FrequentistFramework
Code Walkthrough

Falk Bartels

TLA Statistics Meeting

09.02.2021

FrequentistFramework

- Where to get it?
  - Here: FrequentistFramework

- What changed?
  - Last week I merged my private branch nlofit into master

    → Many new files in the repository now
  - Updates not limited to the NLOfit only, also many additions for analytic fit
  - Existing files untouched, your setups should still work

What changed - Base directory

Name	Last commit	Last update
Input	storing binning tests in separate branch	8 months ago
config	- Added signal Sample to category file;	7 months ago
python	Plot fit from WS and from Bayesian, plus instructions	7 months ago
xmlAnaWSBuilder @ 47b76b10	add submodule	8 months ago
.gitignore	storing binning tests in separate branch	8 months ago
.gitmodules	add submodule	8 months ago
README.md	Update README.md	7 months ago

Name	Last commit		Last update
Input	Falk's branch nlofit into master		1 week ago
atlasstyle-00-04-02	Falk's branch nlofit into master	new	1 week ago
config	Falk's branch nlofit into master		1 week ago
python	Falk's branch nlofit into master		1 week ago
scripts	Falk's branch nlofit into master	new	1 week ago
quickFit @ 59078a5f	Falk's branch nlofit into master	new	1 week ago
xmlAnaWSBuilder @ 47b76b10	add submodule		8 months ago
.gitignore	Falk's branch nlofit into master		1 week ago
.gitmodules	Falk's branch nlofit into master		1 week ago
README.md	Falk's branch nlofit into master		1 week ago

- Included ATLAS Style package (unfortunately no git module)
- Included bash scripts to execute
- Included quickFit submodule

What changed - python

Name	Last commit	Last update
..		
PlotWorkspace	Plot fit from WS and from Bayesian, plus instructions	7 months ago
ExtractTH1FromWS.py	Simple script to attempt extraction of information from workspace	7 months ago
signal_injection.py	added signal recovery & significance	7 months ago
simple_analysis.py	new script for dedicated s+b fits	7 months ago
simple_bkg_fit.py	test fit script	8 months ago

- Mostly new plotting scripts
- Script to extract postFit and fitParameters
- Script to make pseudodata and inject Gaussian
- Rebinning scripts for NLOfit

Name	Last commit	Last update
..		
PlotWorkspace	Falk's branch nlofit into master	1 week ago
ExtractFitParameters.py	Falk's branch nlofit into master	1 week ago
ExtractPostfitFromWS.py	Falk's branch nlofit into master	1 week ago
ExtractTH1FromWS.py	Simple script to attempt extraction of information from works...	7 months ago
InjectGaussian.py	Falk's branch nlofit into master	1 week ago
PlotResiduals.py	Falk's branch nlofit into master	1 week ago
color.py	Falk's branch nlofit into master	1 week ago
createCoverageGraph.py	Falk's branch nlofit into master	1 week ago
createExtractionGraph.py	Falk's branch nlofit into master	1 week ago
createToleranceGraph.py	Falk's branch nlofit into master	1 week ago
generatePseudoData.py	Falk's branch nlofit into master	1 week ago
plotFalseExclusion.py	Falk's branch nlofit into master	1 week ago
plotFalseExclusionCandles.py	Falk's branch nlofit into master	1 week ago
plotLimits.py	Falk's branch nlofit into master	1 week ago
plotLimits_joined.py	Falk's branch nlofit into master	1 week ago
rebin.py	Falk's branch nlofit into master	1 week ago
signal_injection.py	added signal recovery & significance	7 months ago
simple_analysis.py	new script for dedicated s+b fits	7 months ago
simple_bkg_fit.py	test fit script	8 months ago
unifyBinning.py	Falk's branch nlofit into master	1 week ago

What changed – config/dijetTLA

Name	Last commit	Last update
..		
20200630_j75_bkgonly_test	Bkg only (signal and spurious samples removed from c...	7 months ago
20200701_j75_bkgonly_test	- Added signal Sample to category file;	7 months ago
AnaWSBuilder.dtd	preliminary TLA workspaces	8 months ago
background_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
background_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
category_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
category_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
signal_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
signal_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago

Name	Last commit	Last update
..		
20200630_j75_bkgonly_test	Bkg only (signal and spurious samples removed from c...	7 months ago
20200701_j75_bkgonly_test	- Added signal Sample to category file;	7 months ago
signal	Falk´s branch nlofit into master	1 week ago
AnaWSBuilder.dtd	preliminary TLA workspaces	8 months ago
background_dijetTLA_J100yStar06_UA2.xml	Falk´s branch nlofit into master	1 week ago
background_dijetTLA_J100yStar06_fivePar...	Falk´s branch nlofit into master	1 week ago
background_dijetTLA_J100yStar06_fourPa...	Falk´s branch nlofit into master	1 week ago
background_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
background_dijetTLA_J75yStar03_fivePar....	Falk´s branch nlofit into master	1 week ago
background_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
category_dijetTLA_J100yStar06_fivePar.te...	Falk´s branch nlofit into master	1 week ago
category_dijetTLA_J100yStar06_fourPar.te...	Falk´s branch nlofit into master	1 week ago
category_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
category_dijetTLA_J75yStar03.xml	Falk´s branch nlofit into master	1 week ago
category_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
dijetTLA_J100yStar06.template	Falk´s branch nlofit into master	1 week ago
dijetTLA_J100yStar06.xml	Falk´s branch nlofit into master	1 week ago
dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
dijetTLA_J75yStar03.template	Falk´s branch nlofit into master	1 week ago
dijetTLA_J75yStar03.xml	Falk´s branch nlofit into master	1 week ago
dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago
signal_dijetTLA_J100yStar06_v01.xml	preliminary TLA workspaces	8 months ago
signal_dijetTLA_J75yStar03_v01.xml	Configurations to read in J75 signal	7 months ago

