Combine Tutorial: Fit Diagnostics

Nicholas Wardle, Giacomo Ortona,
Andrew Gilbert, David Sperka

Combine Mini-Tutorial | 29 November 2017

Introduction

- In this section we will explore some different tools for getting detailed information about the model and the max. likelihood fit

- Some of the examples make use of the combineTool.py script, available in the separate CombineHarvester repository
  - https://github.com/cms-analysis/CombineHarvester
  - See documentation here for details: http://cms-analysis.github.io/CombineHarvester

- To get started quickly: from a CMSSW_8_1_0 release area with the combine package already checked out according to the instructions here:

```
> cd $CMSSW_BASE/src
> bash <(curl -s https://raw.githubusercontent.com/cms-analysis/CombineHarvester/master/CombineTools/scripts/sparse-checkout-https.sh)
> scram b
```

Impacts

Impacts

- The impact of a nuisance parameter (NP) θ on a parameter of interest (POI) μ is defined as the shift Δμ that is induced as θ is fixed and brought to its +1σ or −1σ post-fit values:
  - $\Delta\mu(\pm) = \hat{\mu}(\hat{\theta}\pm\Delta_\theta) - \hat{\mu}(\hat{\theta})$ (All other parameters profiled as normal)
- In the limit where all uncertainties are gaussian this is a measure of the correlation between the μ and θ
- Useful for determining which NPs have the largest effect on the POI uncertainty
- Possible to calculate impacts with combine using the MultiDimFit mode:

```
> cd data/tutorials/htt/125
> text2workspace.py htt_tt.txt -m 125
> combine -M MultiDimFit -m 125 --algo impact -P CMS_scale_t_tautau_8TeV htt_tt.root
...
 --- MultiDimFit ---
Parameter impacts:
  Parameter                 :   Best-fit              r
  CMS_scale_t_tautau_8TeV   :   -0.731  -0.397/+0.422  +0.150/-0.146
```

$\hat{\theta}\pm\Delta_\theta$ Impact (Δμ)

Impacts

- Typically want to see impacts for all nuisance parameters in the model

- Can use the combineTool.py **Impacts** mode to automate this - also supports models with more than one POI

- Calculating the impacts is done in a few stages. First we just fit for each POI, using the **--doInitialFit** option, and adding the **--robustFit 1** option that will be passed through to combine:

```
> combineTool.py -M Impacts -d htt_tt.root -m 125 --doInitialFit
```

- Next we perform a similar scan for each nuisance parameter with the --doFits options:

```
> combineTool.py -M Impacts -d htt_tt.root -m 125 --doFits --parallel 4
```

- This will run **MultiDimFit --algo impact** for each nuisance parameter in turn

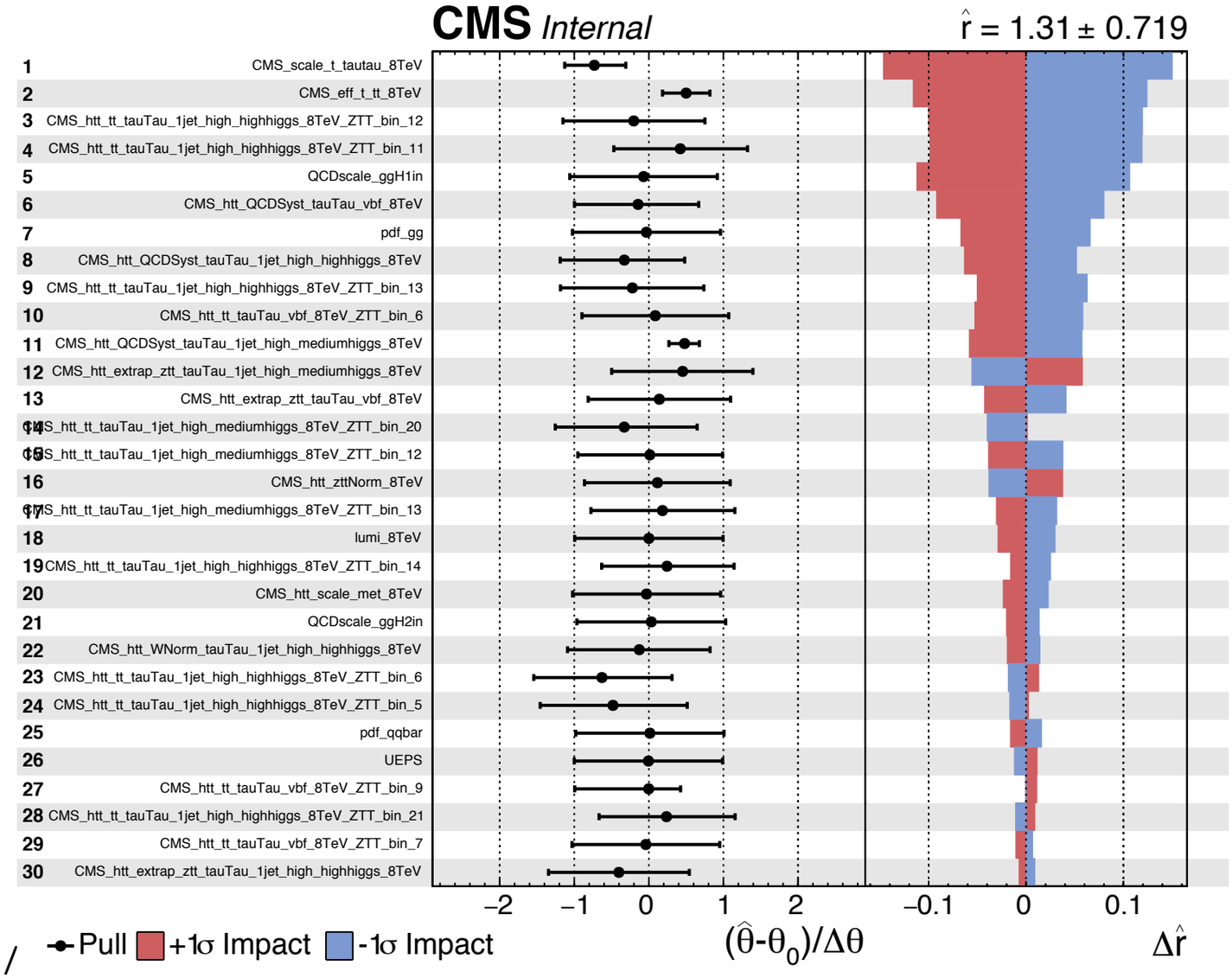
- The names of the POIs and NPs are extracted from the workspace

