Co Documentation

Release 1.0.0

Lars Schöning

June 16, 2014
Co is a Python library for accessing and altering annotated DNA sequences. It keeps track of components and lifts over feature annotations when a component is “mutated” by applying a series of mutations. With `co` you can build new consensus sequences for cloned organisms and trace changes to features within a lineage.
1.1 Installation

Install co using pip:

```
$ pip install co
```

Co works with Python 2.7.x and Python 3.3+ (recommended).

1.2 Quickstart

1.2.1 Simple mutation

To illustrate what co is designed for, let’s begin with a hello world example:

```
>>> from co import Component
>>> from co.mutation import *

>>> hello = Component('Hello X!')
>>> hello.seq
Seq('Hello X!', Alphabet())

>>> hello_world = hello.mutate([Mutation(6, 1, 'world')])
>>> hello_world.seq
Seq('Hello world!', Alphabet())
```

1.2.2 Features & feature inheritance

Component.feature stores feature annotations in Feature format. These feature annotations are attached to the component they are defined in allowing, among other things, easy access to a feature sequence. Features are attached to a component like so:

```
>>> from Bio.SeqFeature import *

>>> slogan = Component('CoPy is for DNA components', features=[
...     SeqFeature(FeatureLocation(0, 4), type='name'),
...     SeqFeature(FeatureLocation(12, 15), id='DNA'))

>>> slogan.features.add(FeatureLocation(16, 26)).seq
Seq('components', Alphabet())
```
>>> [f.seq for f in slogan.features]
[Seq('CoPy', Alphabet()), Seq('DNA', Alphabet()), Seq('components', Alphabet())]

When a component is mutated, `co` automatically translates the feature annotations from the parent to the new coordinate system:

```python
>>> new_slogan = slogan.mutate([DEL(2, 2), DEL(12, 4)])
>>> new_slogan.seq
Seq('Co is for components', Alphabet())
>>> new_slogan.features
ComponentFeatureSet([Feature(FeatureLocation(ExactPosition(0), ExactPosition(2)), type='name'),
Feature(FeatureLocation(ExactPosition(10), ExactPosition(20)), id='DNA')])
```

```python
>>> [f.seq for f in new_slogan.features]
[Seq('Co', Alphabet()), Seq('components', Alphabet())]
```

When a region is affected by a mutation, any features contained in that region are deleted. Features that overlap the mutated region are trimmed. Features that are not affected by any mutation are left as they were—their starting coordinates are rewritten on the fly to map to the coordinate system of any inheriting component.

In the sample above, the `type` 'name' feature is resized by the mutation. The sequence of the `id` 'DNA' feature is deleted in its entirety and so the feature is deleted too. The feature spanning 'components' has not changed at all—but the mutations do affect its coordinates and so they are lifted over when the feature is accessed from within the mutated component.

```python
>>> new_slogan.features.removed
(Feature(FeatureLocation(ExactPosition(0), ExactPosition(9)), type='name'),
Feature(FeatureLocation(ExactPosition(17), ExactPosition(20)), id='DNA'))
>>> list(new_slogan.features.added)
[Feature(FeatureLocation(ExactPosition(0), ExactPosition(5)), type='name')]
```

Feature diffs

`Component.fdiff()` is designed for comparing the features contained in any two components:

```python
>>> diff = new_slogan.fdiff(slogan)
Diff(added=(Feature(FeatureLocation(ExactPosition(0), ExactPosition(9)), type='library'),
Feature(FeatureLocation(ExactPosition(13), ExactPosition(16)), id='DNA')),
removed=(Feature(FeatureLocation(ExactPosition(0), ExactPosition(5)), type='library')))
```

Note: `Component.fdiff()` is currently only implemented for components that directly inherit from one another. Internally, these values are looked up from `Component.features.added` and `Component.features.removed` as shown earlier. Eventually this will work with any two components regardless of ancestry.

