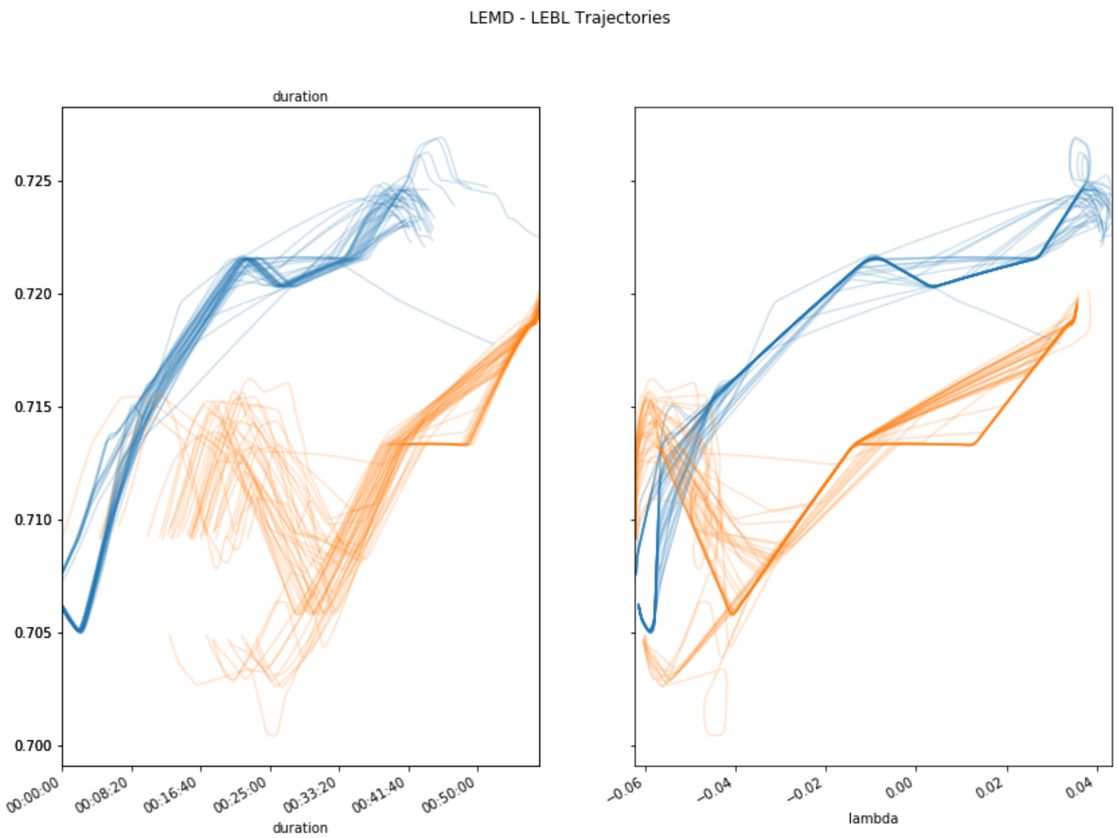
Python Big Data Analytics with Dask

Juan Luis Cano Rodríguez [hello@juanlu.space (mailto:hello@juanlu.space)](mailto:hello@juanlu.space)

