petlx Documentation

Release 0.17

Alistair Miles

August 27, 2014

1 Installation 3

2 Modules 5
- 2.1 Excel .xls (xlrd) 5
- 2.2 Excel .xlsx (openpyxl) 5
- 2.3 SQL (SQLAlchemy) 6
- 2.4 Arrays (numpy) 8
- 2.5 DataFrames (pandas) 9
- 2.6 HDF5 (pytables) 10
- 2.7 Text indexing (Whoosh) 12
- 2.8 iPython 16
- 2.9 Intervals (bx-python) 17
- 2.10 Tabix (pysam) 25
- 2.11 GFF3 25
- 2.12 Variant call format (PyVCF) 25

3 Indices and tables 29

Python Module Index 31

`petlx` is a collection of extensions to petl, a Python package for extracting, transforming and loading tables of data.

- Documentation: http://petlx.readthedocs.org/
- Source Code: https://github.com/alimanfoo/petlx
- Download: http://pypi.python.org/pypi/petlx
- Mailing List: http://groups.google.com/group/python-etl

For an overview of all functions in the package, see the *genindex*.

CHAPTER 1

Installation

This module is available from the Python Package Index. On Linux distributions you should be able to do `easy_install petlx` or `pip install petlx`. On Windows or Mac you can download manually, extract and run `python setup.py install`.

Note that each submodule within the `petlx` package has dependencies on one or more third party modules which will need to be installed separately.

CHAPTER 2

Modules

2.1 Excel .xls (xlrd)

Extension module providing functions for reading from and writing to Excel (.xls) files.

The packages xlrd, xlwt and xlutils are required. Try `pip install xlrd xlwt xlutils`.

`petlx.xls.`**`fromxls`**(*filename*, *sheet=None*, *use_view=True*)
    Extract a table from a sheet in an Excel (.xls) file.

    N.B., the sheet name is case sensitive.

    New in version 0.15.

`petlx.xls.`**`toxls`**(*tbl*, *filename*, *sheet*, *encoding='ascii'*, *style_compression=0*, *styles=None*)
    Write a table to a new Excel (.xls) file.

    New in version 0.15.

2.2 Excel .xlsx (openpyxl)

Extension module providing functions for reading from and writing to Excel (.xlsx) files.

The package openpyxl is required. Instructions for installation can be found at https://bitbucket.org/ericgazoni/openpyxl/wiki/Home or try `pip install openpyxl`.

`petlx.xlsx.`**`fromxlsx`**(*filename*, *sheet=None*, *range=None*, ***kwargs*)
    Extract a table from a sheet in an Excel (.xlsx) file.

    N.B., the sheet name is case sensitive, so watch out for, e.g., 'Sheet1'.

    Changed in version 0.15.

    The `sheet` argument can be omitted, in which case the first sheet in the workbook is used by default.

    The `range` argument can be used to provide a range string specifying a range of cells to extract.

    Any other keyword arguments are passed through to `openpyxl.load_workbook()`.

`petlx.xlsx.`**`toxlsx`**(*tbl*, *filename*, *sheet=None*, *encoding='utf-8'*)
    Write a table to a new Excel (.xlsx) file.

    New in version 0.15.

2.3 SQL (SQLAlchemy)

Extension module providing some convenience functions for working with SQL databases. SQLAlchemy is required, try `apt-get install python-sqlalchemy` or `pip install SQLAlchemy`.

Acknowledgments: much of the code of this module is based on the `csvsql` utility in the csvkit package.

`petlx.sql.`**`todb`**(*table*, *dbo*, *tablename*, *schema=None*, *commit=True*, *create=False*, *drop=False*, *constraints=True*, *metadata=None*, *dialect=None*, *sample=1000*)

Drop-in replacement for `petl.todb()` which also supports automatic table creation.

Parameters **table** : sequence of sequences (petl table)

Table data to load

dbo : database object

DB-API 2.0 connection, callable returning a DB-API 2.0 cursor, or SQLAlchemy connection, engine or session

tablename : string

Name of the table

schema : string

Name of the database schema to create the table in

commit : bool

If True commit the changes

create : bool

If True attempt to create the table before loading, inferring types from a sample of the data

drop : bool

If True attempt to drop the table before recreating (only relevant if create=True)

constraints : bool

If True use length and nullable constraints (only relevant if create=True)

metadata : sqlalchemy.MetaData

Custom table metadata (only relevant if create=True)

dialect : string

One of {'access', 'sybase', 'sqlite', 'informix', 'firebird', 'mysql', 'oracle', 'maxdb', 'postgresql', 'mssql'} (only relevant if create=True)

sample : int

Number of rows to sample when inferring types etc. Set to 0 to use the whole table.

`petlx.sql.`**`create_table`**(*table*, *dbo*, *tablename*, *schema=None*, *commit=True*, *constraints=True*, *metadata=None*, *dialect=None*, *sample=1000*)

Create a database table based on a sample of data in the given table.

Parameters **table** : sequence of sequences (petl table)

Table data to load

dbo : database object

DB-API 2.0 connection, callable returning a DB-API 2.0 cursor, or SQLAlchemy connection, engine or session

tablename : string

Name of the table

schema : string

Name of the database schema to create the table in

commit : bool

If True commit the changes

constraints : bool

If True use length and nullable constraints (only relevant if create=True)

metadata : sqlalchemy.MetaData

Custom table metadata (only relevant if create=True)

dialect : string

One of {'access', 'sybase', 'sqlite', 'informix', 'firebird', 'mysql', 'oracle', 'maxdb', 'postgresql', 'mssql'} (only relevant if create=True)

sample : int

Number of rows to sample when inferring types etc., set to 0 to use the whole table (only relevant if create=True)

`petlx.sql.`**`drop_table`**(*dbo*, *tablename*, *schema=None*, *commit=True*)

Drop a database table if it exists.

Parameters **dbo** : database object

DB-API 2.0 connection, callable returning a DB-API 2.0 cursor, or SQLAlchemy connection, engine or session

tablename : string

Name of the table

schema : string

Name of the database schema the table is in

commit : bool

If True commit the changes

`petlx.sql.`**`make_create_table_statement`**(*table*, *tablename*, *schema=None*, *constraints=True*, *metadata=None*, *dialect=None*)

Generate a CREATE TABLE statement based on a `petl` table.

