbarorient <str>
--colorbarorient=<str>
    Orientation of colorbar, either 'horizontal' (default) or 'vertical'.
--barwidth <uint>
--barwidth=<uint>
    Width (in pixels) of colormap 'bar' (default 16).
--barlength <uint>
--barlength=<uint>
    Length (in pixels) of colormap 'bar' (default 256).
--maxval <double>
--maxval=<double>
    By default, if the input element type is floating-point or if the
    maximum input value is greater than 65535, the maximum value in the
    input will be mapped to 65535 (the highest possible PGM value) in the
    output PGM image. --maxval specifies an alternative maximum input
    value. Input values greater than this will be clipped.
--minval <double>
--minval=<double>
    By default, the minimum value in the input will be mapped to 0 (the
    lowest possible PGM value) in the output PGM image. --minval specifies
    an alternative minimum input value. Input values smaller than this
    will be clipped.
--dimorder <str>
--dimorder=<str>
    Specify dimension order as a comma-separated list of dimension names.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## 0.9 bxh2ppm

Usage:

```
bxh2ppm input.bxh output.ppm
```

This program converts images wrapped with a BXH or XCEDE header into PPM format. 3-D or higher dimensionality images are represented as a sequence of 2-D images.

```
--version
    Print version string and exit.
--colorbar <str>
--colorbar=<str>
    Write a colormap 'bar' to this PPM file.
--colorbarorient <str>
--colorbarorient=<str>
    Orientation of colorbar, either 'horizontal' (default) or 'vertical'.
--barwidth <uint>
--barwidth=<uint>
    Width (in pixels) of colormap 'bar' (default 16).
--barlength <uint>
--barlength=<uint>
    Length (in pixels) of colormap 'bar' (default 256).
--maxval <double>
--maxval=<double>
    By default, if the input element type is floating-point or if the
    maximum input value is greater than 65535, the maximum value in the
    input will be mapped to the color corresponding to the highest value in
    the output PPM image. --maxval specifies an alternative maximum input
    value. Input values greater than this will be clipped.
--minval <double>
--minval=<double>
    By default, the minimum value in the input will be mapped to the color
    corresponding to the lowest value in the output PPM image. --minval
    specifies an alternative minimum input value. Input values smaller
    than this will be clipped.
--colormap <str>
--colormap=<str>
    Use this colormap for converting input values to colors. Valid
    colormaps are 'hot', 'bluered', 'grayhot', and 'gray' (default).
--dimorder <str>
--dimorder=<str>
    Specify dimension order as a comma-separated list of dimension names.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START, START+STEP, START+(2*STEP), ..., END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
```

---

```
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## 0.10 bxhabsorb

Usage:

```
bxhabsorb [ --fromtype type ] [ type-specific-opts... ] inputfiles...
bxhoutputfile
bxhabsorb [ --fromtype type ] [ --inputsfromfile inputlistfile ] [
type-specific-opts...] bxhoutputfile
```

This program creates a BXH or XCEDE wrapper for the specified input images. If the --fromtype option is not specified, it attempts to auto-detect the format of the input image files. If it cannot auto-detect the format, it will exit with an error.

```
--fromtype <str>
--fromtype=<str>
    Type of the input data (pfile, signa5, signafive, iowa-signa5,
    iowa-signafive, ximg, analyze, afni, dicom, minc). If this option is
    not specified, attempt to autodetect format of inputfiles.
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize x <size_t>
--hintsize x=<size_t>
--hintsize y <size_t>
--hintsize y=<size_t>
--hintsize z <size_t>
--hintsize z=<size_t>
--hintsize t <size_t>
--hintsize t=<size_t>
--hintorigin x <double>
--hintorigin x=<double>
--hintorigin y <double>
--hintorigin y=<double>
--hintorigin z <double>
--hintorigin z=<double>
--hintorigin t <double>
--hintorigin t=<double>
--hintspacing x <double>
--hintspacing x=<double>
--hintspacing y <double>
--hintspacing y=<double>
--hintspacing z <double>
--hintspacing z=<double>
--hintspacing t <double>
--hintspacing t=<double>
--hintgap x <double>
--hintgap x=<double>
--hintgap y <double>
--hintgap y=<double>
--hintgap z <double>
--hintgap z=<double>
--hintgap t <double>
--hintgap t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, size x and size y
```

are assumed correct in most image headers, but they, as well as sizez and sizet options may be useful with image type 'pfile'. Origin and spacing hints will be used by most image types.

--forceorientation <str>

--forceorientation=<str>

This option will force the labeled orientation of the image to match the given three letter orientation code. Each letter must come from the following groups in any order: R(ight) or L(ef); A(nterior) or P(osterior); S(uperior) or I(nferior). Only one letter from each group is allowed.

--xcede

Write XCEDE-style XML files.

#### PFILE USAGE

bxhabsorb --fromtype pfile [opts] [pfilehdr imagedata1...] output.bxh

#### PFILE OPTIONS

--forceversion <float>

--forceversion=<float>

Force version of P-file to be interpreted as this number.

--msbfirst

Indicates that data is big-endian (default: little-endian).

--dimorder <str>

--dimorder=<str>

Comma-separated names of dimensions from fastest-moving to slowest-moving (default: x,y,z,t).

#### SIGNA 5.X USAGE

bxhabsorb --fromtype signa5 [opts] [signa5files...] output.bxh

bxhabsorb --fromtype signafive [opts] [signa5files...] output.bxh

#### SIGNA 5.X OPTIONS

--dimzsize <size\_t>

--dimzsize=<size\_t>

Specifies the size of the z dimension (i.e. number of slices per timepoint). Default is to use the number of input files. Equivalent to (and overrides) --hintsizez.

--dimtsize <size\_t>

--dimtsize=<size\_t>

Specifies the size of the t dimension (i.e. number of timepoints). Default is number of input files divided by number of slices per timepoint (as specified by --dimzsize). Equivalent to (and overrides) --hintsize\_t.

#### IOWA SIGNA 5.X USAGE

bxhabsorb --fromtype iowa-signa5 imagedir output.bxh

bxhabsorb --fromtype iowa-signafive imagedir output.bxh

NOTE: the bxhabsorb option --inputsfromfile is not available for the iowa-signa5 format

imagedir is a directory containing I.\* images

#### XIMG USAGE

bxhabsorb --fromtype ximg [opts] [ximgfiles...] output.bxh

#### XIMG OPTIONS

--dimzsize <size\_t>

--dimzsize=<size\_t>

Specifies the size of the z dimension (i.e. number of slices per timepoint). Default is to use the number of input files. Equivalent to (and overrides) --hintsizez.

```

--dimtsize <size_t>
--dimtsize=<size_t>
    Specifies the size of the t dimension (i.e. number of timepoints).
    Default is number of input files divided by number of slices per
    timepoint (as specified by --dimzsize). Equivalent to (and overrides)
    --hintsizet.

ANALYZE/SPM USAGE
    bxhabsorb --fromtype analyze [opts] [analyzefiles...] output.bxh
ANALYZE/SPM OPTIONS:
--orientation <str>
--orientation=<str>
    Orientation of image, letters indication which way theX, Y, and Z
    dimensions (in that order) are pointing (e.g. LPS, IRP). Default is
    RAS (i.e. orientation used by SPM), or that specified in accompanying
    SPM .mat files. This option overrides all info in SPM .mat files
    and/or the Analyze 'orient' field (if --strictanalyze is specified).
    Equivalent to (and overrides) --forceorientation.
--strictanalyze
    Don't use SPM .mat files and use Analyze convention for orientation.
    The 'orient' field in the analyze header is parsed, and the default
    case (i.e. it is zero) means 'LAS'.
--avwbyteorder <str>
--avwbyteorder=<str>
    Specify byte order for AVW files (which don't store this info). This
    field should be 'l' for little-endian or 'b' for big-endian.

AFNI USAGE
    bxhabsorb --fromtype afni [opts] afnifile.HEAD output.bxh

MINC USAGE
    bxhabsorb --fromtype minc [opts] [mincfiles...] output.bxh

DICOM USAGE
    bxhabsorb --fromtype dicom [opts] [dicomfiles...] output.bxh
general options:
--debug
-d
    debug mode, print debug information
input options:
--force-concat
    If the input images have different orientation, Study UID, Series UID,
    ImageType, etc., then this option nevertheless forces them to be
    concatenated into the same volume. (They would otherwise be
    encapsulated within separate XML files.) This option may result in XML
    files that do not correctly describe the DICOM data -- use only if you
    know what you're doing!
    input file format:
--search-for-others
-s
    search for matching files in the same directory
--read-dataset
-f
    read data set without file meta information
input transfer syntax (only with --read-dataset):
--read-xfer-auto
-t

```

---

```
        use TS recognition (default)
--read-xfer-little
-te
        read with explicit VR little endian TS
--read-xfer-big
-tb
        read with explicit VR big endian TS
--read-xfer-implicit
-ti
        read with implicit VR little endian TS
output options:
converting:
--load-short
-M
        do not load very long values (e.g. pixel data)
error handling:
--ignore-errors
-E
        attempt to convert even if file is damaged
```

## 0.11 bxh\_brainmask

Usage:

```
bxh_brainmask [opts] inputfile outputfile
```

This program will attempt to create a simple (thresholded) brain mask given a BXH- or XCEDE-wrapped input image. Output is also a BXH- or XCEDE-wrapped input image. Calculation of the threshold is modified using various options.

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'. Default is all timepoints
    (:).
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--method <str>
--method=<str>
    Method to use for creating the brain mask.
    'threshold' marks those voxels whose mean value over time are not less
    than a given threshold (provided by --filterthresh).
    'rank' chooses the largest threshold that allows at least the n
    highest-valued voxels (as determined by the voxel's mean value over
    time) where n is specified by --filterrank.
    'localmin' fits a nth-order polynomial (order optionally specified by
    --filterorder) to an intensity histogram of the minimum value of each
    voxel over time, and chooses the first local minimum (disregarding the
    first histogram bucket) as the threshold. This method assumes the data
    follows an intensity distribution with at least two "humps", the first
    (lower) of which reflects noise.
    Default is 'rank'.
--filterorder <uint>
--filterorder=<uint>
    Order of the polynomial used for --method localmin. Default is 5.
--filterthresh <str>
--filterthresh=<str>
    Threshold used for --method threshold. If value ends with the percent
    sign (%), then this is taken as a percent of maximum intensity.
    Default is '50%'.
--filterrank <str>
--filterrank=<str>
    Threshold used for --method rank. If value ends with the percent sign
```



(%), then this is taken as a percent of the number of total voxels.  
Default is '20%'.

