

***Supporting Information – CcsCal***

***Large-scale Structural Characterization of Drug and Drug-Like Compounds by  
High-Throughput Ion Mobility-Mass Spectrometry***

Kelly M. Hines,<sup>1</sup> Dylan H. Ross,<sup>1</sup> Kimberly L. Davidson,<sup>2</sup> Matthew F. Bush,<sup>2</sup> Libin Xu<sup>1</sup>

<sup>1</sup>Department of Medicinal Chemistry, University of Washington, Seattle, WA

<sup>2</sup>Department of Chemistry, University of Washington, Seattle, WA

Address correspondence to:  
Libin Xu, Ph.D.  
Department of Medicinal Chemistry  
University of Washington  
Tel: (206) 543-1080  
Fax: (206) 685-3252  
Email: [libinxu@uw.edu](mailto:libinxu@uw.edu)

## OVERVIEW

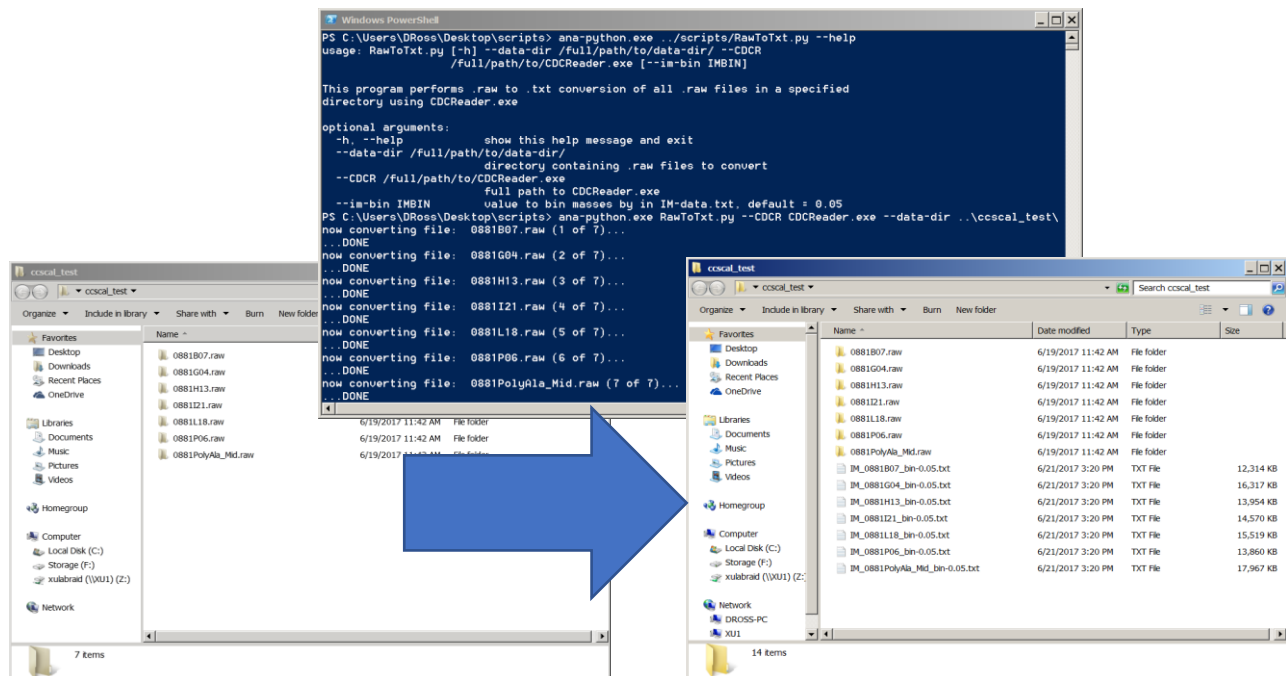
CcsCal is a program written primarily in python that automates extraction of drift times from raw IM-MS data, and construction and application of a CCS calibration curve. The program is organized into a single python package containing sub-modules for performing the various steps in the data analysis protocol. The module may be run as a single program, or its components may be used in isolation by importing them from separate python scripts. Additionally, a convenient API for processing data from external sources is provided in the form of the CcsCalibrationExt class, available in the CcsCal.processing.CcsCalibration module. Extensive documentation of all modules is available in the form of linked HTML documents (see section: **DOCUMENTATION**). CcsCal is available upon request.

