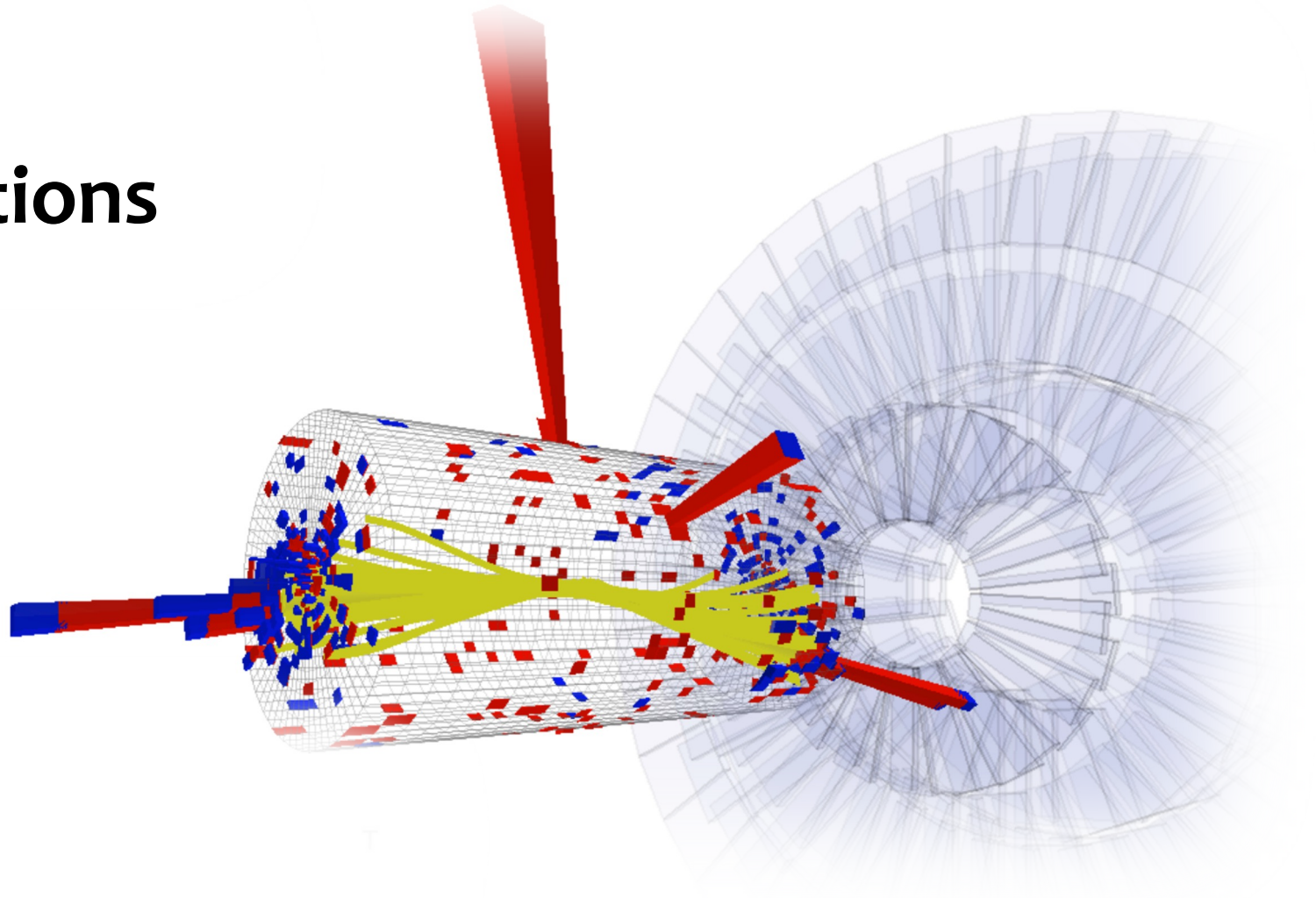
Imperial College London

Combine & CMS Likelihood Publications

Nicholas Wardle



HSF DAWG

10th June 2024

CMS Combine

RooStats / RooFit - based software tool used for **statistical analysis within the CMS experiment.**

→ **Ideally** should re-use (benefit) as much as possible the functionality of RooFit/RooStats without the user needing to remember lots of "tricks"/"gotchas" and write scripts

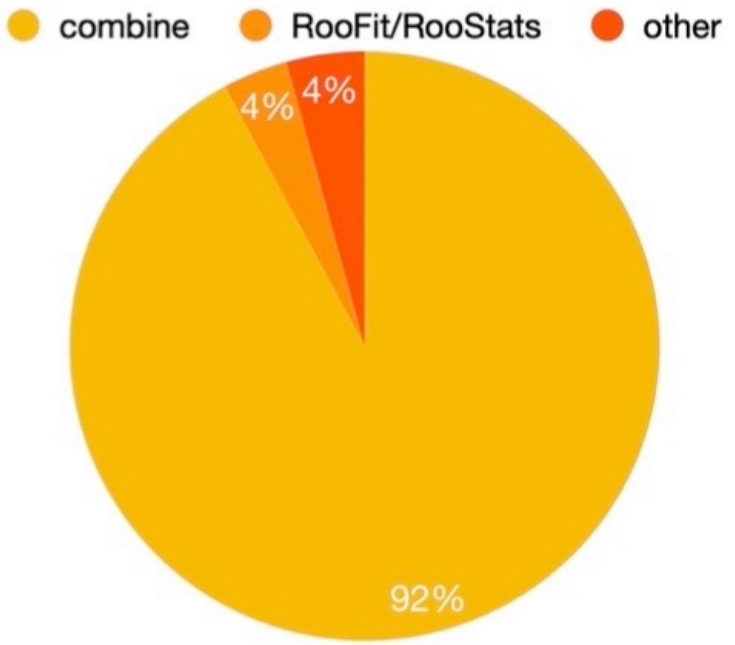
→ Combine mostly provides **"workflows"** for the most **common statistical methods** and uses LHC/CMS statistics committee **best-practices**

General features

- Human readable inputs (**datacards**)
- Build **binned / unbinned likelihood models**
- Intuitive and powerful **models combination**
- **Statistical tests** Bayesian/frequentist/hybrid
- **Diagnostics tools** for fits inspection

Documentation: https://cms-analysis.github.io/HiggsAnalysis-CombinedLimit/latest/

- **Summaries** of **what the tool is doing** (brief statistics)
- Detailed examples of tool functionality for **common and more advanced methods**
- **Tutorials** designed to get new users up to speed **on the basics**