- Old files: ending with _v01
- Everything else is new

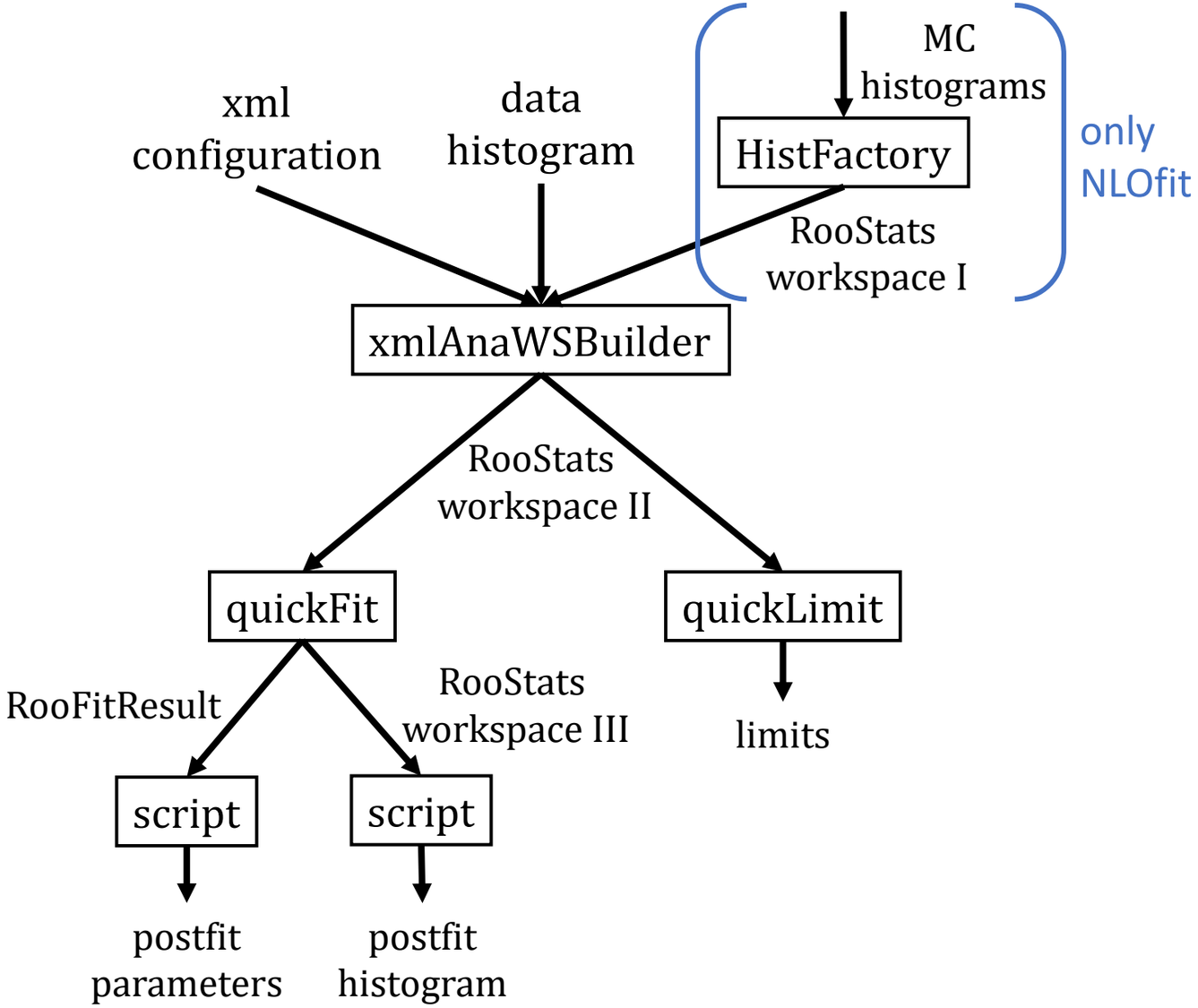
What changed – scripts

Name	Last commit	Last update
..		
run_buildAndFit_loop_nlofit.sh	Falk's branch nlofit into master	1 week ago
run_buildAndFit_loop_swift.sh	Falk's branch nlofit into master	1 week ago
run_buildAndFit_nlofit.sh	Falk's branch nlofit into master	1 week ago
run_buildAndFit_swift.sh	Falk's branch nlofit into master	1 week ago
setup_buildAndFit.sh	Falk's branch nlofit into master	1 week ago

- Bash script to setup xmlAnaWSBuilder and quickFit
- Bash scripts to build workspace, fit and set limits

How to Run

- Initial setup hopefully understandable in README
- We need to provide xmlAnaWSBuilder config files and data histogram
- Rest is out-of-the-box application of xmlAnaWSBuilder & quickFit
- HistFactory path only necessary for NLOfit, not yet in this repository



