scikit-validate: Automating analysis validation

•••

Luke Kreczko for F.A.S.T.

IRIS-HEP Meeting, 04.03.2019

scikit-validate

- First, let's clarify the "validate" part
  - Validation can mean different things in the computing context
  - Here I mean: verification of observables after code changes
- Big experiments put a lot of effort into physics validation for releases
  - E.g. running Jenkins for merge/pull requests
- But what about small scripts, analysis frameworks or small experiments?
  - Usually not enough person power for full-fledged validation system
  - Nevertheless validation is needed for reliable scientific results
- This project started from a set of scripts from CMS analysis & LZ simulation

scikit-validate: Validation in two parts

Observables: compare to a reference

- values in ROOT trees
  - stored values
  - 10 biggest consumers in ntuples
- High-level observables
  - E.g. $(d_{xy})^2 = x^2 + y^2$ where x, y in TTree
- Generic metrics
  - Performance measurements
  - File sizes
- Code issues
- Software versions

Report:

- Tested environments
  - E.g. different GCC, OS, Python versions
- Continuous Integration pipeline
- validation outcomes
  - Mis-/match in distributions
- Generic metric monitoring
  - increase/decrease in metrics

Automate with commands

Comparison

```
root_diff out.root out_ref.root --out-dir=/tmp
pandas_diff* out.csv out_ref.csv --out-dir=/tmp
geometry_diff* geo.xml geo_ref.xml --out-dir=/tmp
```

Performance & Resource Metrics

```
execute_with_metrics <command>
add_file_metric <file>
inspect_root_file* <file>
```

Environment

```
detect_software_versions
```

Reporting

```
make_demo_report
make_report config.yml -o report.md
Submit_report_to_mr   *.md
```

Utility

```
get_target_branch
get_artifact_url
merge_json
remove_from_env
lint*
```

```
<command> --help
```

*planned/not yet public

Minimal example for validation of high-level observables

Takes in an input ROOT file (structure) & produces an output ROOT file

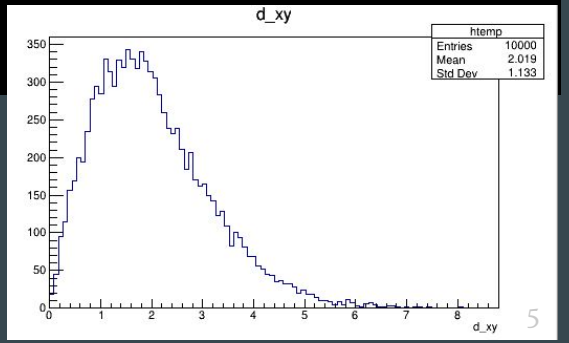
```
if __name__ == '__main__':
    args = sys.argv[1:]
    input_file = args[0]
    output_file = args[1]

    results = analyze(input_file)
    write(results, output_file)
```

```
def analyze(input_file):
    f = uproot.open(input_file)
    tree = f['test']
    x = tree['x'].array()
    y = tree['y'].array()
    d_xy = np.sqrt(x**2 + y**2)
    return d_xy
```

```
def write(results, output_file):
    f = ROOT.TFile.Open(output_file, 'recreate')
    tree = ROOT.TTree( 'test', 'high-level observables' )
    d_xy = array( 'f', [ 0. ] )
    tree.Branch('d_xy', d_xy, 'd_xy/F')
    for r in results:
        d_xy[0] = r
        tree.Fill()
    tree.Write()
    f.Close()
```

```
python high_level.py \
test_1.root out_ref.root
```

```
d_xy: [1.8860719 1.464918
2.4873514 2.042773
1.1673971 2.641898
1.135193
 1.0439259 2.6820924
1.5561566]
```

