DUNE OA using GUNDAM

GUNDAM Oscillation Analysis Tools

Ciro Riccio on behalf of Tristan Doyle, Vlada Yevarouskaya
and Clark McGrew

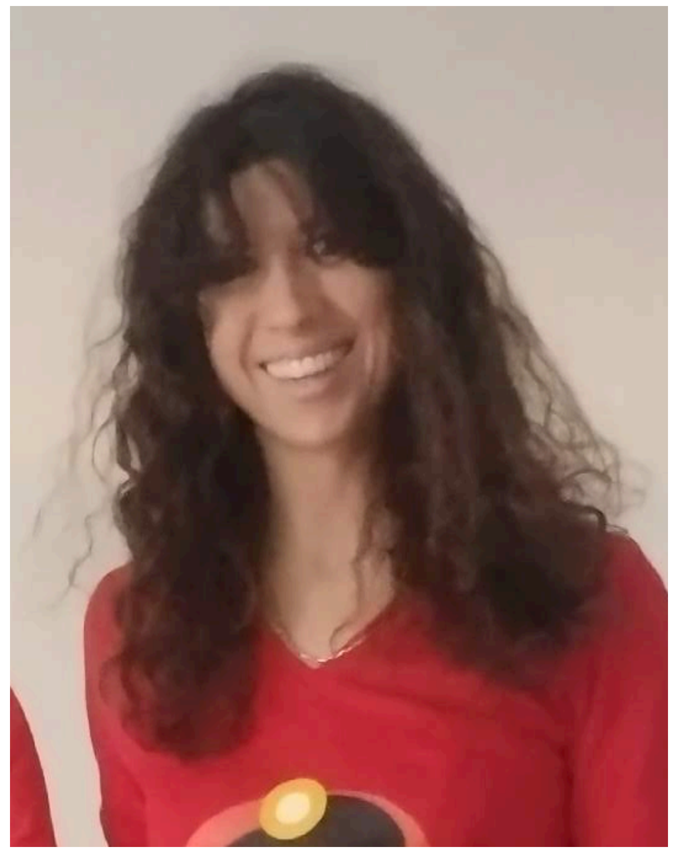
DUNE LBL meeting
August, 19th 2024

Stony Brook
University

The GUNDAM group

Adrien Blanchet (CERN)

Margherita Buizza Avanzini (LLR)



Stephen Dolan (CERN)

Laura Munteanu (CERN)

Vlada Yevarouskaya (SBU)

Tristan Doyle (SBU)

Ciro Riccio (SBU)

Clark McGrew (SBU)

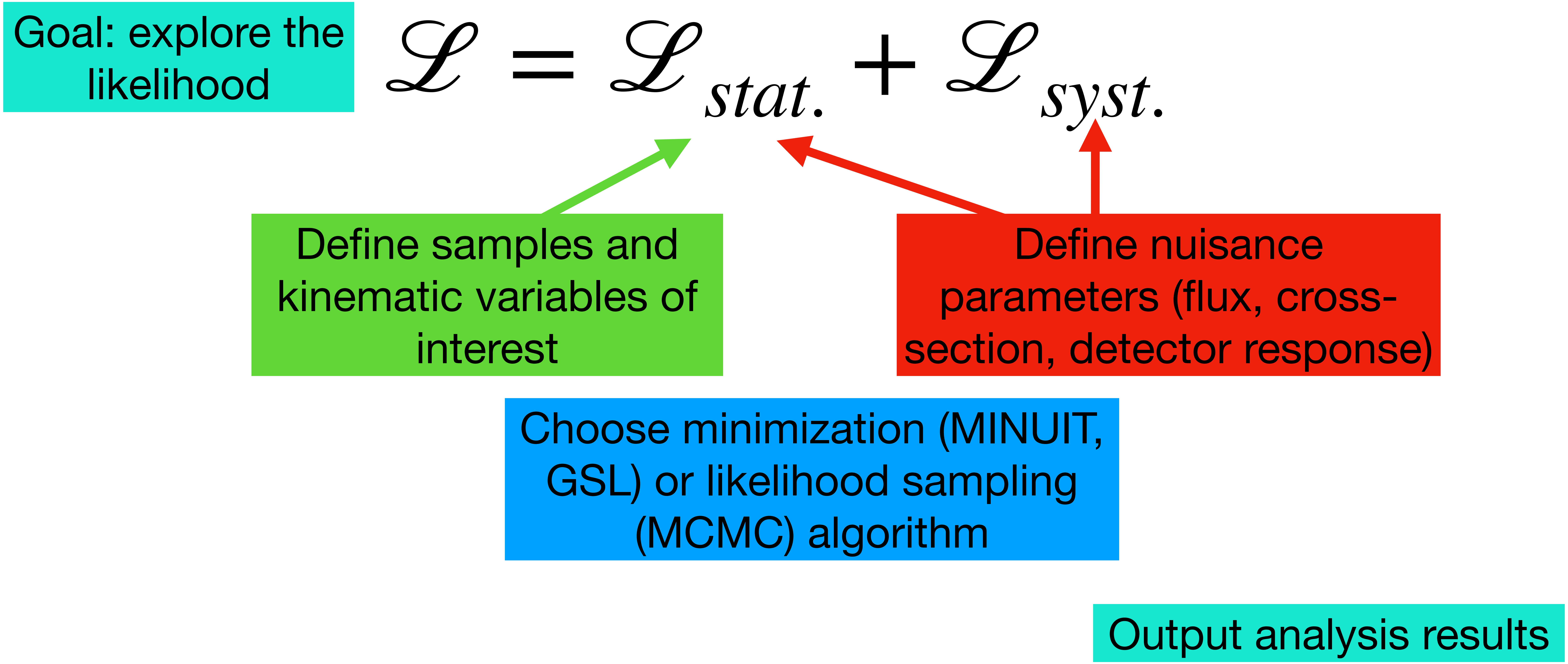
Other contributors and users:

- Yassine Alj Hakim
- Jafaar Chakrani
- Eric Chong
- Benjamin Demarles
- Lorenzo Giannessi
- Jiayu Ji
- Shivam Joshi
- Andres Munoz
- Lena Osu
- Casper Schloesser
- Adam Speer
- Ulysse Virginet
- Joe Walsh
- Rowan Zaki
- Xinguy Zhao

What is GUNDAM?

- GUNDAM, which stand for *Generalized and Unified Neutrino Data Analysis Methods*, is an analysis tool developed in the context of the upgrade of the off-axis near detector of T2K
- It is a synthesis of two tools used in the T2K cross-section and OA WGs (xsLLhFitter and BANFF)
- Its goal is to explore a likelihood function that can be defined by using a set of YAML/JSON configuration files
- Code structure designed for handling different analyses and as much as possible analysis independent
- Code optimized for both CPUs and GPUs
  - 10s - 100s Hz depending on the analysis, system and if using CPUs only or GPUs
  - Example: analysis with ~4000 sample bins and ~600 parameters few hours (~3d previous generation tool)
- Collaborative code developed on GitHub under same GPL as ROOT

GUNDAM Workflow



Goal: explore the likelihood

$$\mathscr{L} = \mathscr{L}_{stat.} + \mathscr{L}_{syst.}$$

Define samples and kinematic variables of interest

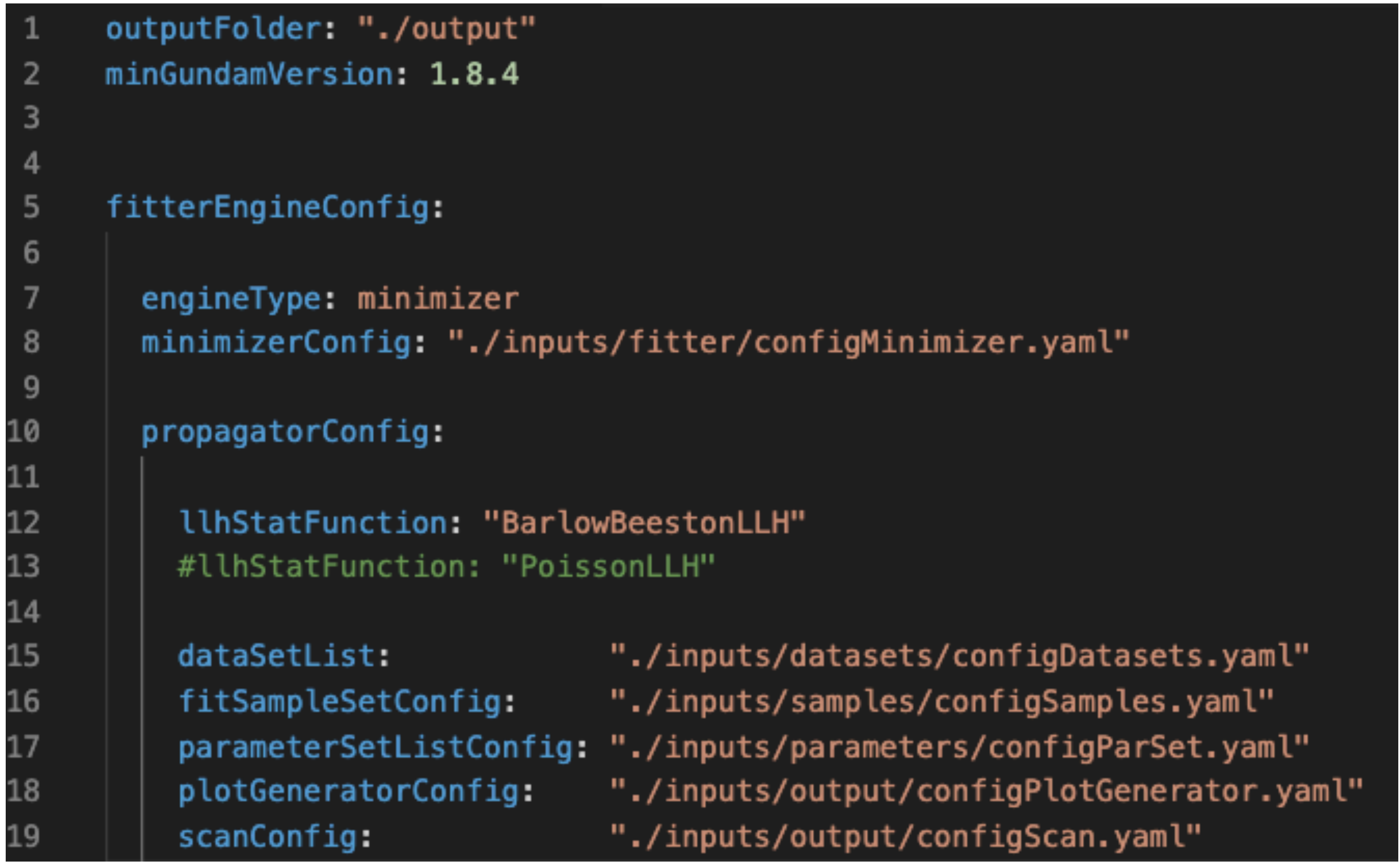
Define nuisance parameters (flux, cross-section, detector response)

Choose minimization (MINUIT, GSL) or likelihood sampling (MCMC) algorithm

Output analysis results

Config examples

mainConfig.yaml



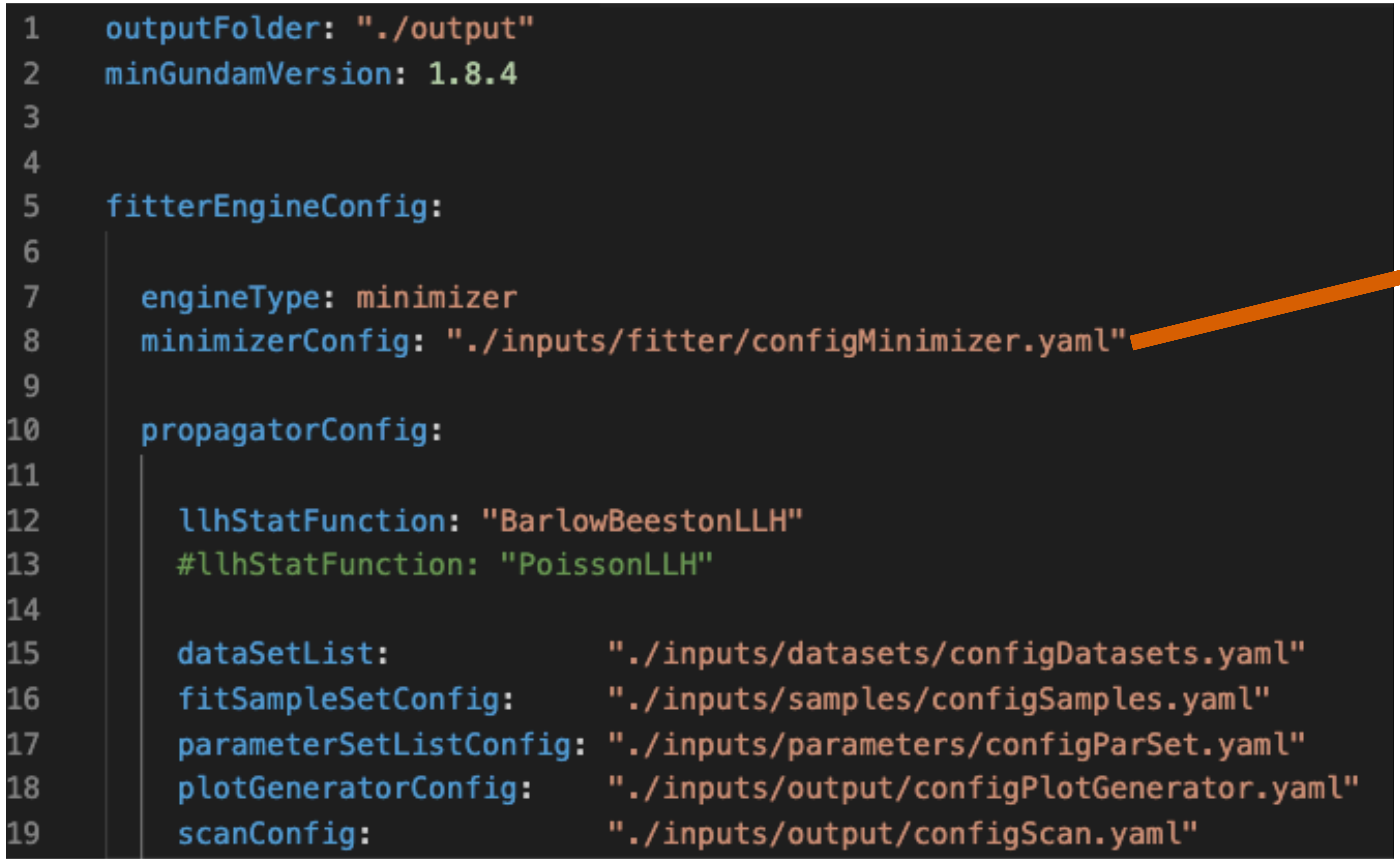
```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:             "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:      "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:     "./inputs/output/configPlotGenerator.yaml"
    scanConfig:              "./inputs/output/configScan.yaml"
```