xmlAnaWSBuilder – Top-Level XML

- One top-level xml per analysis:
  - Include one category xml per channel
  - Specify all parameters of interest

```
<!DOCTYPE Combination  SYSTEM 'AnaWSBuilder.dtd'>
<Combination WorkspaceName="combWS" ModelConfigName="ModelConfig" DataName="combData" OutputFile="workspace/dijetTLA/dijetTLA_J100yStar06.root">
  <Input>config/dijetTLA/category_dijetTLA_J100yStar06.xml</Input>
  <POI>nsig_mean450_width5,nsig_mean450_width7,nsig_mean450_width10,nsig_mean450_width12,nsig_mean450_width15,nsig_mean500_width5,nsig_mean500_width7,nsig_mean500_width10,nsig_mean500_width12,
nsig_mean500_width15,nsig_mean550_width5,nsig_mean550_width7,nsig_mean550_width10,nsig_mean550_width12,nsig_mean550_width15,nsig_mean600_width5,nsig_mean600_width7,nsig_mean600_width10,nsig_mean600_width12,
nsig_mean600_width15,nsig_mean650_width5,nsig_mean650_width7,nsig_mean650_width10,nsig_mean650_width12,nsig_mean650_width15,nsig_mean700_width5,nsig_mean700_width7,nsig_mean700_width10,nsig_mean700_width12,
nsig_mean700_width15,nsig_mean750_width5,nsig_mean750_width7,nsig_mean750_width10,nsig_mean750_width12,nsig_mean750_width15,nsig_mean800_width5,nsig_mean800_width7,nsig_mean800_width10,nsig_mean800_width12,
nsig_mean800_width15,nsig_mean850_width5,nsig_mean850_width7,nsig_mean850_width10,nsig_mean850_width12,nsig_mean850_width15,nsig_mean900_width5,nsig_mean900_width7,nsig_mean900_width10,nsig_mean900_width12,
nsig_mean900_width15,nsig_mean950_width5,nsig_mean950_width7,nsig_mean950_width10,nsig_mean950_width12,nsig_mean950_width15,nsig_mean1000_width5,nsig_mean1000_width7,nsig_mean1000_width10,nsig_mean1000_width12,
nsig_mean1000_width15,nsig_mean1050_width5,nsig_mean1050_width7,nsig_mean1050_width10,nsig_mean1050_width12,nsig_mean1050_width15,nsig_mean1100_width5,nsig_mean1100_width7,nsig_mean1100_width10,
nsig_mean1100_width12,nsig_mean1100_width15,nsig_mean1150_width5,nsig_mean1150_width7,nsig_mean1150_width10,nsig_mean1150_width12,nsig_mean1150_width15,nsig_mean1200_width5,nsig_mean1200_width7,
nsig_mean1200_width10,nsig_mean1200_width12,nsig_mean1200_width15,nsig_mean1300_width5,nsig_mean1300_width7,nsig_mean1300_width10,nsig_mean1300_width12,nsig_mean1300_width15,nsig_mean1400_width5,
nsig_mean1400_width7,nsig_mean1400_width10,nsig_mean1400_width12,nsig_mean1400_width15,nsig_mean1500_width5,nsig_mean1500_width7,nsig_mean1500_width10,nsig_mean1500_width12,nsig_mean1500_width15,
nsig_mean1600_width5,nsig_mean1600_width7,nsig_mean1600_width10,nsig_mean1600_width12,nsig_mean1600_width15,nsig_mean1700_width5,nsig_mean1700_width7,nsig_mean1700_width10,nsig_mean1700_width12,
nsig_mean1700_width15,nsig_mean1800_width5,nsig_mean1800_width7,nsig_mean1800_width10,nsig_mean1800_width12,nsig_mean1800_width15</POI>
  <Asimov Name="POISnap"  Setup=""  Action="savesnapshot" SnapshotPOI="nominalPOI"/>
  <Asimov Name="NPSnap"   Setup="nsig_mean450_width5=0,nsig_mean450_width7=0,nsig_mean450_width10=0,nsig_mean450_width12=0,nsig_mean450_width15=0,nsig_mean500_width5=0,nsig_mean500_width7=0,
nsig_mean500_width10=0,nsig_mean500_width12=0,nsig_mean500_width15=0,nsig_mean550_width5=0,nsig_mean550_width7=0,nsig_mean550_width10=0,nsig_mean550_width12=0,nsig_mean550_width15=0,nsig_mean600_width5=0,
nsig_mean600_width7=0,nsig_mean600_width10=0,nsig_mean600_width12=0,nsig_mean600_width15=0,nsig_mean650_width5=0,nsig_mean650_width7=0,nsig_mean650_width10=0,nsig_mean650_width12=0,nsig_mean650_width15=0,
nsig_mean700_width5=0,nsig_mean700_width7=0,nsig_mean700_width10=0,nsig_mean700_width12=0,nsig_mean700_width15=0,nsig_mean750_width5=0,nsig_mean750_width7=0,nsig_mean750_width10=0,nsig_mean750_width12=0,
nsig_mean750_width15=0,nsig_mean800_width5=0,nsig_mean800_width7=0,nsig_mean800_width10=0,nsig_mean800_width12=0,nsig_mean800_width15=0,nsig_mean850_width5=0,nsig_mean850_width7=0,nsig_mean850_width10=0,
nsig_mean850_width12=0,nsig_mean850_width15=0,nsig_mean900_width5=0,nsig_mean900_width7=0,nsig_mean900_width10=0,nsig_mean900_width12=0,nsig_mean900_width15=0,nsig_mean950_width5=0,nsig_mean950_width7=0,
nsig_mean950_width10=0,nsig_mean950_width12=0,nsig_mean950_width15=0,nsig_mean1000_width5=0,nsig_mean1000_width7=0,nsig_mean1000_width10=0,nsig_mean1000_width12=0,nsig_mean1000_width15=0,nsig_mean1050_width5=0,
nsig_mean1050_width7=0,nsig_mean1050_width10=0,nsig_mean1050_width12=0,nsig_mean1050_width15=0,nsig_mean1100_width5=0,nsig_mean1100_width7=0,nsig_mean1100_width10=0,nsig_mean1100_width12=0,
nsig_mean1100_width15=0,nsig_mean1150_width5=0,nsig_mean1150_width7=0,nsig_mean1150_width10=0,nsig_mean1150_width12=0,nsig_mean1150_width15=0,nsig_mean1200_width5=0,nsig_mean1200_width7=0,
nsig_mean1200_width10=0,nsig_mean1200_width12=0,nsig_mean1200_width15=0,nsig_mean1300_width5=0,nsig_mean1300_width7=0,nsig_mean1300_width10=0,nsig_mean1300_width12=0,nsig_mean1300_width15=0,
nsig_mean1400_width5=0,nsig_mean1400_width7=0,nsig_mean1400_width10=0,nsig_mean1400_width12=0,nsig_mean1400_width15=0,nsig_mean1500_width5=0,nsig_mean1500_width7=0,nsig_mean1500_width10=0,
nsig_mean1500_width12=0,nsig_mean1500_width15=0,nsig_mean1600_width5=0,nsig_mean1600_width7=0,nsig_mean1600_width10=0,nsig_mean1600_width12=0,nsig_mean1600_width15=0,nsig_mean1700_width5=0,
nsig_mean1700_width7=0,nsig_mean1700_width10=0,nsig_mean1700_width12=0,nsig_mean1700_width15=0,nsig_mean1800_width5=0,nsig_mean1800_width7=0,nsig_mean1800_width10=0,nsig_mean1800_width12=0,
nsig_mean1800_width15=0"      Action="fixsyst:fit:float:savesnapshot:nominalPOI" SnapshotNuis="nominalNuis" SnapshotGlob="nominalGlobs"/>
</Combination>
```

