RooWorkspace ⇄ JSON/YAML

Carsten Burgard

huge thanks to Nicolas Morange and Jonas Rembser for their help with getting this together!

special thanks also to the whole pyhf team as well as Jonas Eschle for valuable input

for the ROOT Users Workshop 2022

Disclaimer: This talk has an ATLAS bias!

Disclaimer: This talk draws some inspiration from pyhf!

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Introduction

Where this is coming from

- pyhf has been extremely successful in attracting users
- important reason (among others): ability to define models in a declarative language
  - pyhf JSON is readable, editable, and feature-complete!
- however, limited to HistFactory use-case
  - no “complicated” models, only stacks of homogeneously binned histograms in non-overlapping regions
- push towards publishing likelihoods is getting stronger
  - as far as ATLAS in concerned, so far mostly done by the SUSY group, using pyhf json
  - a declarative, software-independent format would be ideal
- HistFactory XML can also be used, but it’s not very user-friendly
  - does not incorporate all data, templates are stored externally
  - also cannot be obtained easily from existing workspace

a **round-trip-capable**, **human-readable declarative** format for statistical models was missing

Illustration: HistFactory XML

```
    <Sample Name="OtherH0jet"  HistoPath="HWWRun2GGF_histograms/Nominal/SR_bin1/"  HistoName="OtherH0jet"  InputFile="histFactory_tmp.root"  NormalizeByTheory="True" >
      <HistoSys Name="ATLAS_EG_RESOLUTION_ALL"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="HWWRun2GGF_hi
stograms/ATLAS_EG_RESOLUTION_ALL/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"  HistoPathHigh="HWWRun2GG
F_histograms/ATLAS_EG_RESOLUTION_ALL/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EG_SCALE_AF2"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="HWWRun2GGF_histogr
ams/ATLAS_EG_SCALE_AF2/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"  HistoPathHigh="HWWRun2GGF_histogra
ms/ATLAS_EG_SCALE_AF2/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EG_SCALE_ALL"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="HWWRun2GGF_histogr
ams/ATLAS_EG_SCALE_ALL/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"  HistoPathHigh="HWWRun2GGF_histogra
ms/ATLAS_EG_SCALE_ALL/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EL_EFF_ID_CorrUncertainty_NP0"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="H
WWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP0/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"
HistoPathHigh="HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP0/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EL_EFF_ID_CorrUncertainty_NP1"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="H
WWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP1/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"
HistoPathHigh="HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP1/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EL_EFF_ID_CorrUncertainty_NP10"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="
HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP10/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"
  HistoPathHigh="HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP10/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EL_EFF_ID_CorrUncertainty_NP11"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="
HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP11/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"
  HistoPathHigh="HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP11/Up/SR_bin1/"   />
      <HistoSys Name="ATLAS_EL_EFF_ID_CorrUncertainty_NP12"  HistoFileLow="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameLow="OtherH0jet"  HistoPathLow="
HWWRun2GGF_histograms/ATLAS_EL_EFF_ID_CorrUncertainty_NP12/Down/SR_bin1/"  HistoFileHigh="/home/cburgard/Physics/hww/HWWAnalysisCode/share/histFactory_tmp.root"  HistoNameHigh="OtherH0jet"
```