From Statistics Committee Questionnaires 2021-2022

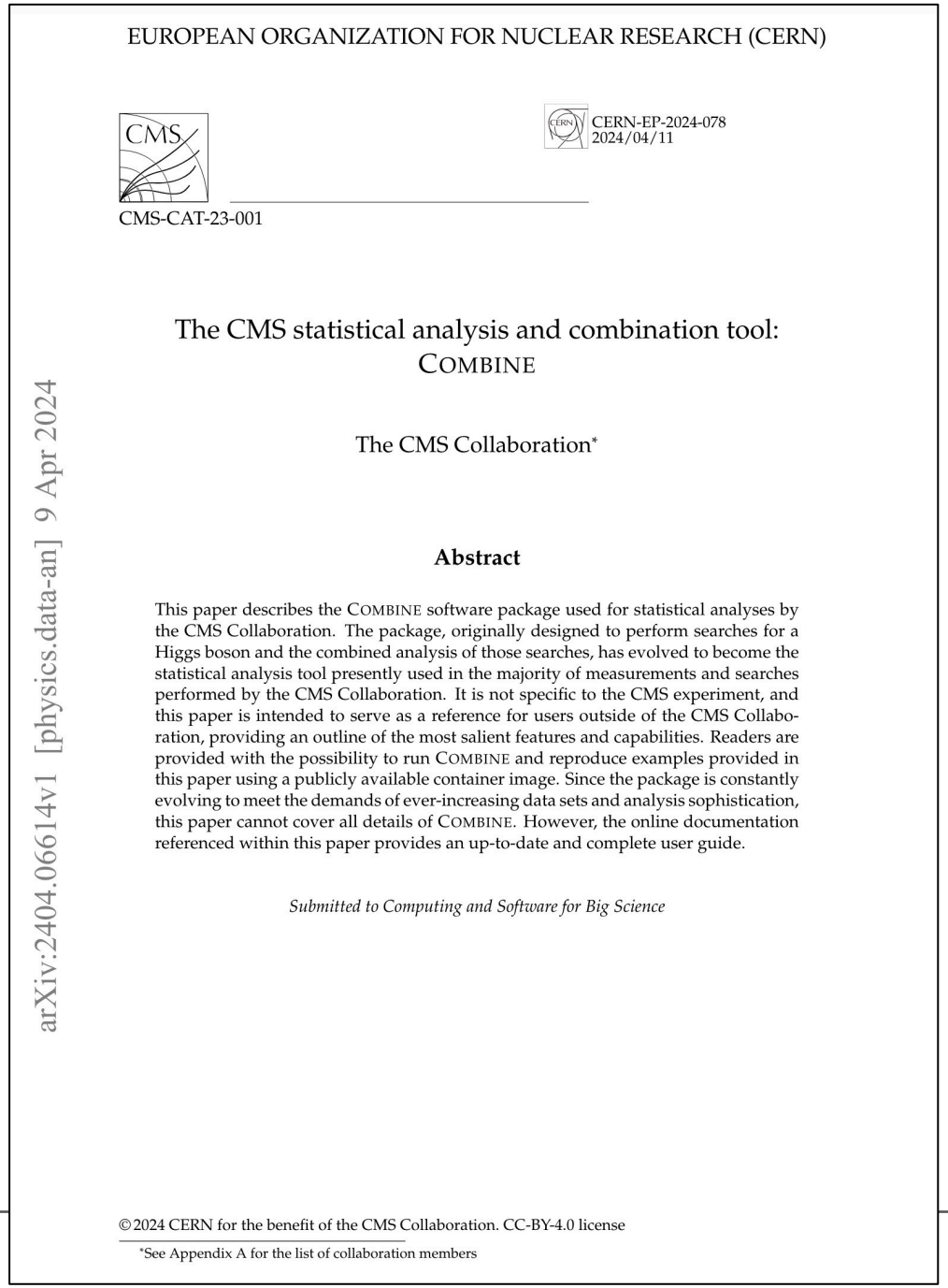
CMS Combine Publication

Publication for Combine:
https://arxiv.org/abs/2404.06614 (sub. to CSBS)

- Guide on **installation via Docker** (pre-compiled version 9.1.0) – other installation strategies described online

- Definition of **statistical model** and how to construct them in combine – **Datacards**
  - Examples for counting, template and parameteric models
  - Physics models

- **Common statistical routines** available
  - Pseudo-data generation
  - Upper limits (frequentist + Bayesian) and significance calculations
  - Maximum likelihood fits and scans
  - Fitting diagnostics (plotting, GoF tests, impacts)



EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH (CERN)

CERN-EP-2024-078
2024/04/11

CMS-CAT-23-001

The CMS statistical analysis and combination tool: COMBINE

The CMS Collaboration*

Abstract

This paper describes the COMBINE software package used for statistical analyses by the CMS Collaboration. The package, originally designed to perform searches for a Higgs boson and the combined analysis of those searches, has evolved to become the statistical analysis tool presently used in the majority of measurements and searches performed by the CMS Collaboration. It is not specific to the CMS experiment, and this paper is intended to serve as a reference for users outside of the CMS Collaboration, providing an outline of the most salient features and capabilities. Readers are provided with the possibility to run COMBINE and reproduce examples provided in this paper using a publicly available container image. Since the package is constantly evolving to meet the demands of ever-increasing data sets and analysis sophistication, this paper cannot cover all details of COMBINE. However, the online documentation referenced within this paper provides an up-to-date and complete user guide.

Submitted to Computing and Software for Big Science

*See Appendix A for the list of collaboration members

Statistical models

Statistical model

- Primary observables $\vec{x}$
- Parameters of interest $\vec{\mu}$
- Nuisance parameters $\vec{\nu}$
- Auxiliary observables $\vec{y}$

$$p(\vec{x},\vec{y};\vec{\Phi}) = p(\vec{x};\vec{\mu},\vec{\nu})\prod_k p_k(y_k;\nu_k)$$

Each element of the vector $\vec{x}$ is referred to as a "channel" (each of which is statistically independent) so that

$$p(\vec{x};\vec{\mu},\vec{\nu}) \rightarrow \prod_i p_i(x_i;\vec{\mu},\vec{\nu})$$

Need to define probability distributions *p* to specify statistical model.

Likelihood constructed by
(in Combine –Log(L)= `CachingNLL`)

$$\mathcal{L}(\vec{\Phi}) = \prod_d p(\vec{x}_d;\vec{\mu},\vec{\nu})\prod_k p_k(y_k;\nu_k)$$

Statistical models

Extremely basic example – a counting experiment Datacard

```
imax 1
jmax 2
kmax 3
# A single channel - ch1 - in which 0 events are observed in data
bin             ch1
observation     0
# ----------
bin             ch1    ch1    ch1
process         ppX    WW     tt
process         0      1      2
rate            1.47   0.64   0.22
# ----------
lumi     lnN    1.11   1.11    1.11
xs       lnN    1.20     -       -
nWW      gmN 4    -    0.16      -
```

Observable = Observed number of events

Expected contributions

‘Systematic” uncertainties

Statistical models

Extremely basic example – a counting experiment Datacard

```
imax 1
jmax 2
kmax 3
# A single channel - ch1 - in which 0 events are observed in data
bin             ch1
observation     0
# ----------
bin             ch1    ch1    ch1
process         ppX    WW     tt
process         0      1      2
rate            1.47   0.64   0.22
# ----------
lumi    lnN     1.11   1.11    1.11
xs      lnN     1.20     -       -
nWW     gmN 4     -    0.16      -
```

Observable = Observed number of events

Expected contributions

‘Systematic” uncertainties

Apply default Physics model

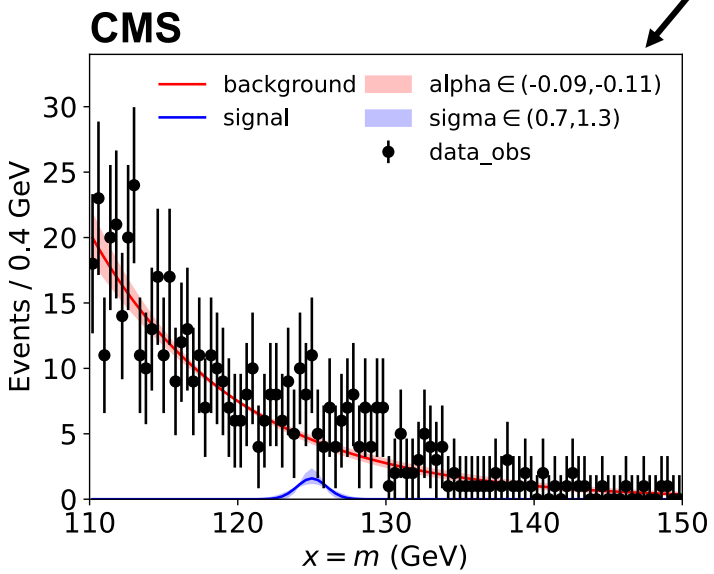
$$p(n,\vec{y};r,\vec{\nu}) = \frac{\lambda(r,\vec{\nu})^{n}}{n!}\mathrm{e}^{-\lambda(r,\vec{\nu})}\frac{1}{2\pi}\mathrm{e}^{-(\nu_{\mathtt{lumi}}-y_{\mathtt{lumi}})^2}\mathrm{e}^{-(\nu_{\mathtt{xs}}-y_{\mathtt{xs}})^2}\frac{(\nu_{\mathtt{nWW}})^{y_{\mathtt{nWW}}}}{y_{\mathtt{nWW}}!}\mathrm{e}^{-\nu_{\mathtt{nWW}}},$$

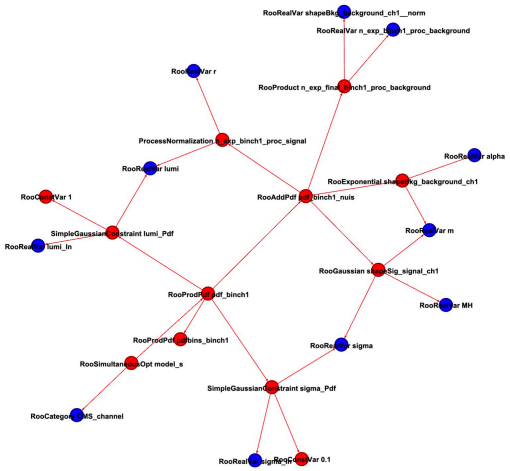
$$\lambda(r,\vec{\nu}) = r\,1.47\,(1.11)^{\nu_{\mathtt{lumi}}}(1.2)^{\nu_{\mathtt{xs}}} + 0.22\,(1.11)^{\nu_{\mathtt{lumi}}} + 0.64\,(1.11)^{\nu_{\mathtt{lumi}}}\frac{\nu_{\mathtt{nWW}}}{0.64}.$$