xmlAnaWSBuilder – Category XML

- One category xml per channel – for us: 1
  - 1 <Data> specifier for input histogram
  - 1 <Sample> specifier for background xml
  - Many <Sample> specifiers for all the signal xml‘s
  - Normalisation: Input samples normalized to 1, scaled by Normfactor (x Lumi)

```
<!DOCTYPE Channel SYSTEM 'AnaWSBuilder.dtd'>
<Channel Name="J75yStar03" Type="shape" Lumi="1">
  <Data InputFile="Input/data/dijetTLA/lookInsideTheBoxWithUniformMjj.root" FileType="histogram" HistName="Nominal/DSJ75yStar03_TriggerJets_J75_yStar03_mjj_finebinned_all_data"
Observable="obs_x_channel[400,2079]" Binning="1679" InjectGhost="1"/>

  <Sample Name="background" InputFile="config/dijetTLA/background_dijetTLA_J75yStar03_fivePar.xml" MultiplyLumi="0" ImportSyst=":self:">
    <NormFactor Name="nbkg[4.1E7,0,6E7]"/>
  </Sample>

  <Sample Name="signal_mean450_width5" InputFile="config/dijetTLA/signal/signal_dijetTLA_J75yStar03_mean450_width5.xml" MultiplyLumi="0" >
    <NormFactor Name="nsig_mean450_width5[0,0,1E6]" />
  </Sample>
  <Sample Name="signal_mean450_width7" InputFile="config/dijetTLA/signal/signal_dijetTLA_J75yStar03_mean450_width7.xml" MultiplyLumi="0" >
    <NormFactor Name="nsig_mean450_width7[0,0,1E6]" />
  </Sample>
  <Sample Name="signal_mean450_width10" InputFile="config/dijetTLA/signal/signal_dijetTLA_J75yStar03_mean450_width10.xml" MultiplyLumi="0" >
    <NormFactor Name="nsig_mean450_width10[0,0,1E6]" />
  </Sample>
```

⋮

xmlAnaWSBuilder – Background XML

- One background xml – for us: 1
  - <Model> specifies fit function and parameter start values / limits

```
<!DOCTYPE Model  SYSTEM 'AnaWSBuilder.dtd'>
<Model Type="UserDef">
  <!-- <ModelItem Name="EXPR::background('@1*TMath::Power(1-@0/13000.,@2)*TMath::Power(@0/13000, -1*(@3*@0/13000. + @4*TMath::Log(@0/13000.) + @5*TMath::Power(TMath::Log(@0/13000),2.))
)', :observable:, p1[1,1,1], p2[-0.22, -100, 100], p3[9.15, -100, 100], p4[0.694, -100, 100], p5[0.22,-100,100])"/> -->
  <!-- <ModelItem Name="EXPR::background('@1*TMath::Power(1-@0/13000.,10*@2)*TMath::Power(@0/13000, -1*(@3*@0/13000. + @4*TMath::Log(@0/13000.) + 0.1*@5*TMath::Power(TMath::Log(@0/
13000),2.)))', :observable:, p1[1,1,1], p2[1.421, -30, 30], p3[-3.43, -30, 30], p4[-1.41, -10, 10], p5[-1.5,-10,10])"/> -->
  <ModelItem Name="EXPR::background('@1*TMath::Power(1-@0/13000.,@2)*TMath::Power(@0/13000, -1*(@3 + @4*TMath::Log(@0/13000.) + @5*TMath::Power(TMath::Log(@0/13000),2.)))',
:observable:, p1[1,1,1], p2[11.6, -30, 30], p3[2.2, -30, 30], p4[-1.0, -10, 10], p5[-0.1,-10,10])"/>
</Model>
```

  - For NLOfit: <Model> reads external HistFactory histograms for each systematic (generation scripts not yet in this repo)

```
<!DOCTYPE Model SYSTEM 'AnaWSBuilder.dtd'>
<Model Type="External" Input="Input/model/dijetTLAnlo/LO_CT14nnlo_reducedNPs_scaledOnly_reweightedNLO/HistFactory_dijetTLAnlo_J100yStar06_bkg_ws.root" WSName="combined"
ModelName="J100yStar06_model" ObservableName="obs_x_J100yStar06">
  <Item Name="unit[1]"/>
  <Rename OldName="Lumi" NewName="unit"/>

  <ExtSyst ConstrName="alpha_var_alpha1Constraint" NPName="alpha_var_alpha1" GOName="nom_alpha_var_alpha1" />
  <ExtSyst ConstrName="alpha_var_scale1Constraint" NPName="alpha_var_scale1" GOName="nom_alpha_var_scale1" />
  <ExtSyst ConstrName="alpha_var_pdf1Constraint" NPName="alpha_var_pdf1" GOName="nom_alpha_var_pdf1" />
```

⋮

xmlAnaWSBuilder – Signal XML

- One signal xml per mass and width, e.g. mass 1000, width 5%:
  - Syntax identical to bkg xml

```
<!DOCTYPE Model  SYSTEM 'AnaWSBuilder.dtd'>
<Model Type="UserDef">
  <Item Name="mass_1000[1000]"/>
  <Item Name="prod::width_1000_5(mass_1000,0.05)"/>
  <ModelItem Name="RooGaussian::signal(:observable:, mass_1000, width_1000_5)"/>
</Model>
```

  - For NLOfit: Again reading Gaussian signal histograms from HistFactory output files

```
<!DOCTYPE Model SYSTEM 'AnaWSBuilder.dtd'>
<Model Type="External" Input="Input/model/dijetTLAnlo/LO_CT14nnlo_reducedNPs_scaledOnly_reweightedNLO/signal/HistFactory_dijetTLAnlo_J100yStar06_sig_mean1000_width5_ws.root"
WSName="combined" ModelName="J100yStar06_model" ObservableName="obs_x_J100yStar06">
  <Item Name="unit[1]"/>
  <Rename OldName="Lumi" NewName="unit"/>
</Model>
```

xmlAnaWSBuilder

- Execute xmlAnaWSBuilder by specifying top-level xml:

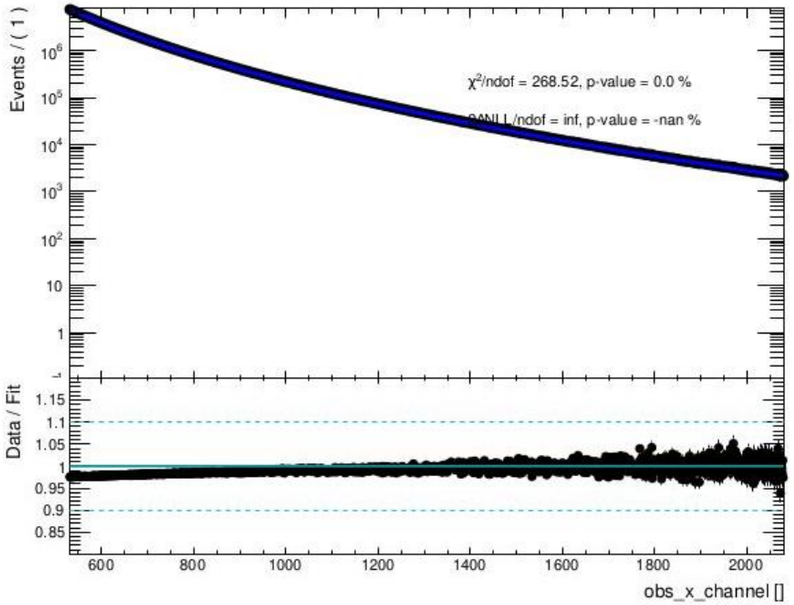
```
./xmlAnaWSBuilder/exe/XMLReader -x config/dijetTLA/dijetTLA_J75yStar03.xml -s 0 --plotOption logy
```