Impacts

- Once all jobs are completed the output can be collected and written into a json file:

```
combineTool.py -M Impacts -d htt_tt.root -m 125 -o impacts.json
```

```
plotImpacts.py -i impacts.json -o impacts
```




```
{
  "POIs": [
    {
      "fit": [
        0.6110729575157166,
        0.9938536882400513,
        1.423936367034912
      ],
      "name": "r"
    }
  ],
  "params": [
    {
      "fit": [
        -0.9803842306137085,
        0.004973331466317177,
        0.9920346140861511
      ],
      "impact_r": 0.003230690956115,
      "name": "CMS_eff_b_7TeV",
      "prefit": [
        -1.0,
        0.0,
        1.0
      ],
      "r": [
        0.9911616444587708,
        0.9940463304519653,
        0.9976230263710022
      ]
    },
    ...
  ]
}
```

Goodness-of-fit

Goodness-of-fit

- As mentioned yesterday combine supports several goodness-of-fit tests:
  - **Saturated model** (Baker & Cousins) has been supported for a while
  - **Kolmogorov-Smirnov** and **Anderson-Darling** tests recently added

- Basic documentation here

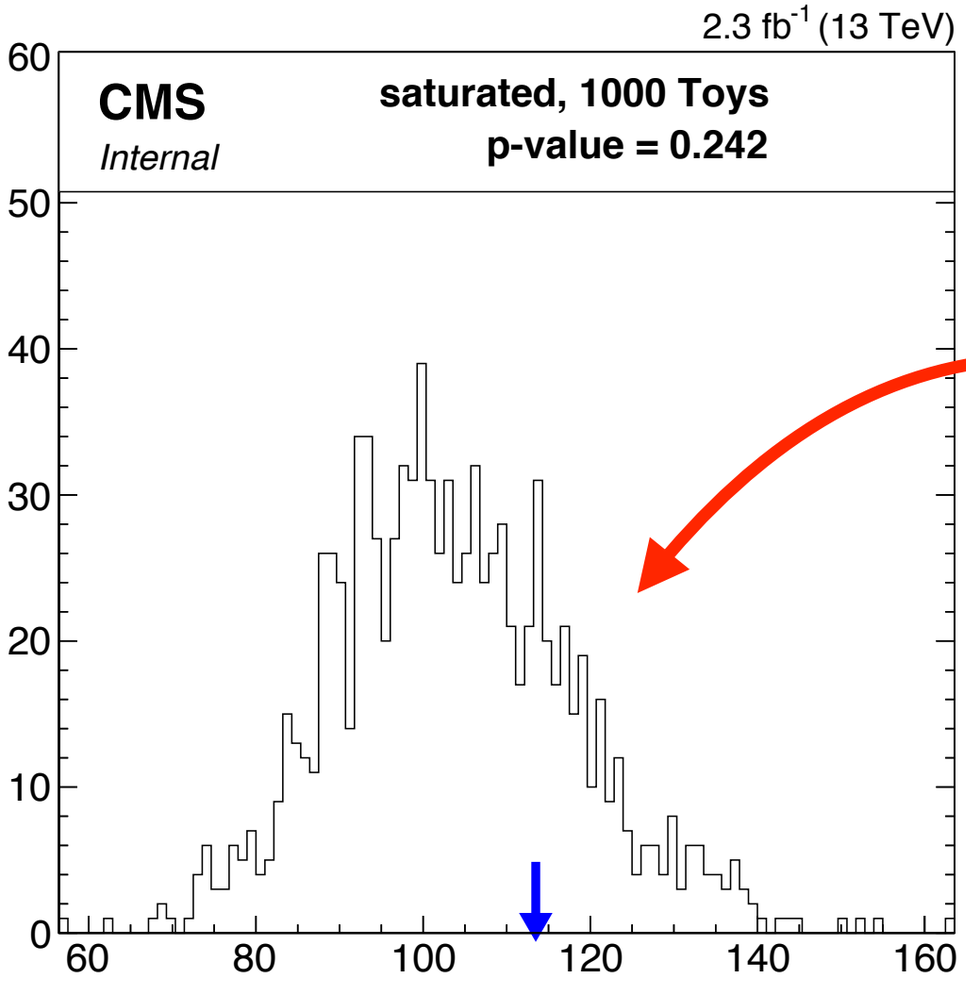
- All methods can utilise combine's toy dataset generating routine to build test-stat distributions ⇒ p-values

