Documentation for: # xcat()

Version 1.2

Requires: Numpy Python Module

This toolbox provides a Python based category analysis engine modeled off of the original MS DOS CATANAL program, designed for the automated processing of behavioral data files. The xcat function has a built in check to see if the input file exists, returning 'nan' for all variables if the file does not exist. The function can take a number of formats, including Neuroscan Stim2:

Column 1 corresponds to the Trial

Column 2 corresponds to the Response

Column 3 corresponds to the Event Code

Column 4 corresponds to the Response Accuracy (1 is correct, 0 is incorrect, -1 is a response prior to the response window, -2 is a response after the response window)

Column 5 corresponds to the Reaction Time Latency

Included in the toolbox are a number of basic functions that allow for manipulation of the data files:

mergedatfiles – Exports the data from two different DAT files as a single file.

```
File1 = 'C:\Studies\Demo\Raw\File1a.dat'
File2 = 'C:\Studies\Demo\Raw\File1b.dat'
CompleteFile = 'C:\Studies\Demo\Raw\File1Complete.dat'
mergedatfiles(inputfile1 = File1, inputfile2 = File2, outputfile = CompleteFile)
```

splitdatfiles – Splits a data file into two different DAT files. The default action is to split the file in half for earlier trials vs later trials. An optional method parameter takes a numeric input and overrides this default action to split the data every N trials.

```
File = 'C:\Studies\Demo\Raw\File1.dat'
Part1 = 'C:\Studies\Demo\Raw\File1part1.dat'
Part2 = 'C:\Studies\Demo\Raw\File1part2.dat'
splitdatfiles(inputfile = File, outputfile1 = Part1, outputfile2 = Part2, method = [])
```

obtaindatheaderinfo – Parses a DAT file to return the requested Header information

```
File = 'C:\Studies\Demo\Raw\File1.dat'
Filetime = obtaindatheaderinfo(inputfile = File, content = 'time........=')
```

The toolbox also provides functions for transforming data to enable its use with other programs:

createboldoutputfile – Uses the PSYDAT data to create files for BOLD fMRI data processing. The custom PSYDAT format is outlined below:

Column 1 corresponds to the Trial

Column 2 corresponds to the Event Type ('Stimulus' or 'Response')

Column 3 corresponds to the Stimulus Duration

Column 4 corresponds to the Inter-Stimulus Interval

Column 5 corresponds to the Inter-Trial Interval

Column 6 corresponds to the Event Code

Column 7 corresponds to the Response

Column 8 corresponds to the Response Accuracy (1 is correct, 0 is incorrect, -1 is a response prior to the response window, -2 is a response after the response window)

Column 9 corresponds to the Reaction Time Latency

Column 10 corresponds to the Global Clock Time from the task onset

Column 11 corresponds to if a Trigger was sent (1 is sent, 0 is not sent)

Column 12 corresponds to the Response Window Minimum

Column 13 corresponds to the Response Window Maximum

Column 14 corresponds to the Stimulus that was presented

```
File = 'C:\Studies\Demo\Raw\File2.psydat'
Part1 = 'C:\Studies\Demo\Raw\File2BoldCorrect.dat'
Part2 = 'C:\Studies\Demo\Raw\File2BoldIncorrect.dat'
createboldoutputfile(inputfile = File, correctoutputfile = Part1, incorrectoutputfile = Part2,
trialtypes = [10, 11, 12, 13], method = 'duration')
```

createneuroscanoutputfile – Uses the PSYDAT data to create Neuroscan Stim2 DAT files which can be merged with Neuroscan Edit 4.

```
FileIn = 'C:\Studies\Demo\Raw\File2.psydat'
FileOut = 'C:\Studies\Demo\Raw\File2.dat'
createneuroscanoutputfile(inputfile = FileIn, outputfile = FileOut, enableresponseoutput = False,
markalleventsasstim = True)
```