Data-Enhanced Trajectory Based Operations Workshop @ ICRAT, 2018-06-25



Outline

1. Introduction
2. Dask
3. Application to Trajectory Prediction
4. Future work
5. Conclusions

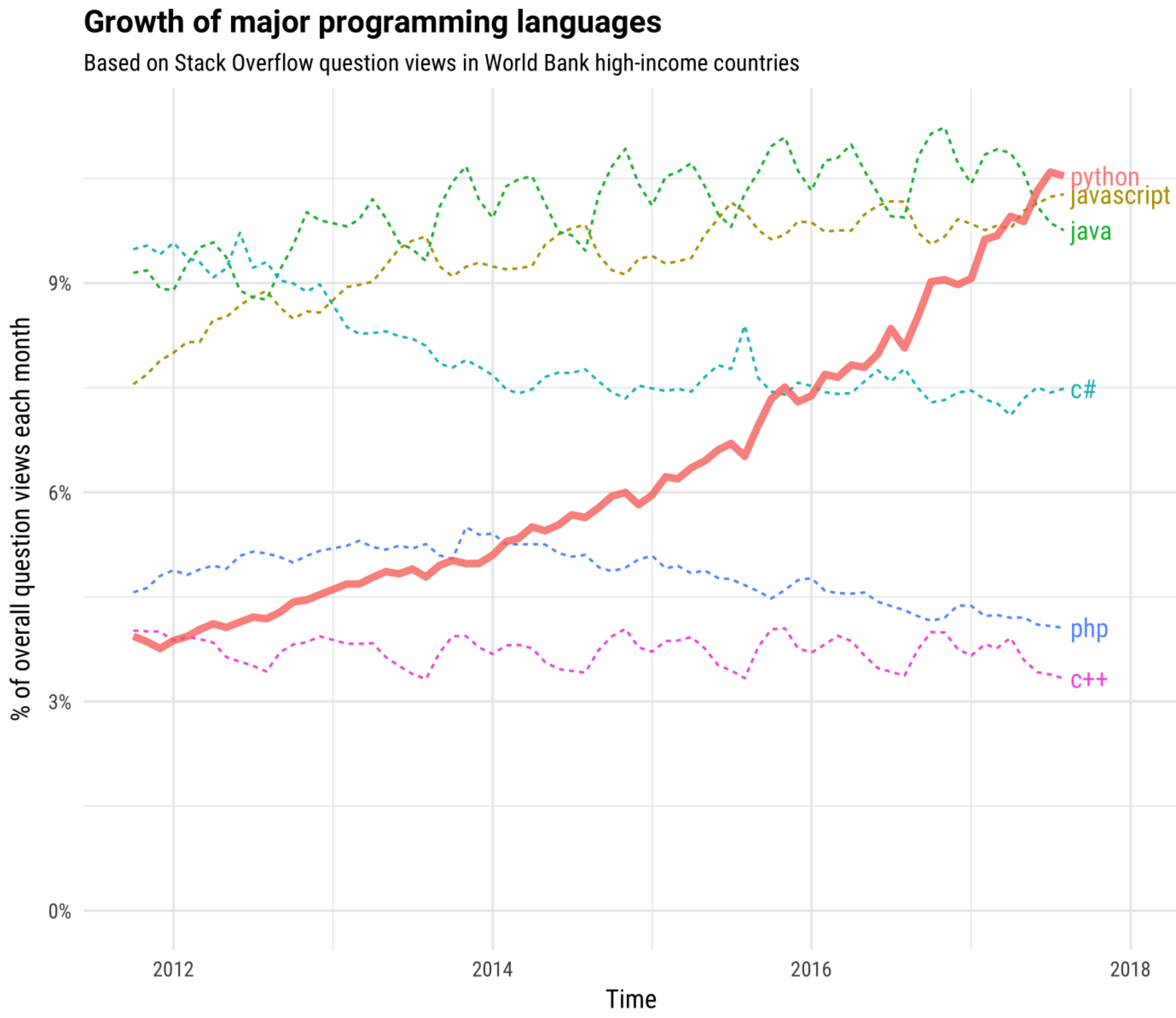
About me

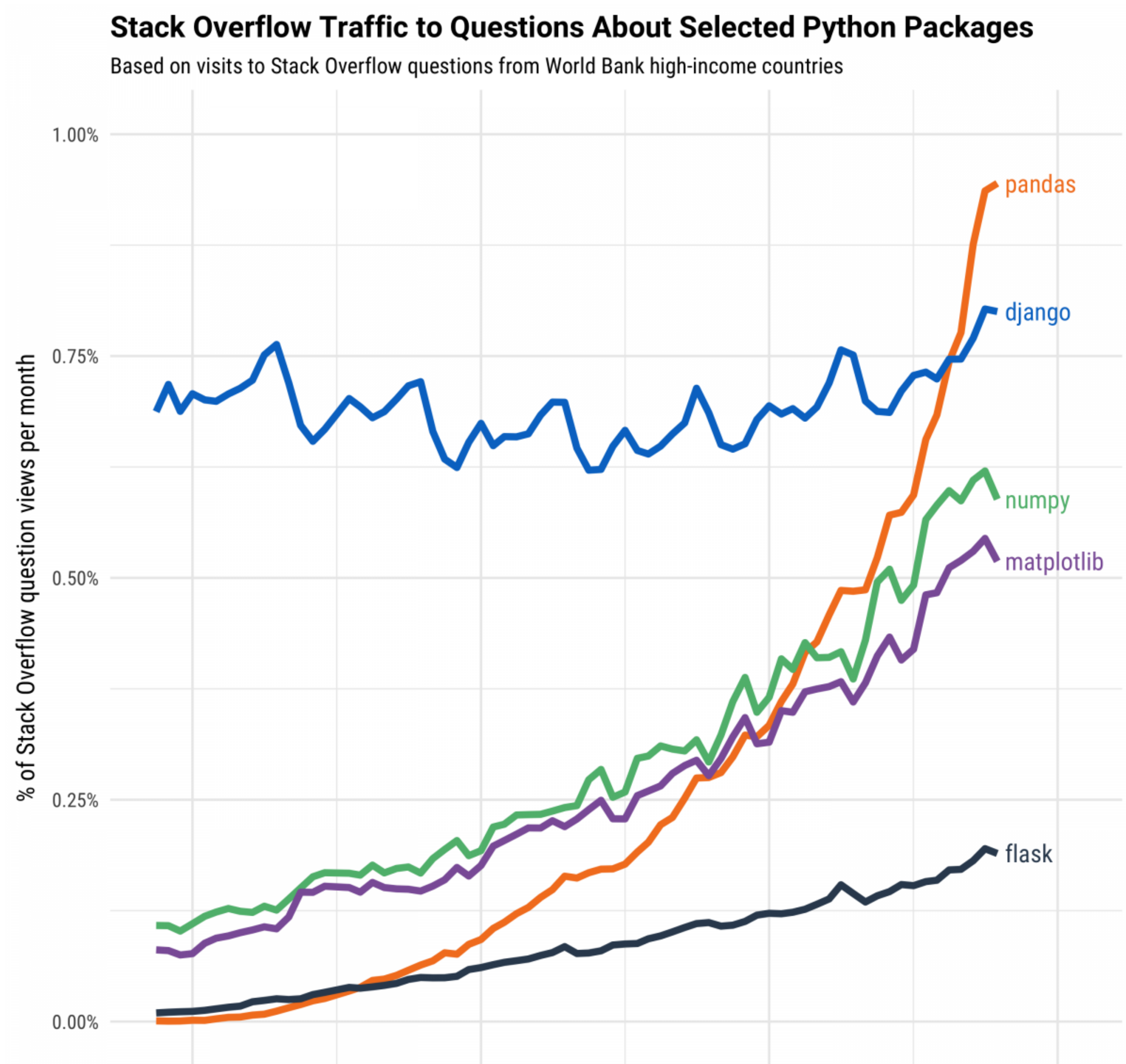
- Aeronautical Engineer specialized in Orbital Mechanics 🛰
- Founder and president of the **Python España** non profit, as well as co-organizer of **PyConES** 🐍
    - Next edition in Málaga, tickets selling out soon [https://2018.es.pycon.org/ (https://2018.es.pycon.org/)](https://2018.es.pycon.org/)
- **Software Engineer** at the geospatial infrastructure team in **Satellogic** 🌍
- **Freelancer** for R&D projects
- Open Source advocate and specially about Python for scientific computing