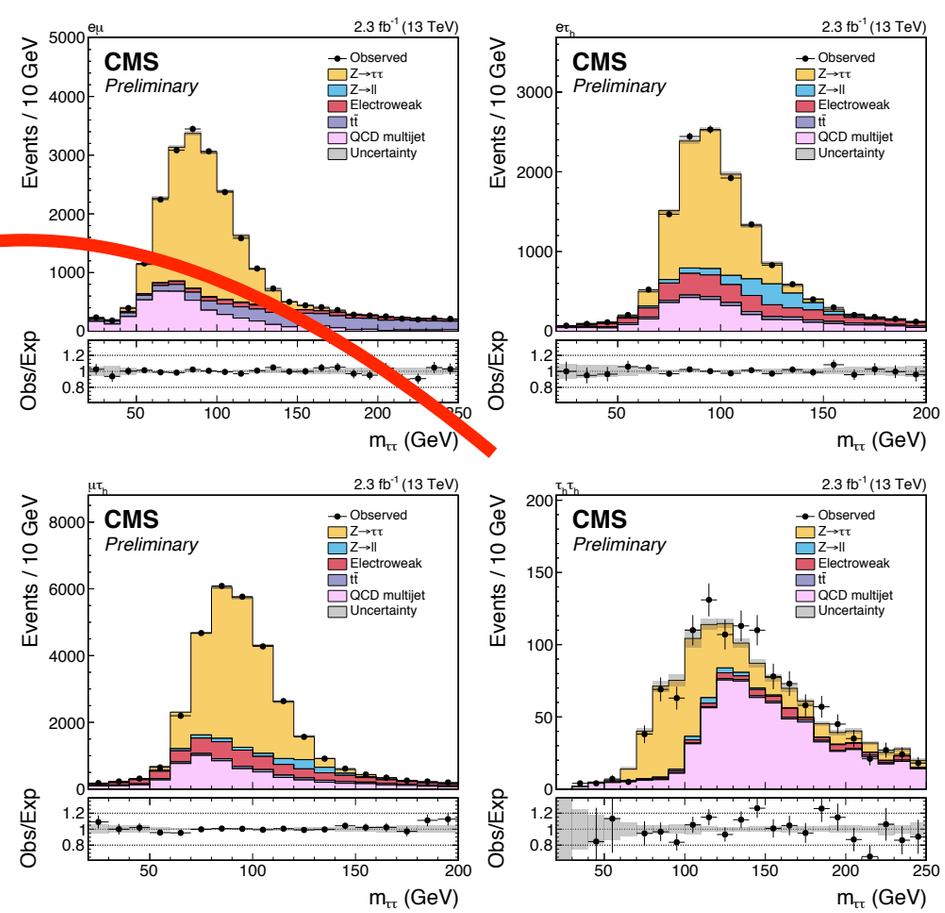
Saturated model

$$-2\ln\lambda = 2\sum_i f_i - d_i + d_i \ln(d_i/f_i).$$

- Likelihood ratio between the nominal model with expectation f_i (at values of parameters θ maximising L) and model in which expectation is replaced by observation d_i
- Straightforward usage in combine - performs test on real data by default, or on toys using **-t** option

```
combine -M GoodnessOfFit --algorithm saturated workspace.root [-t 1000]
```





Example

Z→ττ in four di-τ channels

KS-AD tests

Kolmogorov-Smirnov

$$q_{GoF,KS} = \sup \left| F_c(x) - F_e(x) \right|$$

F_c: Cumulative distribution function

F_e: Empirical distribution function

- Currently implemented for binned analyses only
- Unbinned possible - if someone wants to work on it
- q is determined (and saved) for each category in the simultaneous fit
- Option to save histogram of $|F_c - F_e|$ for each category

Anderson-darling

$$q_{GoF,AD} = n \cdot \int dF_e(x) \frac{\left(F_c(x) - F_e(x)\right)^2}{F_e(x) \cdot \left(1 - F_e(x)\right)}$$