Parameters **table** : sequence of sequences (petl table)

Table data to use to infer types etc.

tablename : string

Name of the table

schema : string

Name of the database schema to create the table in

constraints : bool

If True use length and nullable constraints

metadata : sqlalchemy.MetaData

Custom table metadata

dialect : string

One of {'access', 'sybase', 'sqlite', 'informix', 'firebird', 'mysql', 'oracle', 'maxdb', 'postgresql', 'mssql'}

`petlx.sql.`**`make_sqlalchemy_table`**(*table*, *tablename*, *schema=None*, *constraints=True*, *metadata=None*)

Create an SQLAlchemy table based on a `petl` table.

Parameters **table** : sequence of sequences (petl table)

Table data to use to infer types etc.

tablename : string

Name of the table

schema : string

Name of the database schema to create the table in

constraints : bool

If True use length and nullable constraints

metadata : sqlalchemy.MetaData

Custom table metadata

`petlx.sql.`**`make_sqlalchemy_column`**(*col*, *colname*, *constraints=True*)

Infer an appropriate SQLAlchemy column type based on a sequence of values.

Parameters **col** : sequence

A sequence of values to use to infer type, length etc.

colname : string

Name of column

constraints : bool

If True use length and nullable constraints

2.4 Arrays (numpy)

`petlx.array.`**`toarray`**(*table*, *dtype=None*, *count=-1*, *sample=1000*)

Convenience function to load data from the given *table* into a numpy structured array. E.g.:

```
>>> from petl import look
>>> from petlx.array import toarray
>>> look(table)
+-----------+-------+-------+
| 'foo'     | 'bar' | 'baz' |
+===========+=======+=======+
| 'apples'  | 1     | 2.5   |
+-----------+-------+-------+
| 'oranges' | 3     | 4.4   |
+-----------+-------+-------+
| 'pears'   | 7     | 0.1   |
+-----------+-------+-------+

>>> a = toarray(table)
>>> a
array([('apples', 1, 2.5), ('oranges', 3, 4.4), ('pears', 7, 0.1)],
      dtype=[('foo', '|S7'), ('bar', '<i8'), ('baz', '<f8')])
>>> a['foo']
array(['apples', 'oranges', 'pears'],
      dtype='|S7')
>>> a['bar']
array([1, 3, 7])
>>> a['baz']
array([ 2.5,  4.4,  0.1])
>>> a['foo'][0]
'apples'
>>> a['bar'][1]
3
>>> a['baz'][2]
0.10000000000000001
```

If no datatype is specified, *sample* rows will be examined to infer an appropriate datatype for each field.

The datatype can be specified as a string, e.g.:

```
>>> a = toarray(table, dtype='a4, i2, f4')
>>> a
array([('appl', 1, 2.5), ('oran', 3, 4.400000095367432),
       ('pear', 7, 0.10000000149011612)],
      dtype=[('foo', '|S4'), ('bar', '<i2'), ('baz', '<f4')])
```

The datatype can also be partially specified, in which case datatypes will be inferred for other fields, e.g.:

```
>>> a = toarray(table, dtype={'foo': 'a4'})
>>> a
array([('appl', 1, 2.5), ('oran', 3, 4.4), ('pear', 7, 0.1)],
      dtype=[('foo', '|S4'), ('bar', '<i8'), ('baz', '<f8')])
```

`petlx.array.`**`torecarray`**`(`**args*, ***kwargs*`)`

Convenient shorthand for `toarray(...).view(np.recarray)`.

New in version 0.5.1.

`petlx.array.`**`fromarray`**`(`*a*`)`

Extract a table from a numpy structured array.

New in version 0.4.

2.5 DataFrames (pandas)

The package pandas is required. Instructions for installation can be found at http://pandas.pydata.org/pandas-docs/dev/install.html or try apt-get install python-pandas.

`petlx.dataframe.`**`todataframe`**`(`*table*, *index=None*, *exclude=None*, *columns=None*, *coerce_float=False*, *nrows=None*`)`

Convenience function to load data from the given *table* into a pandas DataFrame.

New in version 0.14.

`petlx.dataframe.`**`fromdataframe`**`(`*df*, *include_index=False*`)`

Extract a table from a pandas DataFrame.

New in version 0.14.

2.6 HDF5 (pytables)

The package pytables is required. Instructions for installation can be found at http://pytables.github.com/usersguide/installation.html or try apt-get install python-tables.

`petlx.hdf5.`**`fromhdf5`**`(`*source*, *where=None*, *name=None*, *condition=None*, *condvars=None*, *start=None*, *stop=None*, *step=None*`)`

Provides access to an HDF5 table. E.g.:

```
>>> from petl import look
>>> from petlx.hdf5 import fromhdf5
>>> table1 = fromhdf5('test1.h5', '/testgroup', 'testtable')
>>> look(table1)
+-------+----------+
| 'foo' | 'bar'    |
+=======+==========+
| 1     | 'asdfgh' |
+-------+----------+
| 2     | 'qwerty' |
+-------+----------+
| 3     | 'zxcvbn' |
+-------+----------+
```

Some alternative signatures:

```
>>> # just specify path to table node
... table1 = fromhdf5('test1.h5', '/testgroup/testtable')
>>>
>>> # use an existing tables.File object
... import tables
>>> h5file = tables.openFile('test1.h5')
>>> table1 = fromhdf5(h5file, '/testgroup/testtable')
>>>
>>> # use an existing tables.Table object
... h5tbl = h5file.getNode('/testgroup/testtable')
>>> table1 = fromhdf5(h5tbl)
>>>
>>> # use a condition to filter data
... table2 = fromhdf5(h5tbl, condition="(foo < 3)")
>>> look(table2)
+-------+----------+
| 'foo' | 'bar'    |
+=======+==========+
| 1     | 'asdfgh' |
+-------+----------+
| 2     | 'qwerty' |
+-------+----------+
```

New in version 0.3.