1. Introduction

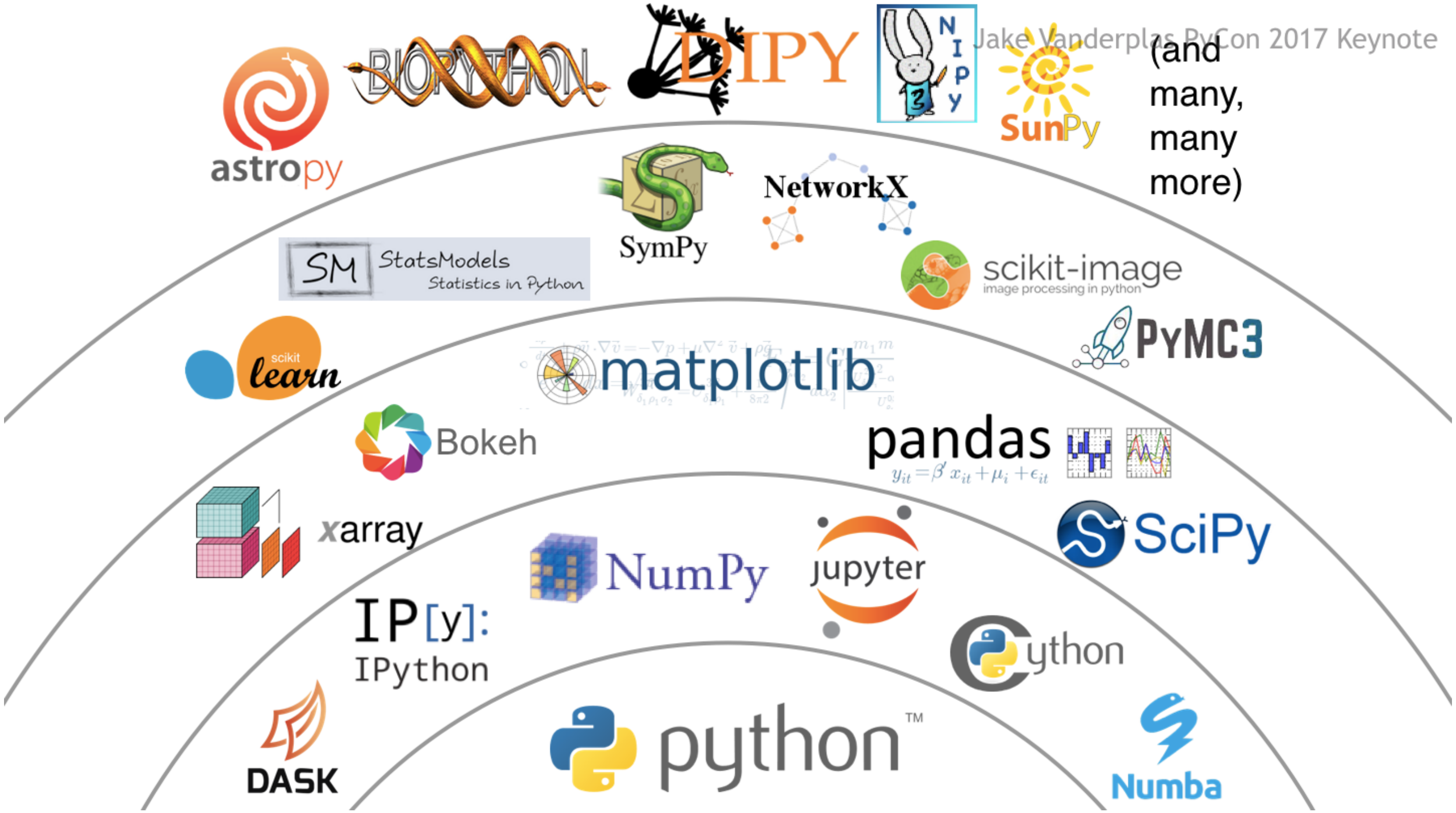
Python for Data Science

- Python is a **dynamic**, **relatively easy to learn**, **general purpose language**
- There is a vast ecosystem of **commercial-friendly, open source libraries** around it
- Growth in the latest years mainly due to adoption in **Data Science**


Growth of major programming languages
Based on Stack Overflow question views in World Bank high-income countries
% of overall question views each month
9%
6%
3%
0%
2012
2014
2016
2018
Time
python
javascript
java
c#
php
c++



