Yandex

On computational experiment analysis infrastructure

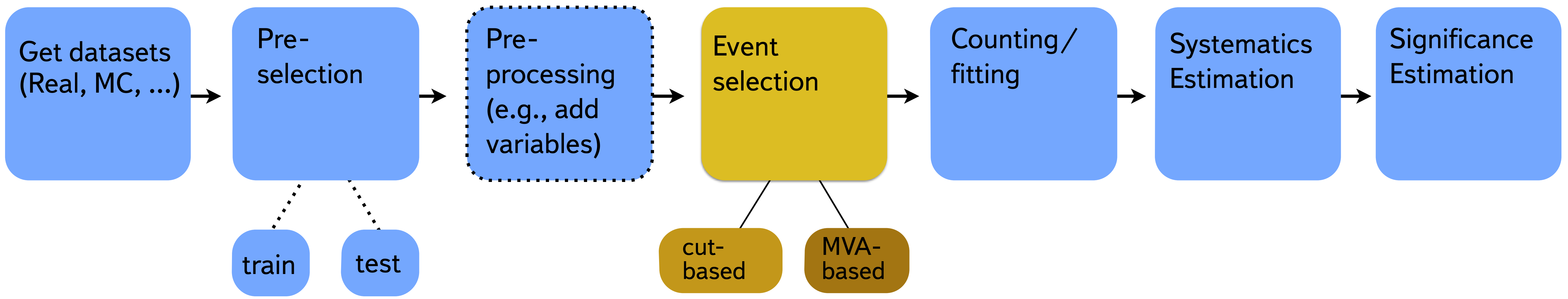
Andrey Ustyuzhanin
June 4 2014

Computational experiment

- **Characteristics**
  - Big datasets
  - Complicated (complex) processing
- **Science**
  - HEP/Cosmology
  - Neuro-medicine
  - Microbiology/Genetics
- **Industry**
  - predictive modeling (machine learning)
  - analysis & reporting

Quest for sensitivity

Analysis Value Chain

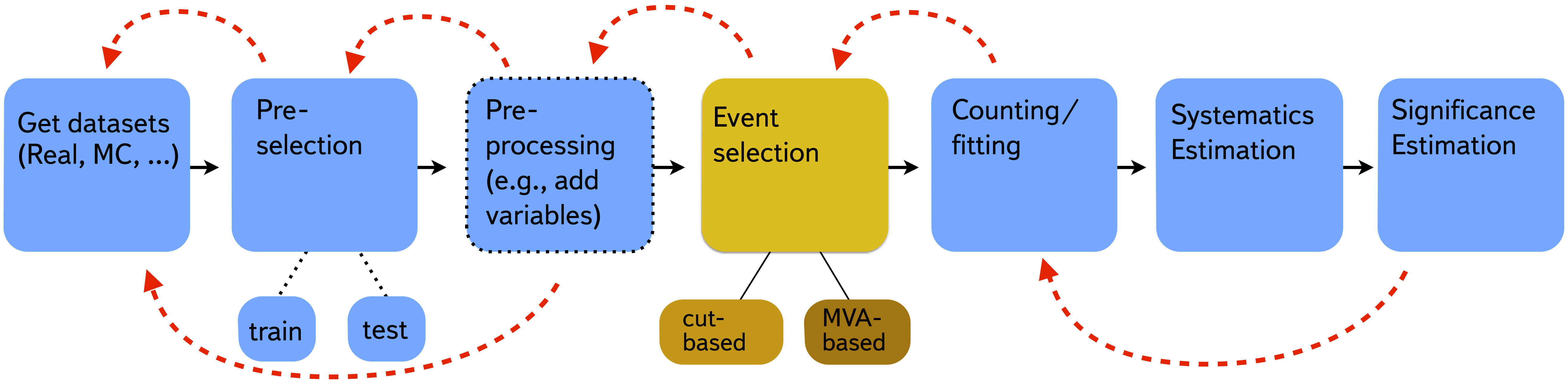


Complexity indicators

- ‘I can’t remember which version of the code I used to generate figure 13’
- ‘The new student wants to reuse that model I published three years ago but he can’t reproduce the figures’
- ‘I thought I used the same parameters but I’m getting different results!?’
- ‘It worked yesterday!’
- ‘Why did I do that?’
- ‘Where are events selected with previous version of reconstruction software?’

Analysis complexity

Case: $\tau \to 3\mu$ (LHCb)



Get datasets (Real, MC, ...)

Pre-selection

train

test

Pre-processing (e.g., add variables)

Event selection

cut-based

MVA-based

Counting/ fitting

Systematics Estimation

Significance Estimation

Repeat count: 10^2 10^2 10^3 10^2 10^2 10^2

Trained models: ~1500

Requires dedicated framework!