## PREREQUISITES

- Windows 7 or newer (also works on Linux/MacOS, see section: **RUNNING ON LINUX/MACOS**)
- Python 2.7 (not compatible with Python 3.x)
- Python libraries:
  - numpy
  - scipy
  - matplotlib
- CDCReader.exe and cdt.d11 (from UniDec)

## SETUP

Before running analysis with CcsCal all IM-MS data files must be converted from Waters .raw to a usable plain-text format, and collected into a single directory. This conversion can be performed using CDCReader.exe and cdt.d11 (from UniDec). Provided with CcsCal is a python script RawToTxt.py that will perform a batch conversion of all .raw data files in a specified directory into a plain-text format using these tools. An example of this conversion using RawToTxt.py is shown below:



A single configuration file is used to provide all of the input parameters required for analysis. An example configuration file is shown to the right. All lines beginning with a semicolon are treated as comments and ignored, with the exception of lines containing a semicolon and a three-letter identifier followed by a number or string. These identifier parameters are required. The `edc` and `tpi` identifiers are instrumental parameters that can be obtained for a given data file by going into the `.raw` directory and looking at `_etern.inf`. The `edc` parameter is denoted by “EDC Delay Coefficient” and the `tpi` parameter is denoted by “ADC Pusher Frequency ( $\mu\text{s}$ )”. The rest of the identifiers specify paths to various input/output files and some parameters for data manipulation. Importantly, the `mwn` identifier must be set to a value larger than that used for the `--im-bin` parameter during the data conversion with `CDCReader.exe`, not doing so may lead to missing or shifted peaks. Additionally, there are two sets of listed parameters: masses and reference CCS values for the CCS calibrants, and masses and data filenames for the compounds to be analyzed. An example configuration file is provided with `CcsCal` and may be used as a template for other runs.

```

1 ; [ general ]
2 ;
3 ; full path and name to same the report file under
4 ;rfn          = /Users/DRoss/Desktop/ccscal_test/ccscal-report.txt
5 ;
6 ; mass window to extract drift time data from
7 ;mwn         = 0.5
8 ;
9 ; edc parameter
10 ;edc         = 1.35
11 ;
12 ; TOF pusher interval
13 ;tpi        = 69.0
14 ;
15 ; Savitsky-Golay smoothing parameters (set both to 0 for no smoothing)
16 ; filter window
17 ;sgw        = 0
18 ; filter polynomial order
19 ;sgp        = 0
20 ;
21 ; [ calibrants ]
22 ;
23 ; full path and name to save calibration curve file under
24 ;cfn         = /Users/DRoss/Desktop/ccscal_test/cal-curve.png
25 ;
26 ; full path and name of the CCS calibration data file
27 ;cdf         = /Users/DRoss/Desktop/ccscal_test/IM_0881PolyAla_Mid.txt
28 ;
29 ; information for CCS calibrants
30 ; mass      literature ccs
31 161.0926   136.05
32 232.13     150.77
33 303.167    163.28
34 374.204    177.55
35 445.241    190.80
36 516.278    205.95
37 587.315    222.68
38 658.352    236.40
39 729.389    249.13
40 800.426    262.60
41 871.463    275.98
42 942.501    289.25
43 1013.537   300.78
44 1084.574   312.80
45 1155.611   327.13
46 1226.648   338.15
47 1297.685   350.08
48 1368.722   359.28
49 1439.759   370.23
50 ;
51 ; [ compounds ]
52 ;
53 ; full path to the directory containing the compound data files
54 ;crd         = /Users/DRoss/Desktop/ccscal_test/
55 ;
56 ; information for compounds
57 compound    start
58 ; data file      mass
59 IM_0881L18.txt  195.0877
60 IM_0881P06.txt  304.0866
61 IM_0881G04.txt  406.2337
62 IM_0881I21.txt  497.1322
63 IM_0881H13.txt  611.1607
64 IM_0881B07.txt  734.4685