F_c: Cumulative distribution function

F_e: Empirical distribution function

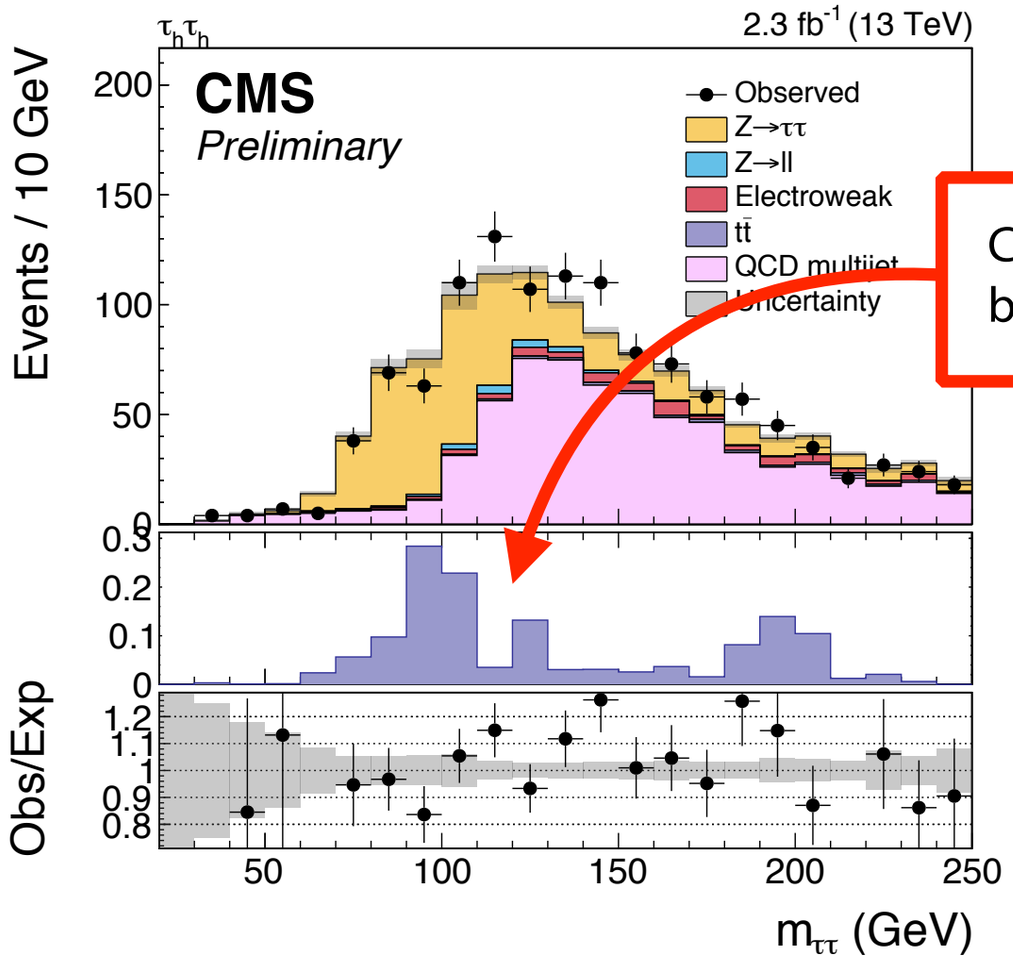
n: Total observed yield

- Gives extra weight to differences in the tails of the distribution
- As for KS only binned supported for now
- q is determined (and saved) for each category in the simultaneous fit
- Option to save histogram of contribution to integral for each bin

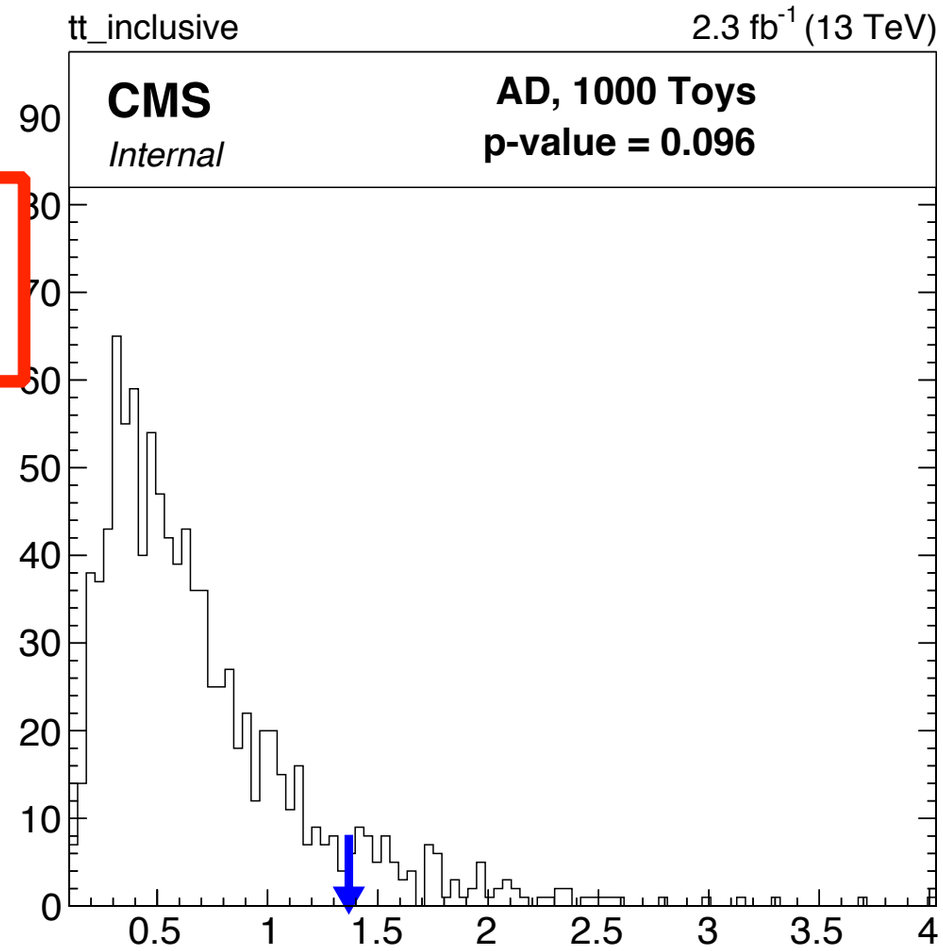
KS-AD tests

- Similar usage to saturated model test:

```
combine -M GoodnessOfFit --algorithm [ad/ks] workspace.root --plots [-t 1000]
```



Contribution of each bin to AD integral



p-value is integral from observation to +inf

FitDiagnostics

FitDiagnostics

- Use "**combine -M FitDiagnostics**" (previously called MaxLikelihoodFit) to get additional information about the model and the behaviour of the fit