Feature search

Features can be filtered using `FeatureSet.find()`. This search function supports filtering by region, type, id, strand as well as any qualifier. Multiple search parameters are interpreted as logical “AND”—i.e. all of them have to match.
>>> from co import *
>>> from Bio.SeqFeature import *

letters = Component('AABBDDEE', features=[
    SeqFeature(FeatureLocation(0, 1), type='vowel'),
    SeqFeature(FeatureLocation(2, 5), type='consonant'),
    SeqFeature(FeatureLocation(5, 6), type='vowel', qualifiers={'gene': 'abcD'})
])

list(letters.features.find(type='vowel'))
[Feature(FeatureLocation(ExactPosition(0), ExactPosition(1)), type='vowel'), Feature(FeatureLocation(ExactPosition(5), ExactPosition(6)), type='vowel')]

list(letters.features.find(between_start=3))
[Feature(FeatureLocation(ExactPosition(5), ExactPosition(6)), type='vowel'), Feature(FeatureLocation(ExactPosition(2), ExactPosition(5)), type='consonant')]

>>> from co.mutation import *

letters = letters.mutate([INS(4, 'CC')])

letters.seq
Seq('AABBCCDDEE', Alphabet())

list(letters.features.find(type='consonant'))
[Feature(FeatureLocation(ExactPosition(2), ExactPosition(7)), type='consonant')]

list(letters.features.find(type='vowel'))
[Feature(FeatureLocation(ExactPosition(0), ExactPosition(1)), type='vowel'), Feature(FeatureLocation(ExactPosition(7), ExactPosition(8)), type='vowel')]

list(letters.features.find(type='consonant', between_end=1))
[]

list(letters.features.find(gene='abcD'))
[Feature(FeatureLocation(ExactPosition(7), ExactPosition(8)), type='vowel')]

Optimization behind the scenes

Feature annotations that are inherited from another component are not copied over in memory — instead they are looked up on the fly. Only added and removed features are stored. A feature is considered changed when its sequence is affected in any way. When a feature is changed, the old feature is removed and the new feature is added.

• On-the-fly coordinate translation of unchanged features is done using translation.TranslationTable—inspired by the UCSC Chain Format.

• Feature locations are indexed using interval.IntervalTree, currently implemented as a BST.

1.2.3 Combining components

Multiple components can be combined using Component.combine(). This function will either create a “source” feature annotation for each of the components that are being merged, or copy over all features from all components if copy_features=True.

>>> a = Component('Co')
>>> b = Component('Py')
>>> b.features.add(FeatureLocation(0, 3), id='python')
>>> c = Component.combine(a, b, copy_features=True)
>>> c.seq
Seq('CoPy', Alphabet())
>>> c.features
ComponentFeatureSet([Feature(FeatureLocation(ExactPosition(2), ExactPosition(5)), id='python')])
1.2.4 Strain inheritance

In addition to DNA components, *co* can track changes in haploid microbial organisms. *HaploidOrganism* can track added, changed, or deleted DNA components—such as chromosomes or plasmids—and aggregate features contained in the strains.

**Strain components**

*HaploidOrganism*.diff() tracks how components have changed across strains:

```python
gene = Component('A')
alpha = HaploidOrganism('alpha')
alpha.set('genome', gene)

beta = HaploidOrganism('beta', parent=alpha)
beta.set('genome', gene.mutate([Mutation(0, 1, 'B')]))
beta.set('plasmid', Component('AGCT'))
beta.diff(alpha)
```

```
Diff(added=('plasmid',), removed=(), changed=('genome',))
```

```python
~beta.diff(alpha)
```

```
Diff(added=(), removed=('plasmid',), changed=('genome',))
```

**Strain features**

*HaploidOrganism*.features returns a *organism FeatureView* which is a searchable and iterable view of all features in all components of a strain.