`petlx.hdf5.`**`fromhdf5sorted`**`(`*source*, *where=None*, *name=None*, *sortby=None*, *checkCSI=False*, *start=None*, *stop=None*, *step=None*`)`

Provides access to an HDF5 table, sorted by an indexed column, e.g.:

```
>>> # set up a new hdf5 table to demonstrate with
... import tables
>>> h5file = tables.openFile("test1.h5", mode="w", title="Test file")
>>> h5file.createGroup('/', 'testgroup', 'Test Group')
/testgroup (Group) 'Test Group'
  children := []
>>> class FooBar(tables.IsDescription):
...     foo = tables.Int32Col(pos=0)
...     bar = tables.StringCol(6, pos=2)
...
>>> h5table = h5file.createTable('/testgroup', 'testtable', FooBar, 'Test Table')
>>>
>>> # load some data into the table
... table1 = (('foo', 'bar'),
...           (3, 'asdfgh'),
...           (2, 'qwerty'),
...           (1, 'zxcvbn'))
>>>
>>> for row in table1[1:]:
...     for i, f in enumerate(table1[0]):
...         h5table.row[f] = row[i]
...     h5table.row.append()
...
>>> h5table.cols.foo.createCSIndex() # CS index is required
0
>>> h5file.flush()
>>> h5file.close()
>>>
>>> # access the data, sorted by the indexed column
... from petl import look
>>> from petlx.hdf5 import fromhdf5sorted
>>> table2 = fromhdf5sorted('test1.h5', '/testgroup', 'testtable', sortby='foo')
>>> look(table2)
+-------+----------+
| 'foo' | 'bar'    |
+=======+==========+
| 1     | 'zxcvbn' |
+-------+----------+
| 2     | 'qwerty' |
+-------+----------+
| 3     | 'asdfgh' |
+-------+----------+
```

New in version 0.3.

petlx.hdf5.**tohdf5**(*table*, *source*, *where=None*, *name=None*, *create=False*, *description=None*, *title=''*, *filters=None*, *expectedrows=10000*, *chunkshape=None*, *byteorder=None*, *createparents=False*, *sample=1000*)

Write to an HDF5 table. If *create* is *False*, assumes the table already exists, and attempts to truncate it before loading. If *create* is *True*, any existing table is dropped, and a new table is created; if *description* is None, the datatype will be guessed. E.g.:

```
>>> from petl import look
>>> look(table1)
+-------+----------+
| 'foo' | 'bar'    |
+=======+==========+
| 1     | 'asdfgh' |
+-------+----------+
| 2      | 'qwerty'  |
+--------+----------+
| 3      | 'zxcvbn'  |
+--------+----------+

>>> from petlx.hdf5 import tohdf5, fromhdf5
>>> tohdf5(table1, 'test1.h5', '/testgroup', 'testtable', create=True, createparents=True)
>>> look(fromhdf5('test1.h5', '/testgroup', 'testtable'))
+-------+----------+
| 'foo' | 'bar'    |
+=======+==========+
| 1     | 'asdfgh' |
+-------+----------+
| 2     | 'qwerty' |
+-------+----------+
| 3     | 'zxcvbn' |
+-------+----------+
```

See also `appendhdf5()`.

New in version 0.3.

`petlx.hdf5.`**`appendhdf5`**(*table*, *source*, *where=None*, *name=None*)

Like `tohdf5()` but don't truncate the table before loading.

New in version 0.3.

2.7 Text indexing (Whoosh)

The package Whoosh is required. To install try `pip install whoosh`.

`petlx.index.`**`fromindex`**(*index_or_dirname*, *indexname=None*, *docnum_field=None*)

Extract all documents from a Whoosh index. E.g.:

```
>>> # set up an index and load some documents via the Whoosh API
... from whoosh.index import create_in
>>> from whoosh.fields import *
>>> schema = Schema(title=TEXT(stored=True), path=ID(stored=True), content=TEXT)
>>> index = create_in('tmp/example', schema)
>>> writer = index.writer()
>>> writer.add_document(title=u"First document", path=u"/a",
...                     content=u"This is the first document we've added!")
>>> writer.add_document(title=u"Second document", path=u"/b",
...                     content=u"The second one is even more interesting!")
>>> writer.commit()
>>> # extract documents as a table
... from petl import look
>>> from petlx.index import fromindex
>>> tbl = fromindex('tmp/example')
>>> look(tbl)
+--------+--------------------+
| 'path' | 'title'            |
+========+====================+
| u'/a'  | u'First document'  |
+--------+--------------------+
| u'/b'  | u'Second document' |
+--------+--------------------+
```

New in version 0.16.

Parameters **index_or_dirname**

Either an instance of *whoosh.index.Index* or a string containing the directory path where the index is stored.

indexname

String containing the name of the index, if multiple indexes are stored in the same directory.

docnum_field

If not None, an extra field will be added to the output table containing the internal document number stored in the index. The name of the field will be the value of this argument.

Returns A table-like object (row container).

`petlx.index.searchindex`(*index_or_dirname*, *query*, *limit=10*, *indexname=None*, *docnum_field=None*, *score_field=None*, *fieldboosts=None*, *search_kwargs={}*)

Search an index using a query. E.g.:

```
>>> # set up an index and load some documents via the Whoosh API
... from whoosh.index import create_in
>>> from whoosh.fields import *
>>> schema = Schema(title=TEXT(stored=True), path=ID(stored=True), content=TEXT)
>>> index = create_in('tmp/example', schema)
>>> writer = index.writer()
>>> writer.add_document(title=u"Oranges", path=u"/a",
...                     content=u"This is the first document we've added!")
>>> writer.add_document(title=u"Apples", path=u"/b",
...                     content=u"The second document is even more "
...                             u"interesting!")
>>> writer.commit()
>>> # demonstrate the use of searchindex()
... from petl import look
>>> from petlx.index import searchindex
>>> look(searchindex('tmp/example', 'oranges'))
+--------+------------+
| 'path' | 'title'    |
+========+============+
| u'/a'  | u'Oranges' |
+--------+------------+

>>> look(searchindex('tmp/example', 'doc*'))
+--------+------------+
| 'path' | 'title'    |
+========+============+
| u'/a'  | u'Oranges' |
+--------+------------+
| u'/b'  | u'Apples'  |
+--------+------------+
```

New in version 0.16.