Fast minimization
(we refit anyways)

- xmlAnaWSBuilder could also fit, but currently disabled in my top-level xml files

  to speed up / avoid crashes at divergence

- Output:
  - 1 sanity check pdf file
  - 1 root file containing RooWorkspace



quickFit

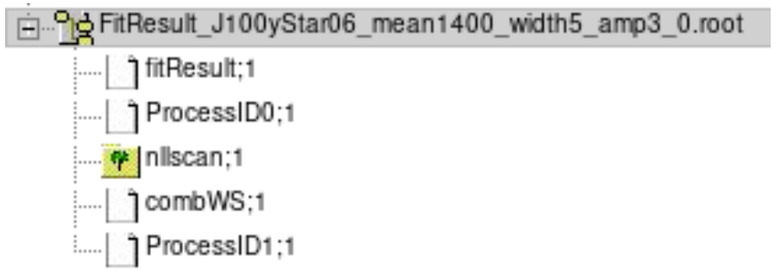
- We now run quickFit on the RooWorkspace file:

Workspace name | Sanity check on input | Save FitResult object to output | Save nuisance parameters in RooWorkspace | Output file

```
quickFit -f ${wsfile} -d combData -p $PARS --checkWS 1 --hesse 1 --savefitresult 1 --saveWS 1 --saveNP 1 --saveErrors 1 -o ${outputfile}
```

Input file | parameter of interest to float | Hessian error calculation | Save RooWorkspace to output | Save errors to FitResult

- Output: 1 root file containing
  - RooFitResult
  - RooWorkspace
  - leaves with final fit parameters (no errors?)
  - <u>no</u> postfit distribution!



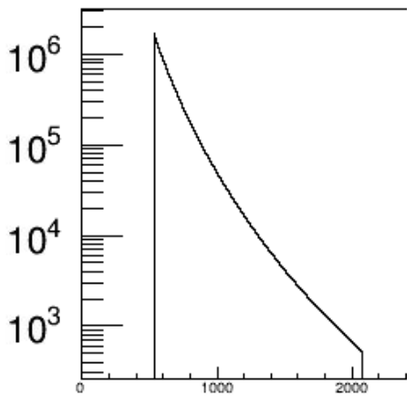
quickFit – Handling the Output

- I added 2 python scripts to deal with quickFit output:

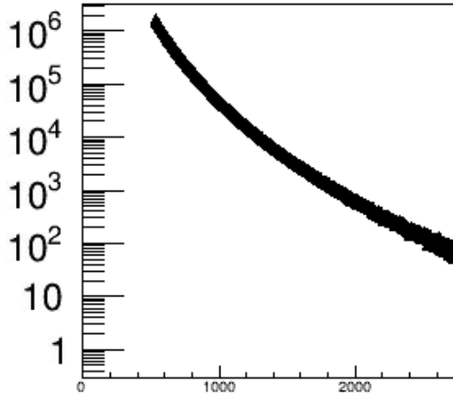
```
python python/ExtractPostfitFromWS.py --datafile $datafile --datahist $datahist --datafirstbin $rangelow --wsfile ${outputfile} --outfile ${outputfile/FitResult/PostFit} || true
```

  - Extracts postfit distribution from RooWorkspace
  - Needs reference data histogram and start bin because binning information not (?) stored in RooStats::ModelConfig
  - Maybe bit inelegant…
  - Produces 3 histograms:

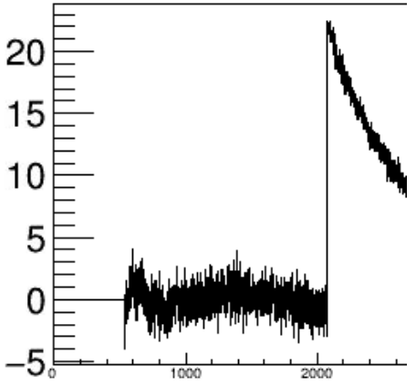
postfit



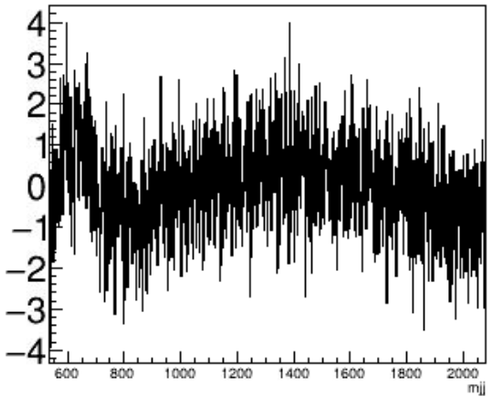
data



(postfit – data) / data error



only sensible within fit range

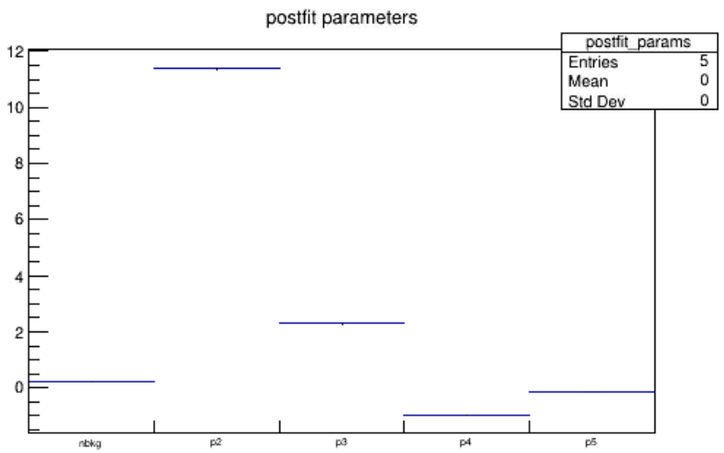


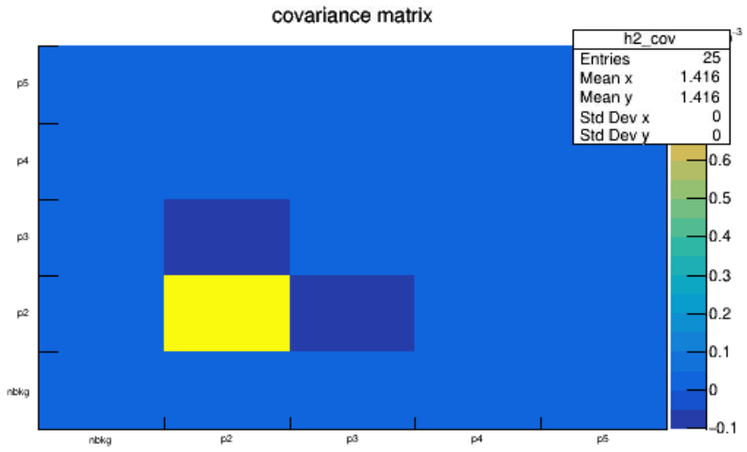
quickFit – Handling the Output

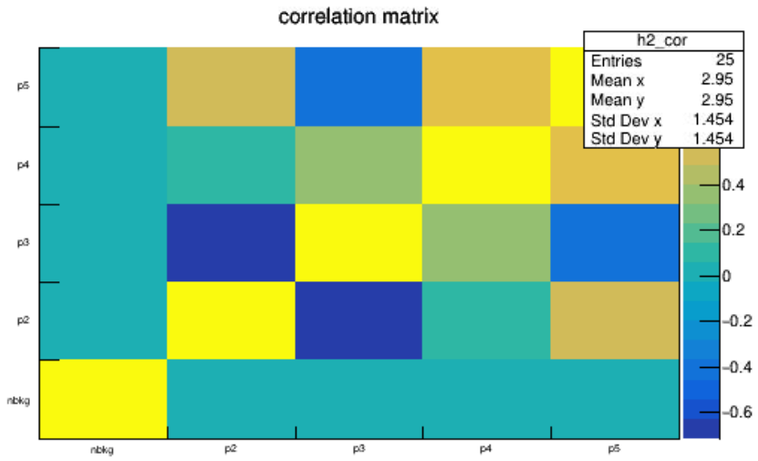
- I added 2 python scripts to deal with quickFit output:

```
python python/ExtractFitParameters.py --wsfile ${outputfile} --outfile ${outputfile/FitResult/FitParameters}
```

  - Extracts postfit parameters & uncertainties from RooFitResult
  - Produces 3 histograms:







quickLimit

- Can run quickLimit on the same RooWorkspace as quickFit:

Workspace name

Sanity check on input

Need approximate guess for limit

```
quickLimit -f ${wsfile} -d combData -p $PARS --checkWS 1 --hesse 1 --initialGuess 100000 -o ${outputfile/FitResult/Limits}
```

Input file

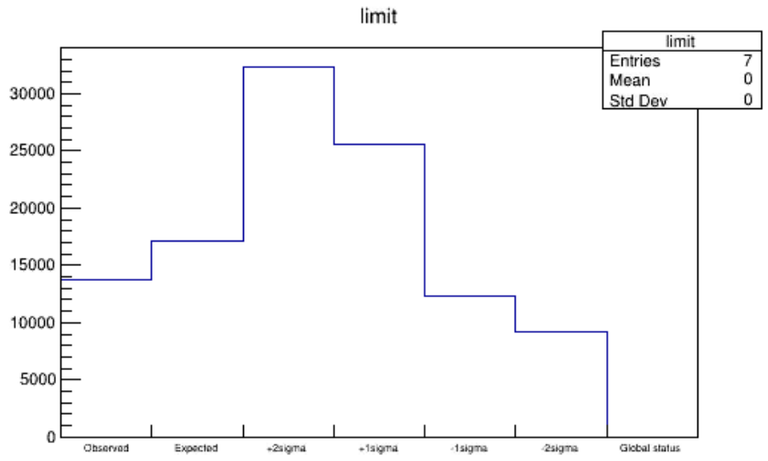
parameter of interest to float

Hessian error calculation

Output file

- Output:
  - 1 txt file
  - 1 root file
  - both same contents

```
13759.9 17177 32374.4 25600.1 12377 9219.33
Limits_J100yStar06_mean1000_width5_amp0_0.txt lines 1-1/1 (END)
```



Automating things

- Setting up the environment via `scripts/setup_buildAndFit.sh`

- Run xmlAnaWSBuilder, quickFit, output extraction, quickLimit in sequence via
  - `scripts/run_buildAndFit_swift.sh` (configured for global analytic fit right now)
  - `scripts/run_buildAndFit_nlofit.sh`

- One would need to modify xmlAnaWSBuilder config xml‘s every time when changing data hist, fit range, fit function, expected #bkg events, …

Automating things

- I use .template files instead: identical to original xml configs, but using placeholder strings

```
<!DOCTYPE Channel SYSTEM 'AnaWSBuilder.dtd'>
<Channel Name="J100yStar06" Type="shape" Lumi="1">
  <Data InputFile="DATAFILE" FileType="histogram" HistName="DATAHIST" Observable="obs_x_channel[RANGELOW,RANGEHIGH]" Binning="BINS" InjectGhost="1"/>

  <!-- <Sample Name="background" InputFile="config/dijetTLA/background_dijetTLA_J100yStar06_fourPar.xml" MultiplyLumi="0" ImportSyst=":self:"> -->
  <Sample Name="background" InputFile="config/dijetTLA/background_dijetTLA_J100yStar06_fivePar.xml" MultiplyLumi="0" ImportSyst=":self:">
    <!-- 29/fb: -->
    <!-- <NormFactor Name="nbkg[0.2E8,0,0.3E8]"/> -->
    <!-- 130/fb: -->
    <!-- <NormFactor Name="nbkg[9E8,0,15E8]"/>  -->
    <NormFactor Name="nbkg[NBKG]"/>
  </Sample>

  <Sample Name="signal_mean450_width5" InputFile="config/dijetTLA/signal/signal_dijetTLA_J100yStar06_mean450_width5.xml" MultiplyLumi="0" >
    <NormFactor Name="nsig_mean450_width5[0,0,1E6]" />
  </Sample>
  <Sample Name="signal_mean450_width7" InputFile="config/dijetTLA/signal/signal_dijetTLA_J100yStar06_mean450_width7.xml" MultiplyLumi="0" >
    <NormFactor Name="nsig_mean450_width7[0,0,1E6]" />
  </Sample>
```