## 0.12 bxh\_correlate

### Usage:

```
bxh_correlate [opts] --template T1,T2,T3... inputxmlfile out_rfile
[out_tfile]
```

This program correlates the time series of each voxel in a 4-D time series of volumes (inputxmlfile) with a given "template" vector (specified with --template option). Output (in out\_rfile) is a 3-D data set storing the correlation coefficient (r). The optional third argument (out\_tfile) is where to write the 3-D data set storing the corresponding t-statistic (derived from r).

```
--version
    Print version string and exit.
--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'. Default is all timepoints
    (:).
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
--template <str>
--template=<str>
    A comma-separated list of numbers making up the template vector to
    correlate with the data. This option or --templatevoxel is required.
--templatevoxel <str>
--templatevoxel=<str>
    A comma-separated x,y,z coordinate (indices start at 0) indicating
    which voxel in the dataset to which to do the correlation. The value
    at that voxel in the output will be 1.0. This option or --template is
    required.
--maskfile <str>
--maskfile=<str>
    Use this 3-D mask (should be an XML file) before doing calculations.
```

## 0.13 bxh\_epochavg

Usage:

```
bxh_epochavg [opts] outputprefix imgfile1 eventfile1[,eventfile1b,...]
[imgfile2 eventfile2[,eventfile2b,...] ...]
```

This program "queries" a 4-D data set (with corresponding event lists) and produces averages of all time courses surrounding each event that match the query. Multiple independent queries may be specified, and the width and position and duration of each time course relative to the event is also user-specified. Multiple event files corresponding to the same image data can be specified separated by commas (the filenames/paths themselves are therefore prohibited from containing commas).

```
--version
    Print version string and exit.
--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--maskfile <str>
--maskfile=<str>
    Use this 3-D mask (should be an XML file) before doing calculations.
--querylanguage <str>
--querylanguage=<str>
    The language used for all queries. Valid values are 'XPath' and
    'event'. Case is irrelevant. Default is 'XPath'.
--query <str>
--query=<str>
    A query string as an XPath boolean expression. Will be applied as a
    predicate filter to each event. Each event node may or may not have
    onset, duration, type, and value elements (as well as others).
    Examples:
    --query "value[@name='color']='red'"
    --query "value[@name='color']='red' or value[@name='color']='blue'"
    --query "(value[@name='color']='red' or value[@name='color']='blue')
    and not value[@name='field']='upper' and onset>12000"
    Note that some characters in queries may need to be protected from the
    shell with quotes (as in the above examples). Separate instances of
    the --query option will result in independent queries, with separate
    outputs. Empty queries match all events. NOTE: At least one query
    must be specified!
--queryfilter <str>
--queryfilter=<str>
    If this option is specified, it is an XPath query (like --query) that
    is applied to a list of pseudo-events, each pseudo-event corresponding
    to an event matching the original query. Each pseudo-event is a
    merging of all events that are simultaneously in effect at the time of
    the onset of the real event. If this query matches the pseudo-event,
    the real event passes through the filter. The n-th instance of this
    option corresponds to the n-th specified query. If any --queryfilter
    options are specified, there should be exactly one --queryfilter per
```

```

--query.Empty or missing filter queries match everything.
--queryepochexclude <str>
--queryepochexclude=<str>
    Like --query, --queryepochexclude specifies an XPath-based event query.
    However, any epoch that includes an event that matches this query will
    be excluded from the analysis. The epoch surrounding an event is
    specified using --ptsbefore and --ptsafter (or --secsbefore and
    --secsafter). The n-th instance of this option corresponds to the n-th
    specified query. If any --queryepochexclude options are specified,
    there should be exactly one --queryepochexclude per --query.Empty or
    missing epoch exclusion queries exclude nothing.
--querylabel <str>
--querylabel=<str>
    A textual label for the corresponding query. The first instance of
    this option corresponds to the first specified query. There should be
    at most one --querylabel per --query. Default label is the query
    number.
--ptsbefore <int>
--ptsbefore=<int>
    How many time points before the event to include in analysis. This
    option (or --secsbefore) is required.
--ptsafter <int>
--ptsafter=<int>
    How many time points after the event to include in analysis. This
    option (or --secsafter) is required.
--secsbefore <double>
--secsbefore=<double>
    How many seconds before the event to include in analysis. This option
    (or --ptsbefore) is required.
--secsafter <double>
--secsafter=<double>
    How many seconds after the event to include in analysis. This option
    (or --ptsafter) is required.
--basestartoffset <int>
--basestartoffset=<int>
    Where to start calculating mean baseline, in number of timepoints (TRs)
    relative to event time. A negative number refers to a timepoint before
    the event, 0 is at the time of the event, and a positive number is
    after the event. Default is 0.
--baseendoffset <int>
--baseendoffset=<int>
    Where to end calculating mean baseline, in number of timepoints (TRs)
    relative to event time. A negative number refers to a timepoint before
    the event, 0 is at the time of the event, and a positive number is
    after the event. Default is 0.
--startpt <uint>
--startpt=<uint>
    This number of time points at the start of the data will be ignored.
    Default is 0.
--endpt <uint>
--endpt=<uint>
    Time points after this point will be ignored. Default is last
    timepoint.
--forcetr <double>
--forcetr=<double>
    If specified, this value (in seconds) will replace the TR specified in
    the input image file, if any.

```

```

--nointerp
    If specified, no interpolation will be done -- events will be assumed
    to occur at the closest TR/image acquisition time.
--scalebl
    If specified, values in each epoch are additionally scaled by dividing
    by (after subtracting) the baseline. This affects the 'avg' and 'std'
    output images. Percent signal-change images are not written. WARNING:
    Know what you are doing before using this option.
--extracttrials
    If this option is specified, the program will write out epochs for
    *all* extracted trials to a file PREFIX_QUERY_trials.bxh. This file
    will be a 5-D image file where the 4th dimension goes across time
    points within an epoch, and the 5th dimension represents the global
    trial number.
--trialmax
    This is an EXPERIMENTAL option. If specified, a 'seed' timepoint and
    voxel is found within the optional ROI specified by --trialmaxroi. The
    seed timepoint is defined as the timepoint within the epoch average
    that has the highest mean intensity. The seed voxel is then defined as
    the voxel with the highest value within the seed timepoint. Then, for
    each voxel, a 'trial sequence' is constructed containing the value of
    that voxel at the seed timepoint within each individual epoch (before
    averaging). The output is a 4-D series of volumes (one for each trial)
    named PREFIX_QUERY_trialmax.bxh that contains the volumes at the seed
    timepoint in each trial. The seed voxel coordinates are written to
    PREFIX_QUERY_trialmaxseed.txt.
--trialmaxroi <str>
--trialmaxroi=<str>
    The ROI used by --trialmax.
--trialmaxseed <str>
--trialmaxseed=<str>
    This specifies an explicit comma-separated coordinate X,Y,Z,T for the
    seed for --trialmax, to be applied to ALL queries. The T coordinate
    must be in the range [0,s-1] where s is the number of time points in
    the epoch. Note that timepoints are indexed from 0.
--extracttimingonly
    If specified, only the PREFIX_LABEL_timing.txt files will be written.
--memorylimit <double>
--memorylimit=<double>
    This specifies the number of megabytes of the input data to read at a
    time. Default is to read the entire data at once. If you are running
    out of memory due to high-resolution data, or large numbers of
    timepoints, this is one way to reduce memory usage. This is not an
    overall memory usage limit -- actual memory usage will surely be much
    higher than this.

```

## 0.14 bxh\_event2table

Usage:

```
bxh_event2table [opts] eventfiles...
```

This program takes XML event files as input, selects events (given user-specified queries), and writes a table of these events and associated metadata to standard output. Each row is one event, and each column represents a different metadata element (like onset, duration, and other values specified in the events file).

```
--version
    Print version string and exit.
--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.
--querylanguage <str>
--querylanguage=<str>
    The language used for all queries. Valid values are 'XPath' and
    'event'. Case is irrelevant. Default is 'XPath'.
--query <str>
--query=<str>
    A query string to match events. This option is required.
--filterquery <str>
--filterquery=<str>
    A query string to filter matched events.
--colsep <str>
--colsep=<str>
    String to separate columns (default is tab).
```

## 0.15 bxh\_eventmerge

Usage:

```
bxh_eventmerge inputquery grabquery inputevents1.xml inputevents2.xml... mergeevents.xml
```

This program takes several input files (inputevents\*.xml) and "merges" the information from another event file (mergeevents.xml) into each input file. The XPath query "inputquery" is applied to events in mergeevents.xml and the input files, and only those events whose results match will be merged. All other events in the input files will be output without change. The output files will be named the same as the inputs, but starting with the prefix "merged-".

## 0.16 bxh\_eventresp

Usage:

```
bxh_eventresp [opts] eventfiles... outputfile
```

This program takes event files as input, and selects stimulus and response events (given user-specified queries). The responses are then merged into the closest stimulus event within a given time interval from the response.

```
--version
    Print version string and exit.
--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--querylanguage <str>
--querylanguage=<str>
    The language used for all queries. Valid values are 'XPath' and
    'event'. Case is irrelevant. Default is 'XPath'.
--stimquery <str>
--stimquery=<str>
    A query string to match stimulus events. This option is required.
--stimfilterquery <str>
--stimfilterquery=<str>
    A query string to filter stimulus events.
--respquery <str>
--respquery=<str>
    A query string to match response events. This option is required.
--respfilterquery <str>
--respfilterquery=<str>
    A query string to filter stimulus events.
--maxresptime <double>
--maxresptime=<double>
    Specifies the longest time interval (in the same units as the onsets in
    the input file) within which a response can be associated with a
    stimulus. A negative value represents infinity (default).
--respdelayname <str>
--respdelayname=<str>
    The name to be used to label the value for response delay (time of
    response minus time of stimulus). Default is not to add this value.
--movevalue <str>
--movevalue=<str>
    By default, all values are moved from matched responses to matched
    stimuli. If this option is specified one or more times, only the
    values specified by instances of this option will be moved. Other
    values will be left alone.
--reversemerge
    This option reverses the merging process -- instead of moving response
    event values into matching stimulus events, it will move the matching
    stimulus event's values into the response event. The response delay
    value (if --respdelayname is specified) is also put into the response
    event. Make sure this is what you really want to do!
```



## 0.17 bxh\_eventstats

### Usage:

```
bxh_eventstats [opts] outputprefix imgfile1 eventfile1a[,eventfile1b...]
               [imgfile2 eventfile2a[,eventfile2b...] ...]
```

This program "queries" a 4-D data set (with corresponding event lists) and produces averages of all time courses surrounding each event that match the query. Multiple independent queries may be specified, and the width and position of each time course relative to the event is also user-specified. Multiple event files corresponding to the same image data can be specified separated by commas (the filenames/paths themselves are therefore prohibited from containing commas). This program also correlates the time series of each voxel in a 4-D time series of volumes (inputxmlfile) with a given "template" vector (specified with --template option). Outputs (in FILE\_cor.bxh and FILE\_tmap.bxh) are 3-D data sets storing the correlation coefficient (r) and the corresponding t-statistic (derived from r). T-statistics of the comparison between two queries is also supported (using the --tcompare option).