Parameters **index_or_dirname**

Either an instance of *whoosh.index.Index* or a string containing the directory path where the index is to be stored.

query

Either a string or an instance of *whoosh.query.Query*. If a string, it will be parsed as a multi-field query, i.e., any terms not bound to a specific field will match **any** field.

limit

Return at most *limit* results.

indexname

String containing the name of the index, if multiple indexes are stored in the same directory.

docnum_field

If not None, an extra field will be added to the output table containing the internal document number stored in the index. The name of the field will be the value of this argument.

score_field

If not None, an extra field will be added to the output table containing the score of the result. The name of the field will be the value of this argument.

fieldboosts

An optional dictionary mapping field names to boosts.

search_kwargs

Any extra keyword arguments to be passed through to the Whoosh *search()* method.

Returns A table-like object (row container).

`petlx.index.`**`searchindexpage`**(*index_or_dirname*, *query*, *pagenum*, *pagelen=10*, *indexname=None*, *docnum_field=None*, *score_field=None*, *fieldboosts=None*, *search_kwargs={}*)

Search an index using a query, returning a result page.

New in version 0.16.

Parameters **index_or_dirname**

Either an instance of *whoosh.index.Index* or a string containing the directory path where the index is to be stored.

query

Either a string or an instance of *whoosh.query.Query*. If a string, it will be parsed as a multi-field query, i.e., any terms not bound to a specific field will match **any** field.

pagenum

Number of the page to return (e.g., 1 = first page).

pagelen

Number of results per page.

indexname

String containing the name of the index, if multiple indexes are stored in the same directory.

docnum_field

If not None, an extra field will be added to the output table containing the internal document number stored in the index. The name of the field will be the value of this argument.

score_field

If not None, an extra field will be added to the output table containing the score of the result. The name of the field will be the value of this argument.

fieldboosts

An optional dictionary mapping field names to boosts.

search_kwargs

Any extra keyword arguments to be passed through to the Whoosh *search()* method.

Returns A table-like object (row container).

petlx.index.**toindex**(*tbl*, *index_or_dirname*, *schema=None*, *indexname=None*, *merge=False*, *optimize=False*)

Load all rows from *tbl* into a Whoosh index. N.B., this will clear any existing data in the index before loading. E.g.:

```
>>> from petl import look
>>> from petlx.index import toindex, fromindex
>>> # here is the table we want to load into an index
... look(tbl)
+--------+------+------+-------+--------------------------------------------------+
| 'f0'   | 'f1' | 'f2' | 'f3'  | 'f4'                                             |
+========+======+======+=======+==================================================+
| u'AAA' |   12 |  4.3 | True  | datetime.datetime(2014, 6, 30, 14, 7, 2, 333199) |
+--------+------+------+-------+--------------------------------------------------+
| u'BBB' |    6 |  3.4 | False | datetime.datetime(1900, 1, 31, 0, 0)             |
+--------+------+------+-------+--------------------------------------------------+
| u'CCC' |   42 |  7.8 | True  | datetime.datetime(2100, 12, 25, 0, 0)            |
+--------+------+------+-------+--------------------------------------------------+

>>> # define a schema for the index
... from whoosh.fields import *
>>> schema = Schema(f0=TEXT(stored=True),
...                 f1=NUMERIC(int, stored=True),
...                 f2=NUMERIC(float, stored=True),
...                 f3=BOOLEAN(stored=True),
...                 f4=DATETIME(stored=True))
>>> # load data
... toindex(tbl, 'tmp/example', schema=schema)
>>> # look what it did
... look(fromindex('tmp/example'))
+--------+------+------+-------+--------------------------------------------------+
| 'f0'   | 'f1' | 'f2' | 'f3'  | 'f4'                                             |
+========+======+======+=======+==================================================+
| u'AAA' |   12 |  4.3 | True  | datetime.datetime(2014, 6, 30, 14, 7, 2, 333199) |
+--------+------+------+-------+--------------------------------------------------+
| u'BBB' |    6 |  3.4 | False | datetime.datetime(1900, 1, 31, 0, 0)             |
+--------+------+------+-------+--------------------------------------------------+
| u'CCC' |   42 |  7.8 | True  | datetime.datetime(2100, 12, 25, 0, 0)            |
+--------+------+------+-------+--------------------------------------------------+
```

New in version 0.16.

Parameters tbl

A table-like object (row container) containing the data to be loaded.

index_or_dirname

Either an instance of *whoosh.index.Index* or a string containing the directory path where the index is to be stored.

indexname

String containing the name of the index, if multiple indexes are stored in the same directory.

merge

Merge small segments during commit?

optimize

Merge all segments together?

`petlx.index.`**`appendindex`**(*tbl*, *index_or_dirname*, *indexname=None*, *merge=True*, *optimize=False*)

Load all rows from *tbl* into a Whoosh index, adding them to any existing data in the index.

New in version 0.16.

Parameters **tbl**

A table-like object (row container) containing the data to be loaded.

index_or_dirname

Either an instance of *whoosh.index.Index* or a string containing the directory path where the index is to be stored.

indexname

String containing the name of the index, if multiple indexes are stored in the same directory.

merge

Merge small segments during commit?

optimize

Merge all segments together?

2.8 iPython

`petlx.ipython.`**`display`**(*tbl*, *limit=None*, ***kwargs*)

Display a table inline within an iPython notebook. E.g.:

```
In [0]: from petlx.ipython import display
        tbl = [['foo', 'bar'], ['a', 1], ['b', 2]]
        display(tbl)
```

Alternatively, using the fluent style:

```
In [0]: import petl.interactive as etl
        import petlx.ipython
        tbl = etl.wrap([['foo', 'bar'], ['a', 1], ['b', 2]])
        tbl.display()
```

New in version 0.5.

`petlx.ipython.`**`displayall`**`(`*tbl*, ***kwargs*`)`

Display *all rows* from a table inline within an iPython notebook. E.g.:

```
In [0]: from petlx.ipython import displayall
        tbl = [['foo', 'bar'], ['a', 1], ['b', 2]]
        displayall(tbl)
```

Alternatively, using the fluent style:

```
In [0]: import petl.interactive as etl
        import petlx.ipython
        tbl = etl.wrap([['foo', 'bar'], ['a', 1], ['b', 2]])
        tbl.displayall()
```

New in version 0.5.

2.9 Intervals (bx-python)

The package bx.intervals is required. Instructions for installation can be found at https://bitbucket.org/james_taylor/bx-python/wiki/Home or try `pip install bx-python`.