Illustration: RooFit tree printout

```
0x55a51b64d050/V- RooProduct::L_x_ggWW1jet_SR_bin3_overallSyst_x_StatUncert = 42.4593 [Auto,Clean]
  0x55a517dd99d0/V- RooConstVar::1 = 1
  0x55a51b64c680/V- RooProduct::ggWW1jet_SR_bin3_overallSyst_x_StatUncert = 42.4593 [Auto,Clean]
    0x55a5199ff5f0/V- ParamHistFunc::mc_stat_SR_bin3 = 1 [Auto,Clean]
      0x55a519a00920/V- RooRealVar::gamma_stat_SR_bin3_bin_0_HWWRun2GGF = 1 +/- 0.0233533
      0x55a519a04ab0/V- RooRealVar::gamma_stat_SR_bin3_bin_1_HWWRun2GGF = 1 +/- 0.0269288
      0x55a519a08b60/V- RooRealVar::gamma_stat_SR_bin3_bin_2_HWWRun2GGF = 1 +/- 0.0193654
      0x55a519a0cc10/V- RooRealVar::gamma_stat_SR_bin3_bin_3_HWWRun2GGF = 1 +/- 0.0175284
      0x55a519a10fd0/V- RooRealVar::gamma_stat_SR_bin3_bin_4_HWWRun2GGF = 1 +/- 0.0215774
      0x55a519a15390/V- RooRealVar::gamma_stat_SR_bin3_bin_5_HWWRun2GGF = 1 +/- 0.0168674
      0x55a519a19750/V- RooRealVar::gamma_stat_SR_bin3_bin_6_HWWRun2GGF = 1 +/- 0.0167513
      0x55a519a1db10/V- RooRealVar::gamma_stat_SR_bin3_bin_7_HWWRun2GGF = 1 +/- 0.00893387
      0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
    0x55a51b5d4a40/V- RooProduct::ggWW1jet_SR_bin3_overallSyst_x_HistSyst = 42.4593 [Auto,Clean]
      0x55a51b5d54b0/V- PiecewiseInterpolation::ggWW1jet_SR_bin3_Hist_alpha = 42.4593 [Auto,Clean]
        0x55a51b5d6560/V- RooHistFunc::ggWW1jet_SR_bin3_Hist_alphanominal = 42.4593 [Auto,Clean]
          0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
        0x55a51b5d9b20/V- RooHistFunc::ggWW1jet_SR_bin3_Hist_alpha_0low = 42.406 [Auto,Clean]
          0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
        0x55a51b5dd0a0/V- RooHistFunc::ggWW1jet_SR_bin3_Hist_alpha_1low = 42.4671 [Auto,Clean]
          0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
        0x55a51b5e0620/V- RooHistFunc::ggWW1jet_SR_bin3_Hist_alpha_2low = 42.8347 [Auto,Clean]
          0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
        0x55a51b5e3ba0/V- RooHistFunc::ggWW1jet_SR_bin3_Hist_alpha_3low = 42.7466 [Auto,Clean]
          0x55a5199fe0b0/V- RooRealVar::obs_x_SR_bin3 = 0.9375
```

- None of these are readable by anyone but experts for realistic scenarios
- XML does not incorporate shapes, which are defined in external files
- Tree printout only useful for inspection not editing / saving

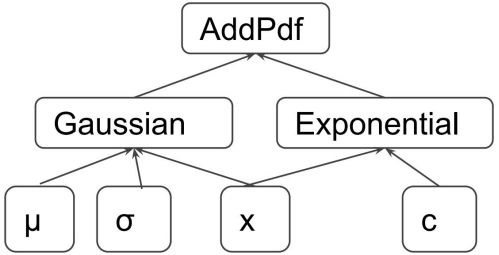
pyhf JSON

- Very specific use-case
  - *closely related to HistFactory*
- Human-readable for simple examples
  - *much more so than the XML version*


```
"channels": [
  {
    "name": "SR",
    "samples": [
      {
        "data": [
          43.20000076293945, 44.79999923706055, 43.79999923706055, 47.70000076293945, 46.79999923706055,
          45.5, 47.20000076293945, 48.20000076293945, 42.29999923706055, 43.400001525878906, 47.099998474121094
        ],
        "modifiers": [
          { "data": null, "name": "norm_bkg", "type": "normfactor" },
          { "name": "staterror_SR", "type": "staterror", "data": [
              1.475652380511632, 1.5036788161546268, 1.4875483617135017, 1.5517731741538603, 1.53855456525139,
              1.516838813006824, 1.5448462487286565, 1.560031983396898, 1.4622072690902237, 1.4799493422007224, 1.5408114680774188
            ], },
          { "data": null, "name": "lumi", "type": "lumi" }
        ],
        "name": "L_x_bkg_SR_overallSyst_x_StatUncert"
      },
      {
        "data": [
          3.509999990463257, 4.210000038146973, 4.559999942779541, 4.599999904632568, 5.429999828338623,
          5.159999847412109, 5.309999942779541, 4.739999771118164, 4.610000133514404, 4.050000190734863, 3.819999933242798
        ],
        "modifiers": [
          { "data": null, "name": "mu", "type": "normfactor" },
          { "name": "staterror_SR", "type": "staterror", "data": [
              1.475652380511632, 1.5036788161546268, 1.4875483617135017, 1.5517731741538603, 1.53855456525139, 1.516838813006824,
              1.5448462487286565, 1.560031983396898, 1.4622072690902237, 1.4799493422007224, 1.5408114680774188
            ], },
          { "data": null, "name": "lumi", "type": "lumi" }
        ],
        "name": "L_x_sig_SR_overallSyst_x_StatUncert"
      }
    ]
  }
],
```

Can we have the cake and eat it?