Convenient to break this is several files but doesn't have to be

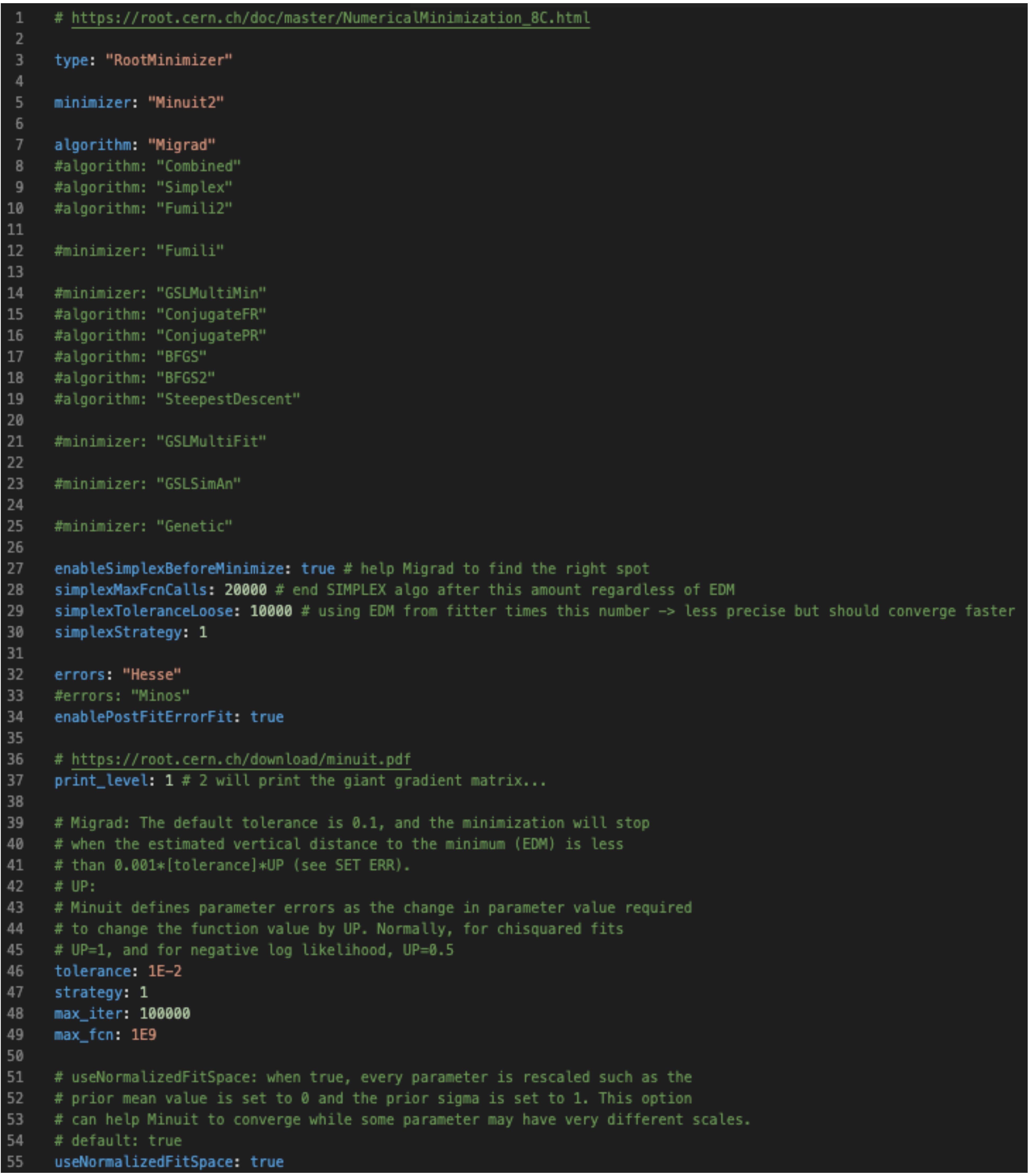
Note: illustrative only examples

Engine configuration

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```

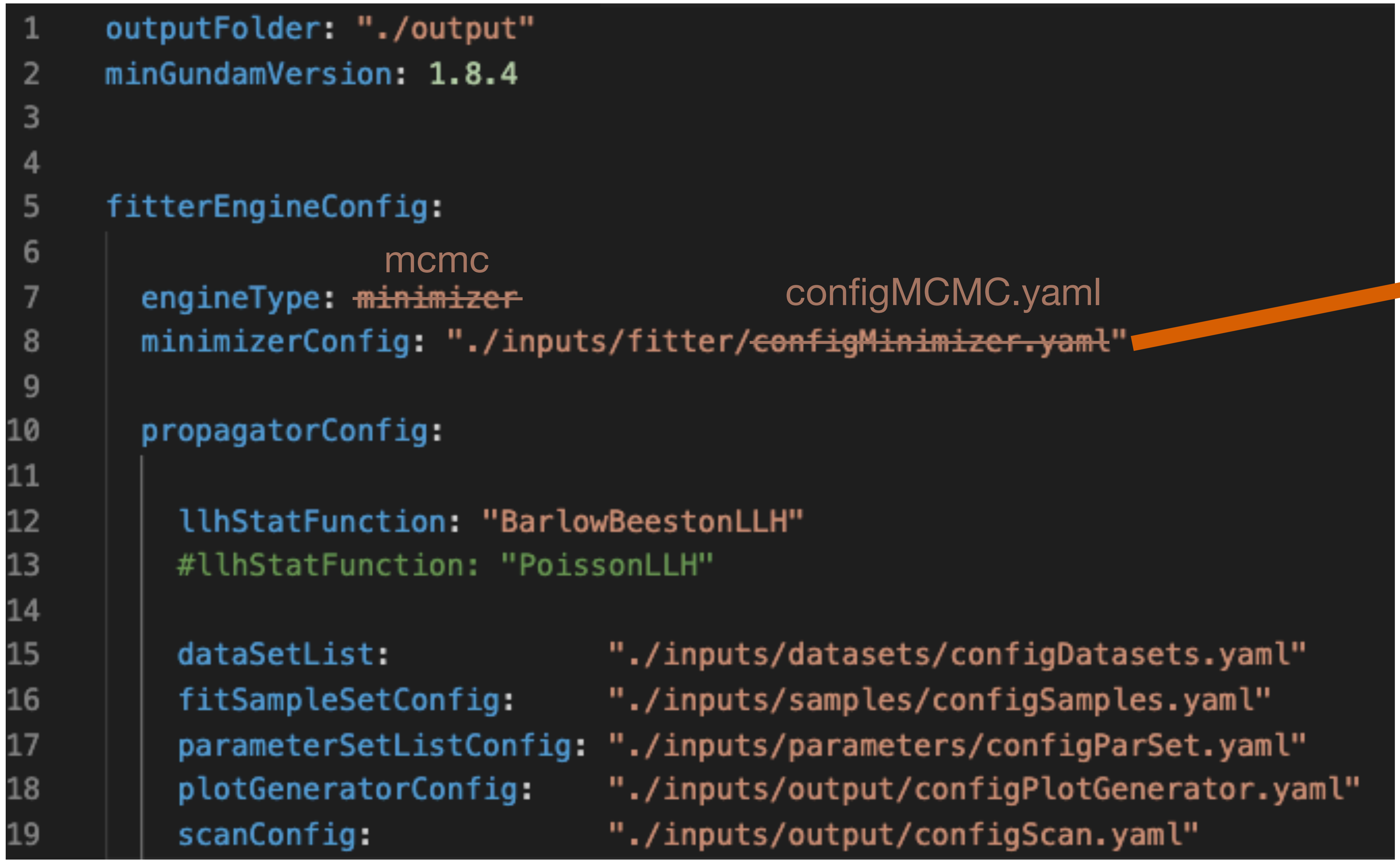


```yaml
# https://root.cern.ch/doc/master/NumericalMinimization_8C.html

type: "RootMinimizer"

minimizer: "Minuit2"

algorithm: "Migrad"
#algorithm: "Combined"
#algorithm: "Simplex"
#algorithm: "Fumili2"

#minimizer: "Fumili"

#minimizer: "GSLMultiMin"
#algorithm: "ConjugateFR"
#algorithm: "ConjugatePR"
#algorithm: "BFGS"
#algorithm: "BFGS2"
#algorithm: "SteepestDescent"

#minimizer: "GSLMultiFit"

#minimizer: "GSLSimAn"

#minimizer: "Genetic"

enableSimplexBeforeMinimize: true # help Migrad to find the right spot
simplexMaxFcnCalls: 20000 # end SIMPLEX algo after this amount regardless of EDM
simplexToleranceLoose: 10000 # using EDM from fitter times this number -> less precise but should converge faster
simplexStrategy: 1

errors: "Hesse"
#errors: "Minos"
enablePostFitErrorFit: true

# https://root.cern.ch/download/minuit.pdf
print_level: 1 # 2 will print the giant gradient matrix...

# Migrad: The default tolerance is 0.1, and the minimization will stop
# when the estimated vertical distance to the minimum (EDM) is less
# than 0.001*[tolerance]*UP (see SET ERR).
# UP:
# Minuit defines parameter errors as the change in parameter value required
# to change the function value by UP. Normally, for chisquared fits
# UP=1, and for negative log likelihood, UP=0.5
tolerance: 1E-2
strategy: 1
max_iter: 100000
max_fcn: 1E9

# useNormalizedFitSpace: when true, every parameter is rescaled such as the
# prior mean value is set to 0 and the prior sigma is set to 1. This option
# can help Minuit to converge while some parameter may have very different scales.
# default: true
useNormalizedFitSpace: true
```