```

## RUNNING

The primary means of running `CcsCal` is by directly calling the module, invoking its `__main__` method. This will go through the full data analysis protocol: parsing the configuration file, pre-processing the plain-text data files, extracting drift times for all analytes, and constructing and applying a CCS calibration curve. Passing

```

PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus> ana-python.exe -m CcsCal --help
usage: __main__.py [-h] [-i "/full/path/to/ccscal_input.txt"] [--test]

This program refers to a specified input file and automatically performs a CCS
calibration then obtains calibrated CCS for all specified masses, finally
printing a full report containing the results of the calibration and CCS of
all compounds

optional arguments:
  -h, --help            show this help message and exit
  -i "/full/path/to/ccscal_input.txt", --input "/full/path/to/ccscal_input.txt"
                        full path to ccscal_input.txt
  --test                runs the included test suite
PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus>

```

the `-h` or `--help` flag when calling `CcsCal` will cause the program to produce a descriptive help message with the expected parameters and exit. An example of this help message is included below. The syntax for calling `CcsCal` is important to take note of, specifically the `-m` flag after the python interpreter that signals `CcsCal` should be interpreted as a module.

To perform the actual data analysis only the `--input` flag must be provided, along with the path to the configuration file input. An example command is shown to the right. As CcsCal runs verbose output is produced to the console indicating what step the program is on. An example of this verbose output is included below.

```
Windows PowerShell
PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus> ana-python.exe -m CcsCal --input C:\Users\DRoss\Documents\cal-plusplus\CcsCalInput.txt
```

```
Windows PowerShell
Getting Calibrated CCS...
Extracting Drift Time for Mass: 406.2337 from Data File: IM_0881G04_bin-0.05.txt (3 of 6)...
Calling PreProcessTxt.exe...
~/Users/DRoss/Documents/GitHub/ccscal-plusplus/CcsCal/input/PreProcessTxt.exe /Users/DRoss/Desktop/ccscal_test/IM_0881G04_bin-0.05.txt 406.2337 1.0
...Done

Getting Calibrated CCS...
Extracting Drift Time for Mass: 497.1322 from Data File: IM_0881I21_bin-0.05.txt (4 of 6)...
Calling PreProcessTxt.exe...
~/Users/DRoss/Documents/GitHub/ccscal-plusplus/CcsCal/input/PreProcessTxt.exe /Users/DRoss/Desktop/ccscal_test/IM_0881I21_bin-0.05.txt 497.1322 1.0
...Done

Getting Calibrated CCS...
Extracting Drift Time for Mass: 611.1607 from Data File: IM_0881H13_bin-0.05.txt (5 of 6)...
Calling PreProcessTxt.exe...
~/Users/DRoss/Documents/GitHub/ccscal-plusplus/CcsCal/input/PreProcessTxt.exe /Users/DRoss/Desktop/ccscal_test/IM_0881H13_bin-0.05.txt 611.1607 1.0
...Done

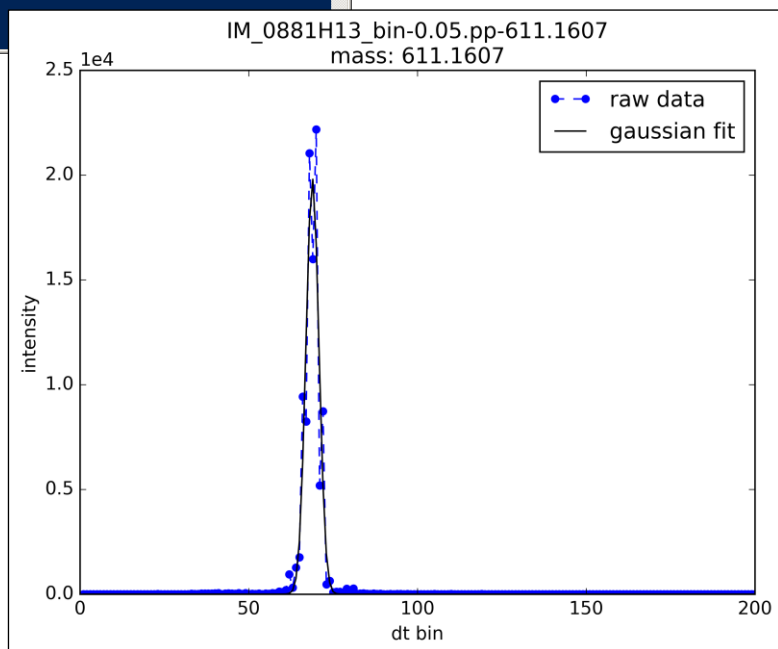
Getting Calibrated CCS...
Extracting Drift Time for Mass: 734.4685 from Data File: IM_0881B07_bin-0.05.txt (6 of 6)...
Calling PreProcessTxt.exe...
~/Users/DRoss/Documents/GitHub/ccscal-plusplus/CcsCal/input/PreProcessTxt.exe /Users/DRoss/Desktop/ccscal_test/IM_0881B07_bin-0.05.txt 734.4685 1.0
...Done

Getting Calibrated CCS...
CcsCal Complete.

total time: 71.0 s
PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus>
```