Statistical models

Statistical model based on **Datacard** + **Object inputs**

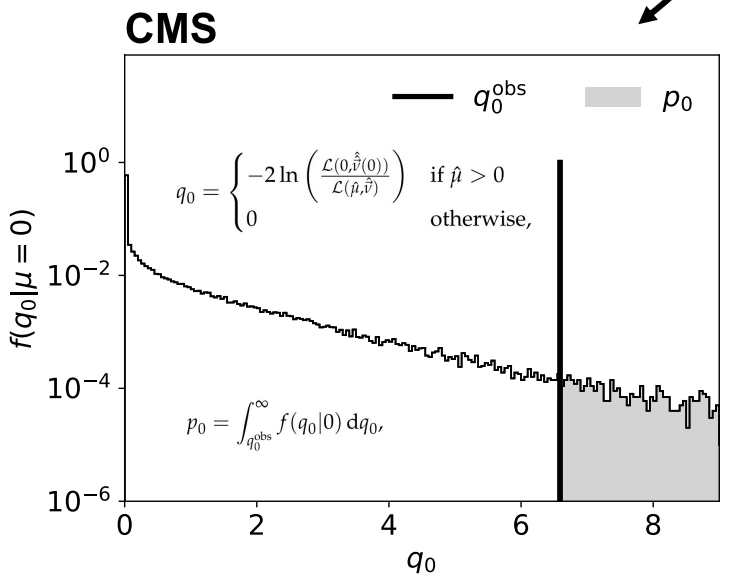
```
imax 1
jmax 1
kmax 2
# ---------------
shapes data_obs   bin1 parametric-analysis-datacard-input.root w:
  ↪ data_obs
shapes signal     bin1 parametric-analysis-datacard-input.root w:sig
shapes background bin1 parametric-analysis-datacard-input.root w:bkg
# ---------------
bin                bin1
observation        567
# ---------------
bin                bin1   bin1
process            signal background
process            0      1
rate               10     1
# ---------------
lumi         lnN   1.1    -
sigma        param 1.0    0.1
alpha        flatParam
bkg_norm     flatParam
```





$$p(\vec{x},\vec{y};\vec{\Phi}) = p(\vec{x};\vec{\mu},\vec{\nu}) \prod_k p_k(y_k;\nu_k)$$

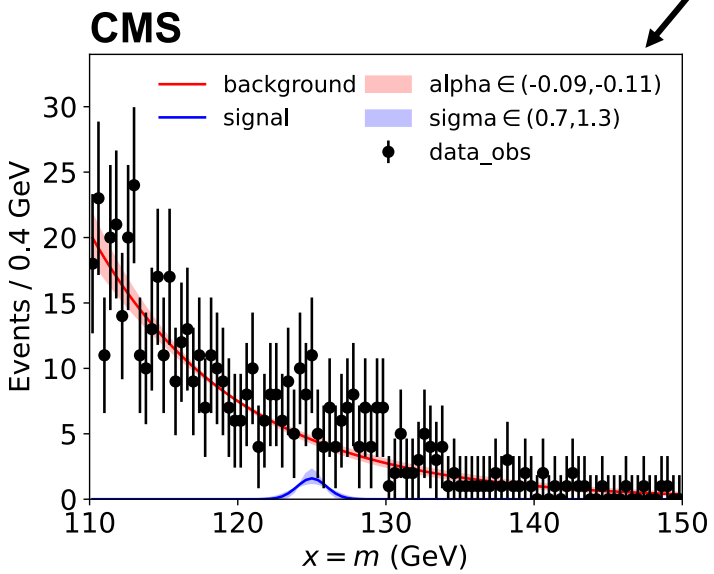
Sampling and Likelihood (= statistical model + data) based methods to extract statistics results

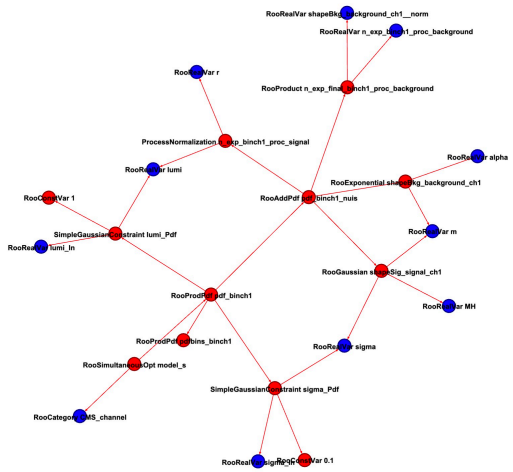


Statistical models

Statistical model based on **Datacard** + **Object inputs**

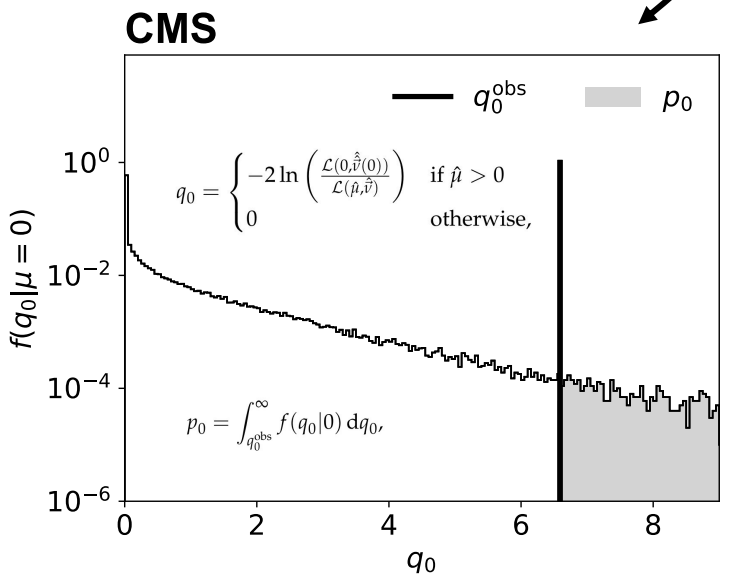
```
imax 1
jmax 1
kmax 2
# ---------------
shapes data_obs   bin1 parametric-analysis-datacard-input.root w:
  ↪ data_obs
shapes signal     bin1 parametric-analysis-datacard-input.root w:sig
shapes background bin1 parametric-analysis-datacard-input.root w:bkg
# ---------------
bin                bin1
observation        567
# ---------------
bin                bin1   bin1
process            signal background
process            0      1
rate               10     1
# ---------------
lumi         lnN   1.1    -
sigma        param 1.0    0.1
alpha        flatParam
bkg_norm     flatParam
```





$$p(\vec{x},\vec{y};\vec{\Phi}) = p(\vec{x};\vec{\mu},\vec{\nu}) \prod_k p_k(y_k;\nu_k)$$

Sampling and Likelihood (= statistical model + data) based methods to extract statistics results



- Combinations based on datacard combination (always perform combinations at “likelihood level”)
- Same statistical model can be used for **multiple statistical routines** in combine!

Physics Models

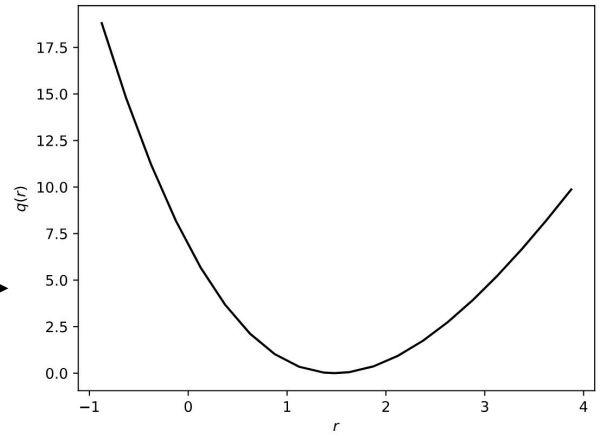
Physics models are the way combine defines a set of **parameters of interest** and uses them to parameterize the **rates of different processes** (*usually* "signal")

Datacard

```
imax 2
jmax 2
kmax *
------------
shapes *  dijet  FAKE
shapes *  incl   FAKE
------------
bin           incl dijet
observation   166  8
------------
bin                incl    incl    incl    dijet   dijet   dijet
process            ggH_hgg qqH_hgg bkg     ggH_hgg qqH_hgg bkg
process             -1     0       1       -1      0       1
rate               21      1.6     140     0.4     0.95    3.2
------------
QCDscale_ggH lnN   1.12    -       -       1.12    -       -
pdf_gg       lnN   1.08    -       -       1.08    -       -
pdf_qqbar    lnN    -      1.025   -       -       1.025   -
bg_incl      lnN    -      -       1.05    -       -       -
```