`petlx.interval.`**`intervaljoin`**`(`*left*, *right*, *lstart='start'*, *lstop='stop'*, *rstart='start'*, *rstop='stop'*, *lfacet=None*, *rfacet=None*, *proximity=0*, *lprefix=None*, *rprefix=None*`)`

Join two tables by overlapping intervals. E.g.:

```
>>> from petl import look
>>> from petlx.interval import intervaljoin
>>> look(left)
+---------+-------+--------+
| 'begin' | 'end' | 'quux' |
+=========+=======+========+
| 1       | 2     | 'a'    |
+---------+-------+--------+
| 2       | 4     | 'b'    |
+---------+-------+--------+
| 2       | 5     | 'c'    |
+---------+-------+--------+
| 9       | 14    | 'd'    |
+---------+-------+--------+
| 9       | 140   | 'e'    |
+---------+-------+--------+
| 1       | 1     | 'f'    |
+---------+-------+--------+
| 2       | 2     | 'g'    |
+---------+-------+--------+
| 4       | 4     | 'h'    |
+---------+-------+--------+
| 5       | 5     | 'i'    |
+---------+-------+--------+
| 1       | 8     | 'j'    |
+---------+-------+--------+

>>> look(right)
+---------+--------+---------+
| 'start' | 'stop' | 'value' |
+=========+========+=========+
| 1         | 4        | 'foo'    |
+-----------+----------+----------+
| 3         | 7        | 'bar'    |
+-----------+----------+----------+
| 4         | 9        | 'baz'    |
+-----------+----------+----------+
```

```
>>> result = intervaljoin(left, right, lstart='begin', lstop='end', rstart='start', rstop='stop'
>>> look(result)
+---------+-------+--------+---------+--------+---------+
| 'begin' | 'end' | 'quux' | 'start' | 'stop' | 'value' |
+=========+=======+========+=========+========+=========+
| 1       | 2     | 'a'    | 1       | 4      | 'foo'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 4     | 'b'    | 1       | 4      | 'foo'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 4     | 'b'    | 3       | 7      | 'bar'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 5     | 'c'    | 1       | 4      | 'foo'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 5     | 'c'    | 3       | 7      | 'bar'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 5     | 'c'    | 4       | 9      | 'baz'   |
+---------+-------+--------+---------+--------+---------+
| 2       | 2     | 'g'    | 1       | 4      | 'foo'   |
+---------+-------+--------+---------+--------+---------+
| 4       | 4     | 'h'    | 3       | 7      | 'bar'   |
+---------+-------+--------+---------+--------+---------+
| 5       | 5     | 'i'    | 3       | 7      | 'bar'   |
+---------+-------+--------+---------+--------+---------+
| 5       | 5     | 'i'    | 4       | 9      | 'baz'   |
+---------+-------+--------+---------+--------+---------+
```

An additional key comparison can be added, e.g.:

```
>>> from petl import look
>>> look(left)
+----------+---------+-------+
| 'fruit'  | 'begin' | 'end' |
+==========+=========+=======+
| 'apple'  | 1       | 2     |
+----------+---------+-------+
| 'apple'  | 2       | 4     |
+----------+---------+-------+
| 'apple'  | 2       | 5     |
+----------+---------+-------+
| 'orange' | 2       | 5     |
+----------+---------+-------+
| 'orange' | 9       | 14    |
+----------+---------+-------+
| 'orange' | 19      | 140   |
+----------+---------+-------+
| 'apple'  | 1       | 1     |
+----------+---------+-------+
| 'apple'  | 2       | 2     |
+----------+---------+-------+
| 'apple'  | 4       | 4     |
+----------+---------+-------+
| 'apple'  | 5       | 5     |
+----------+---------+-------+

>>> look(right)
+----------+---------+--------+---------+
| 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+========+=========+
| 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+--------+---------+
| 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+--------+---------+
| 'orange' | 4       | 9      | 'baz'   |
+----------+---------+--------+---------+

>>> result = intervaljoin(left, right, lstart='begin', lstop='end', rstart='start', rstop='stop'
>>> look(result)
+----------+---------+-------+----------+---------+--------+---------+
| 'fruit'  | 'begin' | 'end' | 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+=======+==========+=========+========+=========+
| 'apple'  | 1       | 2     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 4     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 4     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 5     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 5     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 2       | 5     | 'orange' | 4       | 9      | 'baz'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 2     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 4       | 4     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 5       | 5     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 5       | 5     | 'orange' | 4       | 9      | 'baz'   |
+----------+---------+-------+----------+---------+--------+---------+
```

New in version 0.2.

`petlx.interval.`**`intervalleftjoin`**(*left*, *right*, *lstart='start'*, *lstop='stop'*, *rstart='start'*, *rstop='stop'*, *lfacet=None*, *rfacet=None*, *proximity=0*, *missing=None*, *lprefix=None*, *rprefix=None*)

Like `intervaljoin()` but rows from the left table without a match in the right table are also included. E.g.:

```
>>> from petl import look
>>> from petlx.interval import intervalleftjoin
>>> look(left)
+----------+---------+-------+
| 'fruit'  | 'begin' | 'end' |
+==========+=========+=======+
| 'apple'  | 1       | 2     |
+----------+---------+-------+
| 'apple'  | 2       | 4     |
+----------+---------+-------+
| 'apple'  | 2       | 5     |
+----------+---------+-------+
| 'orange' | 2       | 5     |
+----------+---------+-------+
| 'orange' | 9       | 14    |
+----------+---------+-------+
| 'orange' | 19      | 140   |
+----------+---------+-------+
| 'apple'  | 1       | 1     |
+----------+---------+-------+
| 'apple'  | 2       | 2     |
+----------+---------+-------+
| 'apple'  | 4       | 4     |
+----------+---------+-------+
| 'apple'  | 5       | 5     |
+----------+---------+-------+

>>> look(right)
+----------+---------+--------+---------+
| 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+========+=========+
| 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+--------+---------+
| 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+--------+---------+
| 'orange' | 4       | 9      | 'baz'   |
+----------+---------+--------+---------+

>>> result = intervalleftjoin(left, right, lstart='begin', lstop='end', rstart='start', rstop='s
>>> look(result)
+----------+---------+-------+----------+---------+--------+---------+
| 'fruit'  | 'begin' | 'end' | 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+=======+==========+=========+========+=========+
| 'apple'  | 1       | 2     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 4     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 4     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 5     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 5     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'apple'  | 2       | 5     | 'orange' | 4       | 9      | 'baz'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 2       | 5     | 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 2       | 5     | 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 2       | 5     | 'orange' | 4       | 9      | 'baz'   |
+----------+---------+-------+----------+---------+--------+---------+
| 'orange' | 9       | 14    | None     | None    | None   | None    |
+----------+---------+-------+----------+---------+--------+---------+
```

New in version 0.2.