Stack Overflow Traffic to Questions About Selected Python Packages
Based on visits to Stack Overflow questions from World Bank high-income countries
% of Stack Overflow question views per month
1.00%
0.75%
0.50%
0.25%
0.00%
pandas
django
numpy
matplotlib
flask


Jake Vanderplas PyCon 2017 Keynote
astropy
BIOPYTHON
DIPY
NIPY
SunPy
(and many, many more)
SymPy
NetworkX
StatsModels Statistics in Python
scikit-image image processing in python
scikit learn
matplotlib
PyMC3
Bokeh
pandas
xarray
NumPy
jupyter
SciPy
IP[y]: IPython
Cython
DASK
python™
Numba

Limitations

- All Python libraries were designed for in-memory computing
- On their own, they don't work well with bigger-than-RAM datasets
- Apart from embarrasingly parallel problems, we need other solutions

Current mature tool: PySpark

- Python API for Spark, a complete distributed computing framework written in Scala (Java derivative)
- Pros: Rich ecosystem, good integration with Big Data technologies (Hadoop, Hive)
- Cons: Python to/from Java serialization is slow and fragile, difficult to debug

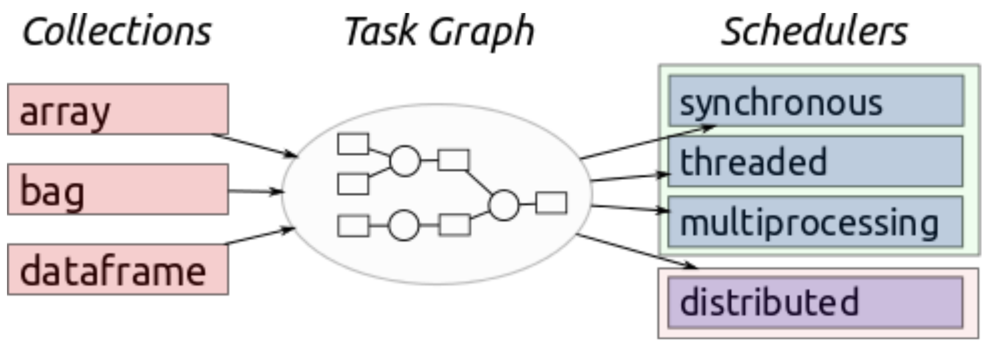
2. Dask

"Dask provides advanced parallelism for analytics, enabling performance at scale for the tools that you love"

1. Dynamic task scheduling optimized for interactive computation
2. "Big Data" collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments

Dask emphasizes the following virtues:

- **Familiar**: Provides parallelized NumPy array and Pandas DataFrame objects
- **Flexible**: Provides a task scheduling interface for more custom workloads and integration with other projects
- **Native**: Enables distributed computing in Pure Python with access to the PyData stack
- **Fast**: Operates with low overhead, low latency, and minimal serialization necessary for fast numerical algorithms
- **Scales up and down**: Runs resiliently on clusters with 1000s of cores or a laptop in a single process
- **Responsive**: Designed with interactive computing in mind it provides rapid feedback and diagnostics to aid humans



In [1]:
```
from distributed import Client, progress

client = Client()
client
```

Out[1]:

Client

- **Scheduler:** tcp://127.0.0.1:32941
- **Dashboard:** [http://127.0.0.1:8787/status (http://127.0.0.1:8787/status)](http://127.0.0.1:8787/status)

Cluster

- **Workers:** 4
- **Cores:** 4
- **Memory:** 8.27 GB

In [2]:
```python
import numpy as np
import dask.array as da

x = np.arange(1000)
y = da.from_array(x, chunks=100)
```