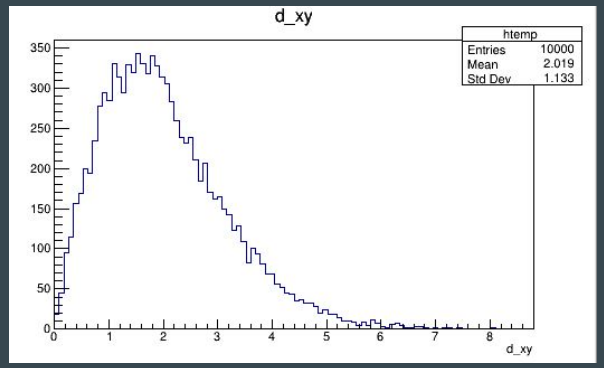


Full example

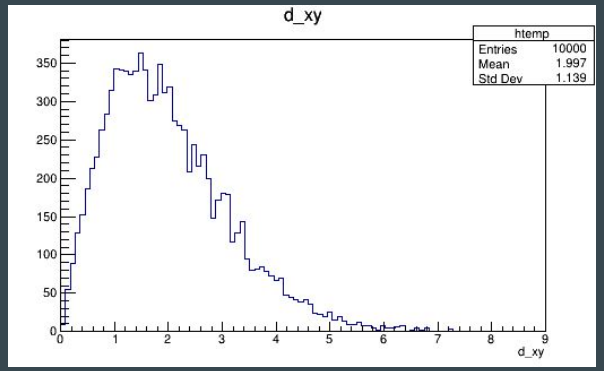
Comparing files - minimal information for a decision

```
root_diff out.root out_ref.root --out-dir=/tmp
```

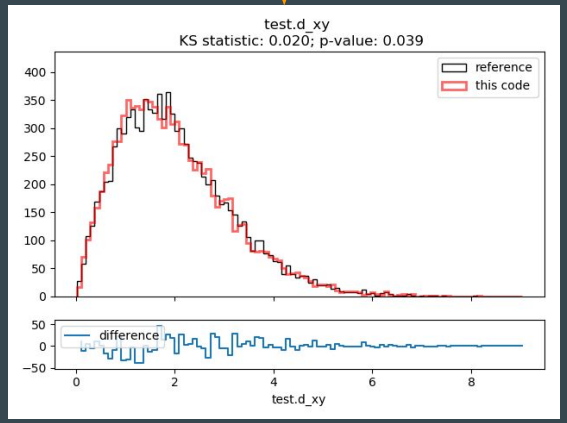
includes Kolmogorov - Smirnov test



out_ref.root



out.root
(input file)



/tmp/test.d_xy.png

Results are stored in JSON file for further processing

General metrics


```
{
"generate": {
        "cpu_time_in_s": {
            "value": 524.510786,
            "unit": "s",
            "lower_is_better": true
        },
        "max_rss_in_MB": {
            "value": 1992.5,
            "unit": "MB",
            "lower_is_better": true
        }
    }
}
```


execute_with_metrics generate


```
{
"mctruth.root": {
        "size_in_bytes": {
            "value": 94735343,
            "unit": "B",
            "lower_is_better": true
        },
        "size_in_mb": {
            "value": 90.3,
            "unit": "MB",
            "lower_is_better": true
        }
    }
}
```


add_file_metric mctruth.root

Add your own

Reporting
(including work-in-progress)

Structure

- Relies entirely on Jinja2 templates & YAML configuration
  - Templates define structure of report (defaults provided)
  - Configuration describes how templates are filled (examples provided)
- Own templated, code, variables can be inserted in config

Header example

```
output_file: {{ output_file | 'report.md' }}
template:
{{scikit_validate}}/data/templates/report/default/summary.md
output_url:
  function: skvalidate.gitlab.get_output_url
pipeline_url:
  function: skvalidate.gitlab.get_pipeline_url
title: "Default Gitlab Report"
sections:
  ...
```

Section example

```
performance_report:
    template:
{{scikit_validate}}/data/templates/report/default/performance.md
    metrics:
      function: skvalidate.report.get_metrics
      metrics_json: metrics.json
      symbol_up: '&#10138;'
      symbol_down: '&#10136;'
      symbol_same: '&#10134;
    sections:
        …. <sub sections>
```

make_demo_report

- Creates a demo report based on report/demo.yml
- Uses data from examples
  - Pipeline & validation summary

scikit-validate DEMO report

Pipeline summary

Current pipeline: https://gitlab.example.com/secret/enigma/pipelines/42

name	status	log	software versions
build1	passed	log (raw)	python=2.7.15, gcc=4.8.4, root=6.04/02, geant4=10.3.2
build2	❌	log (raw)	python=3.6.5, gcc=7.3.0, root=6.14/04, geant4=10.04.02
test1	passed	log (raw)	python=2.7.15, gcc=4.8.4, root=6.04/02, geant4=10.3.2
test2	❌	log (raw)	python=3.6.5, gcc=7.3.0, root=6.14/04, geant4=10.04.02
validation1	❌	log (raw)	python=2.7.15, gcc=4.8.4, root=6.04/02, geant4=10.3.2
validation2	passed	log (raw)	python=3.6.5, gcc=7.3.0, root=6.14/04, geant4=10.04.02