1.3 Mutation types

Mutation types are *Mutation* as well as *INS, DEL, SUB* for substitutions, and *SNP* for SNPs.

```python
class co.mutation.**DEL**(pos, size=1)

class co.mutation.**INS**(pos, new_sequence, replace=False)
```

**Parameters**

- **pos** (*int*) – zero-based insertion index
- **new_sequence** (*str or Bio.Seq*) – insertion sequence
- **replace** (*bool*) – if True, eliminates the original character at the position. Some variant call formats keep the first character of the original sequence in the replacement sequence.

```python
class co.mutation.**Mutation**(position, size=None, new_sequence='', end=None)
```

A *Mutation(start, size, new_sequence)* is similar to the two derived mutations:

```python
DEL(start, size), INS(start, new_sequence)
```

**Parameters**

- **position** (*int*) – start index
- **size** (*int*) – length of deletion
• **new_sequence**(str, Component or Bio.Seq) – insertion sequence

**Note:** Mutation are stored as (position, size) pairs because (start, end) pairs do not allow for unambiguous zero-length mutations (i.e. insertions). It is possible to simulate an insertion by keeping one character of the original sequence, but that would introduce ambiguity to the exact site of the mutated sequence.

**new_sequence**
Replacement sequence inserted at position.

**size**
Length of the stretch of original sequence that is deleted at position.

**position**
Start index of the mutation, zero-based.

**end**
Computed end coordinate of the deletion. Use with caution.

**is_deletion()**
True if the size of the deletion is larger than the size of the insertion

**is_insertion()**
True if the size of the deletion is zero and new_sequence is not empty.

**is_substitution()**
True if the size of the deletion is equal to the size of the insertion.

**new_size**
The length of new_sequence

**start**
Identical to Mutation.position

**class** co.mutation.SNP (pos, new_nucleotide)
**class** co.mutation.SUB (pos, new_sequence)

### 1.4 Components & Features

The Component class is very similar to Bio.SeqRecord.SeqRecord, but does not currently sub-class it—mainly because the features property is implemented differently.

**class** co.Component (seq, parent=None, features=None, removed_features=None, feature_class=None, id=None, name=None, description=None, annotations=None, mutations=None)

**features**
FeatureSet containing the features present in this component.

**id**
A unique identifier for this component; preferably either str or UniqueIdentifier.

**Parameters**

• **features** – A list of additional features (features in addition to those inherited from parent)

• **removed_features** – A list of removed features (features present in parent or one of its parents, but not present in this component)
• **mutations** – A list of mutations that have been applied to `parent` to arrive at `seq`. The mutations will not be applied to `seq` again. Use `Component.mutate()` to mutate a component.