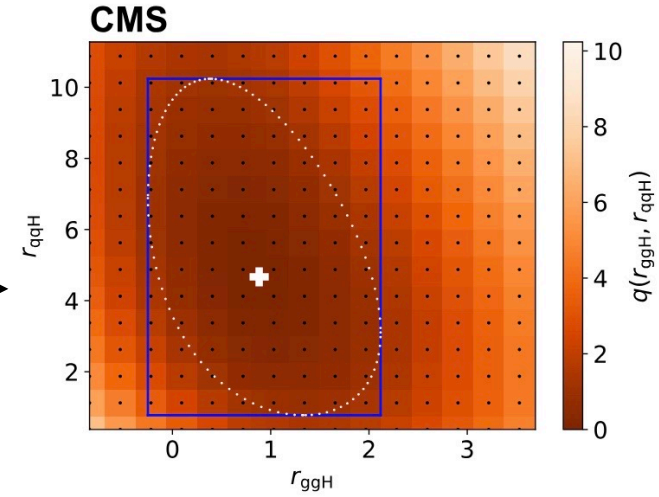
```
def getYieldScale(self, bin, process):
    "Return the name of a RooAbsReal to scale this yield by or the two special values 1
    return "r" if self.DC.isSignal[process] else 1
```

Default Model



-P HiggsAnalysis.CombinedLimit.PhysicsModel:floatingXSHiggs

```
def getHiggsSignalYieldScale(self,production,decay, energy):
  if production == "ggH": return ("r_ggH" if "ggH" in self.modes else 1)
  if production == "qqH": return ("r_qqH" if "qqH" in self.modes else 1)
  if production in [ "WH", "ZH", "VH" ]: return ("r_VH" if "VH" in self.modes else
  ↪ 1)
  if production == "ttH": return ("r_ttH" if "ttH" in self.modes else ("r_ggH" if
  ↪ self.ttHasggH else 1))
  raise RuntimeError, "Unknown production mode '%s'" % production
```



Physics models are Python files that are passed to `text2workspace.py` with

```
$ text2workspace.py <datacard.txt> -P HiggsAnalysis.CombinedLimit.<PythonFile>:<
  ↪ modelInstance> [--PO <options>]
```

Statistical Results

Combine also **handles producing statistical results** once model is constructed. For the most common statistical routines, the idea is that it should be **1 command line.**

```
$ combine datacard-1-counting-experiment.txt -M MarkovChainMC --tries 100
```

```
>  -- MarkovChainMC --
> Limit: r < 2.21031 +/- 0.0133576 @ 95% credibility (100 tries)
> Done in 0.05 min (cpu), 0.05 min (real)
```

```
$ combine datacard-5-multi-signal.root -M MultiDimFit --algo singles --mass 125
```

```
>  --- MultiDimFit ---
> best fit parameter values and profile-likelihood uncertainties:
>    r_ggH :    +0.882   -0.749/+0.795 (68%)
>    r_qqH :    +4.683   -2.746/+3.464 (68%)
> Done in 0.00 min (cpu), 0.04 min (real)
```

```
$ combine datacard-3-parametric-analysis.txt -M Significance --mass 125
```

```
>  -- Significance --
> Significance: 2.56729
> Done in 0.00 min (cpu), 0.00 min (real)
```

+ ROOT file output

Branch name	Type	Description
`limit`	`Double_t`	Main result of the statistical routine being performed.
`limitErr`	`Double_t`	Estimated uncertainty in the result.
`mh`	`Double_t`	Value specified with `--mass` command line option. The default value is 120.
`iToy`	`Int_t`	Pseudo-data set identifier if running with `--toys`.
`iSeed`	`Int_t`	Random seed specified with `-s`.
`t_cpu`	`Float_t`	Estimated processing time.
`t_real`	`Float_t`	Elapsed wall-clock time for routine.
`quantileExpected`	`Float_t`	Quantile identifier for methods that calculate expected and observed results. The meaning is method-dependent. Negative values are reserved for entries that are not related to quantiles of a calculation. The default is set to -1 and specifies that the entry corresponds to the result obtained from the observed data.

More **complicated workflows + many more options** to tweak runtime, sampling strategies, accuracy etc exist in the **online docs.**

Published Statistical Models

The **datacard + its inputs** are publishable products from CMS

→ with Combine now available, these can be (re)interpreted using the same software as the original results

First such publication from CMS is the **Run-1 Higgs discovery model**

CDS Search records... Communities My dashboard Log in

CMS statistical models

Published April 15, 2024 | Version v1.0 Model Open

CMS Higgs boson observation statistical model

CMS Collaboration

1K VIEWS 118 DOWNLOADS

Show more details

Introduction

This resource contains the full statistical model from the Higgs Run-1 combination, which led to the Higgs boson discovery, in the format of `Combine` datacards. The instructions below include a few basic examples on how to extract the significance and signal strength measurements, for more details please consult the `Combine` documentation.

Datacards

Datacards for the combination (and per-decay channel sub-combinations) leading to the Higgs-boson discovery at CMS are in the `125.5` folder. The nuisance parameters corresponding to different sources of systematic uncertainties are described in the `*.html` and `*.yml` files located in that folder.

For the full combination of decay channels, the relevant datacard is `125.5/comb.txt`. The individual datacards for each of the analyses in CMS targeting the main Higgs boson decay modes are also in the `125.5` folder.

Software instructions

General installation instructions for `Combine` can be found in the `Combine` documentation.

> container image is provided to ensure reproducible results. The results in this README are obtained using `v9.2.1`:

```
docker run --name combine -it gitlab-registry.cern.ch/cms-cloud/combine-standalone:v9.2.1
```

> A slim version of the container image is also available at `gitlab-registry.cern.ch/cms-cloud/combine-standalone:v9.2.1-slim`. Versions of packages in the slim container image do not match exactly with the ones in the default container, so small differences in the output of commands with respect to the ones shown below are to be expected.

You can copy files (such as the datacards and other inputs for `combine`) using `docker cp` as documented here.

For the commands below, you may require running `ulimit -s unlimited; ulimit -u unlimited` to avoid memory issues.

Versions

Version v1.0 Apr 15, 2024
10.17181/c2948-e8875

Cite all versions? You can cite all versions by using the DOI 10.17181/2cp5k-ggn24. This DOI represents all versions, and will always resolve to the latest one. Read more.

Communities

CMS statistical models

Details

DOI *(Cite this version - v1.0)*
DOI 10.17181/c2948-e8875

DOI *(Cite all versions)*
DOI 10.17181/2cp5k-ggn24

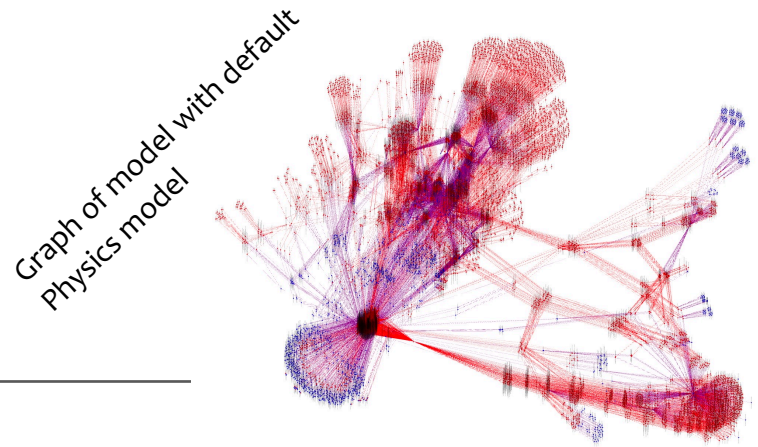
Resource type
Model

Publisher
CERN

https://new-cds.cern.ch/records/c2948-e8875

Release includes

- Datacards + inputs
- Recommended container for combine version
- Instructions for
  - Use with example physics models
  - Deriving important results
- `html` with descriptions of all nuisance parameters in the model



Published Statistical Models

Different results can be produced from the **same published inputs**

- After all, this is how we operate in CMS → No surgery of the binary workspaces needed

```
combine 125.5/comb.txt --mass 125.5 -M Significance
```

Default model → Calculate

```
 -- Significance --
Significance: 4.87557
Done in 1.76 min (cpu), 1.76 min (real)
```

```
text2workspace.py -P HiggsAnalysis.CombinedLimit.HiggsCouplings_ICHEP12:cVcF 125.5/comb.txt -m 125.5 -o comb_kVkF.root
```

Build model

```
combine comb_kVkF.root -m 125.5 -M MultiDimFit --algo singles
```