TranslateBehavioralData – Class of functions that enable transforming e-prime data outputs (.TXT) into the PSYDAT format compatible with the xcat class of functions. Relies on label matching to identify and numerically code trial types.

```
# Label Matching Values
translate = xcat.TranslateBehavioralData()
translate.trial = 'Trial'
translate.durationlabel = 'Duration'
translate.isilabel = 'ISI'
translate.itilabel = 'ITI'
translate.typelabel = 'Type'
translate.responselabel = 'Target.RESP'
translate.accuracylabel = 'Target.ACC'
translate.rtlabel = 'Target.RT'
translate.clocktimelabel = 'Target.OnsetTime'
translate.triggerlabel = 'Trigger'
translate.minresponselabel = 'MinRespWin'
translate.maxresponselabel = 'MaxRespWin'
translate.stimuluslabel = 'ImageTarget'

# If trial type information is not already coded, labels and matching values can be used
# The format: [numeric code for trial type, [label and matching value to search for]]
translate.breakdowns = []
translate.breakdowns.append([10, ['ImageTarget: Go']])
translate.breakdowns.append([20, ['ImageTarget: NoGo']])

# Initialize the Translation
translate.run(inputfile = 'C:\Studies\Demo\Raw\File1.txt', outputfile =
'C:\Studies\Demo\Raw\File1.psydat')
```

The core component of the xcat() toolbox is a Python class function BehavioralAnalysis() with an input file, and a list of the specific trial types of interest as the parameters. The function outputs the following behavioral performance metrics:

'totaltrials' – Total number of trials

'meanrt' – Mean reaction time

'medianrt' – Median reaction time

'sdrt' – Standard deviation of reaction time

'cvrt' – Coefficient of variation of reaction time (SD of RT / Mean RT)

'responseaccuracy' – Response accuracy

'inverseefficiency' – Mean RT divided by the proportion of correct responses (Townsend & Ashby, 1983)

'totalerrors' – Total number of incorrect trials

'totalcommissionerrors' – Total number of trials where the incorrect button was pressed

'totalomissionerrors' – Total number of trials where no button was pressed when one should have been

'totalimpulsiveerrors' – Total number of trials where a response was executed prior to the response window

'totaldelayederrors' – Total number of trials where a response was executed after the response window

'correctruns' – Number of times two or more correct responses occurred

'correctdist' – Mean number of sequential correct responses

'commissionerrorruns' – Number of times two or more incorrect responses occurred

'commissionerrordist' – Mean number of sequential errors of commission

'omissionerrorruns' – Number of times two or more errors of omission occurred

'omissionerrordist' – Mean number of sequential errors of omission

'impulsiveerrorruns' – Number of times two or more impulsive errors occurred

'impulsiveerrordist' – Mean number of sequential impulsive errors

'delayederrorruns' – Number of times two or more delay errors occurred

'delayederrordist' – Mean number of sequential delay errors

'errorlatency' – Mean reaction time for errors of commission

'errorlatencysd' – Standard deviation of reaction time for errors of commission

'matchcorrectlatency' – Mean reaction time for a subset of correct trials matched to error trials based on RT

'matchcorrectlatencysd' – Standard deviation of reaction time for match correct trials

'posterroraccuracy' – Response accuracy for trials immediately following errors of commission

‘postmatchcorrectaccuracy’ – Response accuracy for trials immediately following match correct trials

'posterrorlatency' – Mean reaction time for trials immediately following errors of commission

'postmatchcorrectlatency' - Mean reaction time for trials immediately following match correct trials

Behavioral performance metrics can either be called individually by specifying the specific metric of interest, collectively using ‘fulloutput’, or a subset of these metrics can be specified using ‘shortoutput’ which provides a list of the following metrics:

'totaltrials', 'meanrt', 'medianrt', 'sdrt', 'cvrt', 'responseaccuracy', 'inverseefficiency', 'totalerrors', 'totalcommissionerrors', 'totalomissionerrors', 'totalimpulsiveerrors', 'totaldelayederrors'

An additional component of the xcat() toolbox is a Python class function ConditionalAccuracyFunction () with an input file, and a list of the specific trial types of interest as the parameters, and the number of bins to compute reaction time within. This function computes behavioral performance metrics for trials falling within the percentile bins specified (i.e., if 3 bins are specified the function will output bins corresponding to the trials with a RT below the 33.3rd percentile of the RT distribution, trials with a RT between the 33.3rd and 66.6th percentile of the RT distribution, and trials with a RT above the 66.6th percentile of the RT distribution). The function outputs the following behavioral performance metrics:

'totaltrials' – Total number of trials

'meanrt' – Mean reaction time

'medianrt' – Median reaction time

'sdrt' – Standard deviation of reaction time

'cvrt' – Coefficient of variation of reaction time (SD of RT / Mean RT)

'responseaccuracy' – Response accuracy

Behavioral performance metrics can either be called individually by specifying the specific metric of interest or collectively in the same way the BehavioralAnalysis() function operates. Specifying ‘fulloutput’ will output a list of all metrics for all bins (‘fulloutputlabels’ can be used to output the labels), or the bin specific metrics can be outputted using ‘bin1_output’ (‘bin1_outputlabels’ can be used to output the labels) and specifying the bin of interest.

To implement this toolbox, you must have the appropriate version of Python from https://www.python.org/ for your machine (32 bit vs 64 bit). Python 2.7.9 is recommended.

Because the Numpy Python Package is required, the easiest method for implementing this toolbox is to use PsychoPy (http://www.psychopy.org/), as the standalone version coder window contains all of the necessary packages. To run outside of PsychoPy it is necessary to download the Numpy Package:

Windows users:

1. Go to this website: http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy
2. Download the wheel package for your version of python and operating system to the root of your hard drive (ex. C:).
3. Open the command prompt (type cmd into the start menu search) and type (note that the file name should be the same as the file you downloaded):

```
cd\
pip install numpy-1.9.2+mkl-cp27-none-win_amd64.whl
```

Mac users:

1. Open the terminal and type:

```
pip install numpy
```

Example Implementation Code Running Locally

```
import os
import xcat

taskoutput = xcat.BehavioralAnalysis()
taskoutput.run(inputfile = 'File1.dat', trialtypes = [14, 15, 16, 17, 18, 19, 24, 25, 26, 27, 28, 29])

print('\nFull Behavioral Output')
print(taskoutput.fulloutputlabels)
print(taskoutput.fulloutput)

print('\nShort Behavioral Output')
print(taskoutput.shortoutputlabels)
print(taskoutput.shortoutput)

print('\nSpecific Behavioral Variable')
print(taskoutput.meanrt)

os.remove(os.path.realpath(__file__)[0:-2] + 'pyc') # Remove compiled python file
xcat.cleanupcompiledfiles()
print('Processing Complete')
```

Example Implementation Code Running with the xcat toolbox in another file

```
import os
execfile(r'C:\Studies\Engine\xcat.py')

taskoutput = BehavioralAnalysis()
fullfilepath = 'C:\Studies\Demo\Raw\File1.dat'
taskoutput.run(inputfile = fullfilepath, trialtypes = [14, 15, 16, 17, 18, 19, 24, 25, 26, 27, 28, 29])

print('\nFull Behavioral Output')
print(taskoutput.fulloutputlabels)
print(taskoutput.fulloutput)

print('\nShort Behavioral Output')
print(taskoutput.shortoutputlabels)
print(taskoutput.shortoutput)

print('\nSpecific Behavioral Variable')
print(taskoutput.meanrt)

os.remove(os.path.realpath(__file__)[0:-2] + 'pyc') # Remove compiled python file
print('Processing Complete')
```

Example Implementation Code of the Condition Accuracy Function Running Locally

```python
import os
import xcat

taskoutput = xcat.ConditionalAccuracyFunction()
taskoutput.run(inputfile = 'File1.dat', trialtypes = [14, 15, 16, 17, 18, 19], bins = 3)

print('\nFull Behavioral Output')
print(taskoutput.fulloutputlabels)
print(taskoutput.fulloutput)

print('\nBin 1 Behavioral Output')
print(taskoutput.bin1_outputlabels)
print(taskoutput.bin1_output)

print('\nBin 2 Behavioral Output')
print(taskoutput.bin2_outputlabels)
print(taskoutput.bin2_output)

print('\nBin 3 Behavioral Output')
print(taskoutput.bin3_outputlabels)
print(taskoutput.bin3_output)

print('\nSpecific Behavioral Variable')
print(taskoutput.bin1_meanrt)

os.remove(os.path.realpath(__file__)[0:-2] + 'pyc') # Remove compiled python file
xcat.cleanupcompiledfiles()
print('Processing Complete')
```