In [3]:
```python
y
```

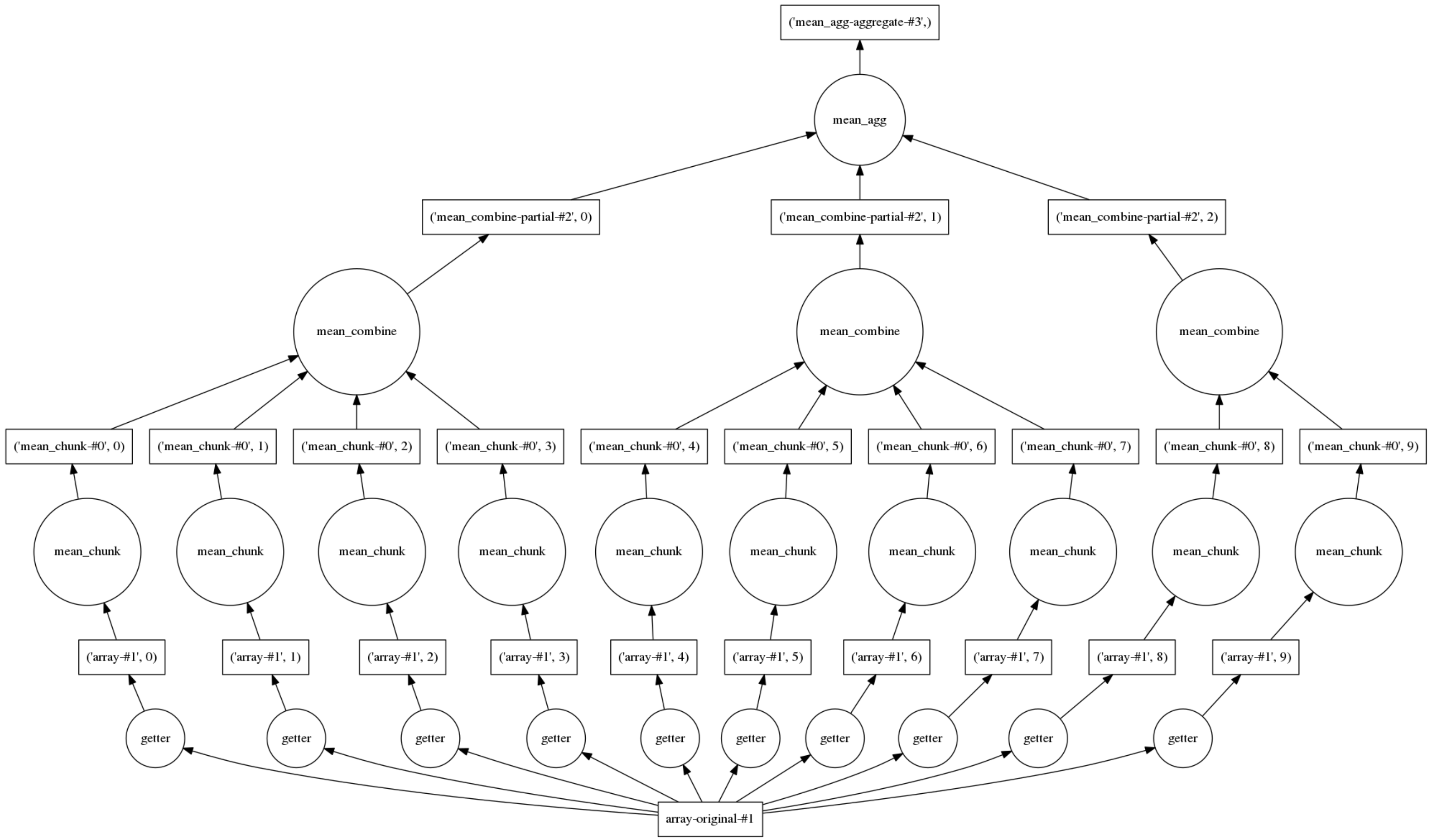
Out[3]:
```
dask.array<array, shape=(1000,), dtype=int64, chunksize=(100,)>
```

In [4]:
```python
op = y.mean()
op
```

Out[4]:
```
dask.array<mean_agg-aggregate, shape=(), dtype=float64, chunksize=()>
```

In [5]:
```
op.visualize()
```

Out[5]:



In [6]:
```
op.compute()
```

Out[6]:
```
499.5
```

In [7]:
```
import dask.dataframe as dd

df = dd.read_csv("data/yellow_tripdata_*.csv", parse_dates=['tpep_pickup_datetime',
'tpep_dropoff_datetime'])
```