Calculate

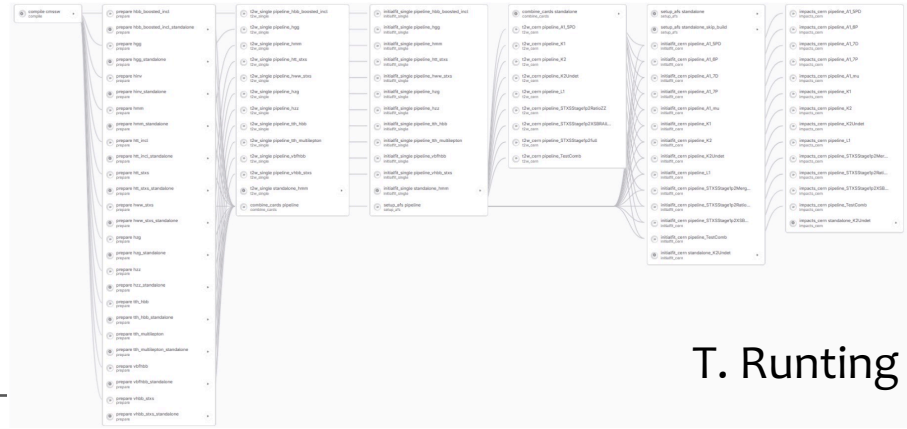
```
 --- MultiDimFit ---
best fit parameter values and profile-likelihood uncertainties:
   CV :    +0.946   -0.120/+0.113 (68%)
   CF :    +0.497   -0.170/+0.203 (68%)
Done in 3.09 min (cpu), 3.09 min (real)
```

CMS plans to **regularly release CMS statistical models** with publications

→ Continuous Integration suite being developed to assist CMS users

→ Several workflow managers now being used (eg `luigi`, `snakemake`) in CMS to perform statistical calculations themselves

→ Makes encapsulation of workflow and validations for publication that much easier



T. Runting

Thoughts on future developments

The rest of these slides are my own personal thoughts on development needed going forward.
Not necessarily the opinion of all of CMS!

These don't include the (obvious?) developments related to improvements in the underlying ROOT/RooFit framework which we benefit from greatly (combine v10 is on its way), or support for different platforms (installation via container, CernVM, CVMFS, standalone, on top of StatAnalysis, conda …)

1. Scripts here, there and everywhere

Over the years, there have (naturally) been a lot **of useful scripts/tools** that have been added to the package

- Do things with / to the datacards
- Do things with outputs of combine (plots, tables …)
- Handle larger workflows (Impacts, job submission …)

```
text2workspace.py data/tutorials/counting/realistic-multi-channel.txt
python test/printWorkspaceNormalisations.py data/tutorials/counting/realistic-mult
```

Process Normalizations

Normalisation Values Evaluated at MH = 125

Channel - bin1

Top-level normalization for process bkg -> n_exp_final_binbin1_proc_bkg
▼expand n_exp_final_binbin1_proc_bkg

RooProduct::n_exp_final_binbin1_proc_bkg[n_exp_binbin1_proc_bkg * shapeBkg_bkg_bin1__norm] = 521.163

... is a product, which contains n_exp_binbin1_proc_bkg
▼expand n_exp_binbin1_proc_bkg

RooRealVar::n_exp_binbin1_proc_bkg = 1 C L(-INF - +INF)

default value = 521.163204829

Top-level normalization for process sig -> n_exp_binbin1_proc_sig
▼expand n_exp_binbin1_proc_sig

Dumping ProcessNormalization n_exp_binbin1_proc_sig @ 0x96d9690
nominal value: 1
log-normals (1):
kappa = 1.1, logKappa = 0.0953102, theta = lumi = 0
asymm log-normals (0):
other terms (1):
term r (class RooRealVar), value = 1

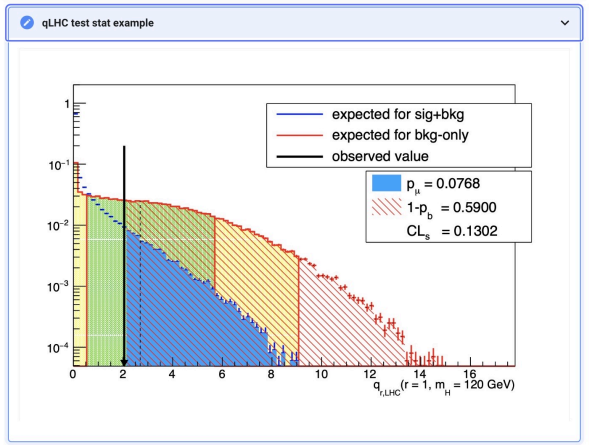
default value = 1.0

```
$ python test/systematicsAnalyzer.py data/tutorials/shapes/simple-shapes-TH1.txt --all -f html > out.html
```

Nuisance Report

Nuisance (types)	Range		Processes	Channels
lumi (lnN)	1.000	1.100	background, signal	bin1(1) [+]
alpha (shape)	1.111	1.150	background	bin1(1) [+]
bgnorm (lnN)	1.000	1.300	background, signal	bin1(1) [+]
sigma (shape)	1.000	1.000	signal	bin1(1) [+]

```
python test/plotTestStatCLs.py --input mygrid.root --poi r --val all --mass MASS
```



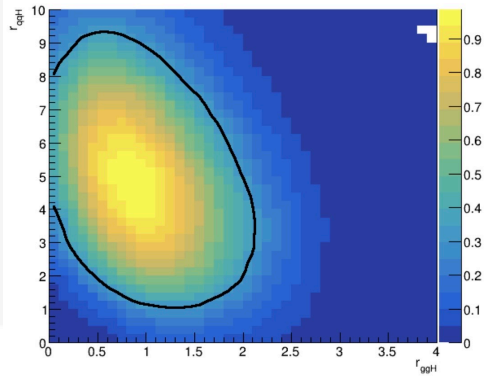
None of these are described in the paper (purposefully omitted) but some are described in the online docs

1. Scripts here, there and everywhere: `combineTool.py`

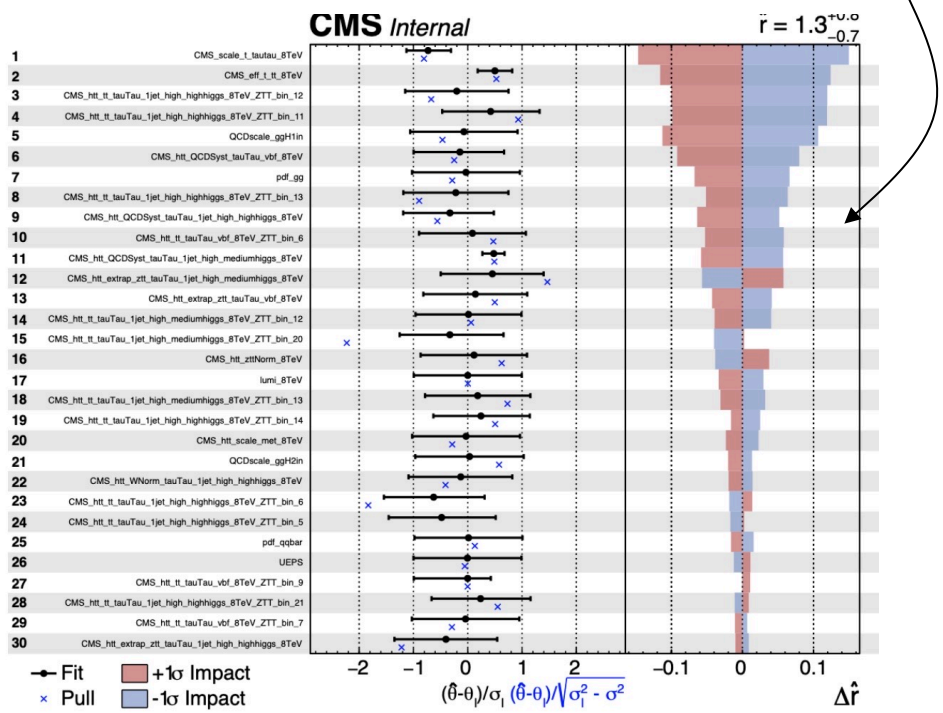
The `combineTool.py*` script is the most powerful and handles a lot of arduous workflows for users

```
combineTool.py -M HybridNewGrid  ./poi_grid_configuration.json -d toy-hgg-125.root
```

```
{
  "verbose" : true,
  "opts" : " --LHCmode LHC-feldman-cousins --saveHybridResult --clsAcc 0 ",
  "POIs" : ["r_ggH", "r_qqH"],
  "grids" : [
    ["0:4|.2","0:10|.5",""]
  ],
  "toys_per_cycle"  : 5000,
  "FC" : true,
  "min_toys": 5000,
  "max_toys": 50000,
  "output_incomplete" : true,
  "make_plots": false,
  "contours":["obs"],
  "CL": 0.68,
  "output": "FeldmanCousins.root",
  "zipfile"         : "collected.zip",
  "statusfile"      : "status.json"
 }
```




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



- It would be useful if all of these useful scripts became part of the main command line interface
- There are plans to move `combineTool` over to the Combine package but as a wrapper, it might make more sense that this becomes the main interface for the user ie:
  `combine --M blah …` → `combineTool(.py) --M blah …`
- Could also facilitate IO captures for better integration into (re)interpretation frameworks?

