datatable Documentation

Release 0.7.0

Pasha Stetsenko

Feb 04, 2019

Installation

H2O's `datatable` is a Python package for manipulating 2-dimensional tabular data structures (aka, data frames). It is close in spirit to pandas or SFrame; however we put specific emphasis on speed and big data support. As the name suggests, the package is closely related to R's data.table and attempts to mimic its core algorithms and API.

Currently `datatable` is in the Alpha stage and is undergoing active development. The API may be unstable; some of the core features are incomplete and/or missing.

Contributing

`datatable` is an open source project released under the Mozilla Public Licence v2. Open Source projects live by their user and developer communities. We welcome and encourage your contributions of any kind!

No matter what your skill set or level of engagement is with `datatable`, you can help others by improving the ecosystem of documentation, bug report and feature request tickets, and code.

We invite anyone who is interested to contribute, whether through pull requests, or tests, or GitHub issues, API suggestions, or generic discussion.

Have Questions?

If you have questions about using `datatable`, post them on Stack Overflow using the `[datatable] [python]` tags at http://stackoverflow.com/questions/tagged/datatable+python.

CHAPTER 1

Installation

This section describes how to install H2O's `datatable`.

1.1 Requirements

- Python 3.5+

1.2 Install on Mac OS X

Run the following command to install `datatable` on Mac OS X.

```
pip install datatable
```

1.3 Install on Linux

Run one of the following commands to retrieve the `datatable` whl file for Linux environments.

```
# Python 3.5
pip install https://s3.amazonaws.com/h2o-release/datatable/stable/datatable-0.3.2/
↪datatable-0.3.2-cp35-cp35m-linux_x86_64.whl

# Python 3.6
pip install https://s3.amazonaws.com/h2o-release/datatable/stable/datatable-0.3.2/
↪datatable-0.3.2-cp36-cp36m-linux_x86_64.whl
```

1.4 Build from Source

The key component needed for building the `datatable` package from source is the Clang/Llvm distribution. The same distribution is also required for building the `llvmlite` package, which is a prerequisite for `datatable`. Note that the `clang` compiler which is shipped with MacOS is too old, and in particular it doesn't have support for the OpenMP technology.

1.4.1 Installing the `Clang/Llvm` distribution

1. Visit https://releases.llvm.org/download.html and **download** the most recent version of Clang/Llvm available for your platform (but no older than version 4.0.0).
2. Extract the downloaded archive into any suitable location on your hard drive.
3. Create one of the environment variables `LLVM4` / `LLVM5` / `LLVM6` (depending on the version of Clang/Llvm that you installed). The variable should point to the directory where you placed the Clang/Llvm distribution.

   For example, on Ubuntu after downloading `clang+llvm-4.0.0-x86_64-linux-gnu-ubuntu-16.10.tar.xz` the sequence of steps might look like:

```
$ mv clang+llvm-4.0.0-x86_64-linux-gnu-ubuntu-16.10.tar.xz  /opt
$ cd /opt
$ sudo tar xvf clang+llvm-4.0.0-x86_64-linux-gnu-ubuntu-16.10.tar.xz
$ export LLVM4=/opt/clang+llvm-4.0.0-x86_64-linux-gnu-ubuntu-16.10
```

You probably also want to put the last `export` line into your `~/.bash_profile`.

1.4.2 Building `datatable`

1. Verify that you have Python 3.5 or above:

```
$ python --V
```

   If you don't have Python 3.5 or later, you may want to download and install the newest version of Python, and then create and activate a virtual environment for that Python. For example:

```
$ virtualenv --python=python3.6 ~/py36
$ source ~/py36/bin/activate
```

2. Build `datatable`:

```
$ make build
$ make install
$ make test
```

3. Additional commands you may find occasionally interesting:

```
# Uninstall previously installed datatable
make uninstall

# Build a debug version of datatable (for example suitable for ``gdb``␣
↪debugging)
make debug
```

(continues on next page)

(continued from previous page)

```
# Generate code coverage report
make coverage
```