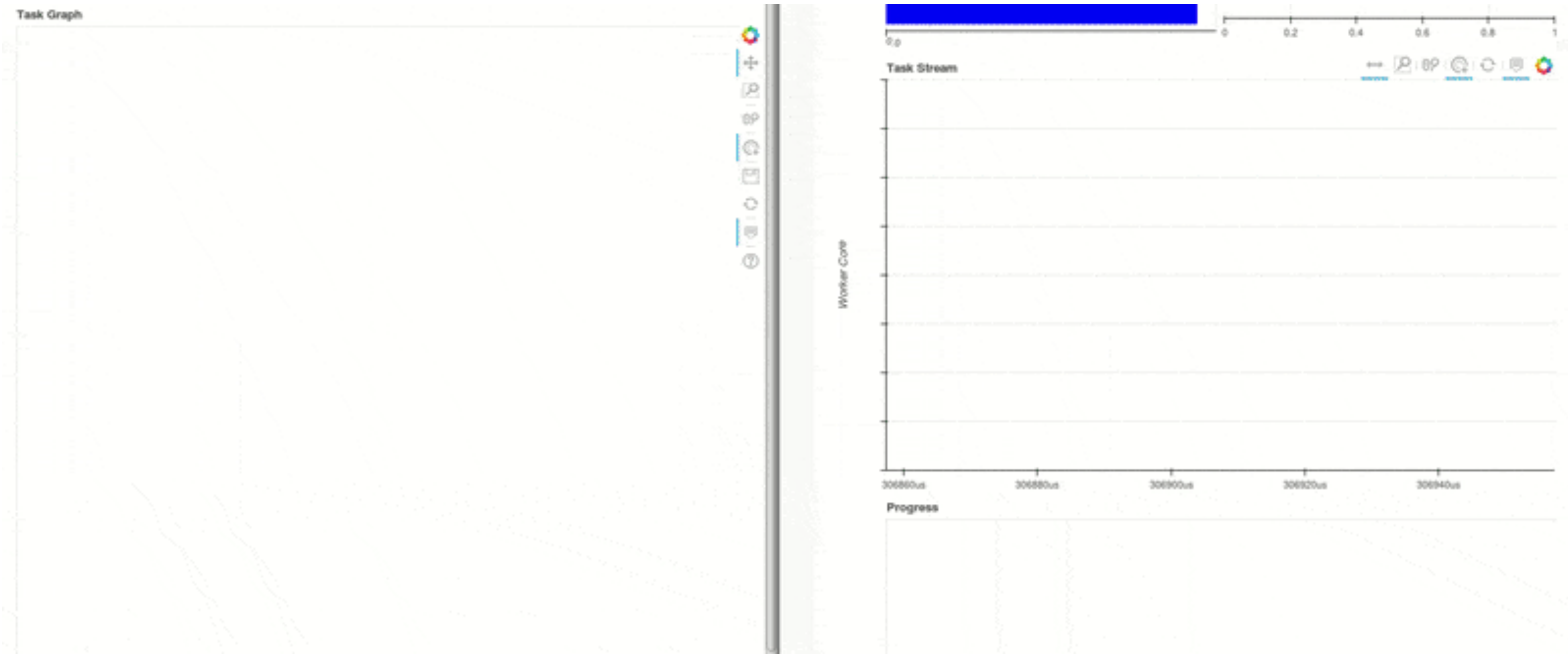
In [8]:
```
df.head()
```

Out[8]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	RateCode
0	2	2015-01-15 19:05:39	2015-01-15 19:23:42	1	1.59	-73.993896	40.750111	1
1	1	2015-01-10 20:33:38	2015-01-10 20:53:28	1	3.30	-74.001648	40.724243	1
2	1	2015-01-10 20:33:38	2015-01-10 20:43:41	1	1.80	-73.963341	40.802788	1
3	1	2015-01-10 20:33:39	2015-01-10 20:35:31	1	0.50	-74.009087	40.713818	1
4	1	2015-01-10 20:33:39	2015-01-10 20:52:58	1	3.00	-73.971176	40.762428	1

In [9]:
```
len(df)
```

Out[9]:
```
12748986
```



3. Application to trajectory prediction

Problem setting

- Complete air traffic data in Spain resampled to 1 second from January to May 2016
- **44264 CSV files, ~98 GiB of data**
- In each file, we have time histories of geometric, aerodynamic and atmospheric variables

Objective: "Explore machine learning algorithms to predict the trajectories"

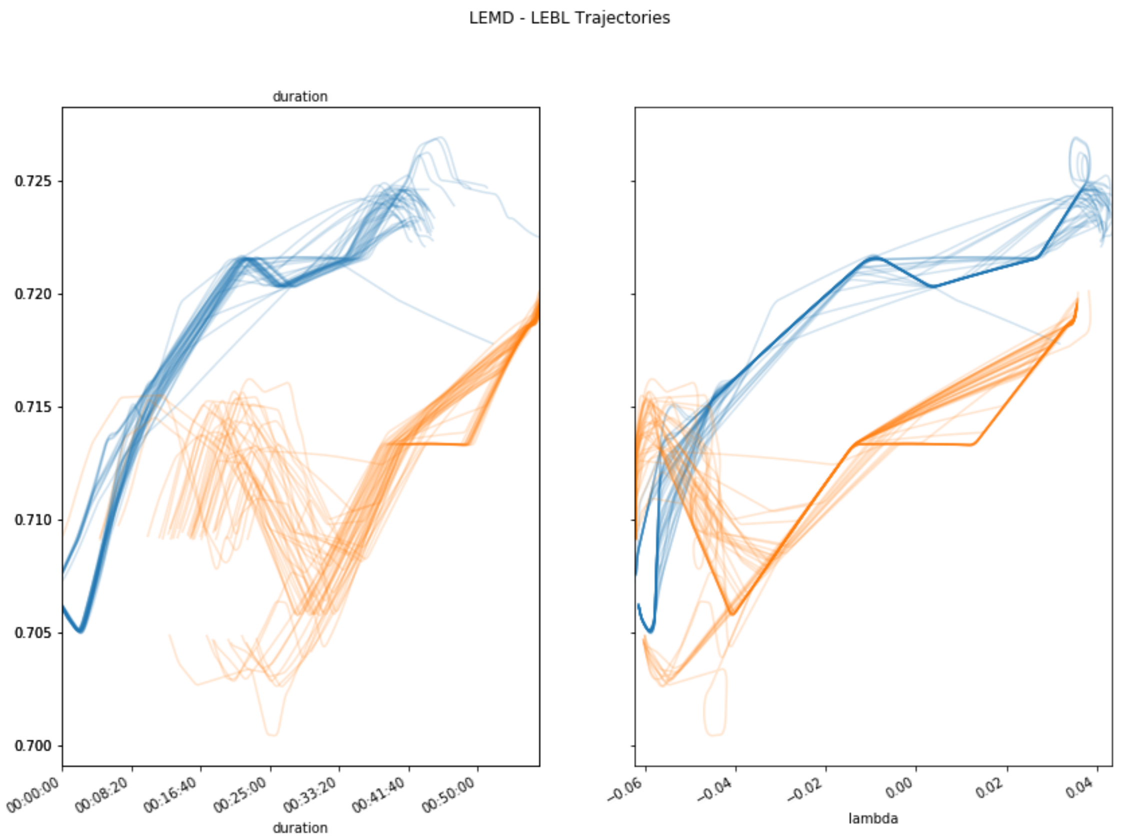
Requirement: Preserve the confidentiality of the data - i.e. don't use cloud resources

Analyzing 100 GiB of data in a 8 GiB RAM laptop? Challenge accepted!