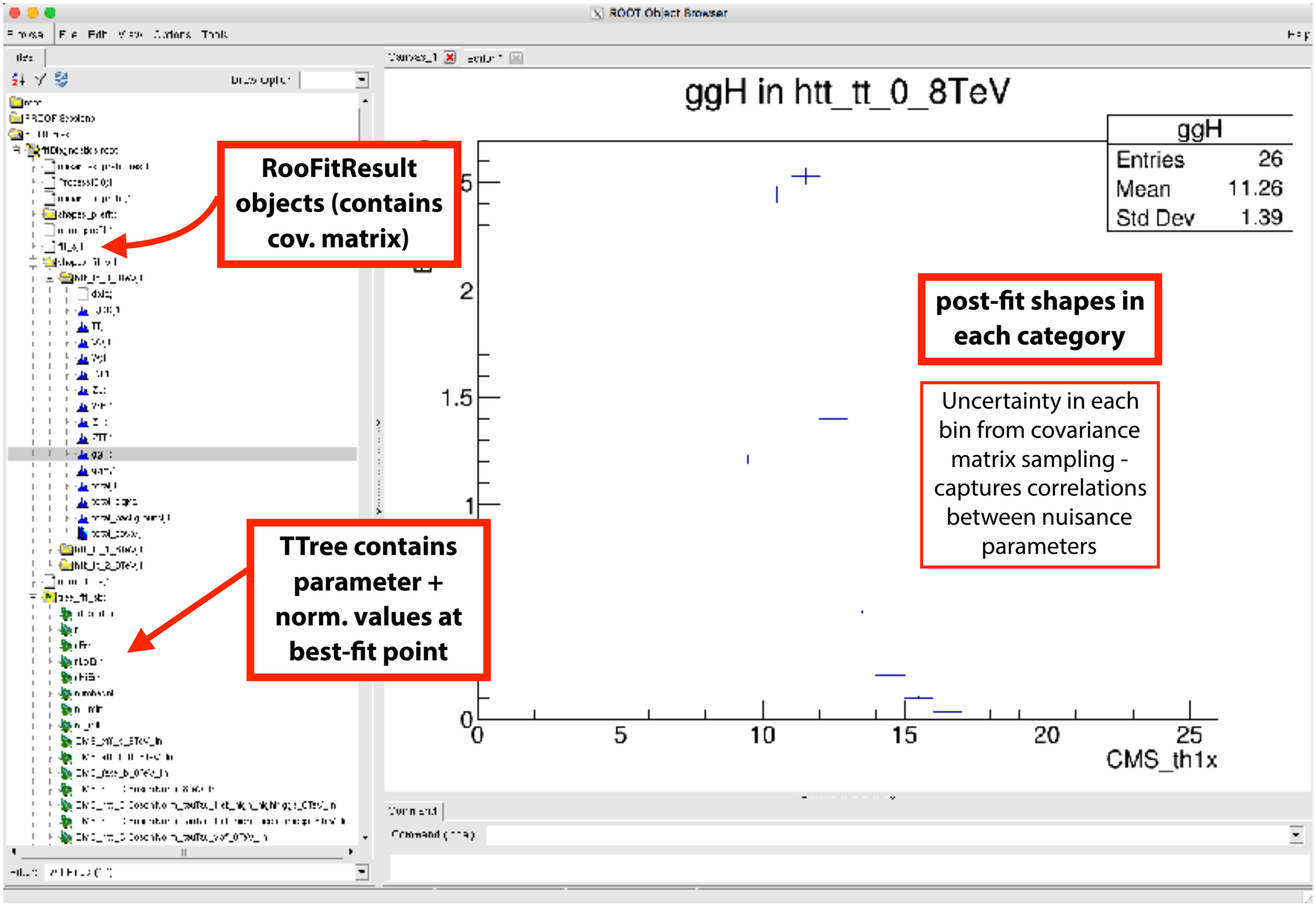
- Performs two fits:
  - "background-only" fit: first POI (usually "r") fixed to zero
  - "signal+background" fit: all POIs are floating

- Saves the covariance matrix at both best-fit points

- Use '**--saveNormalizations**' to save post-fit yields of all processes

- Use the options '**--saveShapes**' and '**--saveWithUncertainties**' and combine will produce pre- and post-fit distributions