Validation report

job	status	summary	details	mismatch
validation1	failed	3/5 distributions differ	details	test;1.x, test;1.y, test;1.z
validation2	failed	4/6 distributions differ	details	test;1.a, test;1.x, test;1.y, test;1.z
validation3	failed	4/6 distributions differ	details	test;1.a, test;1.x, test;1.y, test;1.z

make_demo_report

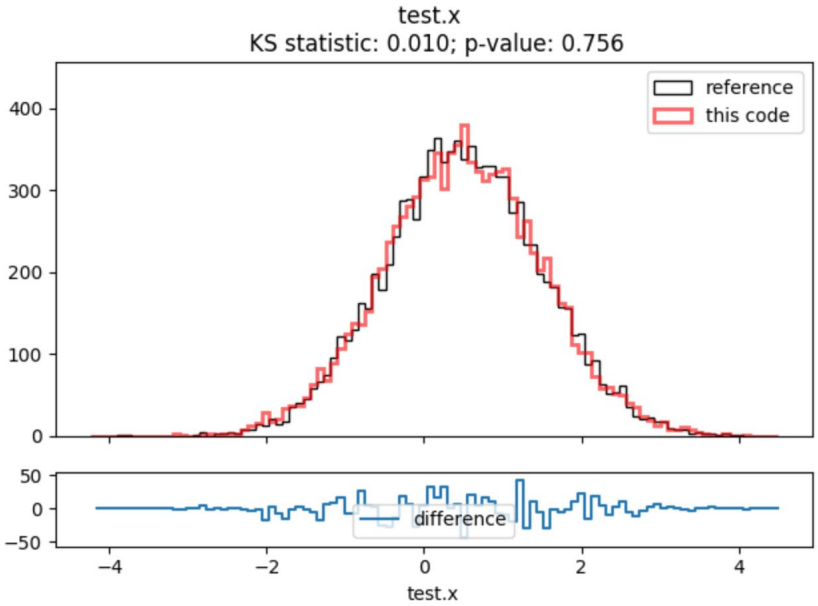
- Creates a demo report based on report/demo.yml
- Uses data from examples
  - Pipeline & validation summary
  - validation report

Detailed validation report

test;1.i (KS statistic: 0.000; p-value: 1.000 - success

test;1.v (KS statistic: 0.005; p-value: 0.211 - success

test;1.x (KS statistic: 0.011; p-value: 0.602 - failed



make_demo_report

- Creates a demo report based on report/demo.yml
- Uses data from examples
  - Pipeline & validation summary
  - validation report
  - Performance & file metrics