The two options at hand



- PDFs are just computational graphs, with nodes are identified by name
  →A completely generic, workspace-like syntax for JSON is possible
  - powerful: any workspace can be translated to and from JSON by virtue of
    - the RooWorkspace factory language
    - RooProxy introspection capabilities
  - bidirectional: converting the JSON back to a workspace yields the same workspace back
  - conceptually simple: just a text-version of the computational graph structure of the workspace itself
  - practically difficult: the JSON files might be huge and hard to read, write and edit
- HistFactory PDFs are (almost) trees
  → A simple, HistFactory JSON syntax is easy to achieve
  - simple: can just use what pyhf is already using and provide reader and writer for this format
  - usable: users can edit, read and write these files pretty easily (more easily than HistFactory XMLs)
  - unidirectional: once your JSON is a workspace, there's no turning back
  - limited: it's only HistFactory, no advanced features like acceptance corrections, morphing functions, …
- Can we have the best of both worlds, somehow?

Let’s make a generic format!

Transcript of a RooArgusBG example, arbitrarily selected analytical example

- few toplevel keys, all of dict-type - workspace-like
  - variables
  - pdfs
  - functions
  - data
- names of the items = names (id’s) of the objects
- variables have predefined properties
  - value, min, max, error, …
- pdfs & functions have one predefined key
  - type: string encoding the type of the pdf (unique!)
  - additional keys are possible
- everything has two additional keys for meta-information:
  - dict: storage for key-value pairs
  - tags: storage for list of strings

This approach allows 1:1 mapping of workspaces, but is not likely to stay readable for large workspaces!


```
"pdfs": {
    "background": {
        "mass": "mes",
        "power": "0.5",
        "resonance": "5.291",
        "slope": "argpar",
        "type": "ARGUS"
    },
    "model": {
        "coefficients": [
            "nsig",
            "nbkg"
        ],
        "dict": {
            "ModelConfig": "ModelConfig"
        },
        "summands": [
            "signal",
            "background"
        ],
        "tags": [
            "toplevel"
        ],
        "type": "pdfsum"
    },
    "signal": {
        "mean": "sigmean",
        "sigma": "sigwidth",
        "type": "Gaussian",
        "x": "mes"
    }
},
```

```
"variables": {
    "argpar": {
        "max": -1.0,
        "min": -100.0,
        "value": -20.0
    },
    "mes": {
        "max": 5.3,
        "min": 5.2,
        "value": 5.25
    },
    "nbkg": {
        "max": 10000.0,
        "min": 0.0,
        "value": 800.0
    },
    "nsig": {
        "max": 10000.0,
        "min": 0.0,
        "value": 200.0
    },
    "sigmean": {
        "max": 5.3,
        "min": 5.2,
        "value": 5.28
    },
    "sigwidth": {
        "max": 1.0,
        "min": 0.001,
        "value": 0.0027
    }
}
```

The remedy: PDF Macros

- Introduce macros as bidirectional mappings for specific common structures
  - HistFactory pdfs can map to a pyhf-like JSON structure
  - this JSON structure will again generate a histfactory-like PDF
- Can be used as a means to “patch” workspaces
  - If at some point in the future the default layout of HistFactory PDFs changes, this could be automatically applied to older workspaces by converting to and from JSON
- Allows to reap the benefits of having a concise representation of real-world workspaces with the need for generality


```
"pdfs": {
    "main": {
        "index": "channelCat",
        "type": "simultaneous",
        "tags": [
            "toplevel"
        ],
        "channels": {
            "channel1": {
                "type": "histfactory",
                "observables": {
                    "obs_x_channel1": {
                        "nbins": 2,
                        "min": 1,
                        "max": 2
                    }
                },
                "samples": {
                    "signal": {
                        "type": "hist-sample",
                        "overallSystematics": {
                            "syst1": {
                                "low": 0.95,
                                "high": 1.05
                            }
                        },
                        "normFactors": [
                            "mu"
                        ],
                        "statError": 0,
                        "data": {
                            "counts": [
                                20,
                                10
                            ]
                        }
                    },
                    "background1": {
                        ...
                    },
                    "background2": {
                        ...
                    }
                }
            }
        }
    }
},
```

```
"variables": {
    "mu": {
        "value": 1,
        "min": -3,
        "max": 5,
        "tags": [
            "poi"
        ]
    }
},
"data": {
    "observed": {
        "index": "channelCat",
        "channel1": {
            "observables": {
                "obs_x_channel1": {
                    "nbins": 2,
                    "min": 1,
                    "max": 2
                }
            },
            "counts": [
                122,
                112
            ]
        }
    }
}
```