`petlx.interval.`**`intervalantijoin`**(*left*, *right*, *lstart='start'*, *lstop='stop'*, *rstart='start'*, *rstop='stop'*, *lfacet=None*, *rfacet=None*, *proximity=0*, *missing=None*)

Return rows from the *left* table with no overlapping rows from the *right* table.

New in version 0.16.

`petlx.interval.`**`intervaljoinvalues`**(*left*, *right*, *value*, *lstart='start'*, *lstop='stop'*, *rstart='start'*, *rstop='stop'*, *lfacet=None*, *rfacet=None*, *proximity=0*)

Convenience function to join the left table with values from a specific field in the right hand table.

New in version 0.5.3.

`petlx.interval.`**`intervalsubtract`**(*left*, *right*, *lstart='start'*, *lstop='stop'*, *rstart='start'*, *rstop='stop'*, *lfacet=None*, *rfacet=None*, *proximity=0*)

Subtract intervals in the right hand table from intervals in the left hand table.

New in version 0.5.4.

`petlx.interval.`**`intervallookup`**(*table*, *start='start'*, *stop='stop'*, *value=None*, *proximity=0*)

Construct an interval lookup for the given table. E.g.:

```
>>> from petlx.interval import intervallookup
>>> table = [['start', 'stop', 'value'],
...          [1, 4, 'foo'],
...          [3, 7, 'bar'],
...          [4, 9, 'baz']]
>>> lkp = intervallookup(table, 'start', 'stop')
>>> lkp.find(1, 2)
[(1, 4, 'foo')]
>>> lkp.find(2, 4)
[(1, 4, 'foo'), (3, 7, 'bar')]
>>> lkp.find(2, 5)
[(1, 4, 'foo'), (3, 7, 'bar'), (4, 9, 'baz')]
>>> lkp.find(9, 14)
[]
>>> lkp.find(19, 140)
[]
>>> lkp.find(1)
[]
>>> lkp.find(2)
[(1, 4, 'foo')]
>>> lkp.find(4)
[(3, 7, 'bar')]
>>> lkp.find(5)
[(3, 7, 'bar'), (4, 9, 'baz')]
```

Note that there must be a non-zero overlap between the query and the interval for the interval to be retrieved, hence *lkp.find(1)* returns nothing. Use the *proximity* keyword argument to find intervals within a given distance of the query.

Some examples using the *proximity* and *value* keyword arguments:

```
>>> table = [['start', 'stop', 'value'],
...          [1, 4, 'foo'],
...          [3, 7, 'bar'],
...          [4, 9, 'baz']]
>>> lkp = intervallookup(table, 'start', 'stop', value='value', proximity=1)
>>> lkp.find(1, 2)
['foo']
>>> lkp.find(2, 4)
['foo', 'bar', 'baz']
>>> lkp.find(2, 5)
['foo', 'bar', 'baz']
>>> lkp.find(9, 14)
['baz']
>>> lkp.find(19, 140)
[]
```

```
>>> lkp.find(1)
['foo']
>>> lkp.find(2)
['foo']
>>> lkp.find(4)
['foo', 'bar', 'baz']
>>> lkp.find(5)
['bar', 'baz']
>>> lkp.find(9)
['baz']
```

New in version 0.2.

`petlx.interval.`**`intervallookupone`**(*table*, *start='start'*, *stop='stop'*, *value=None*, *proximity=0*, *strict=True*)

Construct an interval lookup for the given table, returning at most one result for each query. If `strict=True` is given, queries returning more than one result will raise a *DuplicateKeyError*. If `strict=False` is given, and there is more than one result, the first result is returned.

See also `intervallookup()`.

New in version 0.2.

`petlx.interval.`**`intervalrecordlookup`**(*table*, *start='start'*, *stop='stop'*, *proximity=0*)

As `intervallookup()` but return records.

New in version 0.2.

`petlx.interval.`**`intervalrecordlookupone`**(*table*, *start='start'*, *stop='stop'*, *proximity=0*, *strict=True*)

As `intervallookupone()` but return records.

New in version 0.2.

`petlx.interval.`**`facetintervallookup`**(*table*, *facet*, *start='start'*, *stop='stop'*, *value=None*, *proximity=0*)

Construct a faceted interval lookup for the given table. E.g.:

```
>>> from petl import look
>>> from petlx.interval import facetintervallookup
>>> look(table)
+----------+---------+--------+---------+
| 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+========+=========+
| 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+--------+---------+
| 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+--------+---------+
| 'orange' | 4       | 9      | 'baz'   |
+----------+---------+--------+---------+

>>> lkp = facetintervallookup(table, facet='type', start='start', stop='stop')
>>> lkp['apple'].find(1, 2)
[('apple', 1, 4, 'foo')]
>>> lkp['apple'].find(2, 4)
[('apple', 1, 4, 'foo'), ('apple', 3, 7, 'bar')]
>>> lkp['apple'].find(2, 5)
[('apple', 1, 4, 'foo'), ('apple', 3, 7, 'bar')]
>>> lkp['orange'].find(2, 5)
[('orange', 4, 9, 'baz')]
>>> lkp['orange'].find(9, 14)
```

```
[]
>>> lkp['orange'].find(19, 140)
[]
>>> lkp['apple'].find(1)
[]
>>> lkp['apple'].find(2)
[('apple', 1, 4, 'foo')]
>>> lkp['apple'].find(4)
[('apple', 3, 7, 'bar')]
>>> lkp['apple'].find(5)
[('apple', 3, 7, 'bar')]
>>> lkp['orange'].find(5)
[('orange', 4, 9, 'baz')]
```

New in version 0.2.

petlx.interval.**facetintervallookupone**(*table*, *facet*, *start='start'*, *stop='stop'*, *value=None*, *proximity=0*, *strict=True*)