Note: illustrative only examples

Engine configuration

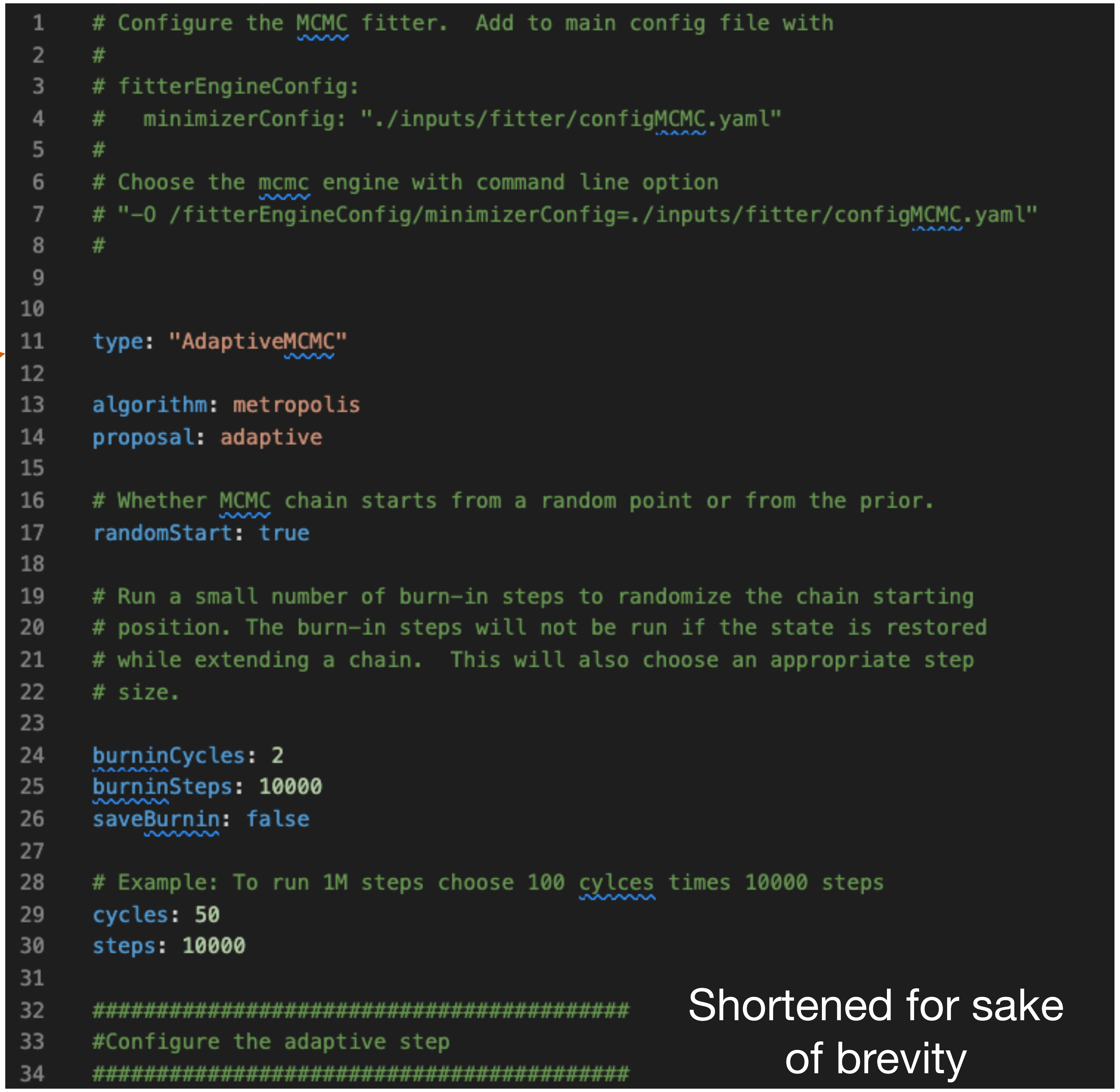
mainConfig.yaml



mcmc

configMCMC.yaml

```
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: ~~minimizer~~
  minimizerConfig: "./inputs/fitter/~~configMinimizer.yaml~~"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:             "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:      "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:     "./inputs/output/configPlotGenerator.yaml"
    scanConfig:              "./inputs/output/configScan.yaml"
```



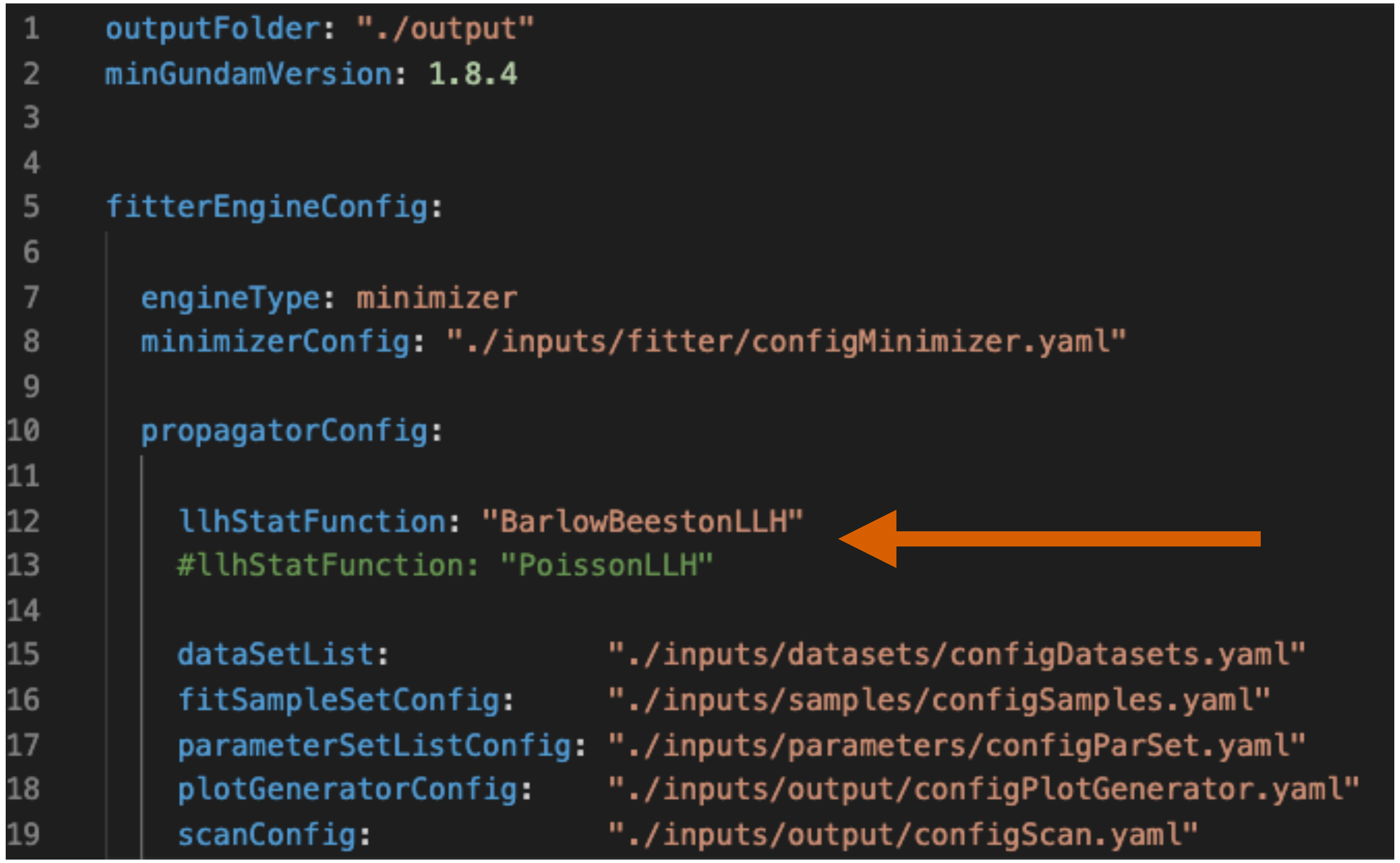
```
# Configure the MCMC fitter.  Add to main config file with
#
# fitterEngineConfig:
#   minimizerConfig: "./inputs/fitter/configMCMC.yaml"
#
# Choose the mcmc engine with command line option
# "-O /fitterEngineConfig/minimizerConfig=./inputs/fitter/configMCMC.yaml"
#



type: "AdaptiveMCMC"

algorithm: metropolis
proposal: adaptive

# Whether MCMC chain starts from a random point or from the prior.
randomStart: true

# Run a small number of burn-in steps to randomize the chain starting
# position. The burn-in steps will not be run if the state is restored
# while extending a chain.  This will also choose an appropriate step
# size.

burninCycles: 2
burninSteps: 10000
saveBurnin: false

# Example: To run 1M steps choose 100 cylces times 10000 steps
cycles: 50
steps: 10000

##########################################
#Configure the adaptive step
##########################################
```

Shortened for sake of brevity

Note: illustrative only examples

Likelihood definition

mainConfig.yaml



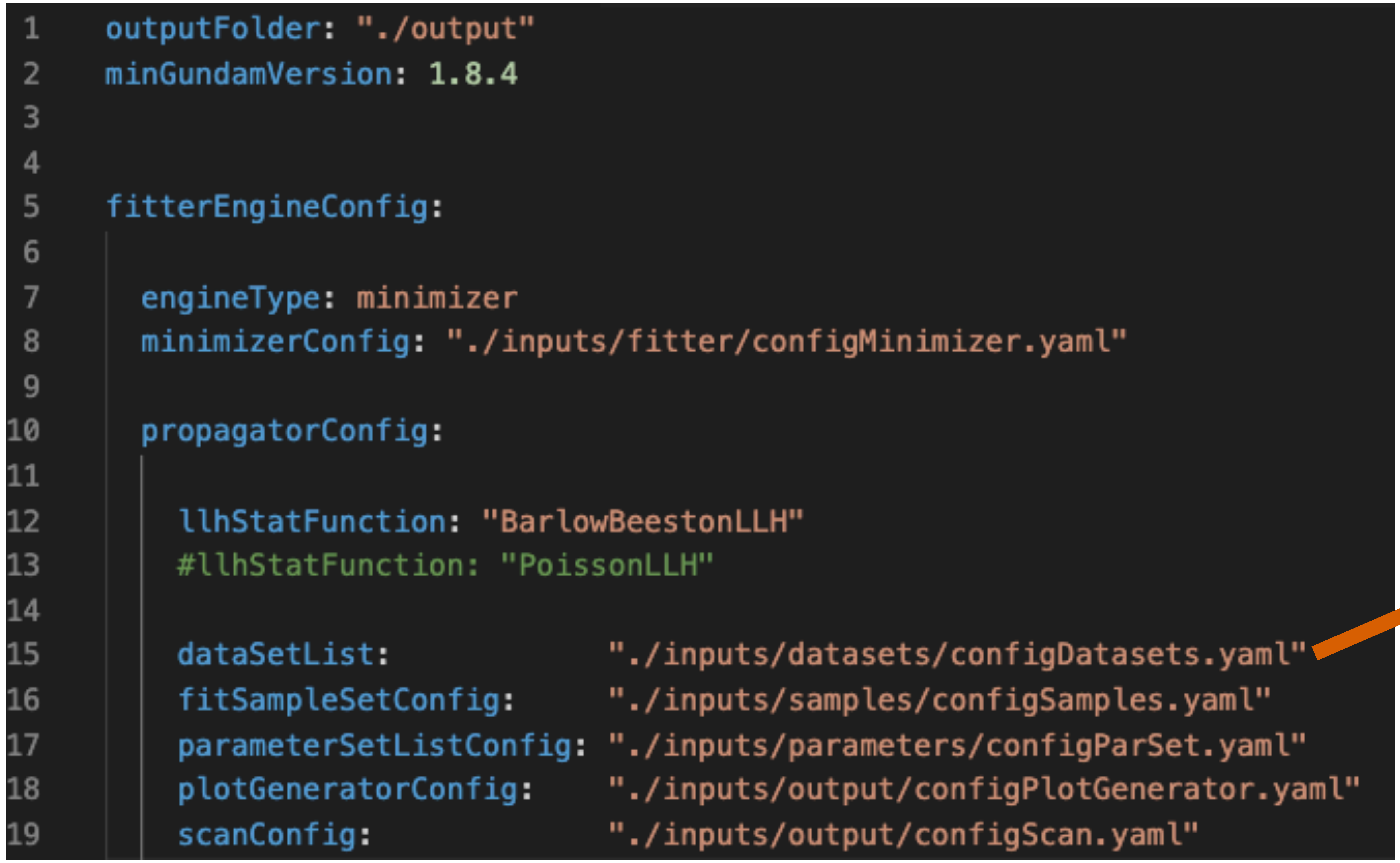
```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```

Different test statistics implemented in the core code

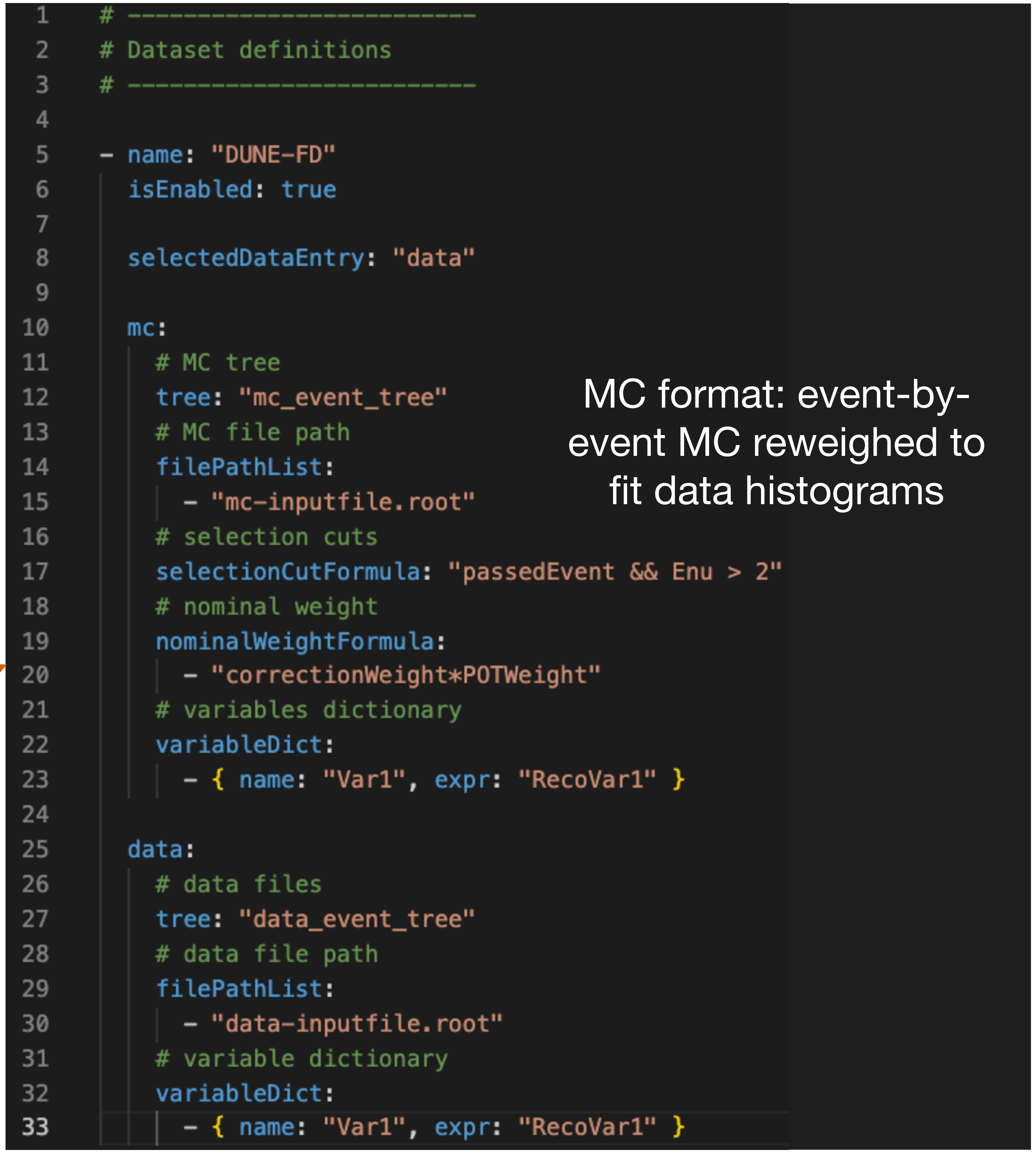
Note: illustrative only examples

Dataset definition

mainConfig.yaml



