AlphaTwirl

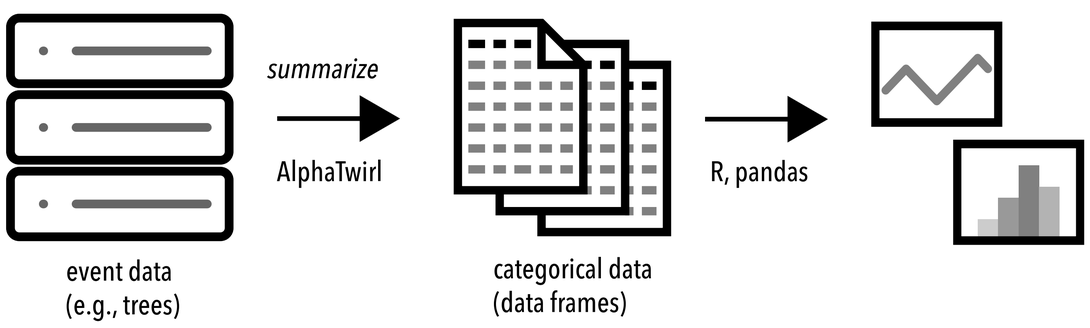
python library for summarizing event data into multi-dimensional categorical data

Tai Sakuma *University of Bristol* 9–13 July, 2018, CHEP 2018

Introduction

AlphaTwirl is a python library that summarizes large **event data** into a set of (multi-dimensional) **categorical data**, which can be loaded as **data frames** in R [1] and pandas.[2] AlphaTwirl is used in multiple analyses to summarize event data in Delphes trees,[3] Heppy trees,[4] and CMS MiniAOD and NanoAOD.[5] AlphaTwirl is released under the BSD license and is available on GitHub at *https://github.com/alphatwirl/alphatwirl*.

- **Event data:** one entry for one event,e.g., ROOT TTree with one entry per proton-proton collision event at LHC. Typically large, stored on a dedicated storage system
- **Categorical data:** one entry for one category, summaries of event data. e.g., histograms. Usually small enough to be loaded into memory on a laptop.



Data frames

- a 2-dimensional tabular data structure: columns for variables, rows for entries.
- implemented as a data type in R and pandas, providing a rich set of data operations: sort by values, concatenate, merge by keys, group by values, reshape between long and wide formats.
- can express multi-dimensional categorical data.

process	htbin	njetbin	minChi	n
QCD	400	2	0.00	8.15e+05
QCD	400	2	0.05	3.49e+05
QCD	400	2	0.1	1.18e+05
QCD	400	2	0.15	3.78e+04
⋮				
TTJets	1200	6	1.45	0.00
TTJets	1200	6	1.5	0.00

An example of categorical data as a data frame showing the number of events in each category specified by 4 variables (4D histogram)

Data frames can be considered as extensions of histograms

Why does AlphaTwirl summarize event data?

Why not simply convert the data type of event data to data frames?

- As data frames can express event data, it is in principle possible to convert the data type.
- However, event data are often too large for R and pandas, which load all data into memory.
- Furthermore, the next step after the conversion in typical binned analyses would be to make histograms or, in more general terms, summarize event data.

➤ AlphaTwirl summarizes event data without converting their data type.

Split-apply-combine strategy

The general idea of AlphaTwirl is to employ the **split-apply-combine** strategy [6] on event data that are too large to be loaded in memory.

- Split event data into groups determined by categories, apply a function to summarize data in each group, and combine the results as a data frame.
- Histograms can be created in this strategy—split data into bins, count the number of entries in each bin, and combine the results.

Summarizing events in AlphaTwirl is a generalization of creating histograms

Dependency injection

- Classes in AlphaTwirl generally operate on arguments of their methods (duck typing). For example, the EventLoop class knows nothing about the nature of the events—not even whether they are loaded from ROOT TTrees.
- Particular implementations of nearly all functionalities are determined at run time: input formats, output formats, a concurrency method, event selections, object selections, categorization, event summarizing methods, summary collecting methods, delivery methods, and even progress bars.
- In addition, conversely, each particular implementation doesn't depend on AlphaTwirl. They can be reused in a different framework with simple adapters. For example, event selection code can be reused in Heppy.

Users can extend and customize almost any functionality with reusable code

Dictionary configuration

Users can specify conditions of event and object selections and definitions of data frames—variables, indices, binnings, and summary method—with python dictionaries

```
dict(All=('ev: ev.ht[0] >= 400',
          'ev: ev.mht[0] >= 200',
          dict(Any=('ev: ev.nJet[0] == 1',
               dict(All=('ev: ev.nJet[0] >= 2',
                         'ev: ev.minChi[0] >= 0.7'))
))))
```

An example configuration of event seleciton—specified as a nested combination of *All* and *Any*

```
htbin = Binning(boundaries=(400, 800, 1200))
dict(keyAttrNames=('ht', 'jet_pt'),
     binnings=(htbin, RoundLog(0.1, 100)),
     keyIndices=(None, 0),
     keyOutColumnNames=('htbin', 'jet_pt'))
```

An example data frame configuration: histogram of the leading jet p_T in bins with equal width in the log scale ($\cdots, 10^{1.9}, 10^{2}, 10^{2.1}, 10^{2.2}, \cdots$) in three ranges of H_T

- Indices of arrays can be flexibly specified with wild cards and back references, e.g., p_T and η of all possible pairs of a jet and a muon:

```
dict(keyAttrNames=('jet_pt', 'jet_eta', 'muon_pt', 'muon_eta'),
     keyIndices=('(*)', '\\1', '(*)', '\\2'), … )
```

Performance, concurrency, optimization for speed

- CPU performance is closely monitored; slow algorithms are regularly replaced with faster ones. In particular, lines of code executed for every variable of every event are carefully optimized for speed.
- Large event data can be split into chuncks and processed concurrently with multiple cores or a batch sytem such as HTCondor.[7]

References

[1] R, *https://www.r-project.org/*
[2] pandas, *https://pandas.pydata.org/*
[3] DELPHES 3 Collaboration, JHEP **02** (2014) 057, *doi:10.1007/JHEP02(2014)057*
[4] Heppy, *https://github.com/cbernet/heppy*
[5] A. Rizzi, G. Petrucciani, *NanoAOD*, CHEP 2018
[6] H. Wickham, JSS 40 (2011) 1, *doi:10.18637/jss.v040.i01*
[7] HTCondor, *https://research.cs.wisc.edu/htcondor/*

9 July, 2018, Tai Sakuma <tai.sakuma@cern.ch>, (correction 2 August 2018)