Construct a faceted interval lookup for the given table, returning at most one result for each query, e.g.:

```
>>> from petl import look
>>> from petlx.interval import facetintervallookupone
>>> look(table)
+----------+---------+--------+---------+
| 'type'   | 'start' | 'stop' | 'value' |
+==========+=========+========+=========+
| 'apple'  | 1       | 4      | 'foo'   |
+----------+---------+--------+---------+
| 'apple'  | 3       | 7      | 'bar'   |
+----------+---------+--------+---------+
| 'orange' | 4       | 9      | 'baz'   |
+----------+---------+--------+---------+

>>> lkp = facetintervallookupone(table, key='type', start='start', stop='stop', value='value')
>>> lkp['apple'].find(1, 2)
'foo'
>>> lkp['apple'].find(2, 4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "petlx/interval.py", line 191, in __getitem__
    raise DuplicateKeyError
petl.util.DuplicateKeyError
>>> lkp['apple'].find(2, 5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "petlx/interval.py", line 191, in __getitem__
    raise DuplicateKeyError
petl.util.DuplicateKeyError
>>> lkp['apple'].find(4, 5)
'bar'
>>> lkp['orange'].find(4, 5)
'baz'
>>> lkp['apple'].find(5, 7)
'bar'
>>> lkp['orange'].find(5, 7)
'baz'
>>> lkp['apple'].find(8, 9)
>>> lkp['orange'].find(8, 9)
'baz'
```

```
>>> lkp['orange'].find(9, 14)
>>> lkp['orange'].find(19, 140)
>>> lkp['apple'].find(1)
>>> lkp['apple'].find(2)
'foo'
>>> lkp['apple'].find(4)
'bar'
>>> lkp['apple'].find(5)
'bar'
>>> lkp['orange'].find(5)
'baz'
>>> lkp['apple'].find(8)
>>> lkp['orange'].find(8)
'baz'
```

If `strict=True` is given, queries returning more than one result will raise a *DuplicateKeyError*. If `strict=False` is given, and there is more than one result, the first result is returned.

See also `facetintervallookup()`.

New in version 0.2.

`petlx.interval.`**`facetintervalrecordlookup`**(*table*, *facet*, *start='start'*, *stop='stop'*, *proximity=0*)

As `facetintervallookup()` but return records.

New in version 0.2.

`petlx.interval.`**`facetintervalrecordlookupone`**(*table*, *facet*, *start*, *stop*, *proximity=0*, *strict=True*)

As `facetintervallookupone()` but return records.

New in version 0.2.

`petlx.interval.`**`collapsedintervals`**(*tbl*, *start='start'*, *stop='stop'*, *facet=None*)

Utility function to collapse intervals in a table.

If no facet key is given, returns an iterator over *(start, stop)* tuples.

If facet key is given, returns an iterator over *(key, start, stop)* tuples.

New in version 0.5.5.

`petlx.interval.`**`tupletree`**(*table*, *start='start'*, *stop='stop'*, *value=None*)

Construct an interval tree for the given table, where each node in the tree is a row of the table.

`petlx.interval.`**`tupletrees`**(*table*, *facet*, *start='start'*, *stop='stop'*, *value=None*)

Construct faceted interval trees for the given table, where each node in the tree is a row of the table.

`petlx.interval.`**`recordtree`**(*table*, *start='start'*, *stop='stop'*)

Construct an interval tree for the given table, where each node in the tree is a row of the table represented as a hybrid tuple/dictionary-style record object.

`petlx.interval.`**`recordtrees`**(*table*, *facet*, *start='start'*, *stop='stop'*)

Construct faceted interval trees for the given table, where each node in the tree is a row of the table represented as a hybrid tuple/dictionary-style record object.

2.10 Tabix (pysam)

`petlx.tabix.fromtabix`(*filename*, *reference=None*, *start=None*, *end=None*, *region=None*, *header=None*)

Extract rows from a tabix indexed file. E.g.:

```
>>> from petlx.tabix import fromtabix
>>> from petl import look
>>> t = fromtabix('test.bed.gz', region='Pf3D7_02_v3:100000-200000')
>>> look(t)
+---------------+----------+----------+-----------------------------+
| '#chrom'      | 'start'  | 'end'    | 'region'                    |
+===============+==========+==========+=============================+
| 'Pf3D7_02_v3' | '23100'  | '105800' | 'SubtelomericHypervariable' |
+---------------+----------+----------+-----------------------------+
| 'Pf3D7_02_v3' | '105800' | '447300' | 'Core'                      |
+---------------+----------+----------+-----------------------------+
```

New in version 0.4.

2.11 GFF3

`petlx.gff3.fromgff3`(*filename*, *region=None*)

Extract feature rows from a GFF3 file.

New in version 0.2.

Changed in version 0.15.

A region query string of the form '[seqid]' or '[seqid]:[start]-[end]' may be given for the `region` argument. If given, requires the GFF3 file to be bgzipped and tabix indexed.

`petlx.gff3.gff3lookup`(*features*, *facet='seqid'*)

Build a GFF3 feature lookup based on interval trees. See also `petlx.interval.facetintervallookup()`.

New in version 0.2.

`petlx.gff3.gff3join`(*table*, *features*, *seqid='seqid'*, *start='start'*, *end='end'*, *proximity=1*)

Join with a table of GFF3 features. See also `petlx.interval.intervaljoin()`.

New in version 0.2.

`petlx.gff3.gff3leftjoin`(*table*, *features*, *seqid='seqid'*, *start='start'*, *end='end'*, *proximity=1*)

Left join with a table of GFF3 features. See also `petlx.interval.intervalleftjoin()`.

New in version 0.2.