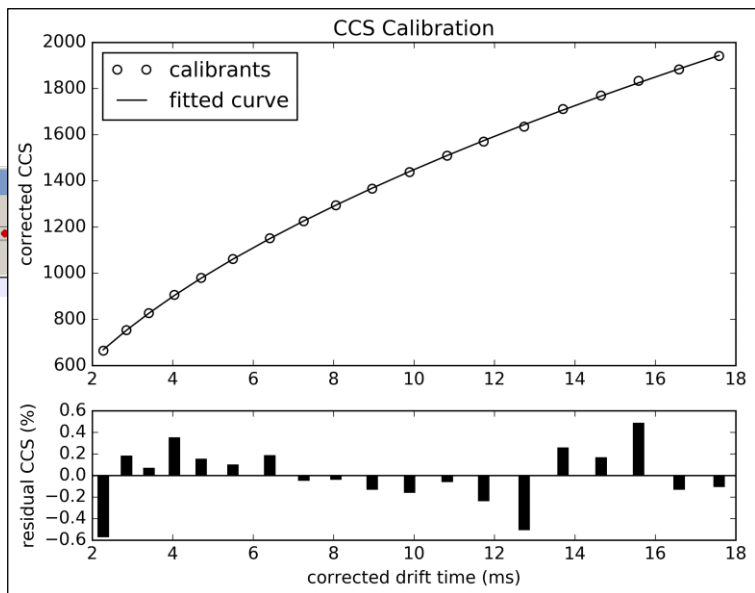
## RESULTS

During execution CcsCal produces several files. For each data file-mass pair, a pre-processed plain-text data file is produced with the same name as the parent data file but with “.pp-mass” appended to the file name (e.g. `IM_0881H13.txt` → `IM_0881H13.pp-611.1607.txt`). Additionally, for each pair a .png image is generated containing a plot of the raw data as well as a gaussian function fit to that data. An example plot is included to the right.



```

C:\Users\DRoss\Desktop\ccscal_test\ccscal-report.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ccscal-report.txt
1 ccscal-report
2 Generated by CcsCal on 06/22/17 10:10:15
3
4 =====
5
6 +-----+
7 | CCS CALIBRATION |
8 +-----+
9
10 CCS calibrants extracted drift times:
11 m/z      drift time (ms)
12 -----
13 161.0926    2.29
14 232.1300    2.87
15 303.1670    3.43
16 374.2040    4.07
17 445.2410    4.74
18 516.2780    5.53
19 587.3150    6.45
20 658.3520    7.29
21 729.3890    8.09
22 800.4260    9.00
23 871.4630    9.93
24 942.5010   10.87
25 1013.5370  11.78
26 1084.5740  12.78
27 1155.6110  13.75
28 1226.6480  14.70
29 1297.6850  15.63
30 1368.7220  16.64
31 1439.7590  17.64
32
33
34 Optimized calibration curve fit parameters:
35 corrected ccs = A * ((corrected drift time) + t0) ** B
36 A = 439.061160888
37 t0 = -0.0273495194704
38 B = 0.519001282396
39
40 Calibrant CCS, calculated vs. literature:
41 m/z      lit ccs (Ang^2)    calc ccs (Ang^2)    residual ccs (Ang^2, %)
42 -----
43 161.0926    136.050                136.821             -0.771   -0.567
44 232.1300    150.770                150.502             0.268   0.178
45 303.1670    163.280                163.175             0.105   0.064
46 374.2040    177.550                176.933             0.617   0.348
47 445.2410    190.800                190.513             0.287   0.150
48 516.2780    205.950                205.754             0.196   0.095
49 587.3150    222.680                222.274             0.406   0.182
50 658.3520    236.400                236.502             -0.102  -0.043
51 729.3890    249.130                249.211             -0.081  -0.032
52 800.4260    262.600                262.927             -0.327  -0.125
53 871.4630    275.980                276.404             -0.424  -0.154
54 942.5010    289.250                289.405             -0.155  -0.054
55 1013.5370   300.780                301.479             -0.699  -0.232