Complexity sources

- Domain
- Datasources
- Analysis strategy (http://bit.ly/SqDDE4)
- Analysis step details
- (Distributed) team communication

Yandex vision

software infrastructure to support a collaborative ecosystem for computational science. It is a solution for team of researchers that allows

- running computational experiments on big shared datasets,
- obtaining reproducible and repeatable results,
- comparing measurable result consistently.

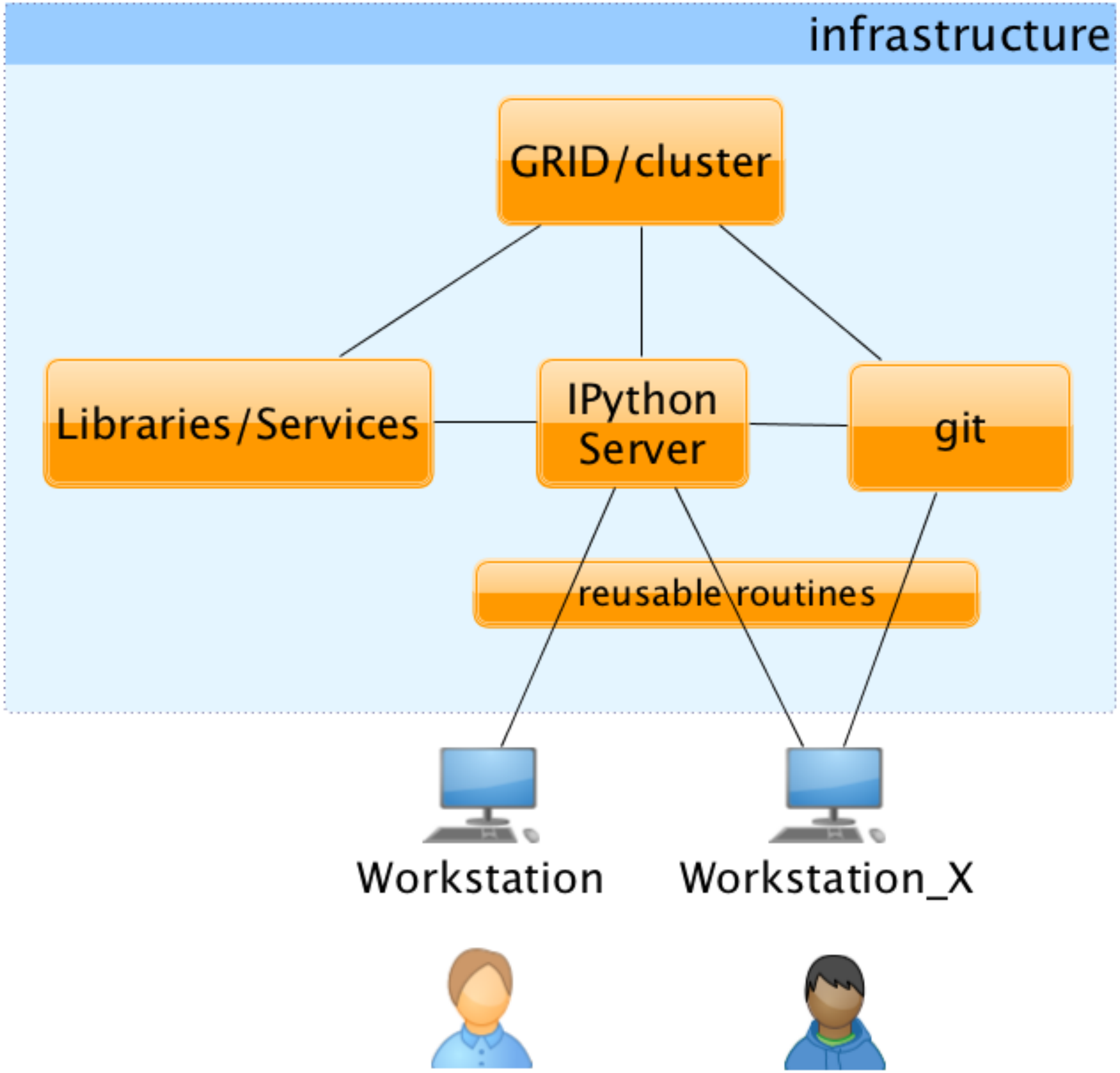
Requirements

- Analysis automation / code reusability
- Consistent cross-checks
- Online visual shared environment
- Reproducibility (provisioning)
- Standard modules support (ROOT, RooFit)
- Scalability
- [flat learning curve]