### Options:

```
--noaverage
    Skip everything up to and including averaging/stddev, just do
    correlation. Assumes averaging was performed previously using this
    script (or the equivalent) with the same outputprefix and queries,
    otherwise it will not be able to find the correct files.

--nocorrelate
    Do not run correlation, just do averaging.

--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.

--createbrainmask
    Create a brain mask using bxh_brainmask (using the default 'localmin'
    histogram method) on the first image and use this for all steps. This
    option is incompatible with the --maskfile option.

--brainmaskmethod <str>
--brainmaskmethod=<str>
    Method to use for creating the brain mask.
    'threshold' marks those voxels whose mean value over time are not less
    than a given threshold (provided by --brainmaskthresh).
    'rank' chooses the largest threshold that allows at least the n
    highest-valued voxels (as determined by the mean value of the voxel
    over time) where n is specified by --brainmaskrank.
    'localmin' fits a 5th-order polynomial to an intensity histogram of the
    minimum value of each voxel over time, and chooses the first local
    minimum (disregarding the first point) as the threshold. This method
    assumes the data follows an intensity distribution with at least two
    "humps", the first (lower) of which reflects noise.
    Default is 'localmin'.

--brainmaskorder <uint>
--brainmaskorder=<uint>
    Order of the polynomial used for --brainmaskmethod localmin. Default
```

is 5.

```
--brainmaskthresh <str>
--brainmaskthresh=<str>
    Threshold used for --brainmaskmethod threshold. If value ends with
    the percent sign (%), then this is taken as a percent of maximum
    intensity. Default is '50%'.
--brainmaskrank <str>
--brainmaskrank=<str>
    Threshold used for --brainmaskmethod rank. If value ends with the
    percent sign (%), then this is taken as a percent of the number of
    total voxels. Default is '20%'.
--tfiltertype <str>
--tfiltertype=<str>
    This option, if present, adds temporal filtering using a Chebyshev
    filter, and chooses which type of filtering to use. Valid choices are
    'lowpass', 'highpass', 'bandpass', or 'bandstop'. Each filter is
    parameterized by one or more instances of --tfilterperiod. 'lowpass'
    or 'highpass' require one --tfilterperiod option, specifying the stop
    or start frequency respectively. 'bandpass' or 'bandstop' require two
    --tfilterperiod options, specifying the start and stop frequencies, in
    any order (larger period/smaller frequency is assumed to be start
    frequency for 'bandpass' and stop frequency for 'bandstop').
--tfilterperiod <double>
--tfilterperiod=<double>
    This option specifies the frequency parameters for the filter in terms
    of the period (i.e. 1/frequency) in seconds per cycle. May be
    specified once for 'lowpass' and 'highpass' filter types, twice for
    'bandpass' and 'bandstop' filter types, and must be greater than 0.
--tfilterripple <double>
--tfilterripple=<double>
    This option specifies the percent ripple for the Chebyshev filter. If
    0 [zero], which is the default, then the filter is a Butterworth
    filter.
--tfilterorder <uint>
--tfilterorder=<uint>
    Order of the temporal filter. Default is 6.
--forcetr <double>
--forcetr=<double>
    If specified, this value (in seconds) will replace the TR specified in
    the input image file, if any.
--querylanguage <str>
--querylanguage=<str>
    The language used for all queries. Valid values are 'XPath' and
    'event'. Case is irrelevant. Default is 'XPath'.
--query <str>
--query=<str>
    A query string as an XPath boolean expression. Will be applied as a
    predicate filter to each event. Each event node may or may not have
    onset, duration, type, and value elements (as well as others).
    Examples:
    --query "value[@name='color']='red'"
    --query "value[@name='color']='red' or value[@name='color']='blue'"
    --query "(value[@name='color']='red' or value[@name='color']='blue')
    and not value[@name='field']='upper' and onset>12000"
    Note that some characters in queries may need to be protected from the
    shell with quotes (as in the above examples). Separate instances of
    the --query option will result in independent queries, with separate
```

outputs. Empty queries match all events. NOTE: At least one query must be specified!

`--queryfilter <str>`  
`--queryfilter=<str>`  
 If this option is specified, it is an XPath query (like `--query`) that is applied to a list of pseudo-events, each pseudo-event corresponding to an event matching the original query. Each pseudo-event is a merging of all events that are simultaneously in effect at the time of the onset of the real event. If this query matches the pseudo-event, the real event passes through the filter. The n-th instance of this option corresponds to the n-th specified query. If any `--queryfilter` options are specified, there should be exactly one `--queryfilter` per `--query`. Empty or missing filter queries match everything.

`--queryepochexclude <str>`  
`--queryepochexclude=<str>`  
 Like `--query`, `--queryepochexclude` specifies an XPath-based event query. However, any epoch that includes an event that matches this query will be excluded from the analysis. The epoch surrounding an event is specified using `--ptsbefore` and `--ptsafter`. The n-th instance of this option corresponds to the n-th specified query. If any `--queryepochexclude` options are specified, there should be exactly one `--queryepochexclude` per `--query`. Empty or missing epoch exclusion queries exclude nothing.

`--querylabel <str>`  
`--querylabel=<str>`  
 A textual label for the corresponding query. The first instance of this option corresponds to the first specified query. There should be at most one `--querylabel` per `--query`. Default label is the query number.

`--forcetr <double>`  
`--forcetr=<double>`  
 If specified, this will replace the TR specified in the input image file, if any.

`--nointerp`  
 If specified, no interpolation will be done -- events will be assumed to occur at the closest TR/image acquisition time.

`--scalebl`  
 If specified, values in each epoch are additionally scaled by dividing by (after subtracting) the baseline. This affects the 'avg' and 'std' output images. Percent signal-change images are not written. WARNING: Know what you are doing before using this option.

`--tcompare <str>`  
`--tcompare=<str>`  
 This specifies an additional t-test comparison between two queries. The string argument is in the form "A-B", where A and B are query labels (as specified using `--querylabel`) or query indices (starting at 1) if no query labels have been specified. Multiple instances of this option are allowed.

`--template <str>`  
`--template=<str>`  
 A comma-separated list of numbers making up the template vector to correlate with the data. This option is required.

`--overwrite`  
 Overwrite existing output files (otherwise error and exit).

`--ptsbefore <uint>`  
`--ptsbefore=<uint>`  
 How many time points before the event to include in analysis. This

```

    option is required.
--ptsafter <uint>
--ptsafter=<uint>
    How many time points after the event to include in analysis. This
    option is required.
--basestartoffset <int>
--basestartoffset=<int>
    Where to start calculating mean baseline, in number of timepoints (TRs)
    relative to event time. A negative number refers to a timepoint before
    the event, 0 is at the time of the event, and a positive number is
    after the event. Default is 0.
--baseendoffset <int>
--baseendoffset=<int>
    Where to end calculating mean baseline, in number of timepoints (TRs)
    relative to event time. A negative number refers to a timepoint before
    the event, 0 is at the time of the event, and a positive number is
    after the event. Default is 0.
--startpt <uint>
--startpt=<uint>
    This number of time points at the start of the data will be ignored.
    Default is 0.
--endpt <uint>
--endpt=<uint>
    Time points after this point will be ignored. Default is last
    timepoint.
--maskfile <str>
--maskfile=<str>
    Use this 3-D mask (should be an XML file) before doing calculations.
    This option is incompatible with the --createbrainmask option.
--extracttrials
    If this option is specified, the program will write out epochs for
    *all* extracted trials to a file PREFIX_QUERY_trials.bxh. This file
    will be a 5-D image file where the 4th dimension goes across time
    points within an epoch, and the 5th dimension represents the global
    trial number.
--trialmax
    This is an EXPERIMENTAL option. If specified, a 'seed' timepoint and
    voxel is found within the ROI specified by --trialmaxroi. The seed
    timepoint is defined as the timepoint within the epoch average that has
    the highest mean intensity. The seed voxel is then defined as the
    voxel with the highest value within the seed timepoint. Then, for each
    voxel, a 'trial sequence' is constructed containing the value of that
    voxel at the seed timepoint within each individual epoch (before
    averaging). The output is a 4-D series of volumes (one for each trial)
    named PREFIX_QUERY_trialmax.bxh that contains the volumes at the seed
    timepoint in each trial. The seed voxel coordinates are written to
    PREFIX_QUERY_trialmaxseed.txt.
--trialmaxroi <str>
--trialmaxroi=<str>
    The ROI used by --trialmax.
--trialmaxseed <str>
--trialmaxseed=<str>
    This specifies an explicit comma-separated coordinate X,Y,Z,T for the
    seed for --trialmax, to be applied to ALL queries. The T coordinate
    must be in the range [0,s-1] where s is the number of time points in
    the epoch. Note that timepoints are indexed from 0.
--trialmaxnodelete

```

If specified, the temporary files used by trialmaxnodelete (PREFIX\_QUERY\_trialmax.bxh and PREFIX\_QUERY\_trialmax.img) are not deleted.

--extracttimingonly  
If specified, only the PREFIX\_LABEL\_timing.txt files will be written.

--memorylimit <double>  
--memorylimit=<double>  
This specifies the number of megabytes of the input data to read at a time. Default is to read the entire data at once. If you are running out of memory due to high-resolution data, or large numbers of timepoints, this is one way to reduce memory usage. This is not an overall memory usage limit -- actual memory usage will surely be much higher than this.

--featinputs  
If specified, all input images are assumed to be FSL/FEAT first-level analysis output directories, and the filtered\_func\_data images will be used. The inputs will be transformed to the selected "averaging space" (see --featavgspace) for averaging, then outputs are transformed to "output space" (see --featoutputspace). The appropriate transformation matrices example\_func2highres, example\_func2standard, or example\_highres2standard must exist in the "reg" subdirectory of all input .feat directories.

--featavgspace <string>  
--featavgspace=<string>  
If specified, this specifies the space in which the averages should be computed. This can be "highres" or "standard". Default is to do the averaging in the same space as the outputs (see --featoutputspace).

--featoutputspace <string>  
--featoutputspace=<string>  
If specified, this specifies the space into which the FEAT-derived outputs should be transformed. This can be "highres" or "standard" (default). Furthermore, if "standard" is used, all example\_highres2standard matrices must match exactly.

--featavgrefvol <string>  
--featavgrefvol=<string>  
--featoutputrefvol <string>  
--featoutputrefvol=<string>  
These options specify the reference volume to use for --featavgspace or --featoutputspace respectively. This must point to a .nii or .hdr file (or just specify the base name without the extension). This volume is only used to determine the resolution and voxel spacing of the outputs. If specified path is not an absolute pathname, the path is relative to the reg subdirectory of the .feat directory. Default is the "example\_func" volume in the reg subdirectory of the .feat directory (i.e. to keep the same resolution as the input functional images). Other typical values are "standard" and "highres".