The Implementation: RooJSONFactoryWSTool

- Factory class that manages import and export of functions, pdfs and variables
- Two-layer approach:
  - some pdfs and functions require code to serialize or deserialize
    - for those, you can write an “Importer” or “Exporter” and register it with the tool
    - the importers/exporters inherit from RooJSONFactoryWSTool and can be implemented in user code and registered with the tool
  - some do not need anything special, all you need to know is the mapping of keys in the JSON to the arguments in the class constructors (import) or the proxies (export)
    - for these, a prescription of how to import/export these objects can be given declaratively
- Both importers/exporters and Import/Export Expressions can be written by users and added/removed at runtime
- Extremely flexible, no need to be feature complete from day 1

Import Expressions

```
"Gaussian": {
    "class": "RooGaussian",
    "arguments": [
        "x",
        "mean",
        "sigma"
    ]
},
"Poisson": {
    "class": "RooPoisson",
    "arguments": [
        "x",
        "mean"
    ]
},
```

Export Expressions

```
'RooGaussian": {
    "type": "Gaussian",
    "proxies": {
        "x": "x",
        "mean": "mean",
        "sigma": "sigma"
    }
},
'RooPoisson": {
    "type": "Poisson",
    "proxies": {
        "x": "x",
        "mean": "mean"
    }
},
```

What is there already

This is sufficient for I/O of most HistFactory workspaces, plus a little bit more

- Importers exist for
  - RooBinSamplingPdf
  - RooBinWidthFunction
  - RooFormulaVar
  - RooGenericPdf
  - RooRealSumPdf
  - RooHistFunc
  - PiecewiseInterpolation
  - RooProdPdf
  - RooAddPdf
  - RooSimultaneous
  - RooRealSumPdf
- ImportExpressions exist for
  - RooGaussian
  - RooExponential
  - RooPoisson
  - RooProduct
  - FlexibleInterpVar
  - RooAddition
  - ParamHistFunc
  - RooArgusBG
- Exporters exist for
  - RooBinWidthFunction
  - RooProdPdf
  - RooProdPdf
  - RooSimultaneous
  - RooBinSamplingPdf
  - RooHistFunc
  - RooGenericPdf
  - RooFormulaVar
  - RooRealSumPdf
  - FlexibleInterpVar
  - PiecewiseInterpolation
- ExportExpressions exist for
  - RooGaussian
  - RooPoisson
  - RooExponential
  - RooProduct
  - RooProdPdf
  - ParamHistFunc
  - RooAddPdf
  - RooAddition
  - RooArgusBG

- List of available classes far from feature-complete
- Implemented in such a way that new imports/exporters can be added by users easily
- Procedure documented extensively in dedicated **README**
- Can dynamically generate documentation with the method `RooJSONWSFactoryTool::gendoc`
- More comprehensive declaration of the format & available PDF types **under development**

Is this useful?

- Many useful applications, predominantly
  - Convenience
    - creating simple fits from scratch & writing files by hand
    - producing non-HistFactory workspaces without having to interact with RooFit directly
    - cross-validating with other fitting frameworks (such as pyhf, zfit, …)
    - editing existing workspaces “by hand” for studies
  - Publication & Preservation
    - deposit and publish likelihoods in a way that is not tied to ROOT
- Whether it will actually be adopted by a broad user base, time will tell
  - great interest from the experimental HEP community already
  - will attempt to harmonize format as much as possible with other frameworks (pyhf, zfit)
  - eagerly awaiting adaptation from theory colleagues

ws2json

```
tool = ROOT.RooJSONFactoryWSTool(myworkspace)
tool.exportJSON(“myworkspace.json”)
```

json2ws

```
ws = ROOT.RooWorkspace(“somename”)
tool =
ROOT.RooJSONFactoryWSTool(ws)
tool.importJSON(“myworkspace.json”)
```

Summary & Conclusions

- ROOT 6.26 is able to read and write workspaces to JSON
- The implementation is “open-world” and intends to cover every possible use-case
  - not restricted to HistFactory-like workspaces
  - for such workspaces, there’s a straightforward mapping to pyhf JSON
- The library of importers & exporters is not feature-complete yet, but user-extendable
  - Current catalogue is sufficient for ATLAS Higgs combination workspaces
- This development has great potential in making our lives easier and improving the quality of the results we publish
- This is still in an early phase, and widespread acceptance is still pending
- Feedback is highly welcome!
- Let’s get the ball rolling!