- To preserve the confidentiality of the data, the analysis was done on a laptop:
    - Linux Mint 18.2 64-bit, kernel 4.10.0-35-generic
    - Intel Core™ i5-6200U CPU @ 2.30 GHz x 2 (4 cores)
    - ~8 GiB of RAM
- We focused on a subset of the data (only LEMD ✈ LEBL trajectories)
- To avoid reading all the CSV files every time, we first built an **index of files** in Apache Parquet format
    - This only contained name of the file and pair of cities

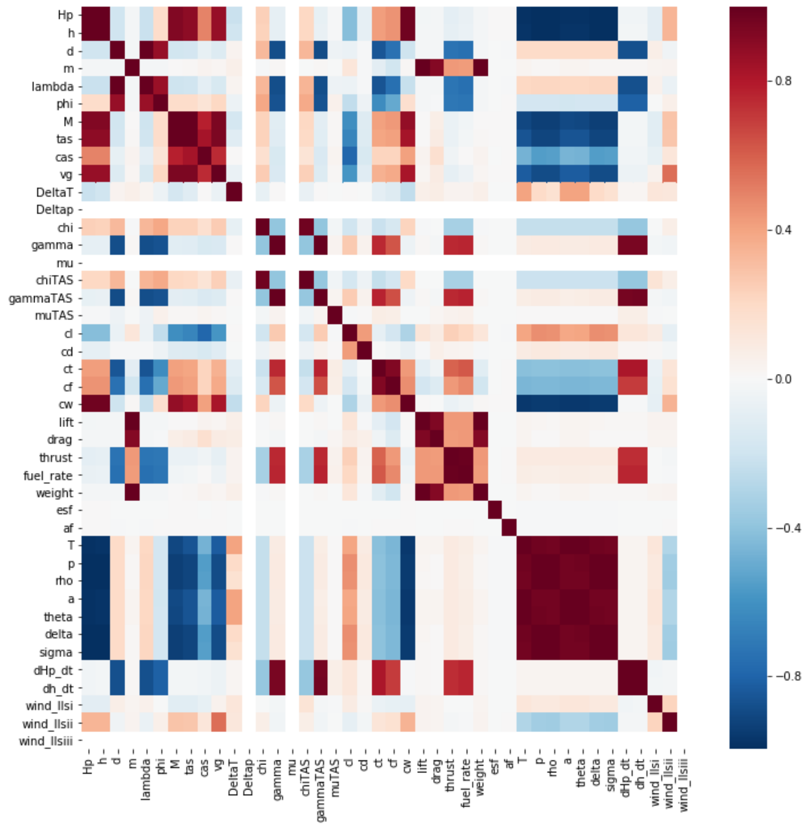
Exploratory analysis

- Normalizing with respect to time seems the most natural option
- However, spatial normalization appears to give less dispersion
- A monotonically increasing variable has to be chosen: imperfect solution



Correlations

Many variables are strongly correlated, so they could be discarded for the analysis