⋮

- These placeholders are then replaced

  in run_buildAndFit_swift.sh:

```
cp ${categoryfile} ${tmpcategoryfile}
sed -i "s@DATAFILE@${datafile}@g" ${tmpcategoryfile} #@ because datafile contains /
sed -i "s@DATAHIST@${datahist}@g" ${tmpcategoryfile}
sed -i "s@RANGELOW@${rangelow}@g" ${tmpcategoryfile}
sed -i "s@RANGEHIGH@${rangehigh}@g" ${tmpcategoryfile}
sed -i "s@BINS@${bins}@g" ${tmpcategoryfile}

cp ${topfile} ${tmptopfile}
sed -i "s@CATEGORYFILE@${tmpcategoryfile}@g" ${tmptopfile}
sed -i "s@OUTPUTFILE@${wsfile}@g" ${tmptopfile}
```

Automating things

- Default values for these replacements set in run_buildAndFit_swift.sh, but they can be changed by setting env variables:

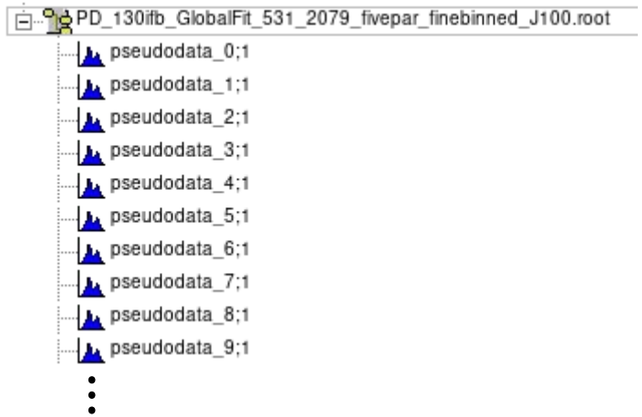
```
if [[ -z $wsfile ]]; then
    wsfile=run/dijetTLAnlo_combWS_swift.root
fi
if [[ -z $sigmean ]]; then
    sigmean=1200
fi
if [[ -z $sigwidth ]]; then
    sigwidth=7
fi
if [[ -z $sigfit ]]; then
    sigfit=false
fi
⋮
```

- One can then execute in a loop over signal means / widths:

```
sigmeans=( 450 500 550 600 650 700 750 800 850 900 950 1000 1050 1100 1150 1200 1300 1400 1500\
 1600 1700 1800 )
sigwidths=( 5 7 10 12 15 )
datafile=Input/data/dijetTLA/lookInsideTheBoxWithUniformMjj.root
datahist=Nominal/DSJ100yStar06_TriggerJets_J100_yStar06_mjj_finebinned_all_data
categoryfile=config/dijetTLA/category_dijetTLA_J100yStar06.template
topfile=config/dijetTLA/dijetTLA_J100yStar06.template

for sigmean in "${sigmeans[@]}"
do
    for sigwidth in "${sigwidths[@]}"
    do
        outputfile=run/FitResult_J100yStar06_mean${sigmean}_width${sigwidth}.root

        env datafile=$datafile datahist=$datahist categoryfile=$categoryfile topfile=$topfile \
sigmean=$sigmean sigwidth=$sigwidth outputfile=$outputfile ./scripts/run_buildAndFit_swift.sh
    done
done
```

Pseudodata

- I've used python/generatePseudoData
  - It reads a postfit histogram, scales by some factor and generates N replicas from that
  - can probably use Juno's PseudoDataGenerator by now



- Process the pseudodata with
  - scripts/run_buildAndFit_swift_loop.sh
  - scripts/run_buildAndFit_nlofit_loop.sh

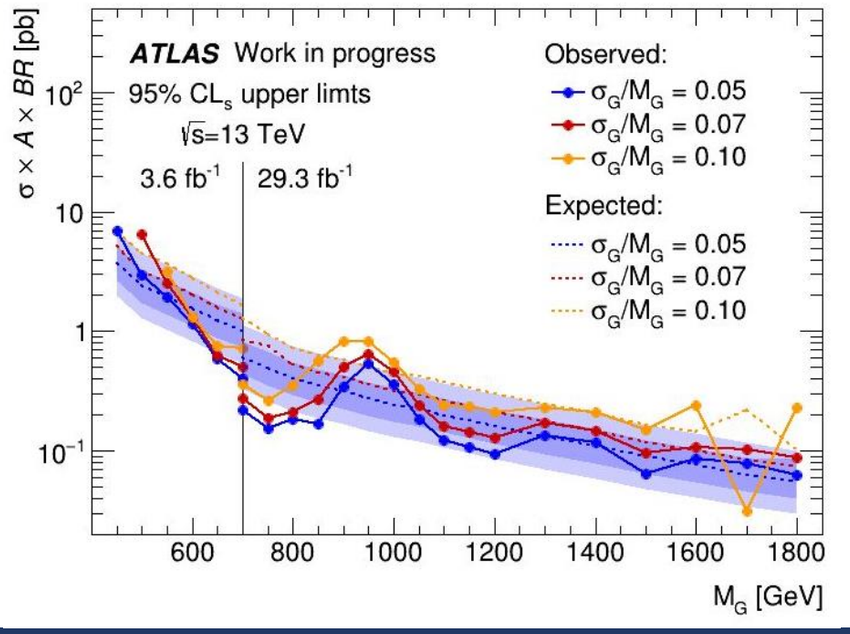
→ identical to previous scripts + 1 hardcoded {0..49} loop right now

Plotting Scripts

- plotLimits_joined.py
  - Reads quickLimit output for different masses / widths for J75, J100