## 0.18 bxh\_mean

Usage:

```
bxh_mean [opts] inputs.bxh... output.bxh
```

This program calculates per-voxel averages across a selected dimension, and produces an output dataset 'collapsed' across that dimension. If `--dimension 'dataset'` is specified, then corresponding voxels in each input dataset are averaged to create an output dataset of the same dimensionality; in this case, all of the dimensions in all input datasets must match. If multiple input datasets are provided and `--dimension 'dataset'` is not specified, then they are concatenated along the last (slowest-moving) dimension; i.e. if one specifies an XYZT 64x64x27x120 time series and an XYZT 64x64x27x130 time series as inputs, they will be considered together as a single 64x64x27x250 time series. In this case, all dimensions except the last dimension must match in all data sets.

```
--version
```

Print version string and exit.

```
--dimension <str>
```

```
--dimension=<str>
```

Select the dimension over which to average. The dimension must be one that exists in the input dataset, or must be 'dataset'. Default is the last (slowest-moving) dimension.

## 0.19 bxhreorient

Usage:

```
bxhreorient [options] inputfile [ outputfile [datafileout] ]
```

This program reorients the image data given by the input BXH or XCEDE file to an orientation specified by the user using the `--orientation` option. It is assumed that the orientation vectors in the BXH/XCEDE file are correct with respect to the image data. `outputfile` is required if not using `--inplace` option. Output is also a BXH or XCEDE file, pointing to an image data file (named by `datafileout` if specified).

```
--version
    Print version string and exit.
--orientation <str>
--orientation=<str>
    This option specifies the new orientation by R/L A/P S/I letters,
    upper- or lower-case, in X,Y,Z order, where R means that dimension
    starts on the left and goes TO THE RIGHT, A means the dimension goes
    from posterior TO ANTERIOR, etc. For example, IPR means X goes S->I, Y
    goes A->P, and Z goes L->R. Default is RAS (neurological axial, as
    used by SPM).
--inplace
    Do the reorientation in-place, overwriting the original files (both BXH
    and data) with new data. Otherwise use the same name as the output
    file, but with extension replaced by '.dat'.
```

## 0.20 bxhselect

Usage:

```
bxhselect [options] inputfile [ outputfile [datafileout] ]
```

This program reorients the image data given by the input BXH or XCEDE file to an orientation specified by the user using the `--orientation` option. It is assumed that the orientation vectors in the BXH/XCEDE file are correct with respect to the image data. `'bxhfileout'` is required if not using `--inplace` option. Output is also a BXH or XCEDE file, pointing to an image data file (named by `datafileout` if specified).

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START, START+STEP, START+(2*STEP), ..., END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```



## 0.21 bxhsetorient

Usage:

```
bxhsetorient [options] orient inputfile [outputfile]
```

This program sets the orientation vectors in the BXH or XCEDE file. NOTE: this program does not reorient or otherwise touch the image data itself. This program is useful to fix incorrect or missing orientation vectors in a BXH/XCEDE file. bxhfileout is required if not using --inplace option (and vice-versa). 'orient' specifies the new orientation by R/L A/P S/I letters, upper- or lower-case, in X,Y,Z order, where R means that dimension starts on the left and goes TO THE RIGHT, A means the dimension goes from posterior TO ANTERIOR, etc. For example, IPR means X goes S->I, Y goes A->P, and Z goes L->R.

--version

Print version string and exit.

--inplace

Do the reorientation in-place, overwriting the original BXH file.

## 0.22 bxh\_tfilter

Usage:

```
bxh_tfilter [opts] input.bxh output.bxh
```

This program runs, on a 4-D data set, a Chebyshev filter across each voxel's fourth dimension (e.g. time course) and writes the results to output.bxh.

```
--version
    Print version string and exit.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--filtertype <str>
--filtertype=<str>
    This required option chooses the filter type. Valid choices are
    'lowpass', 'highpass', 'bandpass', or 'bandstop'. Each filter is
    parameterized by one or more instances of --period. 'lowpass' or
    'highpass' require one --period option, specifying the stop or start
    frequency respectively. 'bandpass' or 'bandstop' require two --period
    options, specifying the start and stop frequencies, in any order
    (larger period/smaller frequency is assumed to be start frequency for
    'bandpass' and stop frequency for 'bandstop').
--period <double>
--period=<double>
    This option specifies the frequency parameters for the filter in terms
    of the period (i.e. 1/frequency) in seconds per cycle. May be
    specified once for 'lowpass' and 'highpass' filter types, twice for
    'bandpass' and 'bandstop' filter types, and must be greater than 0.
--ripple <double>
--ripple=<double>
    This option specifies the percent ripple for the Chebyshev filter. If
    0 [zero], which is the default, then the filter is a Butterworth
    filter.
--order <uint>
--order=<uint>
    Order of the filter. Default is 6.
--forcetr <double>
--forcetr=<double>
    If specified, this value (in seconds) will replace the TR specified in
    the input image file, if any.
--keepdc
    Keep DC component (mean signal). Has no effect for lowpass and
    bandpass filter types (which already keep the DC component).
```

## 0.23 bxh\_ttest

Usage:

```
bxh_ttest [opts] avg1.bxh std1.bxh n1.bxh avg2.bxh std2.bxh n2.bxh out_tfile
```

This program computes a per-voxel t-statistic between two datasets given their 3-D mean, standard deviation, and n images. Output (in out\_tfile) is a 3-D data set storing the t-statistic.

```
--version
    Print version string and exit.
--optsfromfile <str>
--optsfromfile=<str>
    Program options (i.e. those starting with '--') will come from this
    file. If this option is specified, then the options in the file will
    be applied after all command-line options. The options (and their
    arguments) should be specified one per line, with the leading '--'
    omitted.
--overwrite
    Overwrite existing output files (otherwise error and exit).
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'. Default is all timepoints
    (:).
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
--maskfile <str>
--maskfile=<str>
    Use this 3-D mask (should be an XML file) before doing calculations.
```

## 0.24 dicom2bxh

Usage:

```
dicom2bxh [opts] [dicomfiles...] output.bxh
```

This program creates an XML wrapper for DICOM images.

general options:

```
--debug
-d
    debug mode, print debug information
```

input options:

```
--force-concat
    If the input images have different orientation, Study UID, Series UID,
    ImageType, etc., then this option nevertheless forces them to be
    concatenated into the same volume. (They would otherwise be
    encapsulated within separate XML files.) This option may result in XML
    files that do not correctly describe the DICOM data -- use only if you
    know what you're doing!
```

input file format:

```
--search-for-others
-s
    search for matching files in the same directory
--read-dataset
-f
    read data set without file meta information
```

input transfer syntax (only with --read-dataset):

```
--read-xfer-auto
-t
    use TS recognition (default)
--read-xfer-little
-te
    read with explicit VR little endian TS
--read-xfer-big
-tb
    read with explicit VR big endian TS
--read-xfer-implicit
-ti
    read with implicit VR little endian TS
```

output options:

```
converting:
--load-short
-M
    do not load very long values (e.g. pixel data)
```

error handling:

```
--ignore-errors
-E
    attempt to convert even if file is damaged
```

additional options:

```
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize <size_t>
--hintsize=<size_t>
--hintsizey <size_t>
```

```

--hintsizey=<size_t>
--hintsizez <size_t>
--hintsizez=<size_t>
--hintsizez <size_t>
--hintsizez=<size_t>
--hintoriginx <double>
--hintoriginx=<double>
--hintoriginy <double>
--hintoriginy=<double>
--hintoriginz <double>
--hintoriginz=<double>
--hintorigint <double>
--hintorigint=<double>
--hintspacingx <double>
--hintspacingx=<double>
--hintspacingy <double>
--hintspacingy=<double>
--hintspacingz <double>
--hintspacingz=<double>
--hintspacingt <double>
--hintspacingt=<double>
--hintgapx <double>
--hintgapx=<double>
--hintgapy <double>
--hintgapy=<double>
--hintgapz <double>
--hintgapz=<double>
--hintgapt <double>
--hintgapt=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, sizex and sizey
    are assumed correct in most image headers, but they, as well as sizez
    and sizet options may be useful with image type 'pfile'. Origin and
    spacing hints will be used by most image types.
--forceorientation <str>
--forceorientation=<str>
    This option will force the labeled orientation of the image to match
    the given three letter orientation code. Each letter must come from
    the following groups in any order: R(ight) or L(ef); A(nterior) or
    P(osterior); S(uperior) or I(nferior). Only one letter from each group
    is allowed.
--xcede
    Write XCEDE-style XML files.

```

## **0.25 dumpheader**

Usage:

```
../../utils/dumpheader inputfile
```

This program prints a simplistic summary of the BXH or XCEDE file given as input.

## 0.26 eprime2xml

Usage:

```
eprime2xml instructions.txt eprime.txt [outputevents.xml]
```

eprime2xml takes an E-Prime output file as exported as text from the E-Prime software and an instruction file, and creates an XML event file. The instruction file indicates which columns in the E-Prime file are of interest and what they should map to in the output event file. If the output file is not specified, the event data is written to standard output.

Options:

```
--extracttable
    Instead of creating XML as output, output as tabular text.
    This is a no-op for most formats, and is really only useful for
    converting E-Prime "recovery" logs into a tabular form.
--columnnames
    If this options is specified, only the columns of the table are
    printed (one per line) and the program exits.
--subtractonset SECS
    This option subtracts SECS from all onset times (default is 0).
    This is in addition to any other normalization that may occur
    (see use of 'firstmritime' below).
--colsep SEPARATOR
    This option specifies the column separator (default is tab).
```

The instruction file language is defined as follows:

```
COMMAND VALUESPEC [VALUESPEC...]
```

where COMMAND can be event, param, or block. VALUESPEC has one of the following formats:

```
[OUTVALUENAME=]COLUMNNAME[:UNITS]
OUTVALUENAME=@TEXT[:UNITS]
```

Each VALUESPEC defines values that should be passed through to the corresponding event, param, or block. In the first alternative listed above, the value comes from a column in the input file (optionally renamed to OUTVALUENAME) and in the second alternatives, the value is directly specified preceded by a '@' character. If OUTVALUENAME= is missing, then the VALUESPEC is equivalent to:

```
COLUMNNAME=COLUMNNAME[:UNITS]
```

OUTVALUENAME may not contain the equals sign ('='), at sign ('@'), quotes, or whitespace.