```
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```



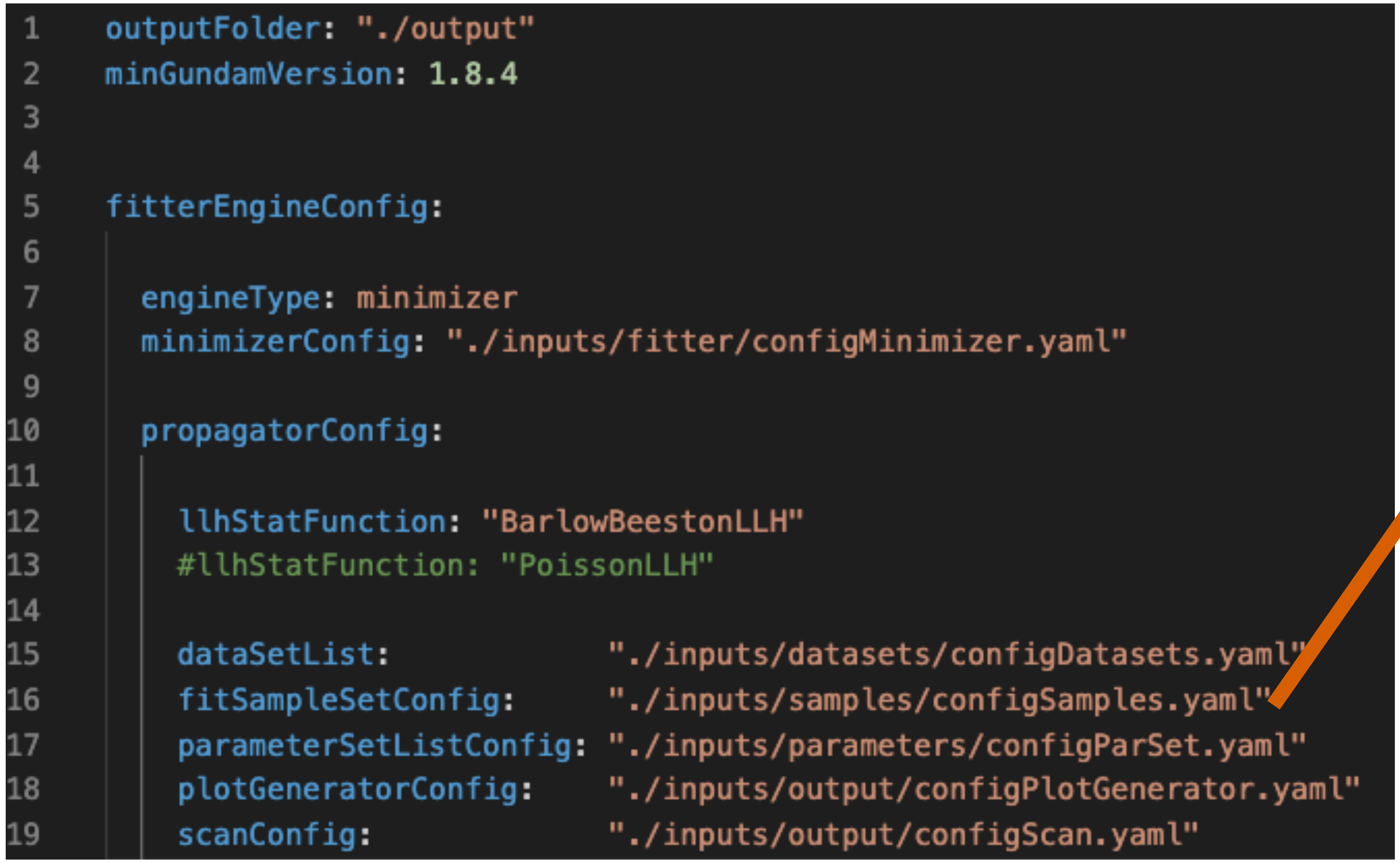
```
# -------------------------
# Dataset definitions
# -------------------------

- name: "DUNE-FD"
  isEnabled: true

  selectedDataEntry: "data"

  mc:
    # MC tree
    tree: "mc_event_tree"
    # MC file path
    filePathList:
      - "mc-inputfile.root"
    # selection cuts
    selectionCutFormula: "passedEvent && Enu > 2"
    # nominal weight
    nominalWeightFormula:
      - "correctionWeight*POTWeight"
    # variables dictionary
    variableDict:
      - { name: "Var1", expr: "RecoVar1" }

  data:
    # data files
    tree: "data_event_tree"
    # data file path
    filePathList:
      - "data-inputfile.root"
    # variable dictionary
    variableDict:
      - { name: "Var1", expr: "RecoVar1" }
```

MC format: event-by-event MC reweighed to fit data histograms

Note: illustrative only examples

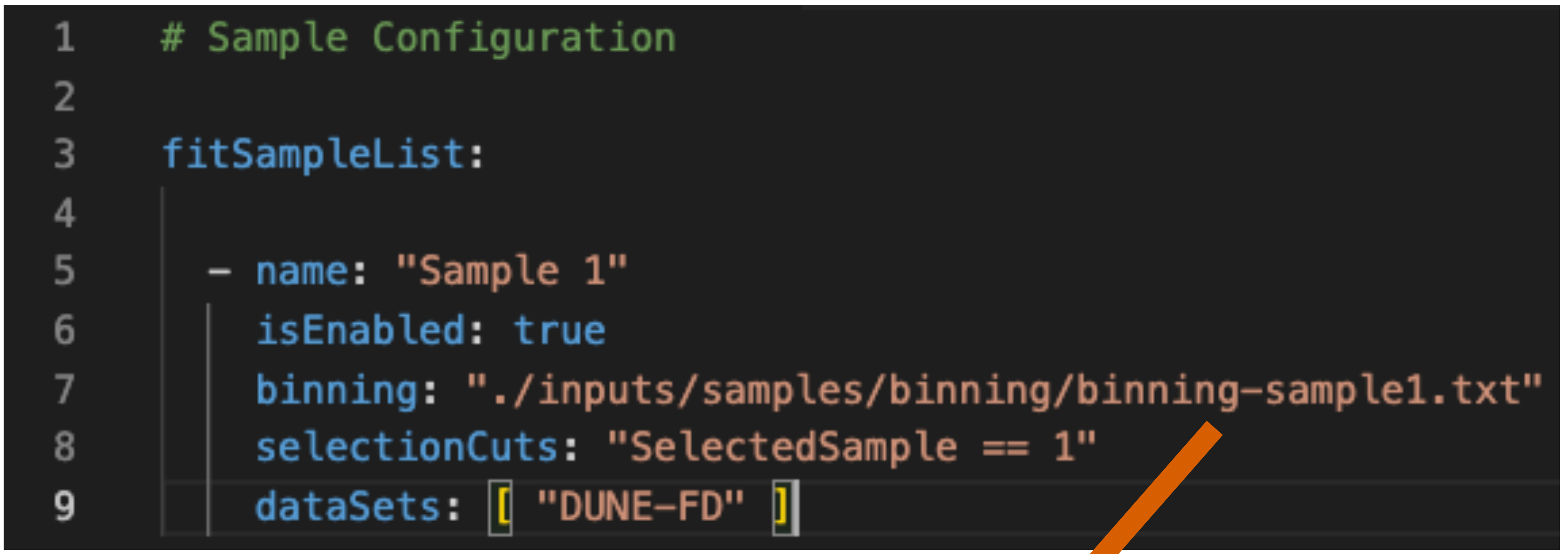
Samples definition

mainConfig.yaml

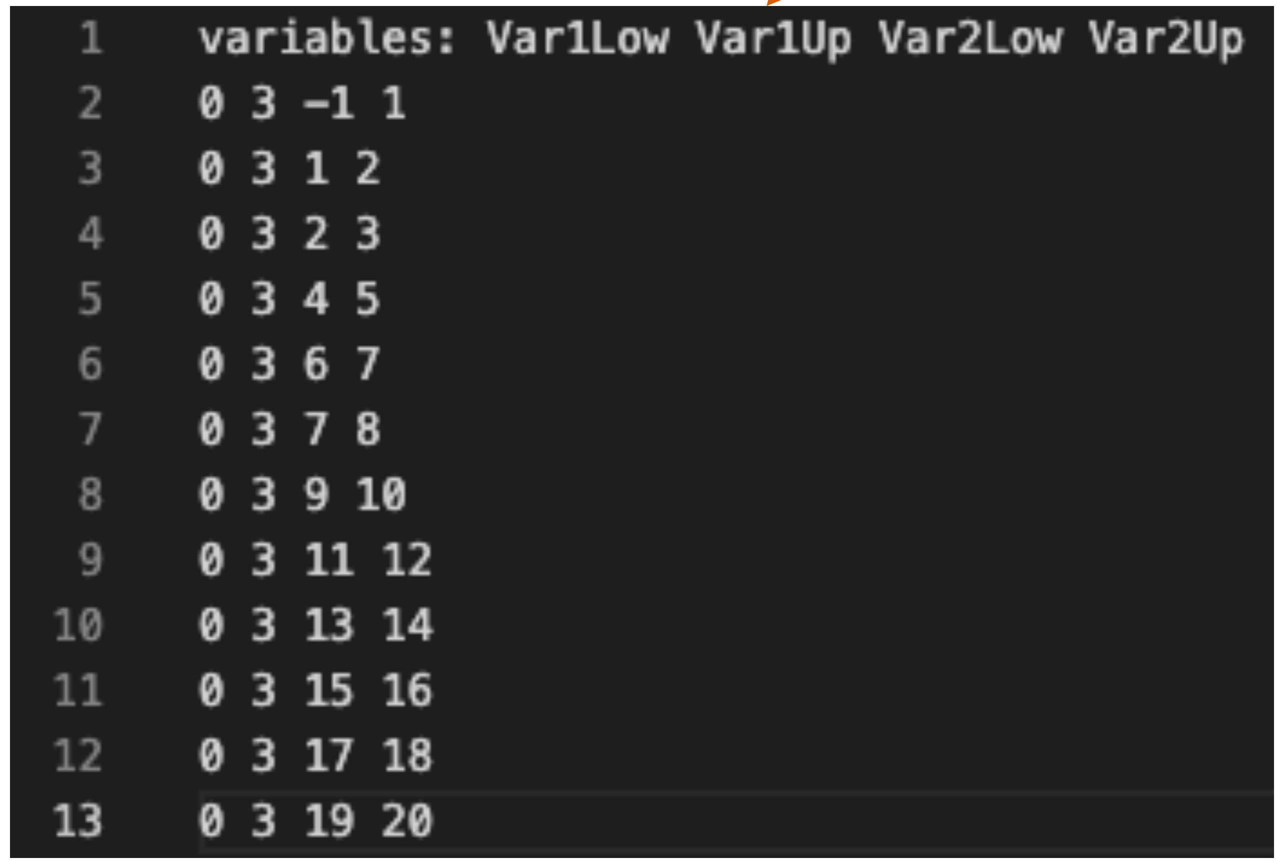


```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```

Any number of samples



```yaml
# Sample Configuration

fitSampleList:

  - name: "Sample 1"
    isEnabled: true
    binning: "./inputs/samples/binning/binning-sample1.txt"
    selectionCuts: "SelectedSample == 1"
    dataSets: [ "DUNE-FD" ]
```



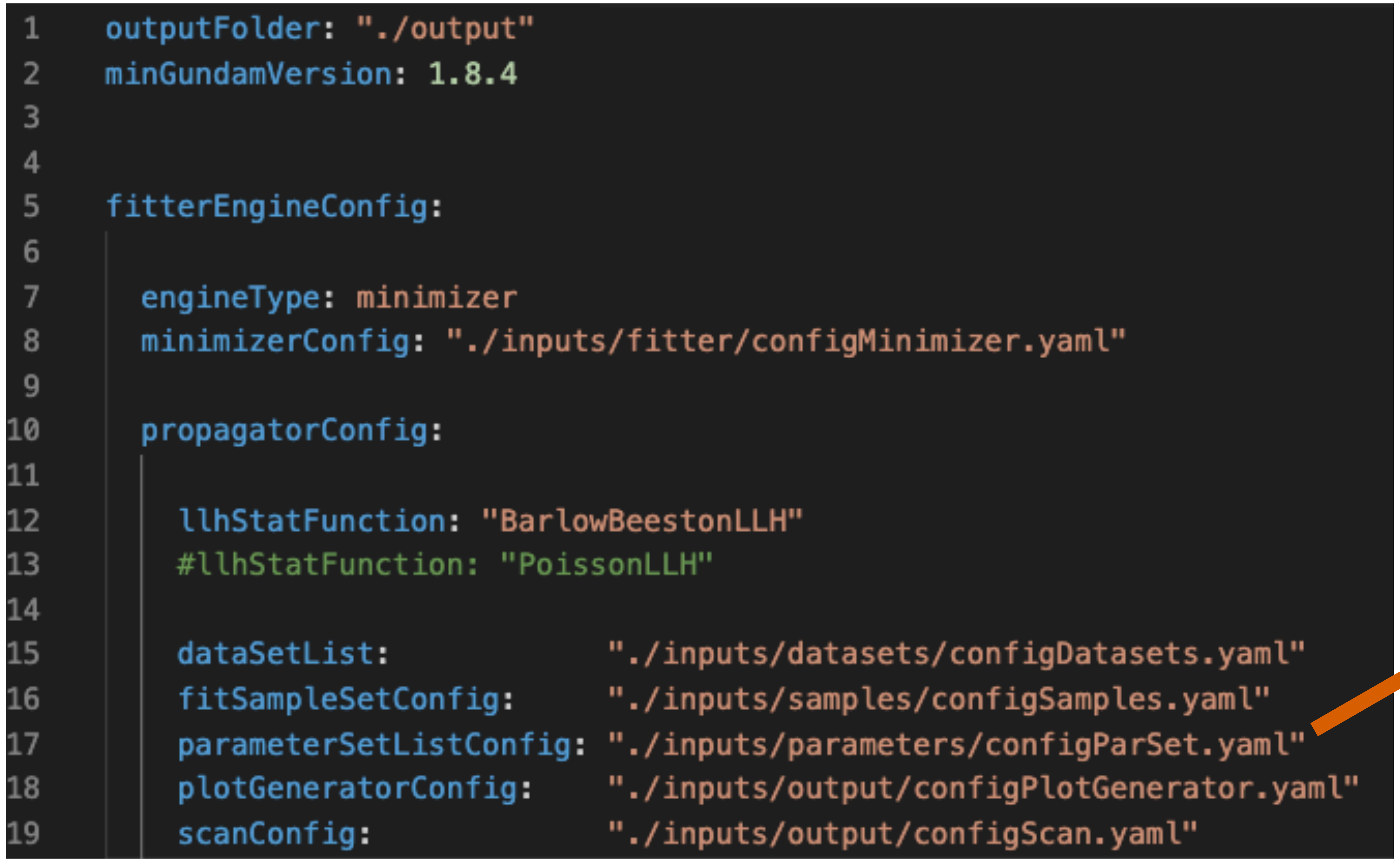
```
variables: Var1Low Var1Up Var2Low Var2Up
0 3 -1 1
0 3 1 2
0 3 2 3
0 3 4 5
0 3 6 7
0 3 7 8
0 3 9 10
0 3 11 12
0 3 13 14
0 3 15 16
0 3 17 18
0 3 19 20
```

Bin in any number of dimension

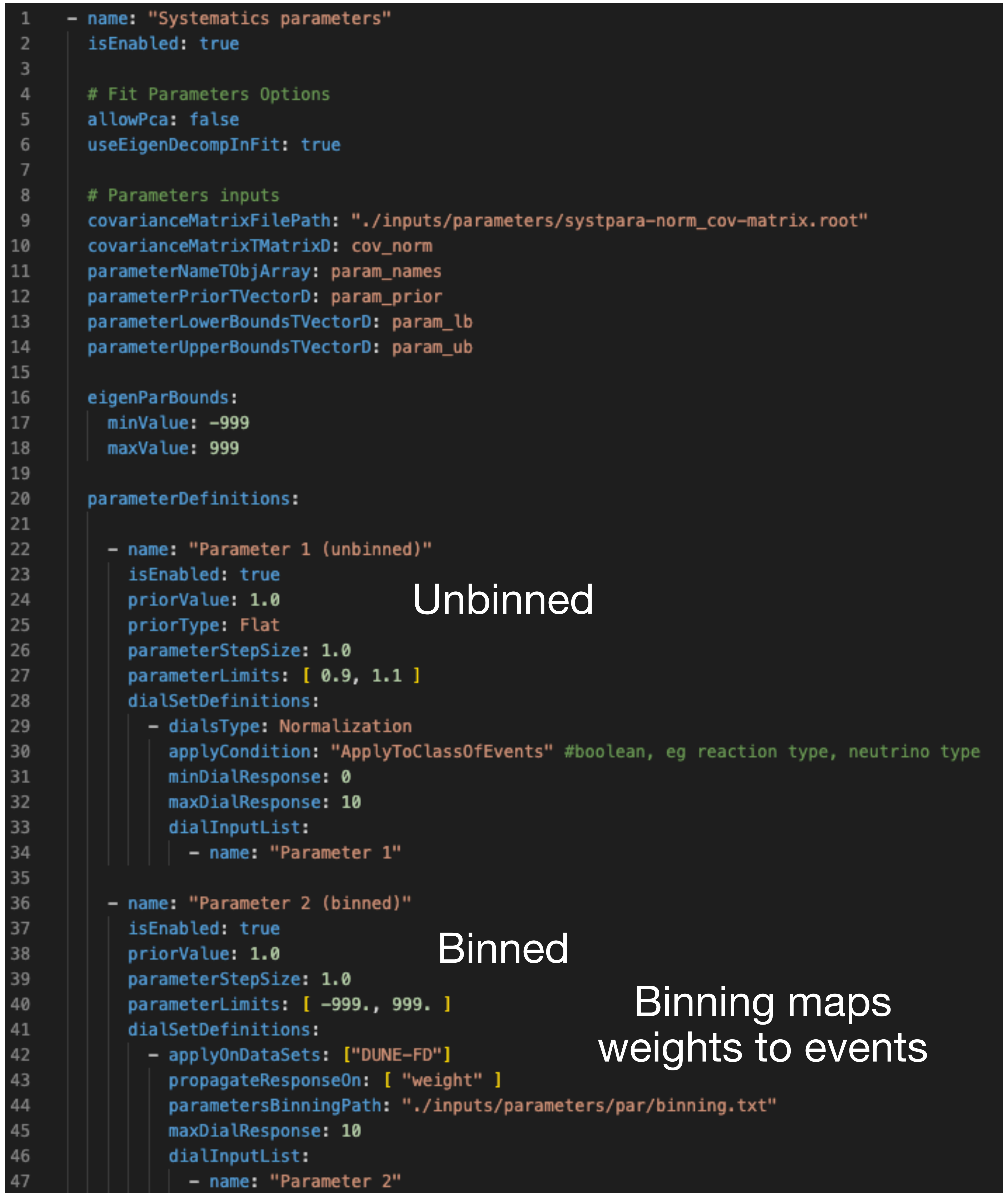
Note: illustrative only examples

Parameter set: normalization

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```



```yaml
- name: "Systematics parameters"
  isEnabled: true

  # Fit Parameters Options
  allowPca: false
  useEigenDecompInFit: true

  # Parameters inputs
  covarianceMatrixFilePath: "./inputs/parameters/systpara-norm_cov-matrix.root"
  covarianceMatrixTMatrixD: cov_norm
  parameterNameTObjArray: param_names
  parameterPriorTVectorD: param_prior
  parameterLowerBoundsTVectorD: param_lb
  parameterUpperBoundsTVectorD: param_ub

  eigenParBounds:
    minValue: -999
    maxValue: 999

  parameterDefinitions:

    - name: "Parameter 1 (unbinned)"
      isEnabled: true
      priorValue: 1.0
      priorType: Flat
      parameterStepSize: 1.0
      parameterLimits: [ 0.9, 1.1 ]
      dialSetDefinitions:
        - dialsType: Normalization
          applyCondition: "ApplyToClassOfEvents" #boolean, eg reaction type, neutrino type
          minDialResponse: 0
          maxDialResponse: 10
          dialInputList:
            - name: "Parameter 1"