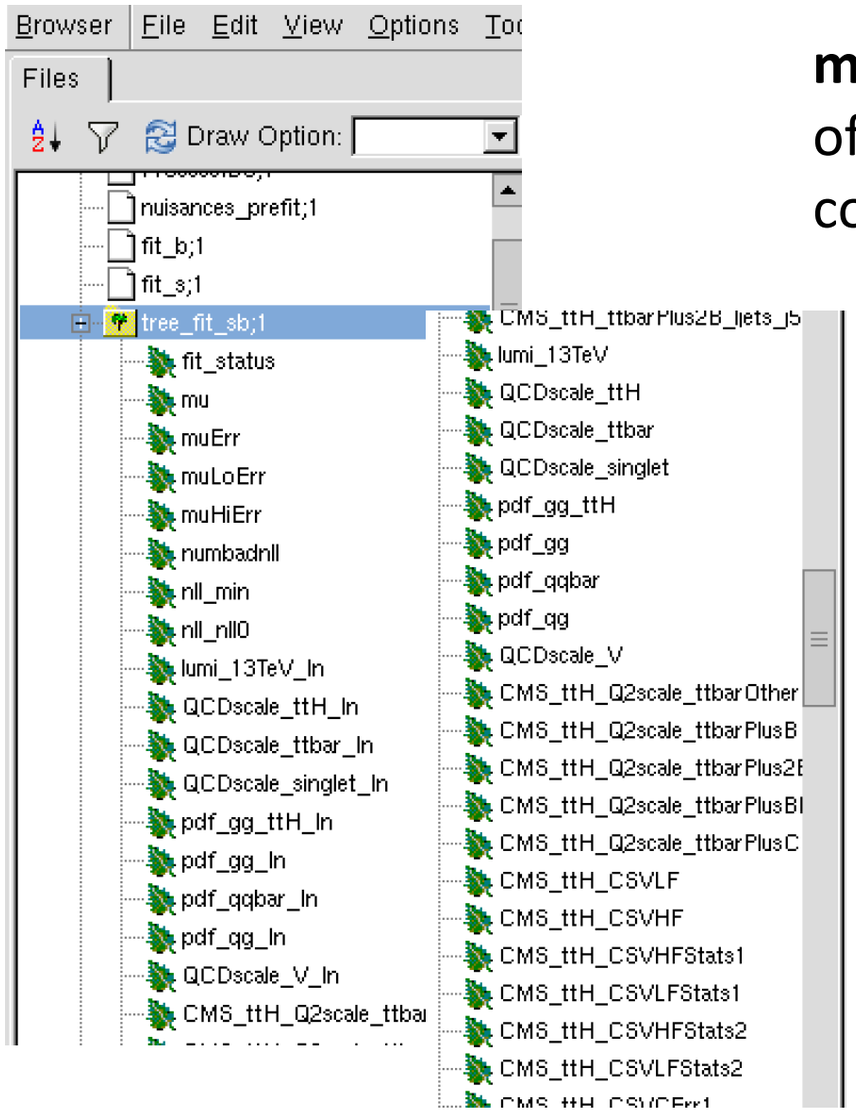
FitDiagnostics



FitDiagnostics - using toys

Generate 200 toys with 0 expected signal ….

```
combine ttH_hbb_13TeV_sl_j5_tge4_high.root -M FitDiagnostics
--toysFrequentist -t 200 --rMin -5 --rMax 5 --expectSignal 0 -n toys
```



mlfit.root contains trees with the results of all nuisance parameters and generated constraints from the fits

Can use trees to investigate strange behavior in fits/correlations between parameters etc…

13

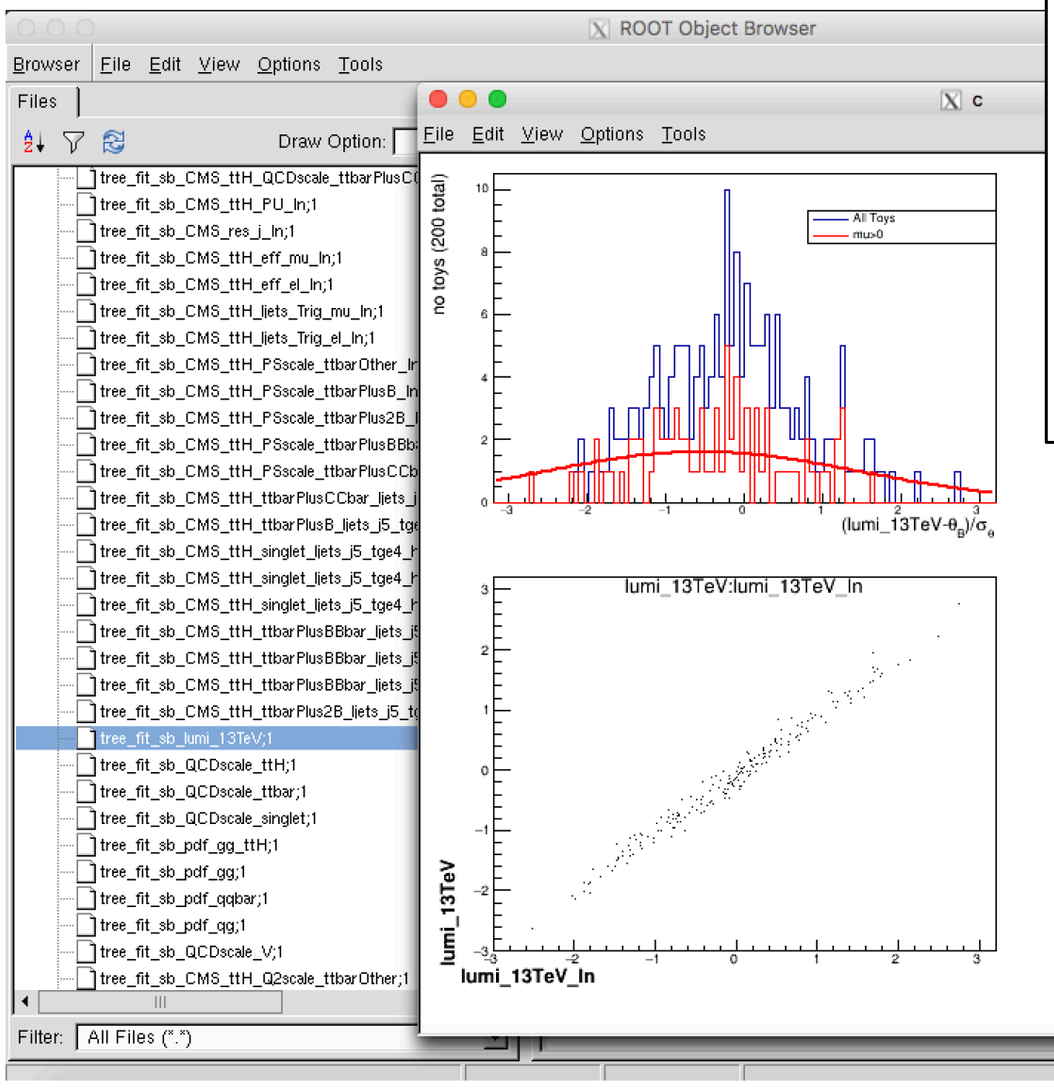
FitDiagnostics - using toys

Result from toys

```
root'test/plotParametersFromToys.C+("mlfittoys.root",
"mlfit.root","ttH_hbb_13TeV_sl_j5_tge4_high.root")'
```