Either COLUMNNAME or TEXT may contain quoted substrings to protect special characters like colon (':'), equals sign ('='), at sign ('@'), or whitespace; otherwise these special characters are prohibited. A single quote will protect all characters until the next single quote, and likewise for double quote. The following examples show equivalent VALUESPECs:

```
description=DESC
description=D'E' "SC"
```

```
onset="Onset Time":secs
onset=Onset' 'Time:secs
onset=Onse't T'ime:secs
```

Unquoted spaces separate VALUESPECs.

```
-----
'event' command:
-----
```

Each 'event' command creates a class of events in the output event file, where the contents of the event are specified by the VALUESPECs. In general, for each matching row (more later), it creates an event with the following contents:

```
<event type="$type" units="$units">
  <onset>$onset</onset>
  <duration>$duration</duration>
  <name>$name</name>
  <description>$description</description>
  <value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
  <value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
  ...
</event>
```

VALUESPECs whose OUTVALUENAMES start with a dollar sign ( \$ ) are "magic", and are interpreted in a value-specific way. VALUESPECs whose OUTVALUENAMES start with a percent sign ( % ) are explicitly non-magic. Any OUTVALUENAME not starting with a % or \$ is assumed to have an implicit % unless it matches a list of pre-defined magic values (below), in which case an implicit \$ is assumed.

Pre-defined magic values '\$type', '\$units', '\$onset', '\$duration', '\$name', and '\$description' are put in the appropriate child element or attribute of <event> (shown above). Only the '\$onset' VALUESPEC is required. Default value for '\$duration' is zero. All non-magic values are placed in <value> elements.

The pre-defined magic value '\$DURUNTIL' indicates that any row in the input used to create an event will have an ending time specified by the value of column COLUMNNAME in the current row. Likewise, the value '\$DURUNTILNEXTROW' does the same thing, but grabs the value from the next row. These are used to calculate the duration of this event. This may be specified more than once, and the first non-NULL column will be used. This option is used when a row does not have a duration column, and it must be calculated based on times in this or the subsequent row.

By default, only those rows whose '\$onset' column is non-empty and non-NULL will be processed as events. Certain magic OUTVALUENAMES further restrict the rows that are used for this event command. '\$MATCH' and '\$MATCHNONZERO' specify a column whose values indicate whether that row should be selected -- for '\$MATCH', the values must be non-empty and non-'NULL'; for '\$MATCHNONZERO', the values must also be non-zero. With '\$MATCHEQUAL', one specifies both a column and an actual value to match -- for the '\$MATCHEQUAL' value name (and only the '\$MATCHEQUAL' value name) the VALUESPEC syntax is extended in the



following way:

```
$MATCHEQUAL=COLUMNNAME@MATCHVALUE
```

where COLUMNNAME and MATCHVALUE are the two relevant parameters.

```
-----
'block' command:
-----
```

The block command has the same usage as the event command. The same magic values apply to block commands as event commands. An '\$onset' value is again required, and '\$duration' is optional (assumed to be zero [0] if missing).

```
-----
'param' command:
-----
```

Each param command specifies a list of columns that should be put in the <params> section of the event file. These represent parameters that are constant (or default) throughout the events file. Each VALUESPEC represents one item to put in the <params> element as such:

```
<params>
...
<value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
<value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
...
</params>
```

Only the first non-empty, non-NULL field in the column specified by a 'param' will be used. Be aware of this if this column does not have the same value in every row.

There is one magic OUTVALUENAME (maybe more later) '\$firstmritime', which will generate the following element:

```
<params>
<firstmritime>0</firstmritime>
</params>
```

If '\$firstmritime' is specified, it (and all '\$onset' VALUESPECs) must have UNITS specified. All '\$onset' columns are normalized by this value, so their units and '\$firstmritime' units must match.

## 0.27 eventstable2xml

Usage:

```
eventstable2xml instructions.txt inputevents.txt [outputevents.xml]
```

eventstable2xml takes a text tabular events file and an instruction file, and creates an XML events file. The instruction file indicates which columns in the original events file are of interest and what they should map to in the output event file. If the output file is not specified, the event data is written to standard output.

Options:

```
--extracttable
    Instead of creating XML as output, output as tabular text.
    This is a no-op for most formats, and is really only useful for
    converting E-Prime "recovery" logs into a tabular form.
--columnnames
    If this option is specified, only the columns of the table are
    printed (one per line) and the program exits.
--subtractonset SECS
    This option subtracts SECS from all onset times (default is 0).
    This is in addition to any other normalization that may occur
    (see use of 'firstmritime' below).
--colsep SEPARATOR
    This option specifies the column separator (default is tab).
```

The instruction file language is defined as follows:

```
COMMAND VALUESPEC [VALUESPEC...]
```

where COMMAND can be event, param, or block. VALUESPEC has one of the following formats:

```
[OUTVALUENAME=]COLUMNNAME[:UNITS]
OUTVALUENAME=@TEXT[:UNITS]
```

Each VALUESPEC defines values that should be passed through to the corresponding event, param, or block. In the first alternative listed above, the value comes from a column in the input file (optionally renamed to OUTVALUENAME) and in the second alternatives, the value is directly specified preceded by a '@' character. If OUTVALUENAME= is missing, then the VALUESPEC is equivalent to:

```
COLUMNNAME=COLUMNNAME[:UNITS]
```

OUTVALUENAME may not contain the equals sign ('='), at sign ('@'), quotes, or whitespace.

Either COLUMNNAME or TEXT may contain quoted substrings to protect special characters like colon (':'), equals sign ('='), at sign ('@'), or whitespace; otherwise these special characters are prohibited. A single quote will protect all characters until the next single quote, and likewise for double quote. The following examples show equivalent VALUESPECs:

```
description=DESC
description=D'E' "SC"

onset="Onset Time":secs
```

```
onset=Onset' 'Time:secs
onset=Onset' t Time:secs
```

Unquoted spaces separate VALUESPECs.

```
-----
'event' command:
-----
```

Each 'event' command creates a class of events in the output event file, where the contents of the event are specified by the VALUESPECs. In general, for each matching row (more later), it creates an event with the following contents:

```
<event type="$type" units="$units">
  <onset>$onset</onset>
  <duration>$duration</duration>
  <name>$name</name>
  <description>$description</description>
  <value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
  <value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
  ...
</event>
```

VALUESPECs whose OUTVALUENAMES start with a dollar sign ( \$ ) are "magic", and are interpreted in a value-specific way. VALUESPECs whose OUTVALUENAMES start with a percent sign ( % ) are explicitly non-magic. Any OUTVALUENAME not starting with a % or \$ is assumed to have an implicit % unless it matches a list of pre-defined magic values (below), in which case an implicit \$ is assumed.

Pre-defined magic values '\$type', '\$units', '\$onset', '\$duration', '\$name', and '\$description' are put in the appropriate child element or attribute of <event> (shown above). Only the '\$onset' VALUESPEC is required. Default value for '\$duration' is zero. All non-magic values are placed in <value> elements.

The pre-defined magic value '\$DURUNTIL' indicates that any row in the input used to create an event will have an ending time specified by the value of column COLUMNAME in the current row. Likewise, the value '\$DURUNTILNEXTROW' does the same thing, but grabs the value from the next row. These are used to calculate the duration of this event. This may be specified more than once, and the first non-NULL column will be used. This option is used when a row does not have a duration column, and it must be calculated based on times in this or the subsequent row.

By default, only those rows whose '\$onset' column is non-empty and non-NULL will be processed as events. Certain magic OUTVALUENAMES further restrict the rows that are used for this event command. '\$MATCH' and '\$MATCHNONZERO' specify a column whose values indicate whether that row should be selected -- for '\$MATCH', the values must be non-empty and non-'NULL'; for '\$MATCHNONZERO', the values must also be non-zero. With '\$MATCHEQUAL', one specifies both a column and an actual value to match -- for the '\$MATCHEQUAL' value name (and only the '\$MATCHEQUAL' value name) the VALUESPEC syntax is extended in the following way:

```
$MATCHEQUAL=COLUMNNAME@MATCHVALUE
```

where COLUMNNAME and MATCHVALUE are the two relevant parameters.

```
-----
'block' command:
-----
```

The block command has the same usage as the event command. The same magic values apply to block commands as event commands. An '\$onset' value is again required, and '\$duration' is optional (assumed to be zero [0] if missing).

```
-----
'param' command:
-----
```

Each param command specifies a list of columns that should be put in the <params> section of the event file. These represent parameters that are constant (or default) throughout the events file. Each VALUESPEC represents one item to put in the <params> element as such:

```
<params>
...
<value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
<value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
...
</params>
```

Only the first non-empty, non-NULL field in the column specified by a 'param' will be used. Be aware of this if this column does not have the same value in every row.

There is one magic OUTVALUENAME (maybe more later) '\$firstmritime', which will generate the following element:

```
<params>
<firstmritime>0</firstmritime>
</params>
```

If '\$firstmritime' is specified, it (and all '\$onset' VALUESPECs) must have UNITS specified. All '\$onset' columns are normalized by this value, so their units and '\$firstmritime' units must match.

**0.28 eventstats**

## 0.29 extractimagedata

Usage:

```
extractimagedata [opts] xmlfile outputfile
```

This program extracts the image data pointed to by the input BXH or XCEDE file and writes it to outputfile.

```
--version
```

```
    Print version string and exit.
```

```
--msbfirst
```

```
    Extract data as big-endian (default: little-endian).
```

## 0.30 extractxyztdata

Usage:

```
extractxyztdata [opts] xmlfile outputfile
```

This program extracts the image data pointed to by the input BXH or XCEDE file and writes it to outputfile. The data is reordered so that the dimensions labeled 'x', 'y', 'z', and 't' are in that order.

```
--version
```

```
    Print version string and exit.
```

```
--msbfirst
```

```
    Extract data as big-endian (default: little-endian).
```

## 0.31 ffile2bxh

Usage:

```
ffile2bxh [ --dimorder "x,y,z,t" ] ffile [datafile1...] outputfile
```

This program takes a Stanford F-file and creates a BXH or XCEDE header using the metadata in the F-file, and points to the image data in the given datafiles.

--dimorder specifies the comma-separated names of the dimensions in the datafiles(s) in order from fastest-moving to slowest-moving  
Default is "x,y,z,t".

--xcede produces an XCEDE file as output.



## 0.32 fmriqa\_count

Usage:

```
fmriqa_count inputfile
```

This program outputs histograms or counts of voxels in a BXH- or XCEDE-wrapped dataset that match the given conditions. Output can be per-slice, per-volume, or for entire dataset (see --granularity). Histogram output requires the --histogram option. If histogram output is not chosen, output is as if there were one histogram 'bucket'. Conditions are specified as command-line options, described below. Default is to 'and' all conditions (but see --aggregate). Default condition for 'and' aggregate is to match all voxels. Default condition for 'or' aggregate is to match no voxels. Thus, with no options, this program prints out the number of voxels in the data.

```
--version
    Print version string and exit.
--granularity <str>
--granularity=<str>
    Print counts at this granularity. Acceptable values are 'timeseries'
    (default), 'volume', 'slice', and 'voxel'.
--aggregate <str>
--aggregate=<str>
    Conditions are aggregated by this operator, either 'and' (default) or
    'or'.
--gt <double>
--gt=<double>
    Match those voxels greater than this value.
--ge <double>
--ge=<double>
    Match those voxels greater than or equal to this value.
--lt <double>
--lt=<double>
    Match those voxels less than this value.
--le <double>
--le=<double>
    Match those voxels less than or equal to this value.
--timeselect <str>
--timeselect=<str>
    Match only those timepoints in this comma-separated list of timepoints
    (first timepoint is 0). Any timepoint can be a contiguous range,
    specified as two numbers separated by a colon, i.e. 'START:END'. An
    empty END implies the last timepoint. The default step of 1 (one) in
    ranges can be changed using 'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' coordinate.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' coordinate.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' coordinate.
--histogram
    Specifies that output should be histogram. See --histobuckets to
    specify number of buckets, --histobucketwidth to specify width of
```

buckets, or --histobounds to specify bucket boundaries.