Ideal workflow

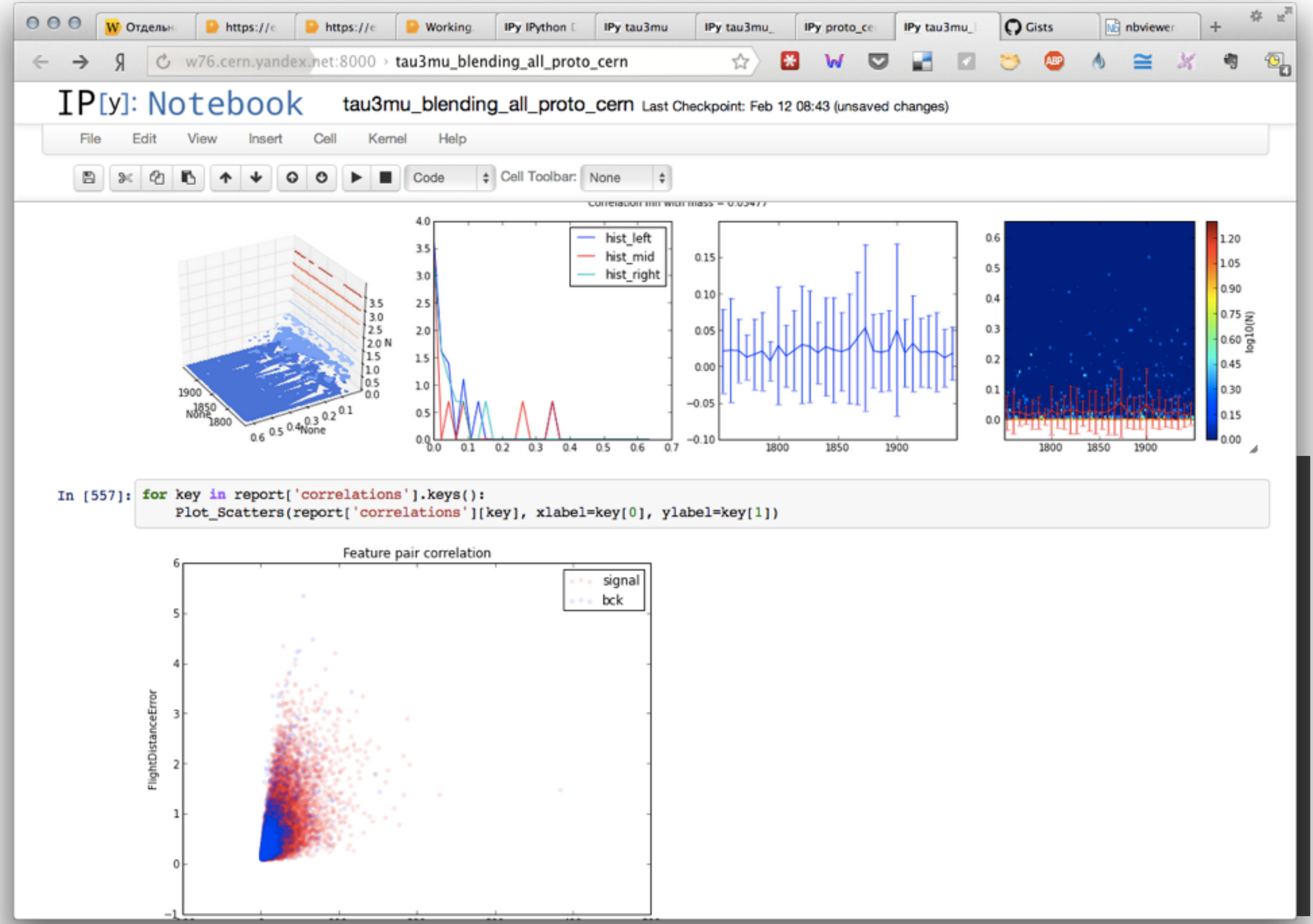
- Prerequisites
  - Git, JIRA, access to computation cluster
- Data preparation
  - MC, real data, stripping to be used
- Analysis strategy definition
- Routines code/modules
- Notebooks to play with code/data
- Analysis preservation
  - analysis note, code/data/environment