    - name: "Parameter 2 (binned)"
      isEnabled: true
      priorValue: 1.0
      parameterStepSize: 1.0
      parameterLimits: [ -999., 999. ]
      dialSetDefinitions:
        - applyOnDataSets: ["DUNE-FD"]
          propagateResponseOn: [ "weight" ]
          parametersBinningPath: "./inputs/parameters/par/binning.txt"
          maxDialResponse: 10
          dialInputList:
            - name: "Parameter 2"
```

Unbinned

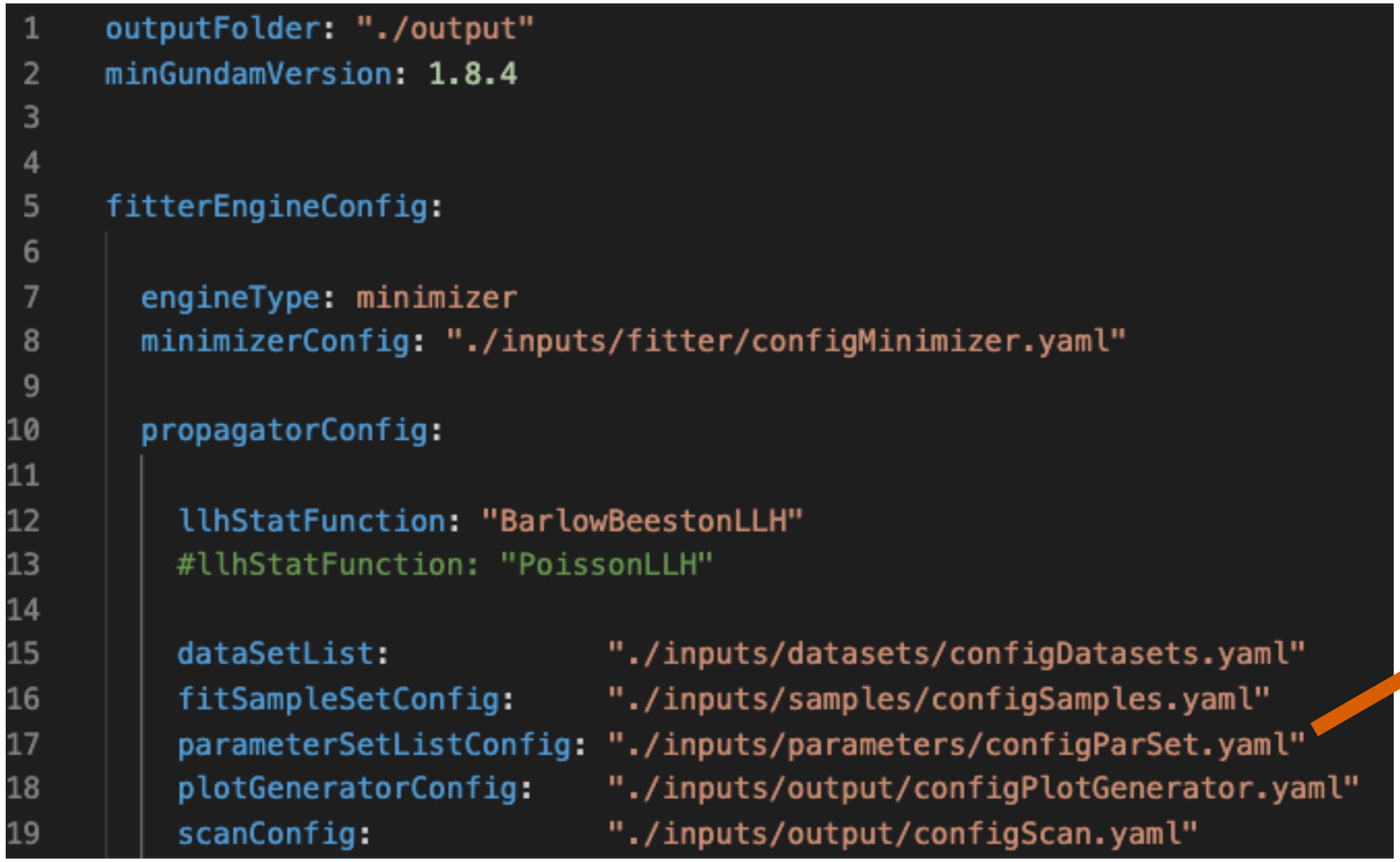
Binned

Binning maps weights to events

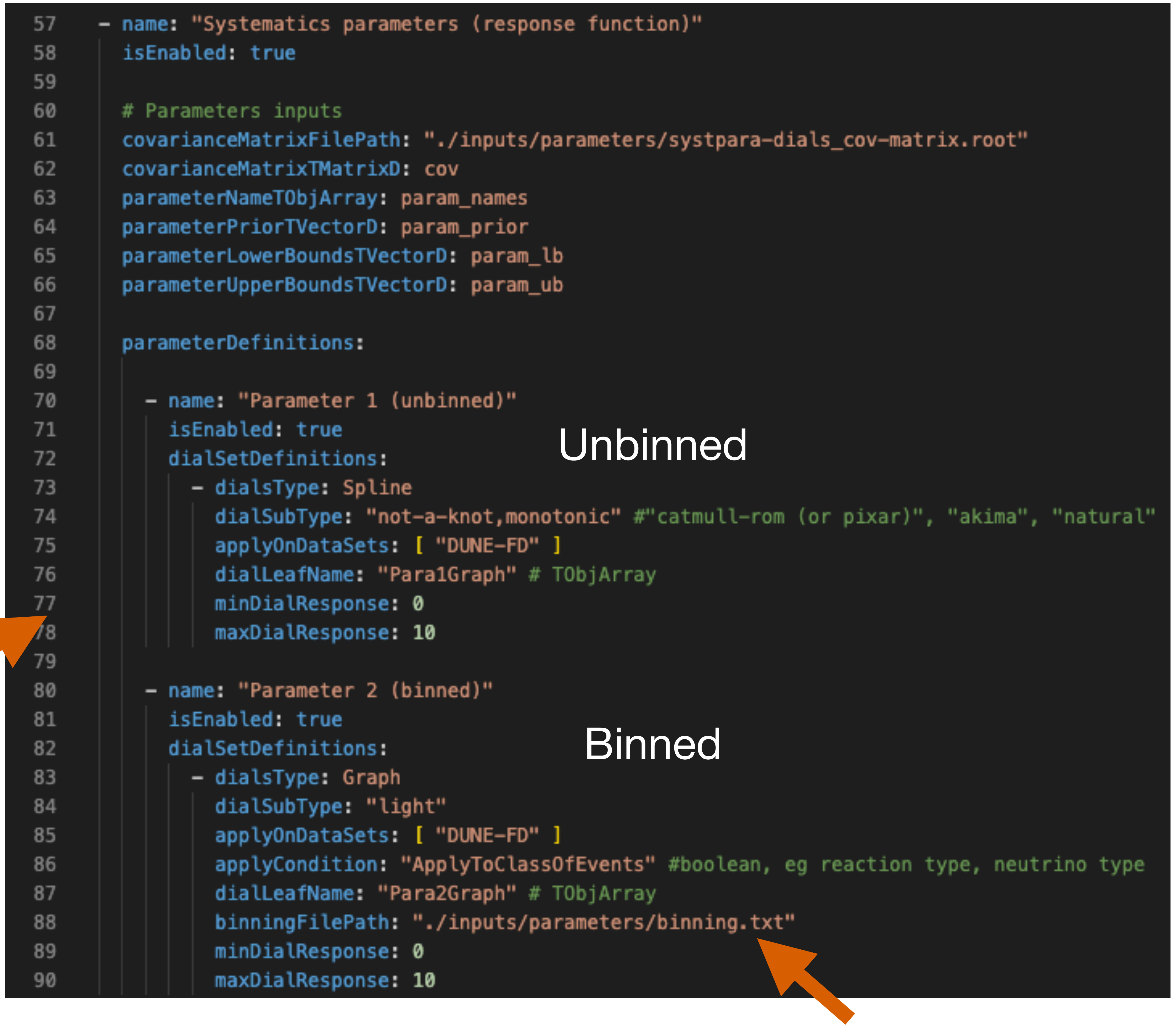
Note: illustrative only examples

Parameter set: response function

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```



```yaml
- name: "Systematics parameters (response function)"
  isEnabled: true

  # Parameters inputs
  covarianceMatrixFilePath: "./inputs/parameters/systpara-dials_cov-matrix.root"
  covarianceMatrixTMatrixD: cov
  parameterNameTObjArray: param_names
  parameterPriorTVectorD: param_prior
  parameterLowerBoundsTVectorD: param_lb
  parameterUpperBoundsTVectorD: param_ub

  parameterDefinitions:

    - name: "Parameter 1 (unbinned)"
      isEnabled: true
      dialSetDefinitions:
        - dialsType: Spline
          dialSubType: "not-a-knot,monotonic" #"catmull-rom (or pixar)", "akima", "natural"
          applyOnDataSets: [ "DUNE-FD" ]
          dialLeafName: "Para1Graph" # TObjArray
          minDialResponse: 0
          maxDialResponse: 10

    - name: "Parameter 2 (binned)"
      isEnabled: true
      dialSetDefinitions:
        - dialsType: Graph
          dialSubType: "light"
          applyOnDataSets: [ "DUNE-FD" ]
          applyCondition: "ApplyToClassOfEvents" #boolean, eg reaction type, neutrino type
          dialLeafName: "Para2Graph" # TObjArray
          binningFilePath: "./inputs/parameters/binning.txt"
          minDialResponse: 0
          maxDialResponse: 10
```

Unbinned

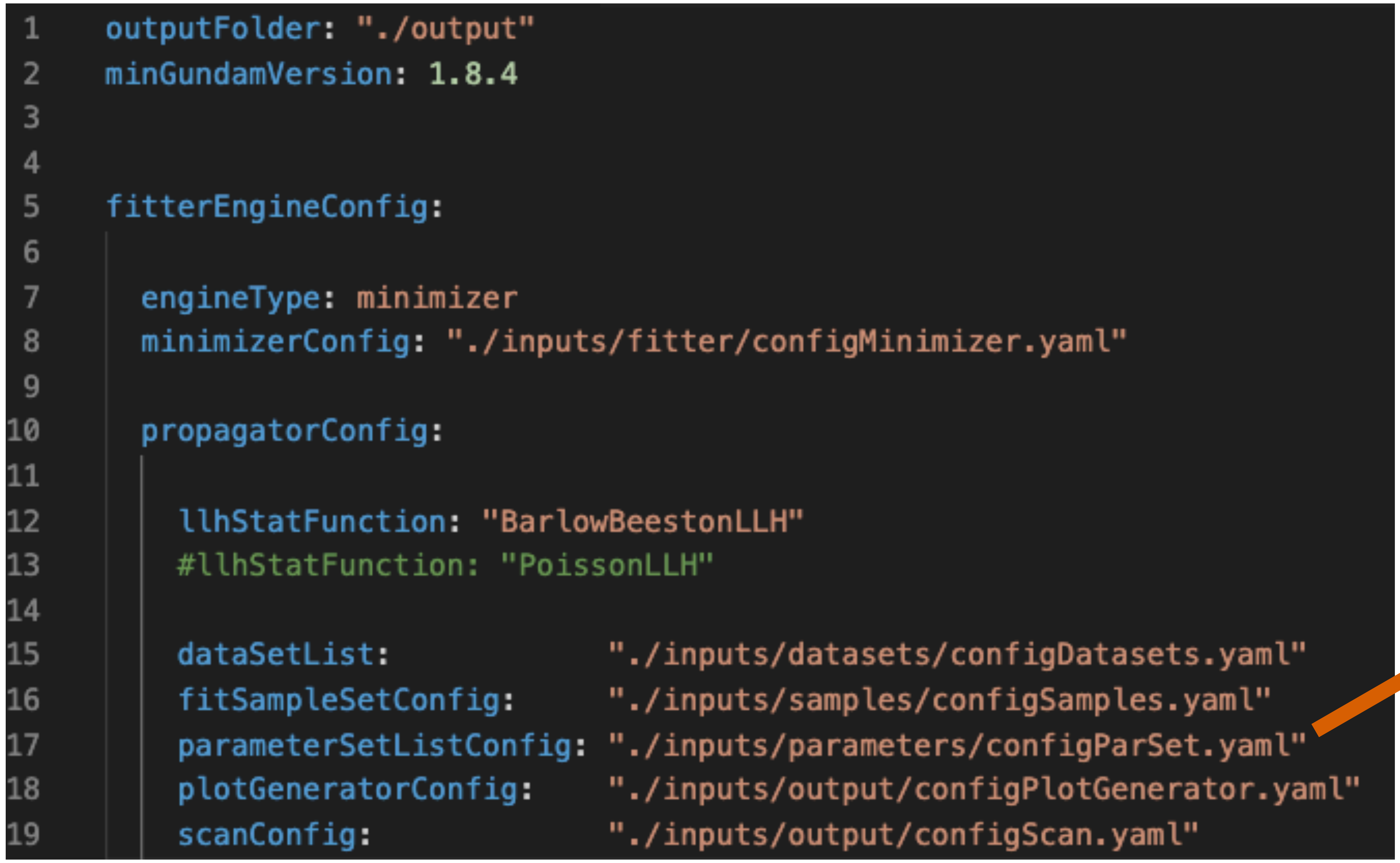
Binned

Binning maps response functions to events

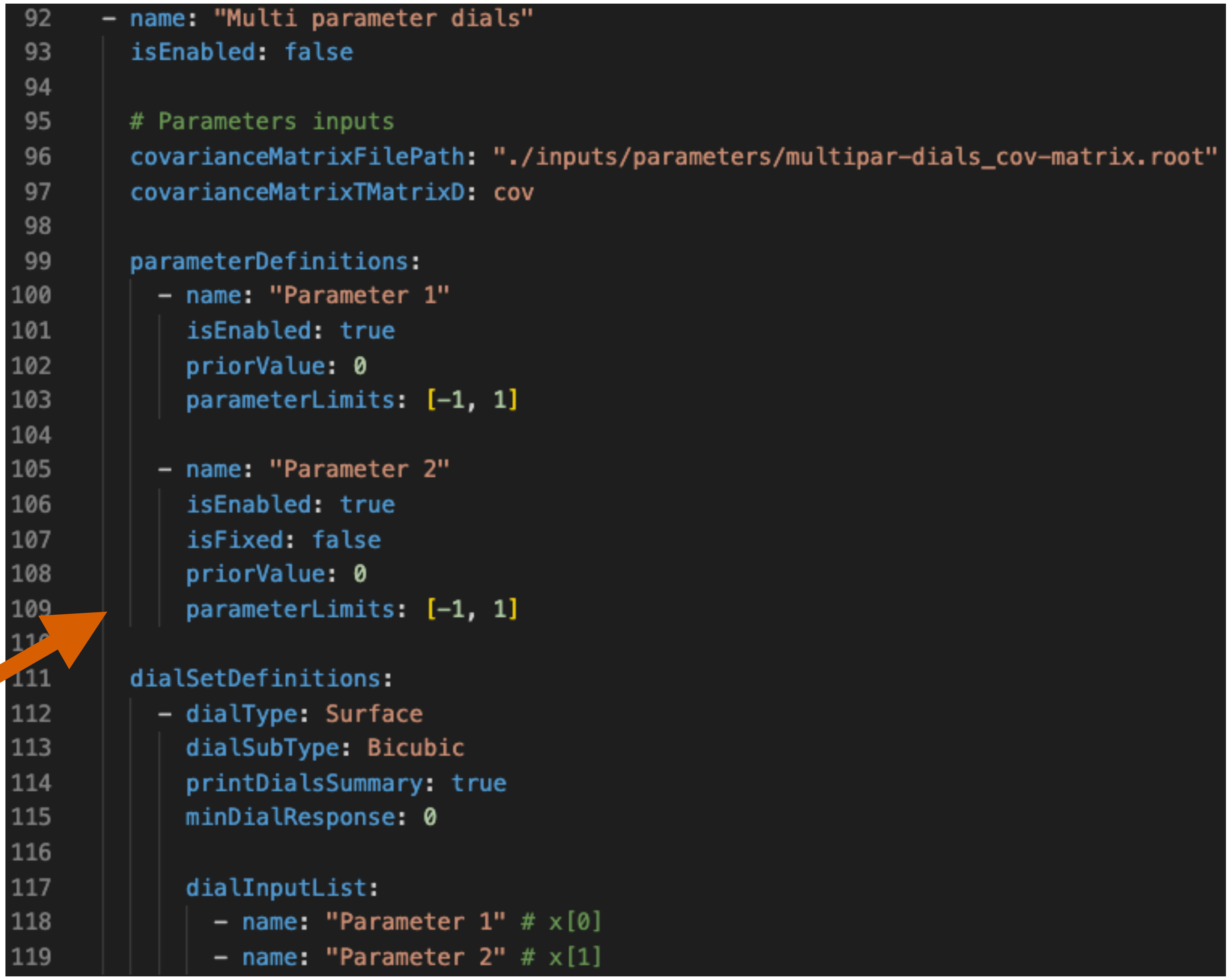
Note: illustrative only examples

Multi-parameter dials

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:             "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:      "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:     "./inputs/output/configPlotGenerator.yaml"
    scanConfig:              "./inputs/output/configScan.yaml"
```

