pandas-msgpack Documentation

Release 0.1.0

PyData Development Team

Mar 30, 2017

Contents

The `pandas_msgpack` module provides an interface from *pandas https://pandas.pydata.org* to the msgpack library. This is a lightweight portable binary format, similar to binary JSON, that is highly space efficient, and provides good performance both on the writing (serialization), and reading (deserialization).

Contents:

CHAPTER 1

Installation

You can install pandas-msgpack with `conda`, `pip`, or by installing from source.

Conda

```
# not enabled YET
$ conda install pandas-msgpack --channel conda-forge
```

This installs pandas-msgpack and all common dependencies, including `pandas`.

Pip

To install the latest version of pandas-msgpack: from the

```
$ pip install pandas-msgpack -U
```

This installs pandas-msgpack and all common dependencies, including `pandas`.

Install from Source

```
$ pip install git+https://github.com/pydata/pandas-msgpack.git
```

Dependencies

The *blosc <https://pypi.python.org/pypi/blosc>* library can be optionally installed as a compressor.

CHAPTER 2

Tutorial

```
In [1]: import pandas as pd

In [2]: from pandas_msgpack import to_msgpack, read_msgpack
```

```
In [3]: df = pd.DataFrame(np.random.rand(5,2), columns=list('AB'))

In [4]: to_msgpack('foo.msg', df)

In [5]: read_msgpack('foo.msg')
Out[5]:
          A         B
0  0.655055  0.695007
1  0.813772  0.803631
2  0.799387  0.985437
3  0.266732  0.459968
4  0.975844  0.200425

In [6]: s = pd.Series(np.random.rand(5),index=pd.date_range('20130101',periods=5))
```

You can pass a list of objects and you will receive them back on deserialization.

```
In [7]: to_msgpack('foo.msg', df, 'foo', np.array([1,2,3]), s)

In [8]: read_msgpack('foo.msg')
Out[8]:
[          A         B
 0  0.655055  0.695007
 1  0.813772  0.803631
 2  0.799387  0.985437
 3  0.266732  0.459968
 4  0.975844  0.200425, 'foo', array([1, 2, 3]), 2013-01-01    0.780439
 2013-01-02    0.475372
 2013-01-03    0.599117
 2013-01-04    0.762336
```

```
 2013-01-05    0.093987
 Freq: D, dtype: float64]
```

You can pass `iterator=True` to iterate over the unpacked results

```
In [9]: for o in read_msgpack('foo.msg',iterator=True):
   ...:     print(o)
   ...:
          A         B
0  0.655055  0.695007
1  0.813772  0.803631
2  0.799387  0.985437
3  0.266732  0.459968
4  0.975844  0.200425
foo
[1 2 3]
2013-01-01    0.780439
2013-01-02    0.475372
2013-01-03    0.599117
2013-01-04    0.762336
2013-01-05    0.093987
Freq: D, dtype: float64
```

You can pass `append=True` to the writer to append to an existing pack

```
In [10]: to_msgpack('foo.msg', df, append=True)

In [11]: read_msgpack('foo.msg')
Out[11]:
[          A         B
 0  0.655055  0.695007
 1  0.813772  0.803631
 2  0.799387  0.985437
 3  0.266732  0.459968
 4  0.975844  0.200425, 'foo', array([1, 2, 3]), 2013-01-01    0.780439
 2013-01-02    0.475372
 2013-01-03    0.599117
 2013-01-04    0.762336
 2013-01-05    0.093987
 Freq: D, dtype: float64,           A         B
 0  0.655055  0.695007
 1  0.813772  0.803631
 2  0.799387  0.985437
 3  0.266732  0.459968
 4  0.975844  0.200425]
```

Furthermore you can pass in arbitrary python objects.

```
In [12]: to_msgpack('foo2.msg', { 'dict' : [ { 'df' : df }, { 'string' : 'foo' }, {
↪'scalar' : 1. }, { 's' : s } ] })

In [13]: read_msgpack('foo2.msg')
Out[13]:
{'dict': ({'df':           A         B
   0  0.655055  0.695007
   1  0.813772  0.803631
   2  0.799387  0.985437
   3  0.266732  0.459968
```

```
 4  0.975844  0.200425},
{'string': 'foo'},
{'scalar': 1.0},
{'s': 2013-01-01    0.780439
 2013-01-02    0.475372
 2013-01-03    0.599117
 2013-01-04    0.762336
 2013-01-05    0.093987
 Freq: D, dtype: float64})}
```

CHAPTER 3

Compression

Optionally, a `compression` argument will compress the resulting bytes. These can take a bit more time to write. The available compressors are `zlib` and blosc.