Components Diagram



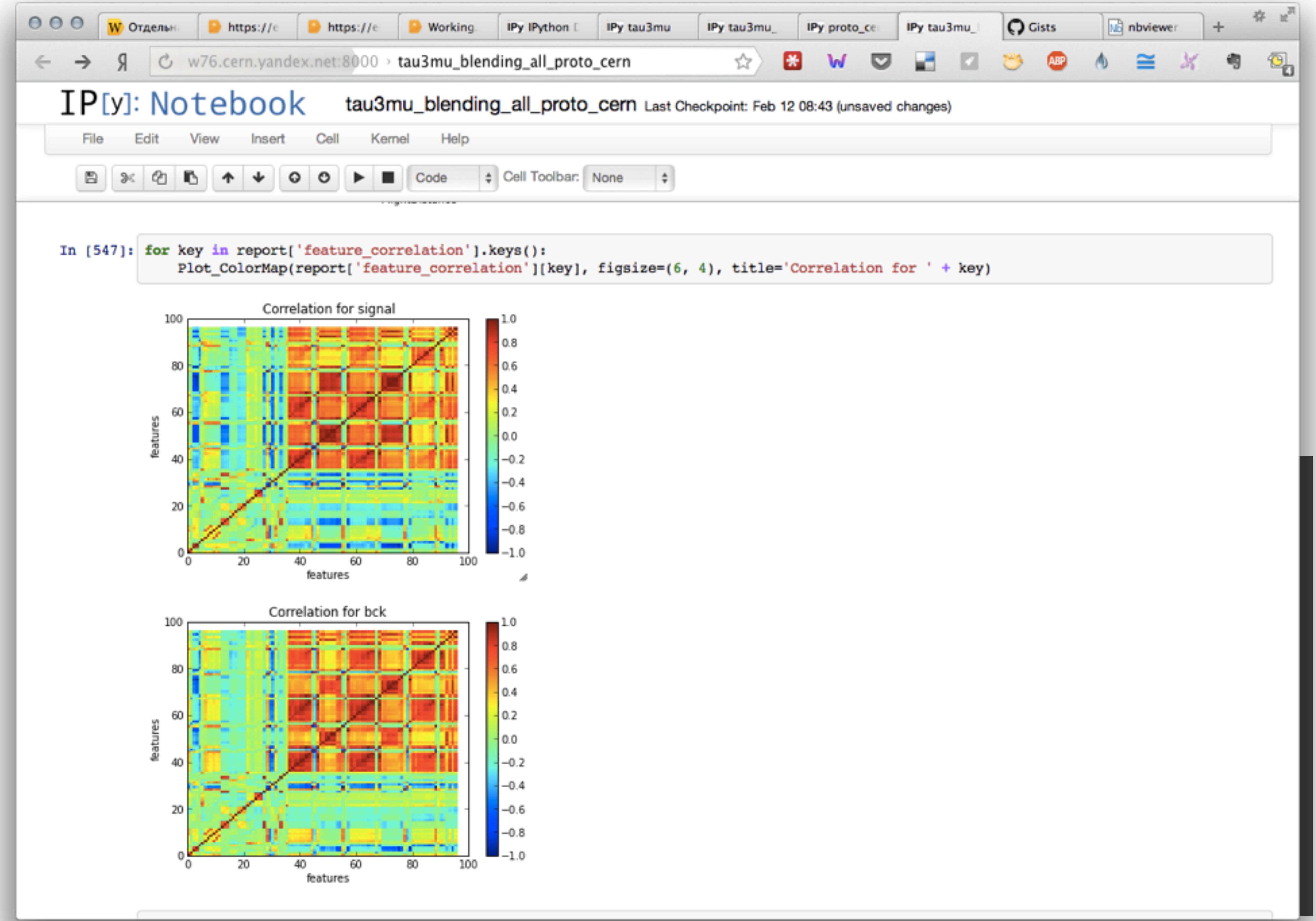
Existing pieces

- IPython (http://bit.ly/1h7zK2d, http://bit.ly/1jQ1vaC)
- Event Filter (https://twiki.cern.ch/twiki/bin/view/LHCb/EventFilterHowtos)
  - TMVA, MatrixNet, …
- Support for:
  - ROOT, PyROOT, scikit-learn, …
- JIRA (https://its.cern.ch/jira/browse/BSTOFOURMU)
- Run on lxplus or CERNVM



tau3mu_blending_all_proto_cern Last Checkpoint: Feb 12 08:43 (unsaved changes)

```
In [557]: for key in report['correlations'].keys():
              Plot_Scatters(report['correlations'][key], xlabel=key[0], ylabel=key[1])
```



```
In [547]: for key in report['feature_correlation'].keys():
              Plot_ColorMap(report['feature_correlation'][key], figsize=(6, 4), title='Correlation for ' + key)
```

Evaluation

- $B_s \to 4\mu$
- $\tau^- \to \mu^+\mu^-\mu^-$
- $B \to \overline{K^*}\mu^+\mu^-$
- $B_u \to J/\psi K K \pi$
- uniform efficiency classifier (extension of Mike Williams’ uBoost)

Next steps

- Provenance tracking (environment tracking, data)
- Running analysis jobs using modern distributed architectures (e.g. LSF, Hadoop, Impala, Drill)
- Analysis workflow visual representation / management
- Analysis preservation
- Integration with EventIndex for event retrieval

Conclusion

- Development is aligned with real analysis needs
- Prototype for a broader analysis ecosystem. Inspired by real industry case
- Start with Event Filter (https://twiki.cern.ch/twiki/bin/view/LHCb/EventFilterHowtos)
- Open-source, supposed to be fun
- Welcome to join! (cases?)