2.12 Variant call format (PyVCF)

`petlx.vcf.fromvcf`(*filename*, *chrom=None*, *start=None*, *end=None*, *samples=True*)

Returns a table providing access to data from a variant call file (VCF). E.g.:

```
>>> from petl import look
>>> from petlx.vcf import fromvcf
>>> t = fromvcf('example.vcf')
>>> look(t)
```

```
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| 'CHROM'  | 'POS'   | 'ID'        | 'REF' | 'ALT'     | 'QUAL' | 'FILTER' | 'INFO'
+==========+=========+=============+=======+===========+========+==========+=====================
| '19'     |     111 | None        | 'A'   | [C]       |    9.6 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '19'     |     112 | None        | 'A'   | [G]       |     10 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     |   14370 | 'rs6054257' | 'G'   | [A]       |     29 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     |   17330 | None        | 'T'   | [A]       |      3 | ['q10']  | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1110696 | 'rs6040355' | 'A'   | [G, T]    |     67 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1230237 | None        | 'T'   | [None]    |     47 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1234567 | 'microsat1' | 'G'   | [GA, GAC] |     50 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1235237 | None        | 'T'   | [None]    | None   | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| 'X'      |      10 | 'rsTest'    | 'AC'  | [A, ATG]  |     10 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
```

New in version 0.5.

petlx.vcf.**unpackinfo**(*tbl*, **keys*, ***kwargs*)

Unpack the INFO field into separate fields. E.g.:

```
>>> from petlx.vcf import fromvcf, unpackinfo
>>> from petl import look
>>> t1 = fromvcf('../fixture/sample.vcf', samples=False)
>>> look(t1)
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| 'CHROM'  | 'POS'   | 'ID'        | 'REF' | 'ALT'     | 'QUAL' | 'FILTER' | 'INFO'
+==========+=========+=============+=======+===========+========+==========+=====================
| '19'     |     111 | None        | 'A'   | [C]       |    9.6 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '19'     |     112 | None        | 'A'   | [G]       |     10 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     |   14370 | 'rs6054257' | 'G'   | [A]       |     29 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     |   17330 | None        | 'T'   | [A]       |      3 | ['q10']  | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1110696 | 'rs6040355' | 'A'   | [G, T]    |     67 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1230237 | None        | 'T'   | [None]    |     47 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1234567 | 'microsat1' | 'G'   | [GA, GAC] |     50 | []       | OrderedDict([('NS',
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| '20'     | 1235237 | None        | 'T'   | [None]    | None   | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------
| 'X'      |      10 | 'rsTest'    | 'AC'  | [A, ATG]  |     10 | []       | {}
+----------+---------+-------------+-------+-----------+--------+----------+---------------------

>>> t2 = unpackinfo(t1)
>>> look(t2)
+----------+---------+-------------+-------+-----------+--------+----------+------+------+-------
| 'CHROM'  | 'POS'   | 'ID'        | 'REF' | 'ALT'     | 'QUAL' | 'FILTER' | 'NS' | 'AN' | 'AC'
+==========+=========+=============+=======+===========+========+==========+======+======+=======
| '19'     |     111 | None        | 'A'   | [C]       |    9.6 | []       | None | None | None
```

```
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '19'    |     112 | None        | 'A'   | [G]       |     10 | []       | None | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    |   14370 | 'rs6054257' | 'G'   | [A]       |     29 | []       |    3 | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    |   17330 | None        | 'T'   | [A]       |      3 | ['q10']  |    3 | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    | 1110696 | 'rs6040355' | 'A'   | [G, T]    |     67 | []       |    2 | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    | 1230237 | None        | 'T'   | [None]    |     47 | []       |    3 | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    | 1234567 | 'microsat1' | 'G'   | [GA, GAC] |     50 | []       |    3 |    6 | [3, 1]
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| '20'    | 1235237 | None        | 'T'   | [None]    | None   | []       | None | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
| 'X'     |      10 | 'rsTest'    | 'AC'  | [A, ATG]  |     10 | []       | None | None | None
+---------+---------+-------------+-------+-----------+--------+----------+------+------+------
```

New in version 0.5.

petlx.vcf.**meltsamples**(*tbl*, **samples*)

Melt the samples columns. E.g.:

```
>>> from petlx.vcf import fromvcf, unpackinfo, meltsamples
>>> from petl import look, cutout
>>> t1 = fromvcf('../fixture/sample.vcf')
>>> t2 = meltsamples(t1)
>>> t3 = cutout(t2, 'INFO')
>>> look(t3)
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| 'CHROM' | 'POS' | 'ID'        | 'REF' | 'ALT' | 'QUAL' | 'FILTER' | 'SAMPLE'  | 'CALL'
+=========+=======+=============+=======+=======+========+==========+===========+==============
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00001' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00002' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00003' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00001' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00002' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00003' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00001' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00002' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00003' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
| '20'    | 17330 | None        | 'T'   | [A]   |      3 | ['q10']  | 'NA00001' | Call(sample=NA
+---------+-------+-------------+-------+-------+--------+----------+-----------+--------------
```

New in version 0.5.

petlx.vcf.**unpackcall**(*tbl*, **keys*, ***kwargs*)

Unpack the call column. E.g.:

```
>>> from petlx.vcf import fromvcf, unpackinfo, meltsamples, unpackcall
>>> from petl import look, cutout
>>> t1 = fromvcf('../fixture/sample.vcf')
>>> t2 = meltsamples(t1)
>>> t3 = unpackcall(t2)
>>> t4 = cutout(t3, 'INFO')
>>> look(t4)
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| 'CHROM' | 'POS' | 'ID'        | 'REF' | 'ALT' | 'QUAL' | 'FILTER' | 'SAMPLE'  | 'GT'  | 'GQ' |
+=========+=======+=============+=======+=======+========+==========+===========+=======+======+
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00001' | '0|0' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00002' | '0|0' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '19'    |   111 | None        | 'A'   | [C]   |    9.6 | []       | 'NA00003' | '0/1' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00001' | '0|0' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00002' | '0|0' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '19'    |   112 | None        | 'A'   | [G]   |     10 | []       | 'NA00003' | '0/1' | None |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00001' | '0|0' |   48 |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00002' | '1|0' |   48 |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '20'    | 14370 | 'rs6054257' | 'G'   | [A]   |     29 | []       | 'NA00003' | '1/1' |   43 |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
| '20'    | 17330 | None        | 'T'   | [A]   |      3 | ['q10']  | 'NA00001' | '0|0' |   49 |
+---------+-------+-------------+-------+-------+--------+----------+-----------+-------+------+
```

New in version 0.5.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

p

petlx, 1
petlx.array, 8
petlx.dataframe, 9
petlx.gff3, 25
petlx.hdf5, 10
petlx.index, 12
petlx.interval, 17
petlx.ipython, 16
petlx.sql, 5
petlx.tabix, 24
petlx.vcf, 25
petlx.xls, 5
petlx.xlsx, 5