* Currently this is part of a separate package "`CombineHarvester`" but there are plans to port it to the `CombinedLimit` package

2. Input formats

Combine supports **non-ROOT** based inputs for **template-based** models (counting experiments already require no inputs)

```
imax 1
jmax 1
kmax *
---------------
shapes * * simple-shapes-TH1_input.root $PROCESS $PROCESS_$SYSTEMATIC

bin bin1
observation 85
-------------------------------
bin             bin1       bin1
process         signal     background
process         0          1
rate            10         100
---------------------------------
lumi     lnN    1.10       1.0
bgnorm   lnN    1.00       1.3
alpha  shape    -          1   u
sigma  shape    0.5        -   u
#
```

```
root [0]
Attaching file simple-shapes-TH1.root as _file0...
root [1] _file0->ls()
TFile**     simple-shapes-TH1.root
 TFile*     simple-shapes-TH1.root
  KEY: TH1F signal;1    Histogram of signal__x
  KEY: TH1F signal_sigmaUp;1    Histogram of signal__x
  KEY: TH1F signal_sigmaDown;1  Histogram of signal__x
  KEY: TH1F background;1    Histogram of background__x
  KEY: TH1F background_alphaUp;1    Histogram of background__x
  KEY: TH1F background_alphaDown;1  Histogram of background__x
  KEY: TH1F data_obs;1  Histogram of data_obs__x
  KEY: TH1F data_sig;1  Histogram of data_sig__x
```

```
imax 1
jmax 1
kmax *
-------------------------------------------------------------------------------
shapes * * simple-shapes-df_input.csv $CHANNEL:$PROCESS $CHANNEL:$PROCESS:$SYSTEMATIC
-------------------------------------------------------------------------------
bin         bin1
observation 85
-------------------------------------------------------------------------------
bin             bin1   bin1
process         signal background
process         0      1
rate            10     100
-------------------------------------------------------------------------------
lumi     lnN    1.10   1.0
bgnorm   lnN    1.00   1.3
alpha  shape    -          1   uncert
sigma  shape    0.5        -   uncert
#                                  so
```

	channel	process	systematic	bin	sum_w	sum_ww
2	bin1	background	alphaDown	0.0	30.224781036376953	30.224781036376957
3	bin1	background	alphaDown	1.0	20.260276794433594	20.26027679443359
4	bin1	background	alphaDown	2.0	13.580870628356934	13.580870628356934
5	bin1	background	alphaDown	3.0	9.103529930114746	9.103529930114746
6	bin1	background	alphaDown	4.0	6.102278709411621	6.102278709411621
7	bin1	background	alphaDown	5.0	4.090479373931885	4.090479373931884
8	bin1	background	alphaDown	6.0	2.7419304847717285	2.741930484771729
9	bin1	background	alphaDown	7.0	1.8379708528518677	1.837970852851868
10	bin1	background	alphaDown	8.0	1.2320287227630615	1.2320287227630617
11	bin1	background	alphaDown	9.0	0.8258535861968994	0.8258535861968994

- I would like to see more use of these alternative input forms (`.json` also supported)
- **ROOT JSON for serialization** would be a natural way to specify inputs to extend to parametric models
  - Note this is different to fully specifying the statistical model in JSON (discuss)
  - Some custom CMS objects (`RooParametricHist, RooMultiPdf, ...`) need to be thought about

2. Input formats – interface?

Datacard mostly acts as a map between objects (eg in a ROOT file), and values to pdfs that should be constructed

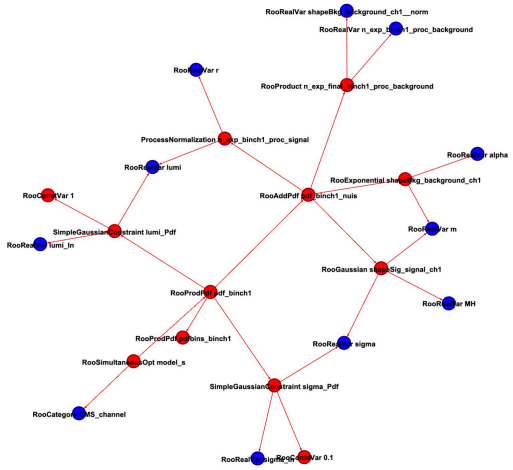
It's possible to simply construct the binary workspace directly by skipping the Datacard parsing step

```
text2workspace.py <datacard.txt>
--dump-datacard > myscript.py
```

No need to write a `.txt` Datacard file since one can just fill/edit the python dictionaries/lists in a (re)interpretation workflow

`myscript.py`

```
from HiggsAnalysis.CombinedLimit.DatacardParser import *
from HiggsAnalysis.CombinedLimit.ModelTools import *
from HiggsAnalysis.CombinedLimit.ShapeTools import *
from HiggsAnalysis.CombinedLimit.PhysicsModel import *

from sys import exit
from optparse import OptionParser
parser = OptionParser()
addDatacardParserOptions(parser)
options,args = parser.parse_args()
options.bin = True # make a binary workspace

DC = Datacard()
MB = None

############## Setup the datacard (must be filled in) ###########################

DC.bins =   ['bin1'] # <type 'list'>
DC.obs =    {'bin1': 0.0} # <type 'dict'>
DC.processes =  ['ggH', 'qqWW', 'ggWW', 'others'] # <type 'list'>
DC.signals =    ['ggH'] # <type 'list'>
DC.isSignal =   {'qqWW': False, 'ggWW': False, 'ggH': True, 'others': False} # <typ
DC.keyline =    [('bin1', 'ggH', True), ('bin1', 'qqWW', False), ('bin1', 'ggWW', F
DC.exp =    {'bin1': {'qqWW': 0.63, 'ggWW': 0.06, 'ggH': 1.47, 'others': 0.22}} # <
DC.systs =  [('lumi', False, 'lnN', [], {'bin1': {'qqWW': 0.0, 'ggWW': 1.11, 'ggH':
DC.shapeMap =   {} # <type 'dict'>
DC.hasShapes =  False # <type 'bool'>
DC.flatParamNuisances =  {} # <type 'dict'>
DC.rateParams =  {} # <type 'dict'>
DC.extArgs =    {} # <type 'dict'>
DC.rateParamsOrder  =  set([]) # <type 'set'>
DC.frozenNuisances  =  set([]) # <type 'set'>
DC.systematicsShapeMap =  {} # <type 'dict'>
DC.nuisanceEditLines    =  [] # <type 'list'>
DC.groups   =  {} # <type 'dict'>
DC.discretes    =  [] # <type 'list'>


###### User defined options ################################################

options.out      = "combine_workspace.root"     # Output workspace name
options.fileName = "./"             # Path to input ROOT files
options.verbose  = "1"              # Verbosity

###########################################################################

if DC.hasShapes:
    MB = ShapeBuilder(DC, options)
else:
    MB = CountingModelBuilder(DC, options)

# Set physics models
MB.setPhysics(defaultModel)
MB.doModel()
```



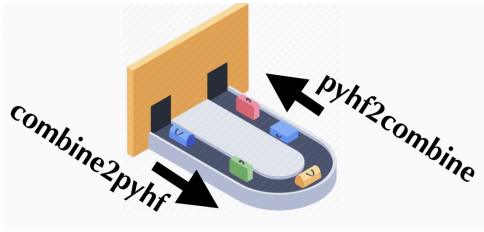
3. Interpretors

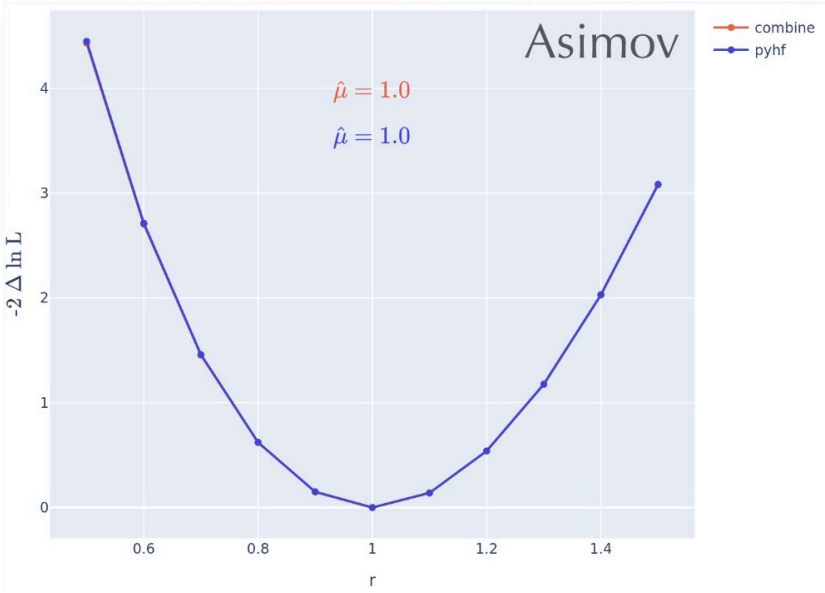
Combine provides the implementation of the model and performs the calculations but work on going to support **other tools** to do this (no ROOT dependence) …