Performance report

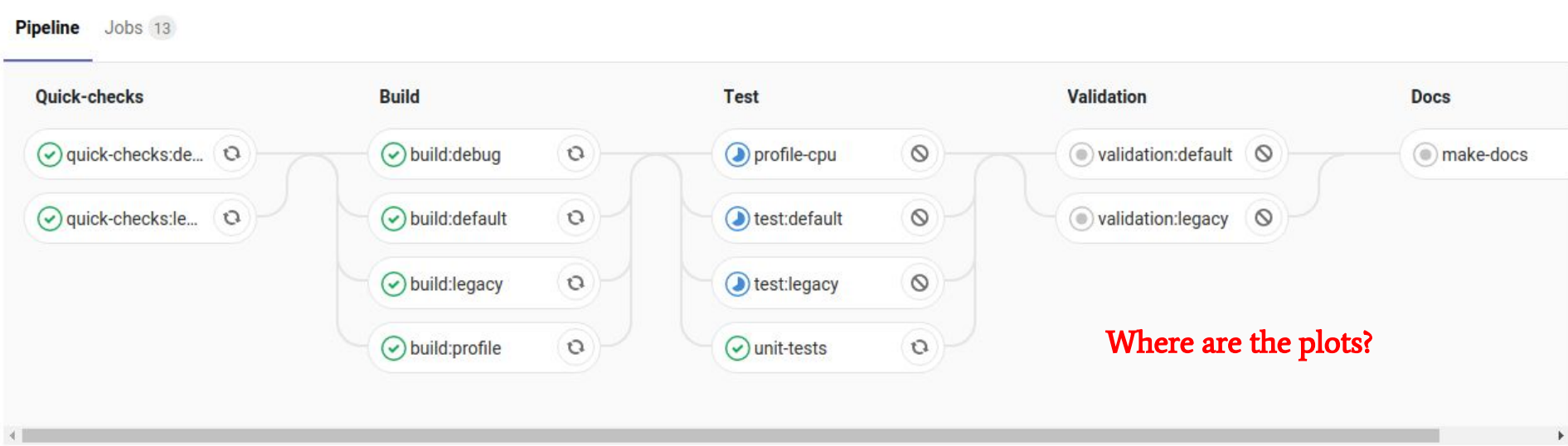
cmd	metric	value	ref value	diff
`MCTruth continuous_integration_101.root`	cpu_time_in_s (s)	187.11	57.67	129.44 (224.44 %) ↗
`MCTruth continuous_integration_101.root`	max_rss_in_MB (MB)	1296.70	1199.80	96.90 (8.08 %) ↗
`RootConverter continuous_integration_101.bin`	cpu_time_in_s (s)	4.95	2.65	2.30 (86.78 %) ↗
`RootConverter continuous_integration_101.bin`	max_rss_in_MB (MB)	330.70	229.30	101.40 (44.22 %) ↗
`make`	cpu_time_in_s (s)	463.70	387.52	76.17 (19.66 %) ↗
	ıax_rss_in_MB /IB)	561.20	605.40	-44.20 (-7.30 %) ↘
	ɒu_time_in_s (s)	524.51	286.20	238.31 (83.26 %) ↗
	ıax_rss_in_MB /IB)	1992.50	1727.20	265.30 (15.36 %) ↗

Disk size report

File	metric	value	ref value	diff
`continuous_integration_101.bin`	size_in_mb (MB)	81.00	39.60	41.40 (104.55 %) ↗
`continuous_integration_101.root`	size_in_mb (MB)	14.30	9.40	4.90 (52.13 %) ↗
`continuous_integration_101_mctruth.root`	size_in_mb (MB)	90.30	31.90	58.40 (183.07 %) ↗

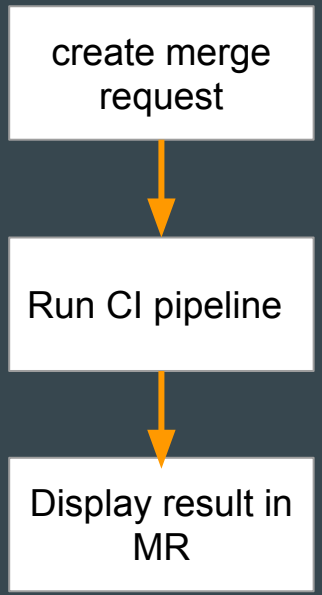
WIP: Making the results visible

It is insufficient to just compile useful information:



Especially newcomers have troubles finding the information

WIP: Making the results visible



For project maintainers:

- Create read-only API access token
  - For now via bot/service account (see GitLab issue #45149)
- Add it to CI variables
- Add `submit_report_to_mr` command to CI
- Report will appear as comment in MR

Summary

- Validation is an important ingredient for reliable scientific
- scikit-validate provides tools to ease the validation effort for individuals, small groups & experiments
  - Comparison of ROOT files, capturing metrics
  - Automatic reporting in Gitlab CI
- Features are still in “exploration mode” - what is useful sticks
  - Used in the Continuous Integration pipelines of the LZ experiment
  - Making validation configurable: definitions of ‘success’ and ‘failure’
- Try it today:

```
pip install scikit-validate
```

Backup slides

The focus on GitLab CI

- Available to both interest groups, currently CMS & LZ
- More flexible compared to free Github alternatives
  - E.g. adding own runners (to expose data, CVMFS, etc)
  - Full docker support
  - Rich configuration syntax
- Accessible API
- Artifact storage

Testing the tester

- Most features are tested in the CI pipeline of the project itself:
  - https://gitlab.cern.ch/fast-hep/public/scikit-validate/pipelines
- Failures also built into the system
  - 1 build, 1 test and 1 validation job always fails
  - Report is meant to be always run
-

```
report:
  stage: report
  when: always
```