1.5 Troubleshooting

- If you get an error like `ImportError: This package should not be accessible on Python 3`, then you may have a `PYTHONPATH` environment variable that causes conflicts. See this SO question for details.
- If you see errors such as `"implicit declaration of function 'PyUnicode_AsUTF8' is invalid in C99"` or `"unknown type name 'PyModuleDef'"` or `"void function 'PyInit__datatable' should not return a value "`, it means your current Python is Python 2. Please revisit step 1 in the build instructions above.
- If you are seeing an error `'Python.h' file not found`, then it means you have an incomplete version of Python installed. This is known to sometimes happen on Ubuntu systems. The solution is to run `apt-get install python-dev` or `apt-get install python3.6-dev`.
- If you run into installation errors with `llvmlite` dependency, then your best bet is to attempt to install it manually before trying to build `datatable`:

  ```
  $ pip install llvmlite
  ```

  Consult the `llvmlite` Installation Guide for additional information.
- On OS X, if you are getting an error `fatal error: 'sys/mman.h' file not found` or similar, this can be fixed by installing the Xcode Command Line Tools:

  ```
  $ xcode-select --install
  ```

CHAPTER 2

Using `datatable`

This section describes common functionality and commands that you can run in `datatable`.

2.1 Create Frame

You can create a Frame from a variety of sources, including `numpy` arrays, `pandas` DataFrames, raw Python objects, etc:

```
import datatable as dt
import numpy as np
np.random.seed(1)
dt.Frame(np.random.randn(1000000))
```

```
import pandas as pd
dt.Frame(pd.DataFrame({"A": range(1000)}))
```

```
dt.Frame({"n": [1, 3], "s": ["foo", "bar"]})
```

2.2 Convert a Frame

Convert an existing Frame into a `numpy` array, a `pandas` DataFrame, or a pure Python object:

```
nparr = df1.tonumpy()
pddfr = df1.topandas()
pyobj = df1.topython()
```

2.3 Parse Text (csv) Files

`datatable` provides fast and convenient parsing of text (csv) files:

```
df = dt.fread("train.csv")
```

The `datatable` parser

- Automatically detects separators, headers, column types, quoting rules, etc.
- Reads from file, URL, shell, raw text, archives, glob
- Provides multi-threaded file reading for maximum speed
- Includes a progress indicator when reading large files
- Reads both RFC4180-compliant and non-compliant files

2.4 Write the Frame

Write the Frame's content into a `csv` file (also multi-threaded):

```
df.to_csv("out.csv")
```

2.5 Save a Frame

Save a Frame into a binary format on disk, then open it later instantly, regardless of the data size:

```
df.save("out.nff")
df2 = dt.open("out.nff")
```

2.6 Basic Frame Properties

Basic Frame properties include:

```
print(df.shape)    # (nrows, ncols)
print(df.names)    # column names
print(df.stypes)   # column types
```

2.7 Compute Per-Column Summary Stats

Compute per-column summary stats using:

```
df.sum()
df.max()
df.min()
df.mean()
df.sd()
df.mode()
```

(continues on next page)

(continued from previous page)

```
df.nmodal()
df.nunique()
```

2.8 Select Subsets of Rows/Columns

Select subsets of rows and/or columns using:

```
df["A"]             # select 1 column
df[:10, :]          # first 10 rows
df[::-1, "A":"D"]   # reverse rows order, columns from A to D
df[27, 3]           # single element in row 27, column 3 (0-based)
```

2.9 Delete Rows/Columns

Delete rows and or columns using:

```
del df["D"]         # delete column D
del df[f.A < 0, :] # delete rows where column A has negative values
```

2.10 Filter Rows

Filter rows via an expression using the following. In this example, `mean`, `sd`, `f` are all symbols imported from `datatable`.

```
df[(f.x > mean(f.y) + 2.5 * sd(f.y)) | (f.x < -mean(f.y) - sd(f.y)), :]
```

2.11 Compute Columnar Expressions

Compute columnar expressions using:

```
df[:, {"x": f.x, "y": f.y, "x+y": f.x + f.y, "x-y": f.x - f.y}]
```

2.12 Sort Columns

Sort columns using:

```
df.sort("A")
```

2.13 Perform Groupby Calculations

Perform groupby calculations using:

```
df(select=mean(f.x), groupby="y")
```

2.14 Append Rows/Columns

Append rows / columns to a Frame using:

```
df1.cbind(df2, df3)
df1.rbind(df4, force=True)
```

CHAPTER 3

Python API

3.1 `Frame`

CHAPTER 4

Indices and tables

- genindex
- modindex
- search