libCCE

Bulid requirements

A Python3 install with dev headers

A C compiler

```
sudo apt-get install python-dev build-essential clang
```

Build Process

Inplace build (local)

```
python3 build.py build_ext --inplace
```

Sytemwide build

This has the potental to require administrator privligaes:

```
python3 build.py install
```

If admin is required use `sudo`'s `-H` flag:

```
sudo -H python3 build.py install
```

Dev Reqirements

```
pip install Cython
```

In `build.py` uncomment the following lines so that any changes made to `.pyx` files will be added to the generated C code:

```
from Cython.Build import cythonize
ext_modules = cythonize(ext_modules)
```

Dev Sugestions

If you are doing dev work on this there are a few things to keep in mind:

First, If you are working on the C code, compiling with `-fsanitize=address` is **highly sugested** during development. It catches issues you don't and saves you lots of hair. Compiling with `-g -fno-omit-frame-pointer` is also recomended because when adsan feels like it it will give you line numbers. If adsan doesn't give you line numbers using IDA or Cutter can be helpful for figuring out where exactly you error is (just jump to the address and poke around).

Second, if you are working on the Cython code, building via `cython <inputfile>.pyx -a` every once in a while and seeing what you can do to minimize interaction with python objects (lines highlighted in yellow).

Also, keep in mind that anything you write in a `cdef` funtion is effectivey straight C with all that comes with it.

Third, adding the followng lines to they `.pyx` files will help with debugging by adding line tracing code:

```
# cython: linetrace=True
# distutils: define_macros=CYTHON_TRACE_NOGIL=1
```

API

The `libCCE` API consists of the `CCE` class and the `localMinCCE` function. All functions are fairly well commented in the source code. The high level docstrings and their asscated methods are placed here for ease of finding.

```
Class CCE:
    This class is used as a wraper for the cCCE implementation.
    As a general rule this implementation will try and fail via
    assertion error before returing incorrect information to
    the caller.

    def __cinit__(self, unsigned int branchingFactor, initalSequences =
None, int subSeqLen = 50):
        Called on creation of object, if initalSequences is supplied
        the sequences will be populated to the tree via faster
        version of insertSequence.
        Args:
            int branchingFactor: The branching factor of the tree,
                this is equlivant to the number of bins you have.
                This argument must be an int.
            Optional:
                initalSequences: A python iterable which contains
                IPDs for insertion.
                int subSeqLen: the length you would like the subsequences
devided into


    def __init__(self, branchingFactor, initalSequence, subSeqLen):
        All real work is done in __cinit__

    def __dealloc__(self):
        Used to cleanup memory on destruction of object. Automaticlly
        called should NOT be manually called.

    def insertSequence(self, pySeq):
        Wrapper for insertSequence, converts a supplied python
        iterable which supports indexing and len into an intiger
        buffer of the same length.
        Args:
            pySeq: A python iterable that supports indexing and len()
```

```
    def calculateCCE(self):
        Wrapper for calcCCEs.  Converts returned array to python list

        Returns:
            CCEs: A list of floats containing the CCE for each layer
                of the tree.  This list will only be as long as the
                number of unique sequences.
```

```
def localMinCCE(seq, int maxLen):
    Calculate the CCE of a sequence using the local minimum as an
    early stoping metric.
    Args:
        seq: A list or tuple which contains the sequence you want
            to calculate he CCE of.
        maxLen: The maximum lookback size.
    Returns:
        minCCE:
            The local minimum CCE for the sequence.
```