--histobuckets <str>  
--histobuckets=<str>  
Valid only with --histogram option. Constructs this many evenly-spaced histogram buckets. This option is incompatible with --histobounds or --histobucketwidth.

--histobounds <str>  
--histobounds=<str>  
Valid only with --histogram option. By default, histogram bucket boundaries are in multiples of standard deviations. This option specifies alternate boundaries for (N + 1) buckets as a space-separated list of N floating point numbers. For example, --histobounds "0.0 5.0 10.0" will separate voxels with values  $-\infty < x < 0.0$ ,  $0.0 \leq x < 5.0$ ,  $5.0 \leq x < 10.0$ , and  $10.0 \leq x < \infty$ . This option is incompatible with --histobucketwidth or --histobucketsize.

--histobucketwidth <double>  
--histobucketwidth=<double>  
Valid only with --histogram option. Constructs evenly-spaced histogram buckets with this width. This option is incompatible with --histobounds or --histobuckets.

## 0.33 fmriqa\_generate.pl

Usage:

```
fmriqa_generate.pl [--overwrite] [--verbose] [--deletestddev]
                  [--deletemean] [--deleteslicevar] [--deletesfnr]
                  [--deletemask] [--forcetr TR]
                  [ --zthresh1 NUM ] [ --zthresh2 NUM ]
                  [ --percthresh1 NUM ] [ --percthresh2 NUM ]
                  [ --qalabel LABEL ] [--standardizeddetrendedmeans]
                  [ --show NAMES ] [ --hide NAMES ]
                  [ --nocalc NAMES ] [ --calc NAMES ]
                  xmlfile(s)... outputdir
```

Given 4-D input BXH- or XCEDE-wrapped image data, this program produces an HTML page with various useful QA plots, images, and measures, such as mean intensity per volume, center of mass per volume, per-slice variation, images of mean and standard deviation (across time), and others. Many of the QA measures are also placed in an XML events file for use by other programs. The index.html file (which should be readable by most Web browsers) and all other files will be put in outputdir. Various BXH- or XCEDE- wrapped images will be written during the process -- to delete these, use the --deleteXXXX options (the JPEG versions of these images displayed in the web page images will still remain).

--forcetr TR

This specifies the TR for the data (and overrides the TR in the image data, if any).

--zthresh1 NUM

--zthresh2 NUM

--percthresh1 NUM

--percthresh2 NUM

A count of images that exceed a given threshold is performed for some metrics. These options specify the two available thresholds for absolute z-score based measurements (i.e. how many standard deviations from the mean) and percent-based measurements (i.e. how many percent from the mean). Defaults are 3 and 4 for the z-score thresholds and 1 and 2 for the percent thresholds.

--qalabel LABEL

This specifies a label to be used in the title of the HTML report.

Default is to use a string derived from the input file name(s).

--standardizeddetrendedmeans

If specified, metrics for detrended data are shifted so that their means are the same.

--show NAMES

--hide NAMES

--calc NAMES

--nocalc NAMES

The --show and --hide options turn on or off the automatic display of the specified plots. Hidden plots are still available in the HTML file, and require only clicking on a checkbox to display them. The --calc and --nocalc options enable or disable the calculation of the data used in the specified plots (uncalculated data will therefore not be available for display). These are used to override the default behavior, which is to calculate and show all data.

Multiple plot names can be specified in the same option by separating them with commas, or can be specified in separate --show or --hide options.

The available basic plot names are:

```
volumemeans, maskedvolumemeans,  
meandiffvolumemeans, maskeddiffvolumemeans,  
cmassx, cmassy, cmassz, maskedcmassx, maskedcmassy, maskedcmassz,  
slicevar, 3dToutcount, 3dFWHM-X, 3dFWHM-Y, 3dFWHM-Z,  
meanstddevsfmr
```

The following additional plot names are convenient shorthands for groups of the above plots:

```
all, unmasked, masked, maskeddetrended, cmass, maskedcmass, fwhm
```

If conflicting options are provided for any particular plot, then the last relevant option is used. Thus, you can use

```
--nocalc all --calc 3dToutcount
```

to disable calculation of all but the voxel outlier plots.

## 0.34 fmriqa\_mean

Usage:

```
fmriqa_mean [opts] xmlfile outputfile
```

This program calculates the mean volume (over time) of a 4-D time series. The input file must be BXH or XCEDE file, and the output will be the same format, written to outputfile.

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## **0.35 fmriqa\_minmax**

Usage:

```
../../fmriqa/fmriqa_minmax xmlfile(s)...
```

This program merely computes the minimum and maximum values in the input BXH- or XCEDE-wrapped dataset and writes them to standard output.

## 0.36 fmriqa\_oediff

Usage:

```
fmriqa_oediff [opts] xmlfile outputfile
```

Given a 4-D BXH- or XCEDE-wrapped time series, this program calculates the cumulative difference between the even images (where the first selected image is 0) and the odd images. The input file must be BXH or XCEDE file, and the output is a 3-D image in the same format, written to outputfile.

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## 0.37 fmriqa\_phantomqa

Usage:

```
fmriqa_phantomqa [opts] xmlfile [outputbase]
```

This program is usually called by fmriqa\_phantomqa.pl, and is not likely to be useful to users on its own. This program takes a 4-D BXH- or XCEDE- wrapped dataset and calculates and writes various QA measures, designed for fMRI images of the BIRN calibration phantom.

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'. Default is to ignore first
    2 timepoints (2:), or 3 if the total number of timepoints is odd.
--zselect <str>
--zselect=<str>
    Chooses the slice number on which to compute the statistics. Must be a
    single unsigned integer within the range 0 <= x <= (numslices-1).
    Default is middle slice.
--summaryonly
    Don't generate ave, nave, std images.
--nofluct
    Don't run fluctuation analysis.
--noroi
    Don't run ROI-based analysis.
--maskfile <str>
--maskfile=<str>
    Use this mask (should be an XML file) instead of 30x30 (for 128x128
    slices) or 15x15. If 2-D, must match slice dimensions of input data.
    If 3-D, all three spatial dimensions must match (but only slice
    specified in zselect will be used).
--forcetr <str>
--forcetr=<str>
    If specified, this value (in seconds) will replace the TR specified in
    the input image file, if any.
```



## 0.38 fmriqa\_phantomqa.pl

Usage:

```
fmriqa_phantomqa.pl [--timeselect timepoints] [--zselect slice]
                    [--overwrite] [--verbose] [--summaryonly]
                    xmlfile [outputdir]
```

Given 4-D input BXH- or XCEDE-wrapped image data, this program produces an HTML page with various QA plots, images, and measures that were designed to be used with BIRN calibration phantom fMRI images. The index.html file (which should be readable by most Web browsers) and all other files will be put in outputdir, if specified, or otherwise will be placed in the same directory as the input file. Various summary measures will be printed to standard output. --summaryonly will only print the summary measures, and will not save any files.

## 0.39 fmriqa\_spikiness

Usage:

```
fmriqa_spikiness [opts] xmlfile [outputbase]
```

This program is usually called by wrapper scripts, and may not be useful to users on its own. This program takes a 4-D BXH- or XCEDE- wrapped dataset and calculates a 'spikiness' metric. Various 'spikiness' metrics are available and are selected using options. The size and meaning of the output data is dependent on the metric being calculated.

```
--version
    Print version string and exit.
--verbose
    More diagnostic output.
--metric <str>
--metric=<str>
    Which metric to return after fitting/detrending data.
    'diff' returns (value-fit) per voxel.
    'zscore' returns (value-fit)/stddev per voxel.
    'abszscore' returns (value-fit)/stddev per voxel.
    'afni' returns abs(value-fit)/mstddev per voxel (i.e. same as returned
    by Robert Cox's AFNI 3dDespike) where mstddev is a modified standard
    deviation that is less influenced by outlier points.
    'abszscoreslice' returns average abs(value-fit)/stddev per slice.
    'jackknife' (default) takes the output of 'abszscoreslice' and finds
    the "jackknife" z-score of each slice (over the volume) where the
    current slice is ignored in calculating mean/stddev.
    'jackknife' and 'abszscoreslice' produce a 2-D result set, whereas
    every other metric produces a 4-D result set.
--brainthresh <double>
--brainthresh=<double>
    Only voxels with a value greater than its_brainthresh are used in the
    calculation. Other voxels will return a metric of 0. Default is minus
    infinity or thereabouts.
--fit_method <str>
--fit_method=<str>
    Which fitting/detrending method to use.
    'mean' (default) simply uses the mean of each voxel's time-course.
    'linear' does a linear L1 fit of each voxel time-course.
    'afni' L1-fits the function used in Robert Cox's AFNI 3dDespike program
    to each voxel's time-course.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
```

```
--zselect=<str>  
    Just like timeselect, but for the 'z' dimension.
```

## 0.40 fmriqa\_stddev

Usage:

```
fmriqa_stddev [opts] xmlfile outputfile
```

This program calculates the standard deviation volume (over time) of a 4-D time series. The input file must be BXH or XCEDE file, and the output will be the same format, written to outputfile.

```
--version
    Print version string and exit.
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## 0.41 fmriqa\_volmeasures

Usage:

```
fmriqa_volmeasures [opts] xmlfile
```

This program calculates various measures per volume of the input 4-D time series, such as mean intensity and center-of-mass in all three dimensions. The input file must be BXH or XCEDE file, and the output will be written to standard output.

```
--version
    Print version string and exit.
--maskfile <str>
--maskfile=<str>
    Use this mask (should be a BXH or XCEDE XML file).
--timeselect <str>
--timeselect=<str>
    Comma-separated list of timepoints to use (first timepoint is 0). Any
    timepoint can be a contiguous range, specified as two numbers separated
    by a colon, i.e. 'START:END'. An empty END implies the last timepoint.
    The default step of 1 (one) in ranges can be changed using
    'START:STEP:END', which is equivalent to
    'START,START+STEP,START+(2*STEP),...,END'.
--xselect <str>
--xselect=<str>
    Just like timeselect, but for the 'x' dimension.
--yselect <str>
--yselect=<str>
    Just like timeselect, but for the 'y' dimension.
--zselect <str>
--zselect=<str>
    Just like timeselect, but for the 'z' dimension.
```

## 0.42 iowa-signafive2bxh

Usage:

```
iowa-signafive2bxh imagedir output.bxh
```

This program creates an XML wrapper for Univ. of Iowa-style GE Signa5 images.

imagedir is a directory containing I.\* images

--xcede

Write XCEDE-style XML files.