The `Feature` class inherits from `Bio.SeqFeature.SeqFeature` but stores some additional information. Proceed with caution when using the two types interchangeably.

```python
class co.Feature (component, *args, **kwargs)
    Feature derives from SeqFeature and binds to a particular Component. Feature does not support the sub_features argument. All other SeqFeature arguments are supported.
```

```python
class co.FeatureSet (feature_class=None)
    An ordered collection of SeqFeature objects.
    Parameters feature_class (type) – type of the features stored in the collection; defaults to SeqFeature and must inherit from it.
```

```python
class co.ComponentFeatureSet (component, removed_features=None, feature_class=None)
    An extended version of FeatureSet that binds to a Component and inherits from any FeatureSet in the parent of a component.
    When iterating over this feature set, any inherited features are also returned.
    removed_features
        Removed features are stored in removed_features if they are present in the parent, but not in component.
```

```python
class co.organism.FeatureView (components)
    Iterates over all features in a set of components – not necessarily in order – and provides search access to these features.
    find(**kwargs)
        Searches all components and yields features matching the constraints.
    See also:
        feature.FeatureSet.find()
```

```python
class co.organism.HaploidOrganism (id, parent=None)
    id
        ID of the organism
    parent
        Parent organism
    diff(other)
        Parameters other (HaploidOrganism) –
        Returns A Diff object with added, changed, and removed component names.
    fdiff(other)
        Returns A Diff object with added and removed features.
features
A read-only view of all the features present in all components of the organism.

Returns FeatureView

get_lineage (inclusive=True)
Iterate over all ancestors of this organism

Parameters inclusive (bool) – whether to include the organism itself in the lineage

Returns iterator over HaploidOrganism objects

remove (name)

Parameters name (str) – key of the component to remove

set (name, component)

Parameters
• name (str) – key for this component e.g. ‘genome’, ‘chr1’, or ‘pLASMID’
• component (Component)

1.6 Translation helpers

class co.translation.MutableTranslationTable (size)

Parameters size – the length of the source sequence

A mutable version of TranslationTable with insert, delete and substitute methods for updating the translation table with the corresponding mutations.

delete (position, size, strict=True)
Insert a gap in the target sequence

Parameters
• position (int) – start of gap site
• size (int) – length of gap
• strict (bool) – use strict mode

Raises OverlapError in various edge cases involving overlapping mutations, particularly in strict mode.

freeze ()
Return an immutable version of this translation table.

Return type TranslationTable

classmethod from_mutations (sequence, mutations, strict=True)

Parameters
• sequence – the source sequence
• mutations – iterable of Mutation objects
• strict (bool) – use strict mode

classmethod from_sequences (reference, query, algorithm=None)

Raises Not Implemented Error
**insert** *(position, size, strict=True)*
Insert a gap in the *source* sequence

**Parameters**
- **position** *(int)* – start of gap site
- **size** *(int)* – length of gap
- **strict** *(bool)* – use strict mode

**Raises** *OverlapError* in various edge cases involving overlapping mutations, particularly in *strict* mode.

**substitute** *(position, size, strict=True)*
Insert two gaps of equal length in the *source* and *target* sequences

**Parameters**
- **position** *(int)* – start of gap site
- **size** *(int)* – length of gap
- **strict** *(bool)* – use strict mode

**Raises** *OverlapError* in various edge cases involving overlapping mutations, particularly in *strict* mode.

**exception** *co.translation.OverlapError*

*OverlapError* is raised when a mutation is applied to a position in a sequence that has been altered by a previous mutation.

In *strict mode*, an *OverlapError* is fired more frequently, such as when a deletion is applied to a range that has previously been modified by an insertion.

**class** *co.translation.TranslationTable*(source_size, target_size, source_start, source_end, target_start, target_end, chain)*

This class is inspired by the UCSC chain format for pairwise alignments documented here:

http://genome.ucsc.edu/goldenPath/help/chain.html

*TranslationTable* encodes an alignment between two sequences, *source* and *target*.

The alignment is encoded in a chain of tuples in the format *(ungapped_size, ds, dt)*, where *ungapped_size* refers to regions that align, and the gaps *dt* and *ds* each refer to regions present only in the other sequence.

**source_start**
The first position in the source sequence that aligns with the target sequence

**source_end**
The last position in the source sequence that aligns with the target sequence.

**alignment()**
Returns an iterator yielding tuples in the form *(source, target)*.

**Returns** an iterator over all coordinates in both the source and target sequence.

**alignment_str()**
Returns a string representation of the alignment between *source* and *target* coordinates.

**Warning:** This function should only be used for debugging purposes.
ge (position)

See also:

The le () function.

- **Raises IndexError** if position does not exist in the source or if it maps to a coordinate after the end of the target sequence alignment.
- **Returns** the first position, equal or greater than position that exists in the query sequence.

invert ()

Creates a copy of the table where source and target are inverted.

- **Returns** a new TranslationTable object.

le (position)

le () attempts to return the coordinate in the target sequence that corresponds to the position parameter in the source sequence. If position falls into a gap in the target sequence, it will instead return the last coordinate in front of that gap.

- **Raises IndexError** if position does not exist in the source or if it maps to a coordinate before the start of the target sequence alignment.
- **Returns** the first position, equal or lower than position that exists in the query sequence.

total_ungapped_size

The total length of the alignment between source and target.
Road map

• Future releases may include a version tracking system to track and propagate updated mutations due to e.g. better re-sequencing. Versioned components will maintain the relationship to the component’s child component. Most likely, versions will be hashes of sequences and their mutations.

• An improved non-strict mode with better tolerance for overlapping mutations is planned.

• Cross-referencing between components through source feature annotations, with use for e.g. parts libraries and BioBricks.

• As this is a very early release of co, there is a long list of general improvements—they will be developed on demand.
Indices and tables

- genindex
- modindex
- search
Python Module Index

C
co, 3
co.mutation, 6
co.organism, 8
co.translation, 9