Result from the data fit

The binary workspace



Output is two root files. One foor for the background only and one for the signal+background fits

Each file has a one canvas per nuisance parameter showing a summary of the fits of that parameter across the toys

Plain likelihood vs that parameter

Uncertainty breakdown and nuisance parameter groups

Uncertainty breakdown

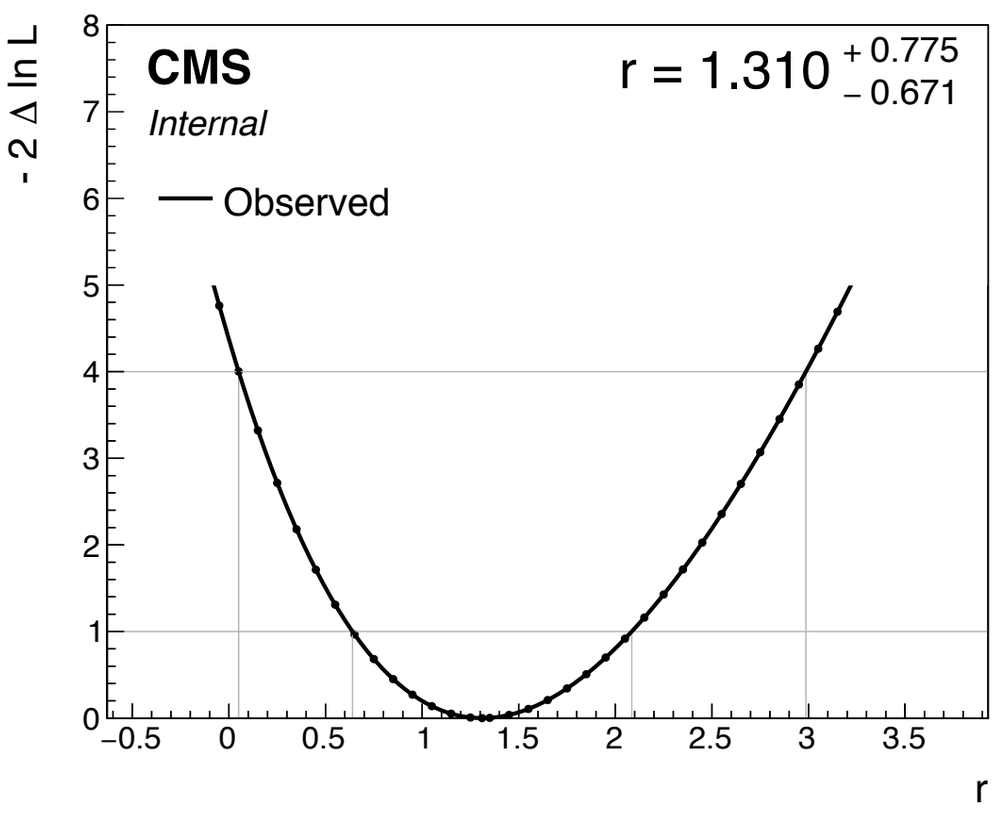
- Can also study how groups of related nuisance parameters affect a measurement: twiki
- **Example**: decompose a signal strength uncertainty into components, e.g. statistical, systematic, theory
- **Step 1**: Perform a likelihood scan using the **MultiDimFit** mode of combine with the **grid** algo
  - This gives us the full uncertainty on our signal strength r

```
cd $CMSSW_BASE/src/HiggsAnalysis/CombinedLimit/data/tutorials/htt/125
text2workspace.py htt_tt.txt -m 125
combine -M MultiDimFit --algo grid --points 50 --rMin -1 --rMax 4 htt_tt.root -m 125 -n nominal
plot1DScan.py higgsCombinenominal.MultiDimFit.mH125.root        # plotting requires the CombineHarvester package
```

```
**********************************
*    Row   *          r *  deltaNLL *
**********************************
*      0 * 1.3102449 *          0 *
*      1 * -0.949999 * 7.8044676 *
*      2 * -0.850000 * 7.0111770 *
*      3 *     -0.75 * 6.2667942 *
*      4 * -0.649999 * 5.5712294 *
*      5 * -0.550000 * 4.9240608 *
*      6 * -0.449999 * 4.3245592 *
*      7 * -0.349999 * 3.7717974 *
*      8 *     -0.25 * 3.2645695 *
*      9 * -0.150000 * 2.8014321 *
*     10 * -0.050000 * 2.3808751 *
*     11 * 0.0500000 * 2.0011189 *
*     12 * 0.1500000 * 1.6606326 *
...
```

First the global best-fit is determined

Then combine steps through 50 points in the given range, fixing r, and profiling to give the ΔNLL with respect to the best-fit



Uncertainty breakdown

- **Step 2**: Next we will repeat the scan but with some or all of the nuisance parameters frozen to their best-fit values.
  - First perform the fit and save a copy of the workspace in the output file with a "snapshot" of the best-fit model parameters included

```
combine -M MultiDimFit --algo none --rMin -1 --rMax 4 htt_tt.root -m 125 -n bestfit --saveWorkspace
```