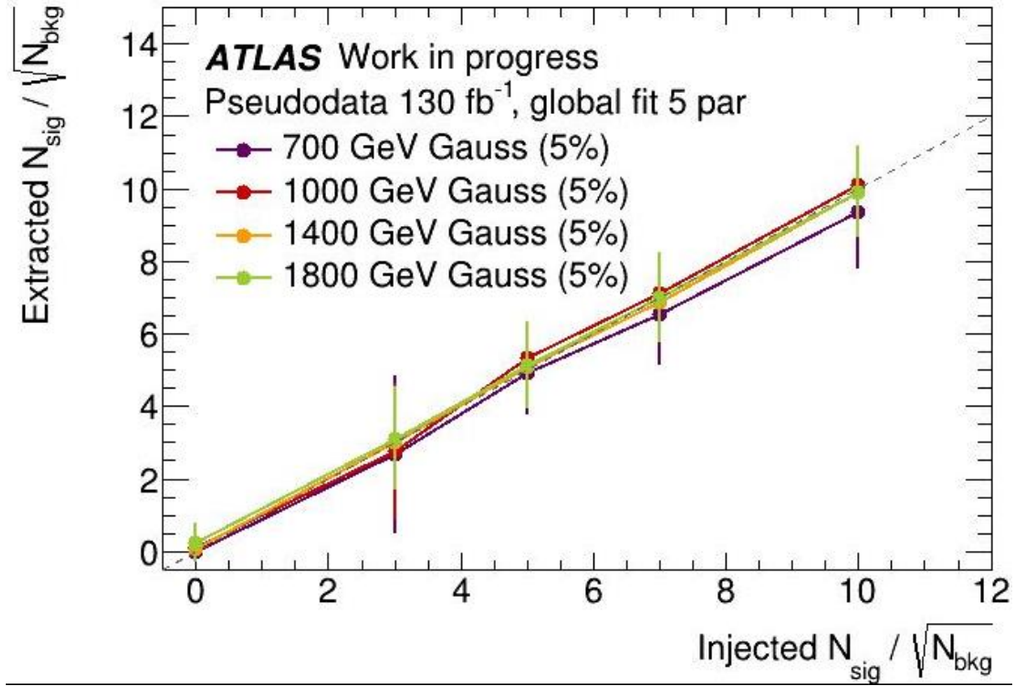
```
paths = [ "../run/Limits/swift_fivepar/Limits_J75yStar03_mean${MEAN}_width${WIDTH}.root",
          "../run/Limits/swift_fivepar/Limits_J100yStar06_mean${MEAN}_width${WIDTH}.root", ]


sigmeans  = [ [ 450, 500, 550, 600, 650, 700, ],
              [ 700, 750, 800, 850, 900, 950, 1000, 1050, 1100, 1150, 1200, 1300, 1400, 1500, 1600, 1700, 1800, ] ]
sigwidths = [ [ 5, 7, 10, ],
              [ 5, 7, 10, ] ]

lumis = [ 3600, 29300 ]
```

  - Produces this:



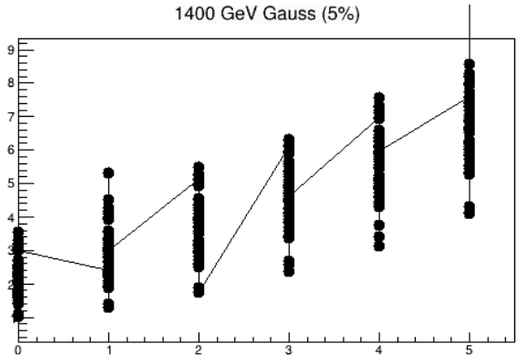
Plotting Scripts

- createExtractionGraph.py
  - Reads quickFit output for different masses, widths, signal amplitudes
  - Produces this:

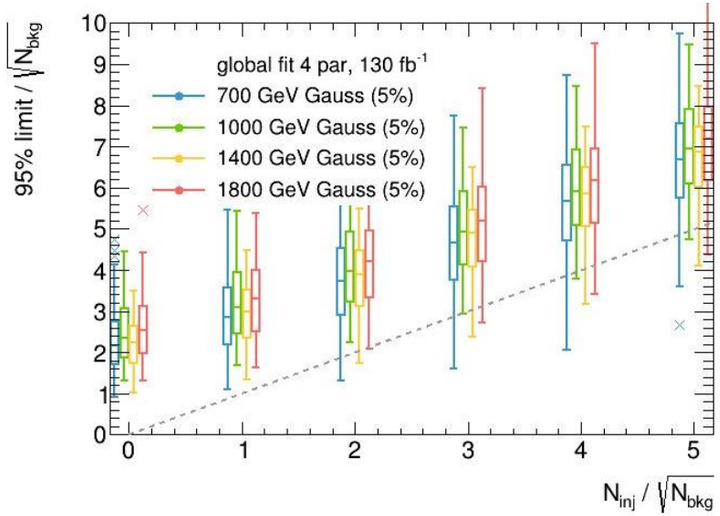


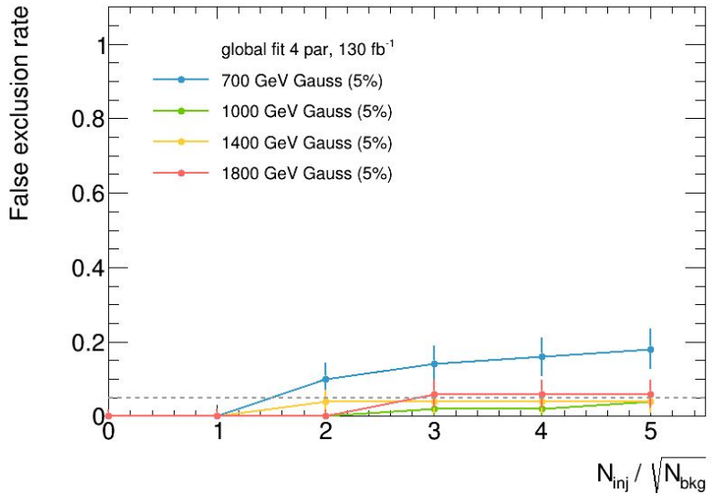
Plotting Scripts

- createCoverageGraph.py
  - Reads quickLimit output for different masses, widths, signal amplitudes
  - Produces TGraphs of upper limit vs injection:



- plotFalseExclusionCandles.py
  - Reads these TGraphs
  - Produces candleplots & false exclusion rate:





Closing Remarks

- If you have questions or encounter problems, please ask!

  (Might not be your fault, local Heidelberg setup a bit different from lxplus)

- Still very much work in progress