Generally compression will increase the writing time.

```
In [1]: import pandas as pd

In [2]: from pandas_msgpack import to_msgpack, read_msgpack

In [3]: df = pd.DataFrame({'A': np.arange(100000),
   ...:                    'B': np.random.randn(100000),
   ...:                    'C': 'foo'})
   ...:
```

```
In [4]: %timeit -n 1 -r 1 to_msgpack('uncompressed.msg', df)
1 loop, best of 1: 23.4 ms per loop
```

```
In [5]: %timeit -n 1 -r 1 to_msgpack('compressed_blosc.msg', df, compress='blosc')
1 loop, best of 1: 28.5 ms per loop
```

```
In [6]: %timeit -n 1 -r 1 to_msgpack('compressed_zlib.msg', df, compress='zlib')
1 loop, best of 1: 160 ms per loop
```

If compressed, it will be be automatically inferred and de-compressed upon reading.

```
In [7]: %timeit -n 1 -r 1 read_msgpack('uncompressed.msg')
1 loop, best of 1: 25.4 ms per loop
```

```
In [8]: %timeit -n 1 -r 1 read_msgpack('compressed_blosc.msg')
1 loop, best of 1: 21.8 ms per loop
```

```
In [9]: %timeit -n 1 -r 1 read_msgpack('compressed_zlib.msg')
1 loop, best of 1: 29.4 ms per loop
```

These can provide storage space savings.

```
In [10]: !ls -ltr *.msg
-rw-r--r-- 1 docs docs 2000582 Mar 30 20:12 uncompressed.msg
-rw-r--r-- 1 docs docs 1188115 Mar 30 20:12 compressed_blosc.msg
-rw-r--r-- 1 docs docs 1320656 Mar 30 20:12 compressed_zlib.msg
```

CHAPTER 4

Read/Write API

Msgpacks can also be read from and written to strings.

```
In [1]: import pandas as pd

In [2]: from pandas_msgpack import to_msgpack, read_msgpack

In [3]: df = pd.DataFrame({'A': np.arange(10),
   ...:                    'B': np.random.randn(10),
   ...:                    'C': 'foo'})
   ...:

In [4]: to_msgpack(None, df)
Out[4]: b
↪'\x84\xa6blocks\x93\x86\xa4locs\x86\xa3typ\xa7ndarray\xa4data\xd7\x00\x01\x00\x00\x00\x00\x00\x00\x
↪1\xac\xe7\xf8\xbfB\\\x9e\xc1\x89\xf1\xf5\xbf\xa0(\x97\x96\xfb\xe0\xf3?\x84^
↪\x1c\xc6f5\xeb\xbf\xbe-\xa3\xcc}\xf0\xe0?^\xfc\x8f#\xd0:\xb5?
↪\x87Y\xbcp1\x81\xd5\xbfM\xf4}
↪\xea\xb5\xc3\xe5\xbf\xa5dtype\xa7float64\x86\xa4locs\x86\xa3typ\xa7ndarray\xa4data\xd7\x00\x00\x00\
↪manager\xa4axes\x92\x86\xa5klass\xa5Index\xa3typ\xa5index\xa4data\x93\xa1A\xa1B\xa1C\xa8compress\xc
↪index\xa4stop\n\xa4step\x01\xa4name\xc0'
```

Furthermore you can concatenate the strings to produce a list of the original objects.

```
In [5]: read_msgpack(to_msgpack(None, df) + to_msgpack(None, df.A))
Out[5]:
[   A         B    C
 0  0 -0.618816  foo
 1  1 -0.483378  foo
 2  2 -1.556561  foo
 3  3 -1.371469  foo
 4  4  1.242427  foo
 5  5 -0.850269  foo
 6  6  0.529357  foo
 7  7  0.082929  foo
 8  8 -0.336010  foo
```

```
9  9 -0.680140  foo, 0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
Name: A, dtype: int64]
```

CHAPTER 5

API Reference

read_msgpack(path_or_buf[, encoding, iterator])	Load msgpack pandas object from the specified
to_msgpack(path_or_buf, *args, **kwargs)	msgpack (serialize) object to input file path

pandas_msgpack.**read_msgpack**(*path_or_buf*, *encoding='utf-8'*, *iterator=False*, ***kwargs*)

Load msgpack pandas object from the specified file path

THIS IS AN EXPERIMENTAL LIBRARY and the storage format may not be stable until a future release.

Parameters **path_or_buf** : string File path, BytesIO like or string

encoding: Encoding for decoding msgpack str type

iterator : boolean, if True, return an iterator to the unpacker

(default is False)

Returns **obj** : type of object stored in file

pandas_msgpack.**to_msgpack**(*path_or_buf*, **args*, ***kwargs*)

msgpack (serialize) object to input file path

Parameters **path_or_buf** : string File path, buffer-like, or None

if None, return generated string

args : an object or objects to serialize

encoding: encoding for unicode objects

append : boolean whether to append to an existing msgpack

(default is False)

compress : type of compressor (zlib or blosc), default to None (no

compression)

CHAPTER 6

Changelog

0.1.3 / 2017-03-30

Initial release of transfered code from pandas

Includes patches since the 0.19.2 release on pandas with the following:

- Bug in `read_msgpack()` in which `Series` categoricals were being improperly processed, see pandas-GH#14901
- Bug in `read_msgpack()` which did not allow loading of a dataframe with an index of type `CategoricalIndex`, see pandas-GH#15487
- Bug in `read_msgpack()` when deserializing a `CategoricalIndex`, see pandas-GH#15487

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index

R

T