## 0.43 minc2bxx

Usage:

```
minc2bxx [opts] [mincfiles...] output.bxx
```

This program creates an XML wrapper for MINC images.

```
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize x <size_t>
--hintsize x=<size_t>
--hintsize y <size_t>
--hintsize y=<size_t>
--hintsize z <size_t>
--hintsize z=<size_t>
--hintsize t <size_t>
--hintsize t=<size_t>
--hintorigin x <double>
--hintorigin x=<double>
--hintorigin y <double>
--hintorigin y=<double>
--hintorigin z <double>
--hintorigin z=<double>
--hintorigin t <double>
--hintorigin t=<double>
--hintspacing x <double>
--hintspacing x=<double>
--hintspacing y <double>
--hintspacing y=<double>
--hintspacing z <double>
--hintspacing z=<double>
--hintspacing t <double>
--hintspacing t=<double>
--hintgap x <double>
--hintgap x=<double>
--hintgap y <double>
--hintgap y=<double>
--hintgap z <double>
--hintgap z=<double>
--hintgap t <double>
--hintgap t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, size x and size y
    are assumed correct in most image headers, but they, as well as size z
    and size t options may be useful with image type 'pfile'. Origin and
    spacing hints will be used by most image types.
--forceorientation <str>
--forceorientation=<str>
    This option will force the labeled orientation of the image to match
    the given three letter orientation code. Each letter must come from
    the following groups in any order: R(ight) or L(ef); A(nterior) or
    P(osterior); S(uperior) or I(nferior). Only one letter from each group
    is allowed.
```

--xcede  
Write XCEDE-style XML files.



## 0.44 pfile2bxh

Usage:

```
pfile2bxh [opts] [pfilehdr imagedata1...] output.bxh
```

This program creates an XML wrapper for GE P-files (and associated reconstructed image data).

```
--forceversion <float>
--forceversion=<float>
    Force version of P-file to be interpreted as this number.
--msbfirst
    Indicates that data is big-endian (default: little-endian).
--dimorder <str>
--dimorder=<str>
    Comma-separated names of dimensions from fastest-moving to
    slowest-moving (default: x,y,z,t).
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize_x <size_t>
--hintsize_x=<size_t>
--hintsize_y <size_t>
--hintsize_y=<size_t>
--hintsize_z <size_t>
--hintsize_z=<size_t>
--hintsize_t <size_t>
--hintsize_t=<size_t>
--hintorigin_x <double>
--hintorigin_x=<double>
--hintorigin_y <double>
--hintorigin_y=<double>
--hintorigin_z <double>
--hintorigin_z=<double>
--hintorigin_t <double>
--hintorigin_t=<double>
--hintspacing_x <double>
--hintspacing_x=<double>
--hintspacing_y <double>
--hintspacing_y=<double>
--hintspacing_z <double>
--hintspacing_z=<double>
--hintspacing_t <double>
--hintspacing_t=<double>
--hintgap_x <double>
--hintgap_x=<double>
--hintgap_y <double>
--hintgap_y=<double>
--hintgap_z <double>
--hintgap_z=<double>
--hintgap_t <double>
--hintgap_t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
    types will not use all these values. In particular, size_x and size_y
```

are assumed correct in most image headers, but they, as well as sizez and sizet options may be useful with image type 'pfile'. Origin and spacing hints will be used by most image types.

--forceorientation <str>

--forceorientation=<str>

This option will force the labeled orientation of the image to match the given three letter orientation code. Each letter must come from the following groups in any order: R(ight) or L(ef); A(nterior) or P(osterior); S(uperior) or I(nferior). Only one letter from each group is allowed.

--xcede

Write XCEDE-style XML files.

## 0.45 presentation2xml

Usage:

```
presentation2xml instructions.txt inputevents.txt [outputevents.xml]
```

presentation2xml takes a Presentation output file and an instruction file, and creates an XML events file. The instruction file indicates which columns in the Presentation file are of interest and what they should map to in the output event file. If the output file is not specified, the event data is written to standard output.

Options:

```
--extracttable
    Instead of creating XML as output, output as tabular text.
    This is a no-op for most formats, and is really only useful for
    converting E-Prime "recovery" logs into a tabular form.
--columnnames
    If this option is specified, only the columns of the table are
    printed (one per line) and the program exits.
--subtractonset SECS
    This option subtracts SECS from all onset times (default is 0).
    This is in addition to any other normalization that may occur
    (see use of 'firstmritime' below).
--colsep SEPARATOR
    This option specifies the column separator (default is tab).
```

The instruction file language is defined as follows:

```
COMMAND VALUESPEC [VALUESPEC...]
```

where COMMAND can be event, param, or block. VALUESPEC has one of the following formats:

```
[OUTVALUENAME=]COLUMNNAME[:UNITS]
OUTVALUENAME=@TEXT[:UNITS]
```

Each VALUESPEC defines values that should be passed through to the corresponding event, param, or block. In the first alternative listed above, the value comes from a column in the input file (optionally renamed to OUTVALUENAME) and in the second alternatives, the value is directly specified preceded by a '@' character. If OUTVALUENAME= is missing, then the VALUESPEC is equivalent to:

```
COLUMNNAME=COLUMNNAME[:UNITS]
```

OUTVALUENAME may not contain the equals sign ('='), at sign ('@'), quotes, or whitespace.

Either COLUMNNAME or TEXT may contain quoted substrings to protect special characters like colon (':'), equals sign ('='), at sign ('@'), or whitespace; otherwise these special characters are prohibited. A single quote will protect all characters until the next single quote, and likewise for double quote. The following examples show equivalent VALUESPECs:

```
description=DESC
description=D'E' "SC"

onset="Onset Time":secs
```

```
onset=Onset' 'Time:secs
onset=Onset' t Time:secs
```

Unquoted spaces separate VALUESPECs.

```
-----
'event' command:
-----
```

Each 'event' command creates a class of events in the output event file, where the contents of the event are specified by the VALUESPECs. In general, for each matching row (more later), it creates an event with the following contents:

```
<event type="$type" units="$units">
  <onset>$onset</onset>
  <duration>$duration</duration>
  <name>$name</name>
  <description>$description</description>
  <value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
  <value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
  ...
</event>
```

VALUESPECs whose OUTVALUENAMES start with a dollar sign ( \$ ) are "magic", and are interpreted in a value-specific way. VALUESPECs whose OUTVALUENAMES start with a percent sign ( % ) are explicitly non-magic. Any OUTVALUENAME not starting with a % or \$ is assumed to have an implicit % unless it matches a list of pre-defined magic values (below), in which case an implicit \$ is assumed.

Pre-defined magic values '\$type', '\$units', '\$onset', '\$duration', '\$name', and '\$description' are put in the appropriate child element or attribute of <event> (shown above). Only the '\$onset' VALUESPEC is required. Default value for '\$duration' is zero. All non-magic values are placed in <value> elements.

The pre-defined magic value '\$DURUNTIL' indicates that any row in the input used to create an event will have an ending time specified by the value of column COLUMNNAME in the current row. Likewise, the value '\$DURUNTILNEXTROW' does the same thing, but grabs the value from the next row. These are used to calculate the duration of this event. This may be specified more than once, and the first non-NULL column will be used. This option is used when a row does not have a duration column, and it must be calculated based on times in this or the subsequent row.

By default, only those rows whose '\$onset' column is non-empty and non-NULL will be processed as events. Certain magic OUTVALUENAMES further restrict the rows that are used for this event command. '\$MATCH' and '\$MATCHNONZERO' specify a column whose values indicate whether that row should be selected -- for '\$MATCH', the values must be non-empty and non-'NULL'; for '\$MATCHNONZERO', the values must also be non-zero. With '\$MATCHEQUAL', one specifies both a column and an actual value to match -- for the '\$MATCHEQUAL' value name (and only the '\$MATCHEQUAL' value name) the VALUESPEC syntax is extended in the following way:

```
$MATCHEQUAL=COLUMNNAME@MATCHVALUE
```

where COLUMNNAME and MATCHVALUE are the two relevant parameters.

```
-----
'block' command:
-----
```

The block command has the same usage as the event command. The same magic values apply to block commands as event commands. An '\$onset' value is again required, and '\$duration' is optional (assumed to be zero [0] if missing).

```
-----
'param' command:
-----
```

Each param command specifies a list of columns that should be put in the <params> section of the event file. These represent parameters that are constant (or default) throughout the events file. Each VALUESPEC represents one item to put in the <params> element as such:

```
<params>
...
<value name="OUTVALUENAME1" units="UNITS1">VALUE1</value>
<value name="OUTVALUENAME2" units="UNITS2">VALUE2</value>
...
</params>
```

Only the first non-empty, non-NULL field in the column specified by a 'param' will be used. Be aware of this if this column does not have the same value in every row.

There is one magic OUTVALUENAME (maybe more later) '\$firstmritime', which will generate the following element:

```
<params>
<firstmritime>0</firstmritime>
</params>
```

If '\$firstmritime' is specified, it (and all '\$onset' VALUESPECs) must have UNITS specified. All '\$onset' columns are normalized by this value, so their units and '\$firstmritime' units must match.

## **0.46 printfrags**

Usage:

```
../../utils/printfrags xmlfile
```

This program prints out the 'frags' in a BXH/XCEDE file.

## 0.47 showplay2xml

Usage:

```
../../utils/showplay2xml [opts] pdigmfile [eventfile.xml]  
../../utils/showplay2xml [opts] run.ppf [show.out] eventfile.xml
```

Options: --snaptotr TR[:offset]  
          --overwrite

This program creates an XML events file from the output of CIGAL/showplay. In the first example, if the second argument (eventfile.xml) is missing, results are sent to standard output. In the second example, eventfile.xml must be specified. --snaptotr indicates that each event time should be shifted to the closest timepoint that is a multiple of TR, with an optional offset, separated from the TR by a colon. Default offset is 0. If --overwrite is not specified, then existing files will not be overwritten.