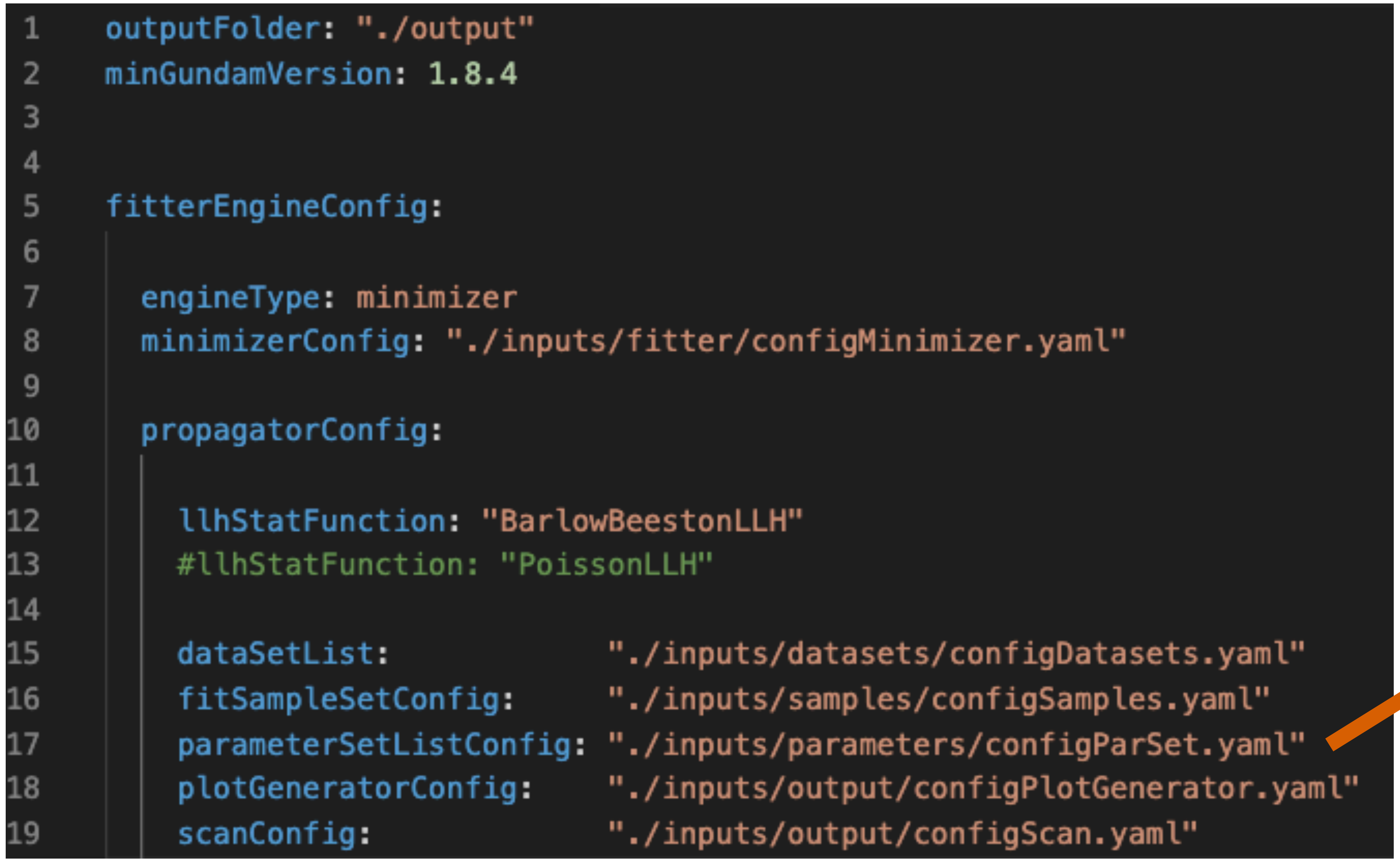


```yaml
- name: "Multi parameter dials"
  isEnabled: false

  # Parameters inputs
  covarianceMatrixFilePath: "./inputs/parameters/multipar-dials_cov-matrix.root"
  covarianceMatrixTMatrixD: cov

  parameterDefinitions:
    - name: "Parameter 1"
      isEnabled: true
      priorValue: 0
      parameterLimits: [-1, 1]

    - name: "Parameter 2"
      isEnabled: true
      isFixed: false
      priorValue: 0
      parameterLimits: [-1, 1]

  dialSetDefinitions:
    - dialType: Surface
      dialSubType: Bicubic
      printDialsSummary: true
      minDialResponse: 0

      dialInputList:
        - name: "Parameter 1" # x[0]
        - name: "Parameter 2" # x[1]
```

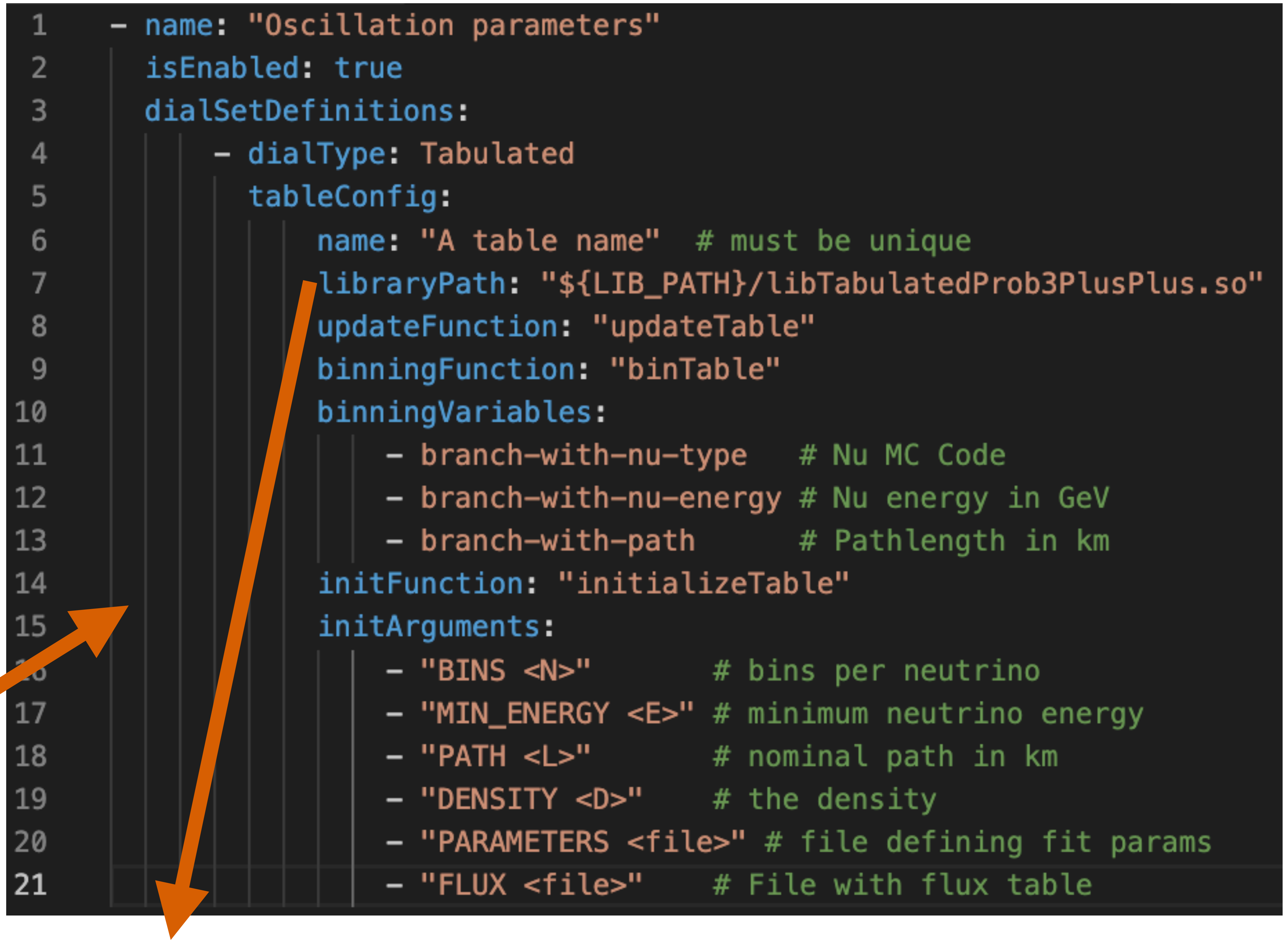
Note: illustrative only examples

Tabulated dials

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:           "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:    "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:   "./inputs/output/configPlotGenerator.yaml"
    scanConfig:            "./inputs/output/configScan.yaml"
```

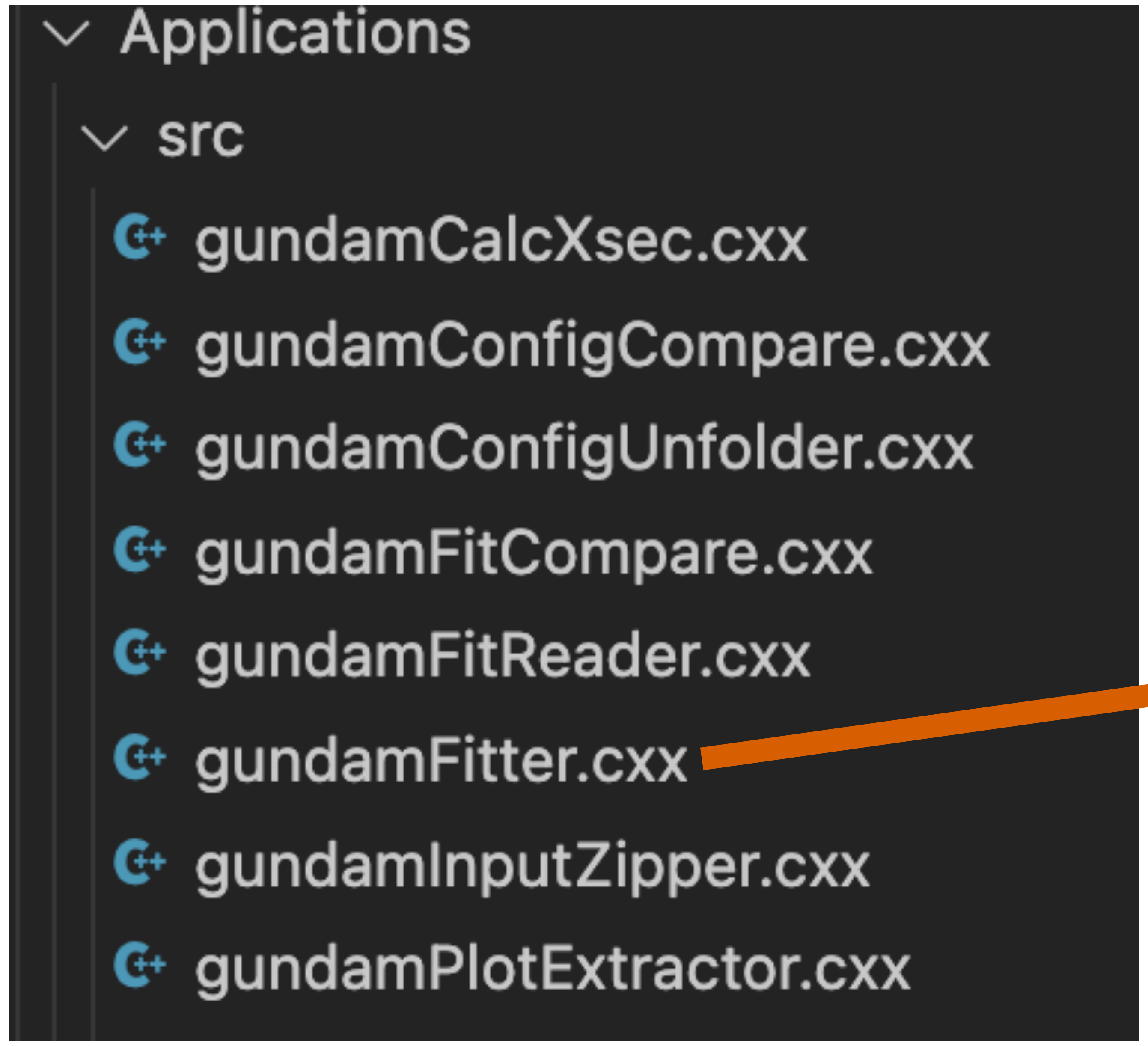