56 1084.5740   312.800                314.365             -1.565  -0.500
57 1155.6110   327.130                326.302             0.828   0.253
58 1226.6480   338.150                337.597             0.553   0.164
59 1297.6850   350.080                348.390             1.690   0.483
60 1368.7220   359.280                359.733             -0.453  -0.126
61 1439.7590   370.230                370.602             -0.372  -0.101
62
63
64 +-----+
65 | COMPOUND DATA |
66 +-----+
67
68 Compounds extracted drift times and calibrated CCS:
69 data file name          m/z      drift time (ms)    ccs (Ang^2)
70 -----
71 IM_0881L18_bin-0.05.txt 195.0877    2.403             138.410
72 IM_0881P06_bin-0.05.txt 304.0866    3.633             168.095
73 IM_0881G04_bin-0.05.txt 406.2337    4.629             188.816
74 IM_0881I21_bin-0.05.txt 497.1322    5.455             204.533
75 IM_0881H13_bin-0.05.txt 611.1607    7.577             241.676
76 IM_0881B07_bin-0.05.txt 734.4685    8.689             258.602
77
Normal text file      length: 3,149  lines: 77      Ln: 1  Col: 1  Sel: 0 | 0      Windows (CR LF)  UTF-8      INS
    
```



For each analysis, a .png image is generated containing a plot of the CCS calibration curve and fit residuals for all of the CCS calibrants. An Example plot is included above. A report file summarizing all of the extracted drift times, CCS calibration curve fit parameters, and calibrated CCS values for all compounds is also generated. An example report file is included to the left.

**CcsCALIBRATIONEXT API**

A simple API is included in *CcsCal* for users who would like to perform CCS calibration and obtain calibrated CCS for compounds using drift time data from external sources. The following example shows how to construct a calibration curve from an external list of masses, drift times, and reference CCS values contained in a file: `external_data.csv`.

```
from CcsCal.processing.CcsCalibration import CcsCalibrationExt
import numpy

# use numpy to load the data from a .csv file into a 2D array:
#      [[mass1, mass2, ...], [dt1, dt2, ...], [ccs1, ccs2, ...]]
ext_data = numpy.genfromtxt("external_data.csv", delimiter=",", unpack=True)

# initialize the CcsCalibrationExt object with the external data
# unpack the array into mass, dt, and ccs arrays with the * operator
cce = CcsCalibrationExt(*ext_data)

# save a figure with the calibration curve and residuals
cce.saveCalCurveFig(figure_file_name="calibration_curve_20170622.png")
```