## 0.48 signafive2bxh

Usage:

```
signafive2bxh [opts] [signa5files...] output.bxh
```

This program creates an XML wrapper for GE Signa5 image files.

```
--dimzsize <size_t>
--dimzsize=<size_t>
    Specifies the size of the z dimension (i.e. number of slices per
    timepoint). Default is to use the number of input files. Equivalent
    to (and overrides) --hintsizez.
--dimtsize <size_t>
--dimtsize=<size_t>
    Specifies the size of the t dimension (i.e. number of timepoints).
    Default is number of input files divided by number of slices per
    timepoint (as specified by --dimzsize). Equivalent to (and overrides)
    --hintsize_t.
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize_x <size_t>
--hintsize_x=<size_t>
--hintsize_y <size_t>
--hintsize_y=<size_t>
--hintsize_z <size_t>
--hintsize_z=<size_t>
--hintsize_t <size_t>
--hintsize_t=<size_t>
--hintorigin_x <double>
--hintorigin_x=<double>
--hintorigin_y <double>
--hintorigin_y=<double>
--hintorigin_z <double>
--hintorigin_z=<double>
--hintorigin_t <double>
--hintorigin_t=<double>
--hintspacing_x <double>
--hintspacing_x=<double>
--hintspacing_y <double>
--hintspacing_y=<double>
--hintspacing_z <double>
--hintspacing_z=<double>
--hintspacing_t <double>
--hintspacing_t=<double>
--hintgap_x <double>
--hintgap_x=<double>
--hintgap_y <double>
--hintgap_y=<double>
--hintgap_z <double>
--hintgap_z=<double>
--hintgap_t <double>
--hintgap_t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
```



types will not use all these values. In particular, `sizeX` and `sizeY` are assumed correct in most image headers, but they, as well as `sizeZ` and `sizeT` options may be useful with image type 'pfile'. Origin and spacing hints will be used by most image types.

--forceorientation <str>

--forceorientation=<str>

This option will force the labeled orientation of the image to match the given three letter orientation code. Each letter must come from the following groups in any order: R(ight) or L(efT); A(nterior) or P(osterior); S(uperior) or I(nferior). Only one letter from each group is allowed.

--xcede

Write XCEDE-style XML files.

## 0.49 ximg2bxh

Usage:

```
ximg2bxh [opts] [ximgfiles...] output.bxh
```

This program creates an XML wrapper for GE Ximg image files.

```
--dimzsize <size_t>
--dimzsize=<size_t>
    Specifies the size of the z dimension (i.e. number of slices per
    timepoint). Default is to use the number of input files. Equivalent
    to (and overrides) --hintsizez.
--dimtsize <size_t>
--dimtsize=<size_t>
    Specifies the size of the t dimension (i.e. number of timepoints).
    Default is number of input files divided by number of slices per
    timepoint (as specified by --dimzsize). Equivalent to (and overrides)
    --hintsize_t.
--inputsfromfile <str>
--inputsfromfile=<str>
    Read list of input files from this file.
--version
    Print version string and exit.
--hintsize_x <size_t>
--hintsize_x=<size_t>
--hintsize_y <size_t>
--hintsize_y=<size_t>
--hintsize_z <size_t>
--hintsize_z=<size_t>
--hintsize_t <size_t>
--hintsize_t=<size_t>
--hintorigin_x <double>
--hintorigin_x=<double>
--hintorigin_y <double>
--hintorigin_y=<double>
--hintorigin_z <double>
--hintorigin_z=<double>
--hintorigin_t <double>
--hintorigin_t=<double>
--hintspacing_x <double>
--hintspacing_x=<double>
--hintspacing_y <double>
--hintspacing_y=<double>
--hintspacing_z <double>
--hintspacing_z=<double>
--hintspacing_t <double>
--hintspacing_t=<double>
--hintgap_x <double>
--hintgap_x=<double>
--hintgap_y <double>
--hintgap_y=<double>
--hintgap_z <double>
--hintgap_z=<double>
--hintgap_t <double>
--hintgap_t=<double>
    These options will provide a hint to the program of the 'size',
    'origin', 'spacing', or 'gap' of the specified dimension. Some image
```

types will not use all these values. In particular, `sizeX` and `sizeY` are assumed correct in most image headers, but they, as well as `sizeZ` and `sizeT` options may be useful with image type 'pfile'. Origin and spacing hints will be used by most image types.

--forceorientation <str>

--forceorientation=<str>

This option will force the labeled orientation of the image to match the given three letter orientation code. Each letter must come from the following groups in any order: R(ight) or L(efT); A(nterior) or P(osterior); S(uperior) or I(nferior). Only one letter from each group is allowed.

--xcede

Write XCEDE-style XML files.

## **Part III**

# **Example files**

## .1 Example bxh\_eventstats options file

```
## This is a sample bxh_eventstats options file.
## To use it, you need to specify it on the command line, e.g.:
##   bxh_eventstats --optsfromfile myOptionFile.txt [other arguments...]
## All lines beginning with # are comments (like this line).
## For more help, try "bxh_eventstats --help"

## This specifies the query language. Valid languages are 'XPath' and
## 'event'. 'event' is a simpler language, 'XPath' is more powerful.
querylanguage event

## These specify the beginning and starting points of the analysis.
## Time points before and after will be ignored. Default is to include
## all points.
# startpt 0
# endpt 100 # example!

## How many time points before/after the event will be considered as
## part of the epoch? These are both in TRs. If both are 0, then
## the event time point itself is the only point in the epoch.
## The number of time points in the epoch is ptsbefore + ptsafter + 1.
## Usually both should be non-negative.
## These options (or secsbefore and secsafter) are required.
ptsbefore 3
ptsafter 8

## Alternatively, you can specify the epoch duration in seconds.
## These options (or ptsbefore and ptsafter) are required.
# secsbefore 3
# secsafter 8

## Which points (relative to the event) should be considered part of
## the average baseline? These are both offsets (in TRs) from the event.
## If both are 0, then the event time point is the baseline.
## The number of time points in the baseline is
## (-1)*basestartoffset + baseendoffset + 1.
## Usually basestartoffset should be non-positive and baseendoffset
## should be non-negative.
## These options are required.
basestartoffset -3
baseendoffset 0

## If specified, this will create a brain mask based on the first
## image file using the 'localmin' method of bxh_brainmask.
createbrainmask

## Alternatively you can specify a brain mask explicitly with the
## maskfile option:
# maskfile /my/mask/file.bxh

## If specified, this will force all events to occur on the closest TR.
## This will decrease computation time, but make the results less
## accurate.
# nointerp

## If the input image files do not specify a TR, or you would otherwise
```

```

## like to override the value, use this option. Please use image file
## formats that specify a TR!
# forcetr 3.0 # example!

#####
# FREQUENCY FILTERING #
#####

## If you would like to do any frequency filtering of each voxel across
## time, use the following options.

## This option, if present, adds temporal filtering using a Chebyshev
## filter, and chooses which type of filtering to use. Valid choices are
## 'lowpass', 'highpass', 'bandpass', or 'bandstop'. Each filter is
## parameterized by one or more instances of "tfilterperiod". 'lowpass'
## or 'highpass' require one tfilterperiod option, specifying the stop
## or start frequency respectively. 'bandpass' or 'bandstop' require two
## "tfilterperiod" options, specifying the start and stop frequencies, in
## any order (larger period/smaller frequency is assumed to be start
## frequency for 'bandpass' and stop frequency for 'bandstop').
#tfiltertype bandpass # example!

## This option specifies the frequency parameters for the filter in terms
## of the period (i.e. 1/frequency) in seconds per cycle. May be
## specified once for 'lowpass' and 'highpass' filter types, twice for
## 'bandpass' and 'bandstop' filter types, and must be greater than 0.
## Must be greater than 2*TR.
#tfilterperiod 7 # example!
#tfilterperiod 100 # example!

## This option specifies the percent ripple for the Chebyshev filter. If
## 0 [zero], which is the default, then the filter is a Butterworth
## filter.
#tfilterripple 0 # example!

## Order of the temporal filter. Default is 6.
#tfilterorder 6

#####
# QUERIES #
#####

# Each query represents a "bin".
# Each query may have the following options:
# querylabel - specifies a text label for this bin
# query - specifies a query that matches events that should fall into this bin
# queryfilter - if specified, filters the events matching the main query
# queryepochexclude - if specified, filters the events by epoch
# There must be at least one "query" option.
# The others are optional, but if they exist, they must be specified for
# every query. i.e. if you specify "queryepochexclude" for one query,
# you must specify it for all queries. Empty strings for queryfilter
# and queryepochexclude do what you expect -- they have no effect.

# Example queries:
# Find all events whose 'code' value is 1, but exclude those events
# whose epochs include a volume whose mean intensity is more than 3

```

```
# standard deviations from the mean (assumes you have a QA event file
# that has this information for the given run).
querylabel Rest
query "code==1"
queryfilter ""
queryepochexclude "volmean_z_indiv > 3"

# Find all events whose 'shape' value is 'square', but only those who
# happened simultaneously with a blue background. Exclude those events
# whose epochs include a volume whose mean intensity is more than 3
# standard deviations from the mean (assumes you have a QA event file
# that has this information for the given run).
querylabel SquareBlueBack
query "shape=='square'"
queryfilter "background=='blue'"
queryepochexclude "volmean_z_indiv > 3"

#####
# CORRELATION #
#####

# The hemodynamic curve template to correlate against for each query/bin.
# r- and t-statistic volumes will be generated for each query/bin.
template 0,0,0,0,0.321601345,0.890377726,1,0.566742211,0.272476597,0.082058953,0.014452811,0

# Bin comparisons, if any, you wish to make. Specify two query labels
# separated by a dash '-'. The labels themselves may not contain dashes,
# of course.
tcompare DevTone-StdTone
```

## Index

afni2bxh, 16  
analyze2bxh, 18  
  
batch\_showplay2xml, 20  
bxh2analyze, 21  
bxh2pgm, 23  
bxh2ppm, 24  
bxh\_brainmask, 30  
bxh\_correlate, 32  
bxh\_epochavg, 33  
bxh\_event2table, 36  
bxh\_eventmerge, 37  
bxh\_eventresp, 38  
bxh\_eventstats, 39  
bxh\_mean, 44  
bxh\_tfilter, 48  
bxh\_ttest, 49  
bxhabsorb, 26  
bxhreorient, 45  
bxhselect, 46  
bxhsetorient, 47  
  
dicom2bxh, 50  
dumpheader, 52  
  
eprime2xml, 53  
eventstable2xml, 56  
eventstats, 59  
extractimagedata, 60  
extractxyztdata, 61  
  
ffile2bxh, 62  
fmriqa\_count, 63  
fmriqa\_generate.pl, 65  
fmriqa\_mean, 67  
fmriqa\_minmax, 68  
fmriqa\_oediff, 69  
fmriqa\_phantomqa, 70  
fmriqa\_phantomqa.pl, 71  
fmriqa\_spikiness, 72  
fmriqa\_stddev, 74  
fmriqa\_volmeasures, 75  
  
iowa-signafive2bxh, 76  
  
minc2bxh, 77  
  
pfile2bxh, 79  
presentation2xml, 81  
printfargs, 84  
  
showplay2xml, 85  
signafive2bxh, 86  
  
ximg2bxh, 88