K. Skovpen

E.G 1: `pyHF`

- For simple template-based modes, able to translate between Combine Datacard and pyHF JSON formats
- Can use pyHF interpretation tools to calculate limits etc from CMS datacards





E.G 2: `combinetf`

- Extension to use TensorFlow for computational graph (+autograd functionality) by converting Datacard to hdf5 format
- Performance boost for extremely complicated template models (many bins)
- Restricted to template-based methods only and not documented (or widely used yet) in CMS

	Likelihood	Likelihood+Gradient	Hessian
Combine, TR1950X 1 Thread	10ms	830ms	-
TF, TR1950X 1 Thread	70ms	430ms	165s
TF, TR1950X 32 Thread	20ms	71ms	32s
TF, 2x Xeon Silver 4110 32 Thread	17ms	54ms	24s
TF, GTX1080	7ms	13ms	10s
TF, V100	4ms	7ms	8s

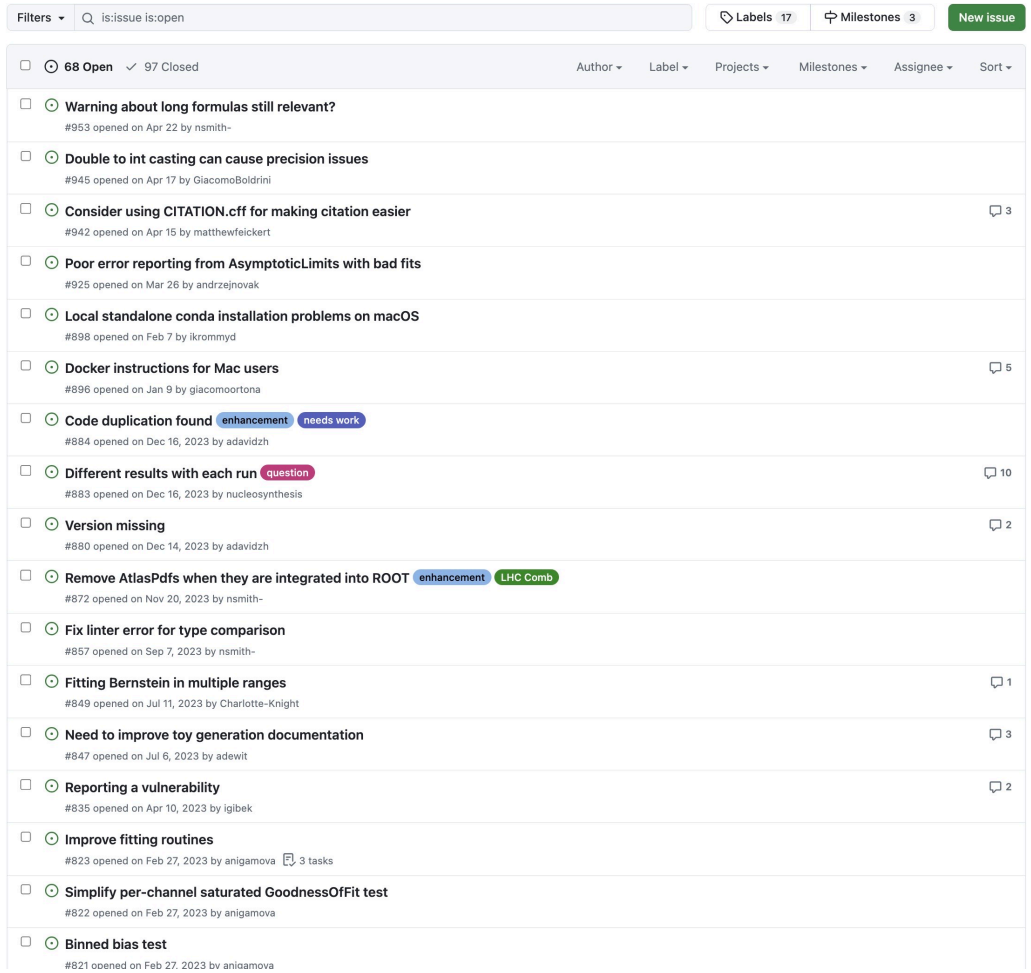
J. Bendavid

4. Lists of (other) issues

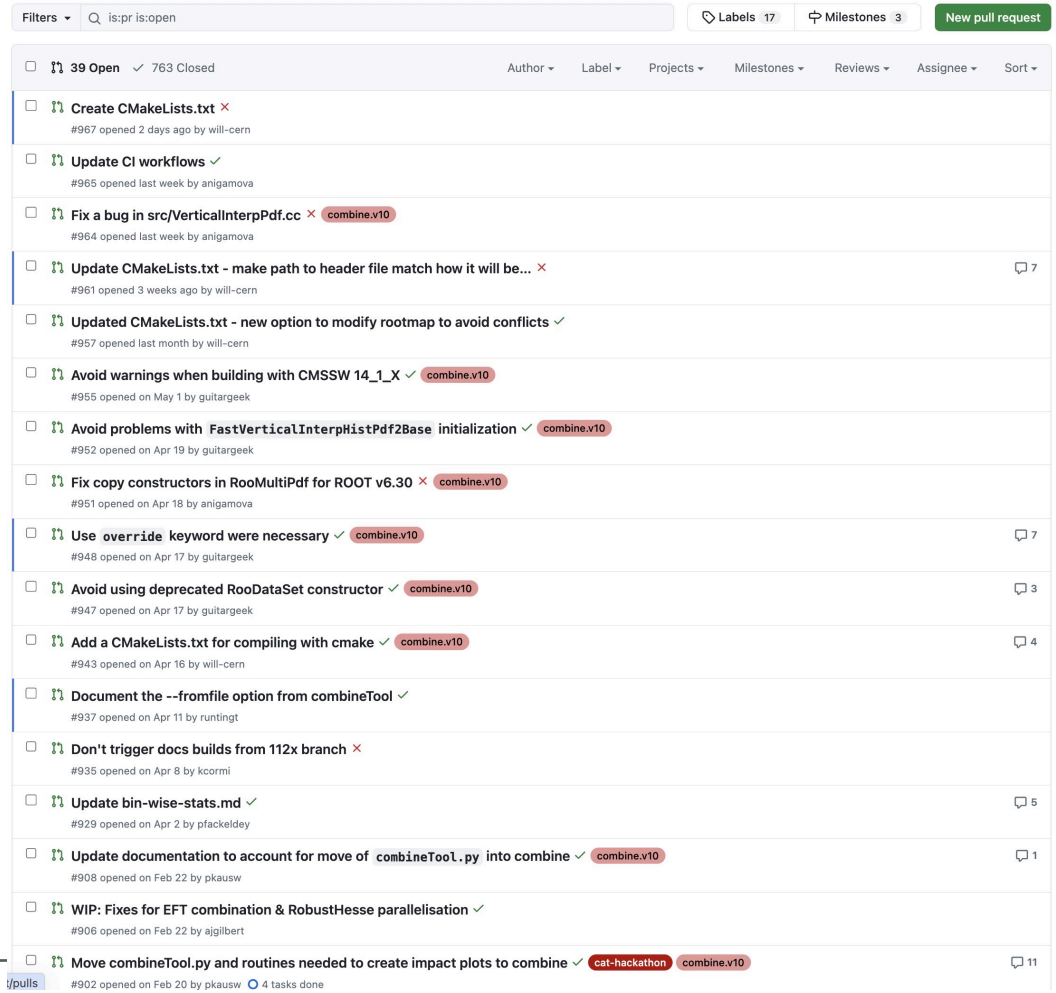
Without going into detail, there are **lots of things that need doing**

→ Lots of room for improvement and publishing statistical models will help development from wider community