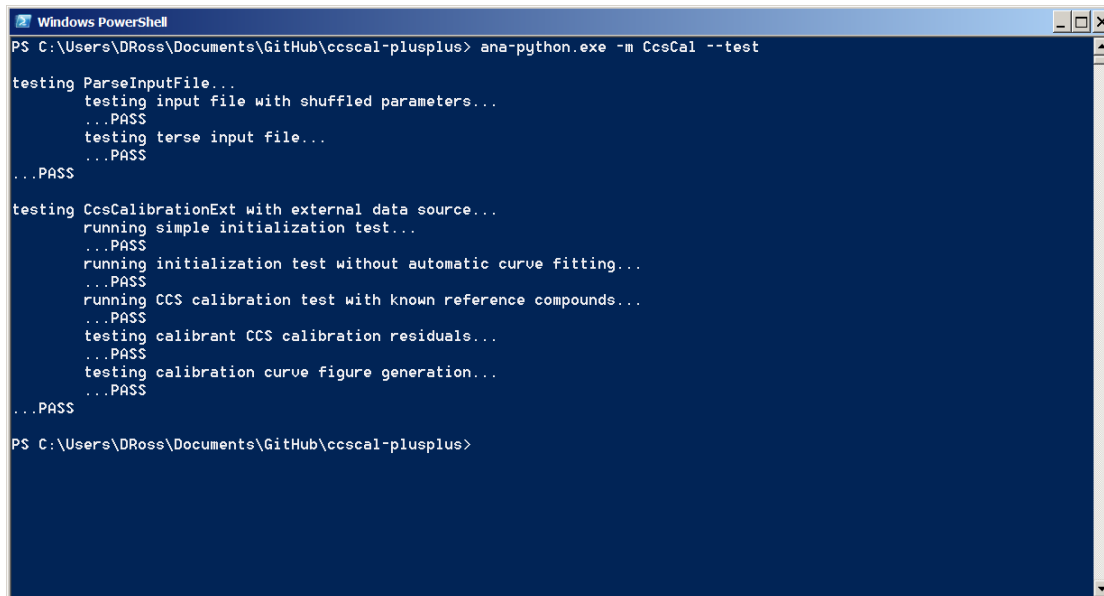
The calibration curve may then be applied to a list of masses and drift times to obtain calibrated CCS.

```
# simple list of masses and drift times
#      [[mass1, dt1], [mass2, dt2]]
masses_and_dts = [[406.2337, 4.68], [611.1607, 7.61]]

# loop through the list, get calibrated CCS for each mass/dt pair and print it
for pair in masses_and_dts:
    # unpack each pair into mass and dt with the * operator
    ccs = cce.getCalibratedCcs(*pair)
    # print with nice formatting
    print "m/z: {: 9.4f} dt: {: 4.2f} CCS: {: 6.2f}".format(pair[0], pair[1], ccs)
```