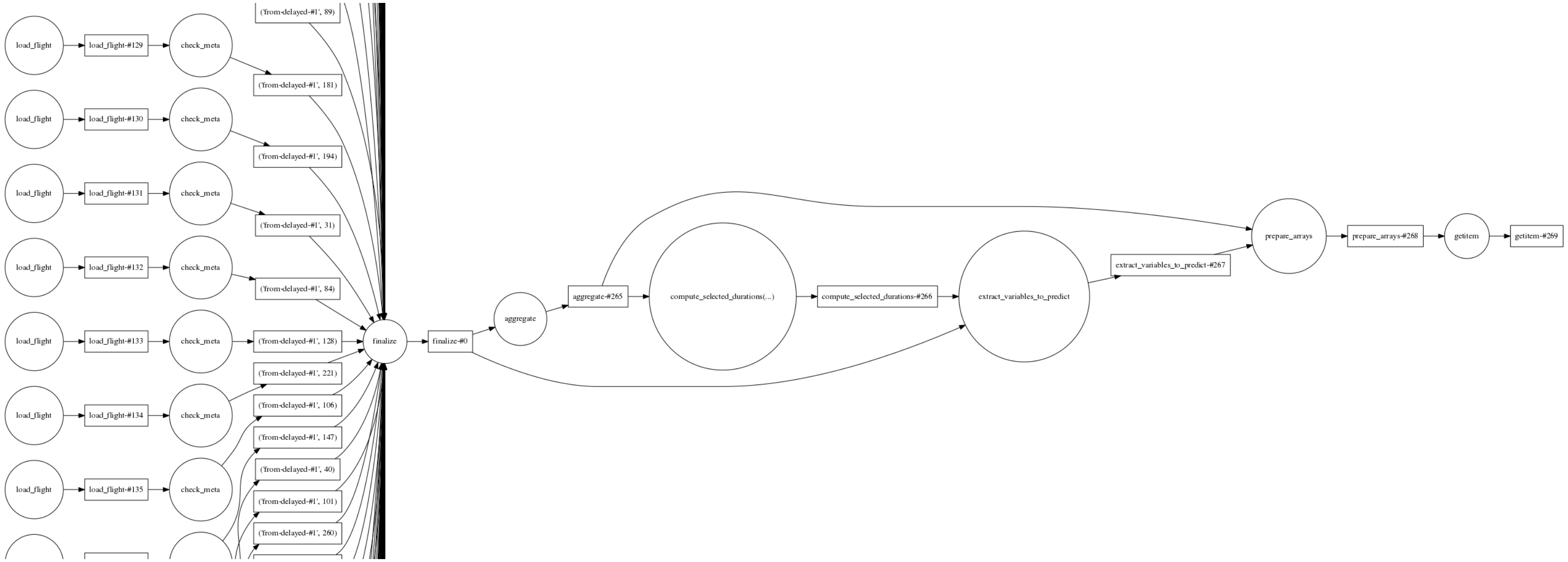
Model

- We tried a simple approach based on computing aggregated variables for each flight
- The target variables were the 3D position, time and distance at an specific % of the total duration
- We used `RandomForestRegressor` and `MultiOutputRegressor(RandomForestRegressor)` from **scikit-learn**
- **Pros:** We only need to compute the aggregated variables once per flight
- **Cons:** Time history information is lost

	\	**Predictors (whole flight)**	\	**Target variables (x %)**
	\	duration_max, T_mean, p_mean, ..., wkday_Sat	\	λ, φ, duration, distance, H_p
Flight 1	\	...	\	...
Flight 2	\	...	\	...
...	\	...	\	...
Flight N	\	...	\	...

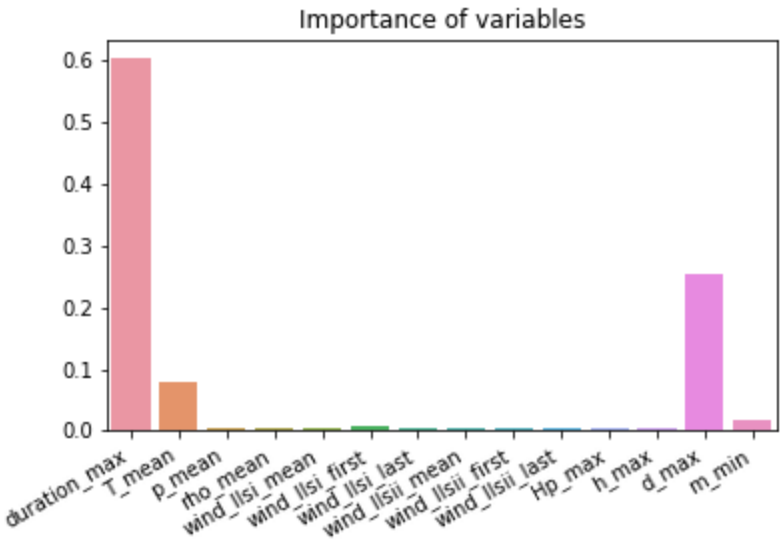
Process

- We first computed the aggregated variables
- Both models were trained independently for each time fraction



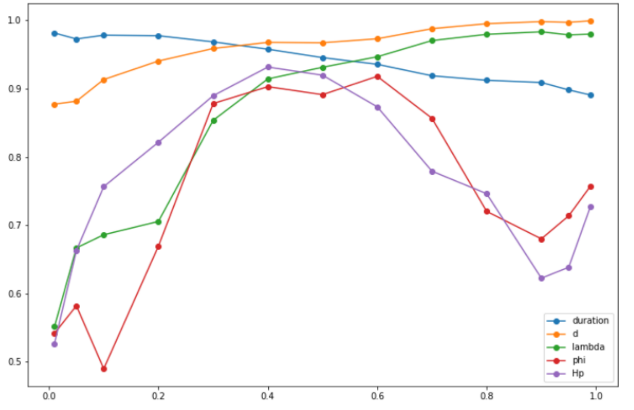
Variables importance

- Using the Random Forest algorithm, the importance of the variables for the prediction was obtained
- Most important variables were **maximum duration** and **distance**, **mean temperature** and **minimum mass**
- We kept the categorical variables for the prediction as well

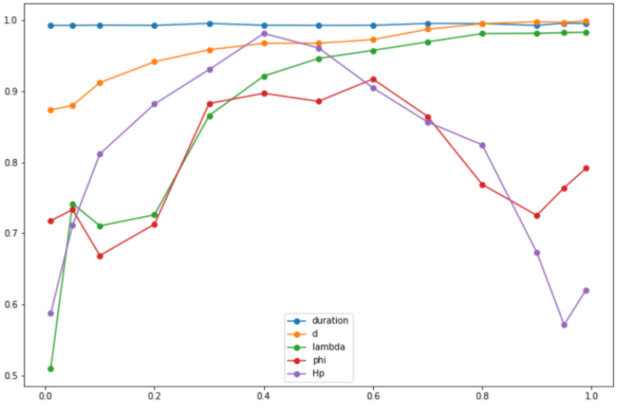


Results

The accuracy was better in the central phase of the flight (R^2 ~ 0.9) and worse in take-off and landing (R^2 ~ 0.7)



RandomForest



MultiOutputRegressor

4. Future work

- Deeper exploratory analysis and feature engineering
- Scale to a cluster for better performance, more models
- Automate the processing
- Dask-ML for training

5. Final thoughts

- Traditional Python libraries are not ready to scale horizontally
- **Dask enables an interactive, familiar workflow easy to scale from a laptop to a cluster**
- This simple model could use the result of a clustering to do the prediction
- Interactive visualization and exploration analysis is crucial

Questions?

- https://github.com/Juanlu001 (https://github.com/Juanlu001)
- hello@juanlu.space (mailto:hello@juanlu.space)