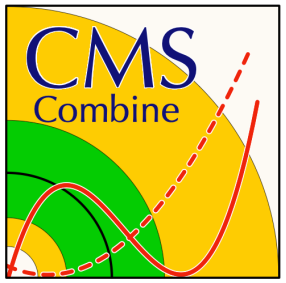
[Issues](#)



[PRs](#)



Summary



Combine is more or less THE statistics tool in CMS

- Datacard + inputs → defines statistical model, Combine constructs it and performs statistical calculations
- Counting, template and parametric modes supported, different physics models and different calculations from the same underlying inputs

Future developments ongoing

- Restructuring of the software package in discussion: Less scripts, more consolidation of workflow management
- Input formats: I think Physics model should stay separate from underlying objects but ROOT JSON could replace our typical inputs and could be used to store the constructed workspace too
- Interpretors: Users might not want to use combine, plain RooFit, pyHF, combinetf should be supported looking forward

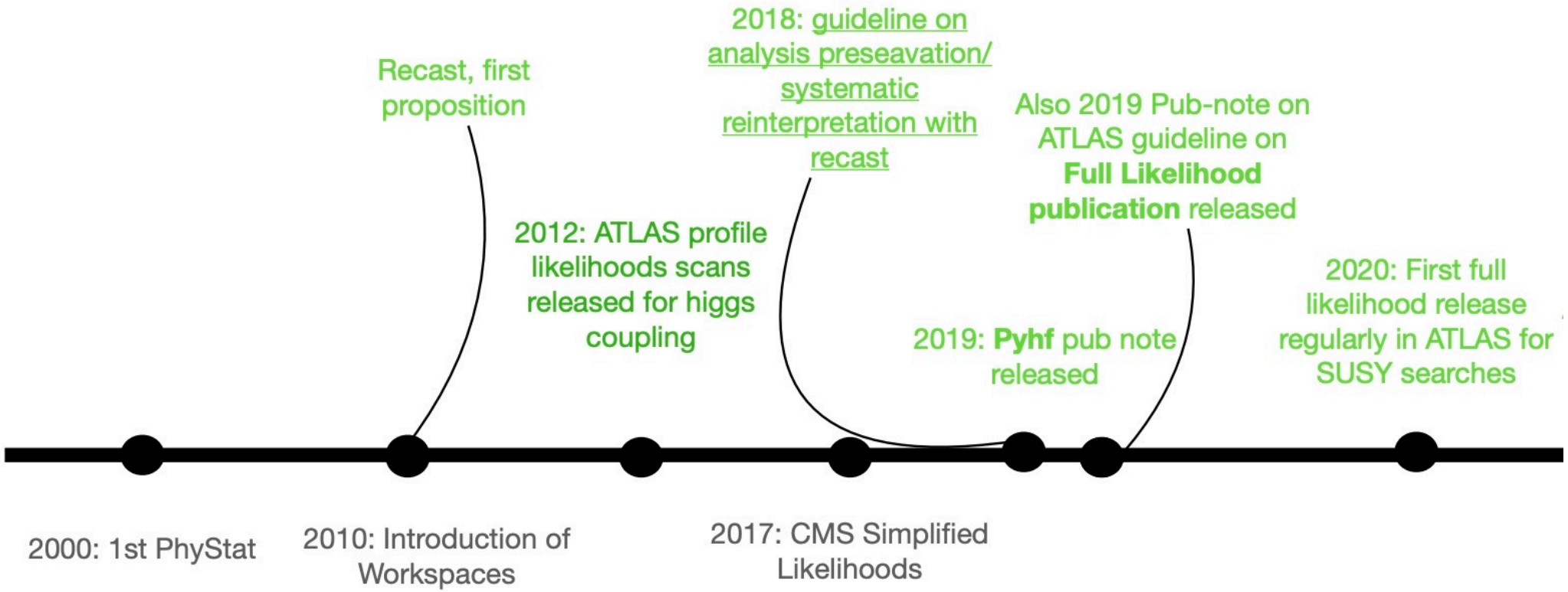
Plenty of room for improvement

- Combine has evolved over a period >10 years since before the Higgs discovery so naturally there are things to be updated/improved
- I have a long list of issues that irritate me and need fixing (ask me about `-t` vs `-T` !)
- However, tool is still a big part of doing CMS analyses and its still (imo) a good tool for statistics

Thanks!

Backup Slides

Timeline for public likelihoods



2024 CMS public statistics models + Combine

L. Heinrich

Template based shape analysis

Observed number of events in channel c and bin b

$$\mathcal{L}(\text{data}|\mathbf{c},\theta,\rho) = \underbrace{\prod_{ch=c}\prod_{bin=k}\frac{(N_{ck}(\mathbf{c},\theta,\rho))^{n_{ck}}}{n_{ck}!}e^{-N_{ck}(\mathbf{c},\theta,\rho)}}_{\text{Poisson}} \times \overbrace{\prod_{syst=j}\pi(\tilde{\theta}_j|\theta_j)}^{\text{Nuisances}}$$

Likelihood

Nuisances affecting shapes are interpolated via fractional bin contents (f)

$$y_b(\theta) = \begin{cases} \sum_b y_b^{nom}\left[f_b^{nom} + \sum_s F\left(\theta_s, \delta_b^{s,+}, \delta_b^{s,-}\right)\right] \\ \sum_b y_b^{nom}\left[\exp\left(\ln f_b^{nom}\right) + \sum_s F\left(\theta_s, \Delta_b^{s,+}, \Delta_b^{s,-}\right)\right] \end{cases}$$

$$\delta_b^{\pm} = f_b^{\pm} - f_b^{nom}, \quad \Delta_b^{\pm} = \ln f_b^{\pm} - \ln f_b^{nom}$$

$$F(\theta,\delta^+,\delta^-) = \begin{cases} \frac{1}{8}\theta'((\theta^+ - \theta^-) + \frac{1}{8}(\delta^+ + \delta^-)(3\bar{\theta}^5 - 10\bar{\theta}^3 + 15\bar{\theta})) & \theta < q \\ \theta'\delta^+ & \theta > q \\ -\theta'\delta^- & \theta < -q \end{cases}$$

$N_{ck}(\mathbf{c},\theta,\rho)$ = Expected number of events in channel c and bin b

POI scaling · Norm · Templates · MC stat

$$N_{ck}(\mathbf{c},\theta,\rho) = \sum_{proc=p} M_{cp}(\mathbf{c}) \times Nr_{cp}(\theta,\rho) \times y_{cbp}(\theta) + E_{cb}(\theta)$$

$$M_{cp}(\mathbf{c}) = \{1|c_i|c_i^2|c_ic_j|\ldots\} \text{ (EFT case)}$$

Gamma · Asymmetric lnN

$$Nr_{cp}(\theta,\rho) = \overbrace{N_{0,cp}(\theta_G)}^{\text{Gamma}} \underbrace{\prod_n k_n^{\theta_L,n}}_{\text{lnN}} \overbrace{\prod_a k_a^A(\theta_{L(S)}^a, k_a^+, k_a^-)}^{\text{Asymmetric lnN}} \underbrace{\prod_r F_r(\rho)}_{\text{rateParams}}$$

Interpolation between up and down variations on normalization of process p

$$k^A = \{k^+ \text{ if } \theta \geq 0.5,\ k^- \text{ if } \theta \leq 0.5,\ \exp\left(\tfrac{1}{8}\left[4\ln(k^+/k^-) + \ln(k^+k^-)(48\theta^5 - 40\theta^3 + 15\theta)\right]\right)\}$$

Statistical uncertainties in MC templates handled via Barlow Beeston approach

$$E_{cb}(\theta) = \theta\left(\sum_p \left(e_{cbp}^2 Nr_{cp} M_{cp}(\mathbf{c})\right)^2\right)^{0.5}$$

Schema from A. Gilbert

-t vs -T

Never ending source of confusion is the difference between –t and –T in combine

Reason is that there are "levels" of toy generation with combine

-t supports ~all methods

→ I want to generate a toy dataset to make plots with/perform a test of a fit …

→ useful as lots of extra diagnostic info can be saved (toys themselves, fit results + propagated yields/bin contents for each toy …)

-T is used specifically for toy-based methods

→ i.e anything using –M HybridNew (limits, p-values, FC intervals)

→ DetailedOutput implemented in certain test-statistics only?

The type of toy generation is defaulted differently between the two

-t → Hybrid by default, frequentist if you add –toysFreq

-T → frequentist by default if using --LHCmode <X> , can use Hybrid if specify –genNuisances

There's even another toy generation for sampling from post-fit covariances for making plots inside FitDiagnostics ☹

Of course, all of this is documented as such but some consolidation