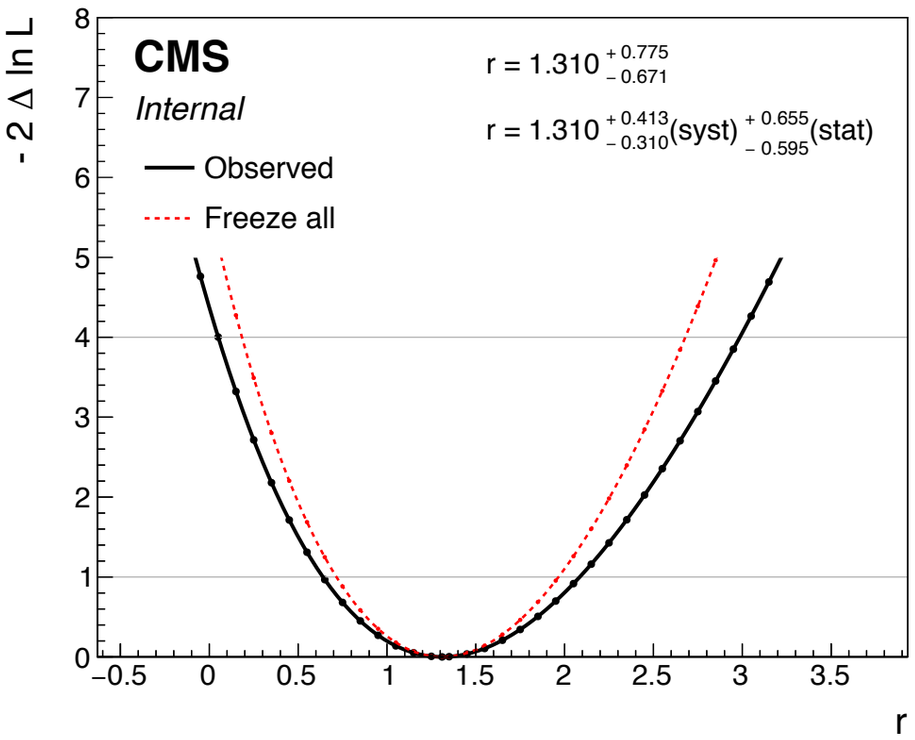
  - Can then repeat the scan by first loading this snapshot, then freezing all the nuisance parameters

```
combine -M MultiDimFit --algo grid --points 50 --rMin -1 --rMax 4 -m 125 -n stat \
higgsCombinebestfit.MultiDimFit.mH125.root --snapshotName MultiDimFit --freezeNuisances all
```

  - The uncertainty from this scan gives us the statistical component of the uncertainty. By defining: $\sigma_{total}^2 = \sigma_{stat}^2 + \sigma_{syst}^2$, we can infer the systematic component

```
plot1DScan.py higgsCombinenominal.MultiDimFit.mH125.root \
--others 'higgsCombinestat.MultiDimFit.mH125.root:Freeze all:2' \
--breakdown syst,stat
```

- **Step 3**: What if we want to split things even further, and treat the theoretical uncertainties as a separate contribution from the experimental systematics?
  - Sequentially freeze groups of parameters - at each step do quadratic subtraction as above to give contribution of the group that was frozen
    - $\sigma_{theory}^2 = \sigma_{total}^2 - \sigma_{freeze\ theory}^2$
    - $\sigma_{syst}^2 = \sigma_{freeze\ theory}^2 - \sigma_{freeze\ theory+syst}^2$
    - $\sigma_{stat}^2 = \sigma_{freeze\ theory+syst}^2 - \sigma_{freeze\ all}^2$



Uncertainty breakdown

- Could just list parameters to freeze with --freezeNuisances x,y,z,...
- But if many parameters to freeze can also define named lists in the datacard:

```
...
QCDscale_VH             lnN                                                 1.04          -             1.04          -
QCDscale_ggH1in         lnN                                                 -             -             -             1.205
QCDscale_ggH2in         lnN                                                 -             -             -             -
QCDscale_qqH            lnN                                                 -             1.012         -             -
UEPS                    lnN                                                 1.025         1.025         1.025         0.975
lumi_8TeV               lnN                                                 1.026         1.026         1.026         1.026
pdf_gg                  lnN                                                 -             -             -             1.097
pdf_qqbar               lnN                                                 1.02          1.036         1.02          -
theory group = QCDscale_VH QCDscale_ggH1in QCDscale_ggH2in QCDscale_qqH UEPS pdf_gg pdf_qqbar
```

- **After adding this line, re-run the steps on the previous slide,** then can then freeze all parameters in this group with:

```
combine -M MultiDimFit --algo grid --points 50 --rMin -1 --rMax 4 -m 125 -n theory \
higgsCombinebestfit.MultiDimFit.mH125.root --snapshotName MultiDimFit --freezeNuisanceGroups theory
```

```
plot1DScan.py higgsCombinenominal.MultiDimFit.mH125.root --others \
'higgsCombinetheory.MultiDimFit.mH125.root:Freeze th.:4' \
'higgsCombinestat.MultiDimFit.mH125.root:Freeze all:2' \
--breakdown theory,syst,stat
```

Can list as many additional scans as desired, in the format FILE:LEGEND:COLOUR. The labels in --breakdown should follow the order of the scans for the quadratic subtractions