## TESTING

CcsCal comes packaged with a small test suite to ensure key portions of the program function properly. To run the tests, simply issue the `--test` flag when calling CcsCal. The tests will run and verbose output will be produced reporting on the success of the tests. The test suite is especially useful if making any changes to the program. A brief example of running the test suite is included below.



```

Windows PowerShell
PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus> ana-python.exe -m CcsCal --test

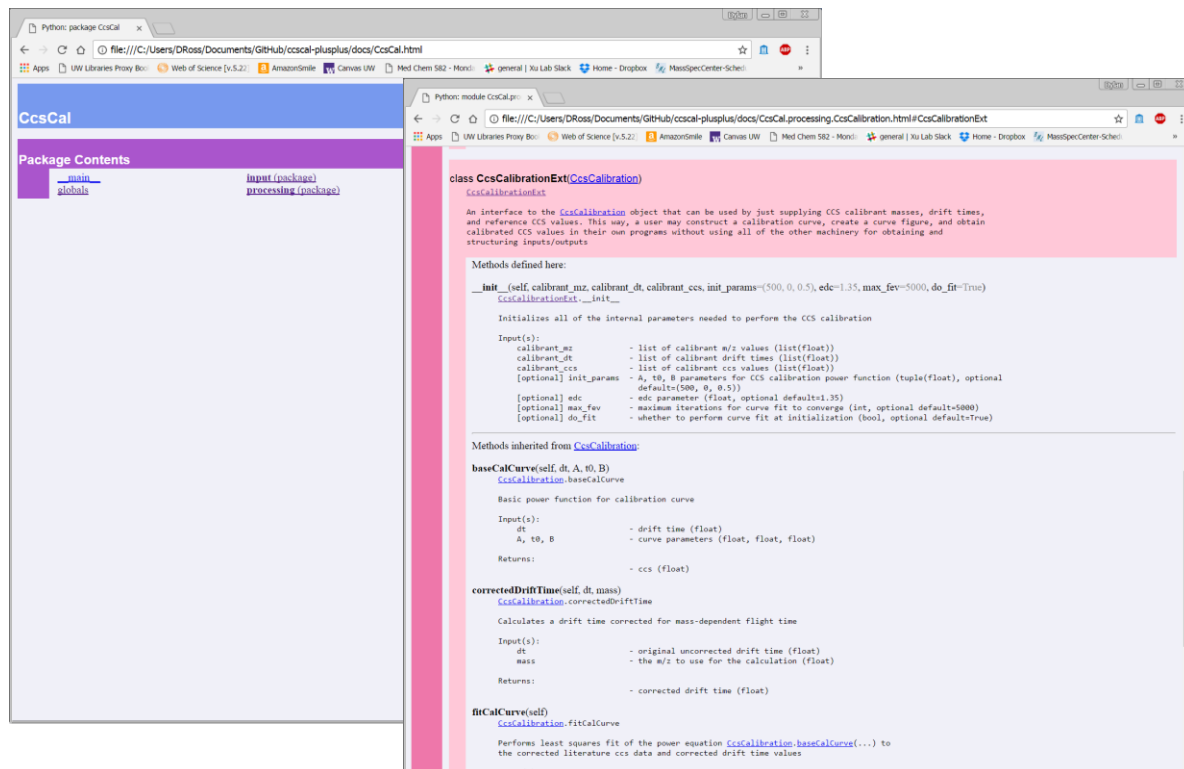
testing ParseInputFile...
  testing input file with shuffled parameters...
  ...PASS
  testing terse input file...
  ...PASS
...PASS

testing CcsCalibrationExt with external data source...
  running simple initialization test...
  ...PASS
  running initialization test without automatic curve fitting...
  ...PASS
  running CCS calibration test with known reference compounds...
  ...PASS
  testing calibrant CCS calibration residuals...
  ...PASS
  testing calibration curve figure generation...
  ...PASS
...PASS

PS C:\Users\DRoss\Documents\GitHub\ccscal-plusplus>
  
```

## DOCUMENTATION

CcsCal comes packaged with module-level documentation automatically produced by python's built in pydoc module. This documentation comes in the form of several linked HTML files, with CcsCal.html serving as the root index for the package. Shown below is CcsCal.html and a page containing an example of documentation for the CcsCalibrationExt submodule.



The screenshot displays two browser windows. The left window shows the root index page for the CcsCal package, titled 'CcsCal', with a 'Package Contents' section listing links to 'main', 'globals', 'input (package)', and 'processing (package)'. The right window shows the documentation for the 'CcsCalibrationExt' class, which inherits from 'CcsCalibration'. The documentation includes a description of the class as an interface to the 'CcsCalibration' object, a list of methods defined here (including an initialization method and several calculation methods), and a list of methods inherited from the parent class. The methods include 'baseCalCurve', 'correctedDriftTime', and 'fitCalCurve', each with detailed input and return parameters.

## **RUNNING ON LINUX/MACOS**

The majority of the code in *CcsCal* is cross-platform compatible by virtue of the python interpreter, however, the portion of the code that interfaces with the C++ text pre-processing extension (and the extension itself) can cause problems with running this program on a Linux or MacOS machine. The process for porting this program to be run on Linux or MacOS is essentially identical and involves recompiling the C++ extension, changing the executable path, and (possibly) modifying how the extension is called by *CcsCal*.

- Recompile the executable using a C++ compiler. Make sure the compiler uses the C++11 standard (with g++ the `--std=c++11` flag should be included in the compile command)
- Check in *CcsCal/globals.py* and make sure the `PP_EXE_PATH` variable is set to the absolute path of the newly compiled executable.
- *CcsCal* uses `subprocess.call` to call the text pre-processing extension executable. If the after the above steps are completed there are still errors trying to call the executable, try removing the `shell=True` parameter from the `call` function in the definition of `RawData.callPreProcessTxt` in *CcsCal/input/RawData.py*.