```yaml
- name: "Oscillation parameters"
  isEnabled: true
  dialSetDefinitions:
    - dialType: Tabulated
      tableConfig:
        name: "A table name"  # must be unique
        libraryPath: "${LIB_PATH}/libTabulatedProb3PlusPlus.so"
        updateFunction: "updateTable"
        binningFunction: "binTable"
        binningVariables:
          - branch-with-nu-type   # Nu MC Code
          - branch-with-nu-energy # Nu energy in GeV
          - branch-with-path      # Pathlength in km
        initFunction: "initializeTable"
        initArguments:
          - "BINS <N>"       # bins per neutrino
          - "MIN_ENERGY <E>" # minimum neutrino energy
          - "PATH <L>"       # nominal path in km
          - "DENSITY <D>"    # the density
          - "PARAMETERS <file>" # file defining fit params
          - "FLUX <file>"    # File with flux table
```

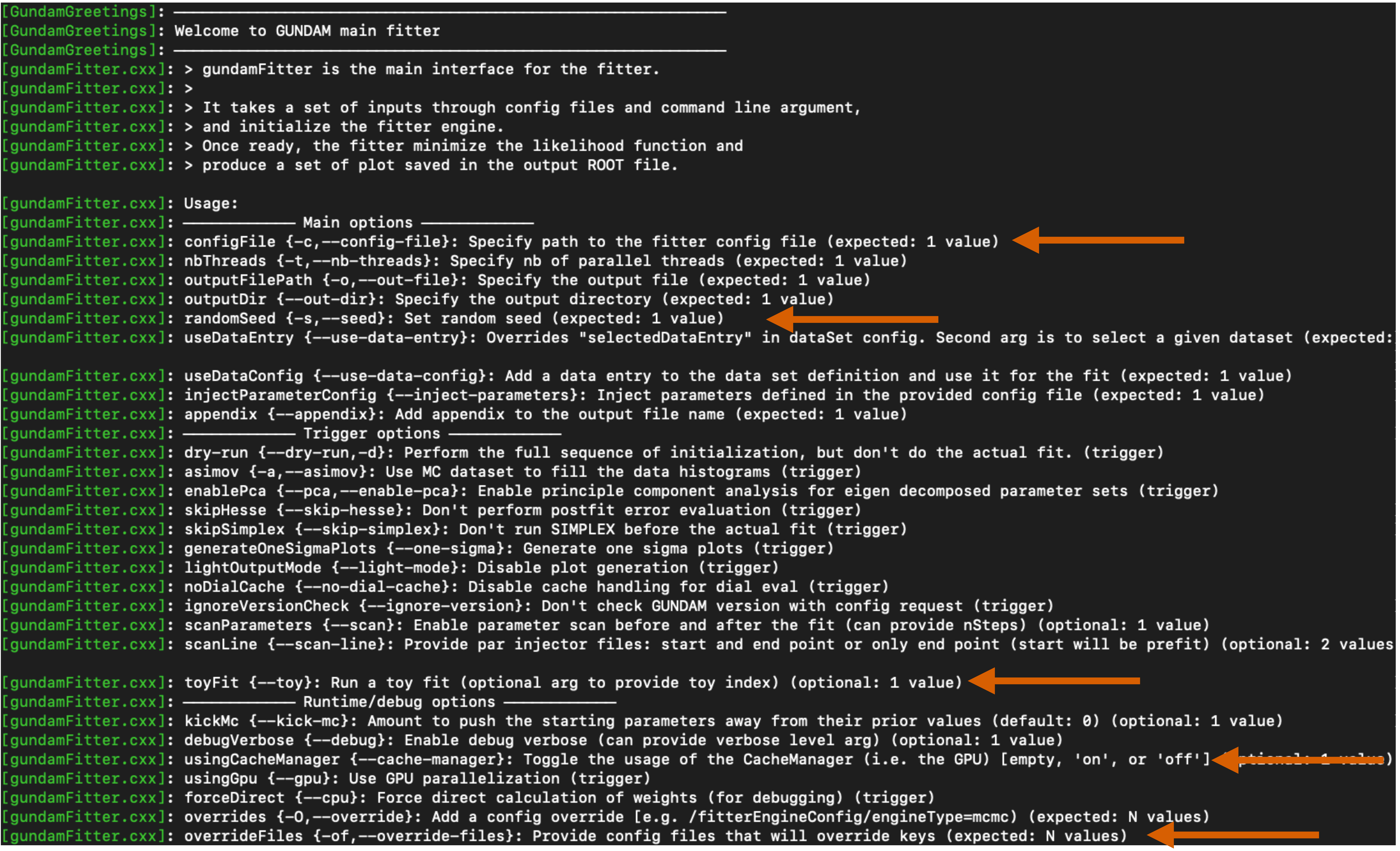
Note: External pre-compiled C++ libraries which belong to the inputs, not the code of GUNDAM itself

Note: illustrative only examples

Applications



```
Applications
  src
    gundamCalcXsec.cxx
    gundamConfigCompare.cxx
    gundamConfigUnfolder.cxx
    gundamFitCompare.cxx
    gundamFitReader.cxx
    gundamFitter.cxx
    gundamInputZipper.cxx
    gundamPlotExtractor.cxx
```

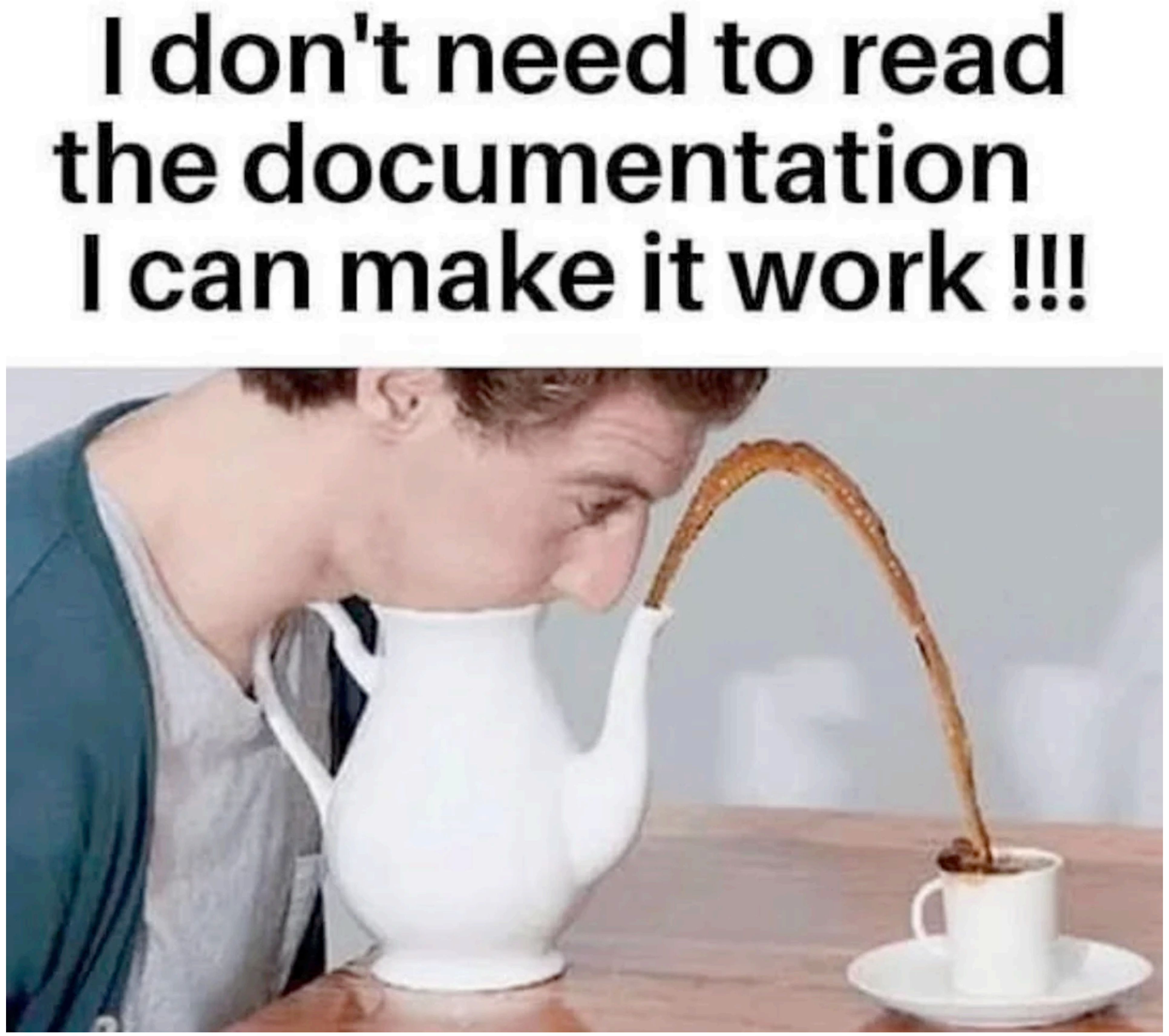


```
[GundamGreetings]: ────────────────────────────────────────────────────────────
[GundamGreetings]: Welcome to GUNDAM main fitter
[GundamGreetings]: ────────────────────────────────────────────────────────────
[gundamFitter.cxx]: > gundamFitter is the main interface for the fitter.
[gundamFitter.cxx]: >
[gundamFitter.cxx]: > It takes a set of inputs through config files and command line argument,
[gundamFitter.cxx]: > and initialize the fitter engine.
[gundamFitter.cxx]: > Once ready, the fitter minimize the likelihood function and
[gundamFitter.cxx]: > produce a set of plot saved in the output ROOT file.

[gundamFitter.cxx]: Usage:
[gundamFitter.cxx]: ──────────── Main options ────────────
[gundamFitter.cxx]: configFile {-c,--config-file}: Specify path to the fitter config file (expected: 1 value)
[gundamFitter.cxx]: nbThreads {-t,--nb-threads}: Specify nb of parallel threads (expected: 1 value)
[gundamFitter.cxx]: outputFilePath {-o,--out-file}: Specify the output file (expected: 1 value)
[gundamFitter.cxx]: outputDir {--out-dir}: Specify the output directory (expected: 1 value)
[gundamFitter.cxx]: randomSeed {-s,--seed}: Set random seed (expected: 1 value)
[gundamFitter.cxx]: useDataEntry {--use-data-entry}: Overrides "selectedDataEntry" in dataSet config. Second arg is to select a given dataset (expected:

[gundamFitter.cxx]: useDataConfig {--use-data-config}: Add a data entry to the data set definition and use it for the fit (expected: 1 value)
[gundamFitter.cxx]: injectParameterConfig {--inject-parameters}: Inject parameters defined in the provided config file (expected: 1 value)
[gundamFitter.cxx]: appendix {--appendix}: Add appendix to the output file name (expected: 1 value)
[gundamFitter.cxx]: ──────────── Trigger options ────────────
[gundamFitter.cxx]: dry-run {--dry-run,-d}: Perform the full sequence of initialization, but don't do the actual fit. (trigger)
[gundamFitter.cxx]: asimov {-a,--asimov}: Use MC dataset to fill the data histograms (trigger)
[gundamFitter.cxx]: enablePca {--pca,--enable-pca}: Enable principle component analysis for eigen decomposed parameter sets (trigger)
[gundamFitter.cxx]: skipHesse {--skip-hesse}: Don't perform postfit error evaluation (trigger)
[gundamFitter.cxx]: skipSimplex {--skip-simplex}: Don't run SIMPLEX before the actual fit (trigger)
[gundamFitter.cxx]: generateOneSigmaPlots {--one-sigma}: Generate one sigma plots (trigger)
[gundamFitter.cxx]: lightOutputMode {--light-mode}: Disable plot generation (trigger)
[gundamFitter.cxx]: noDialCache {--no-dial-cache}: Disable cache handling for dial eval (trigger)
[gundamFitter.cxx]: ignoreVersionCheck {--ignore-version}: Don't check GUNDAM version with config request (trigger)
[gundamFitter.cxx]: scanParameters {--scan}: Enable parameter scan before and after the fit (can provide nSteps) (optional: 1 value)
[gundamFitter.cxx]: scanLine {--scan-line}: Provide par injector files: start and end point or only end point (start will be prefit) (optional: 2 values

[gundamFitter.cxx]: toyFit {--toy}: Run a toy fit (optional arg to provide toy index) (optional: 1 value)
[gundamFitter.cxx]: ──────────── Runtime/debug options ────────────
[gundamFitter.cxx]: kickMc {--kick-mc}: Amount to push the starting parameters away from their prior values (default: 0) (optional: 1 value)
[gundamFitter.cxx]: debugVerbose {--debug}: Enable debug verbose (can provide verbose level arg) (optional: 1 value)
[gundamFitter.cxx]: usingCacheManager {--cache-manager}: Toggle the usage of the CacheManager (i.e. the GPU) [empty, 'on', or 'off'] (optional: 1 value)
[gundamFitter.cxx]: usingGpu {--gpu}: Use GPU parallelization (trigger)
[gundamFitter.cxx]: forceDirect {--cpu}: Force direct calculation of weights (for debugging) (trigger)
[gundamFitter.cxx]: overrides {-O,--override}: Add a config override [e.g. /fitterEngineConfig/engineType=mcmc) (expected: N values)
[gundamFitter.cxx]: overrideFiles {-of,--override-files}: Provide config files that will override keys (expected: N values)
```

```
Example: gundamFitter -c mainConfig.yaml -t 16 --cache-manager -s 123 --toy 1 -of configOverride.yaml
```

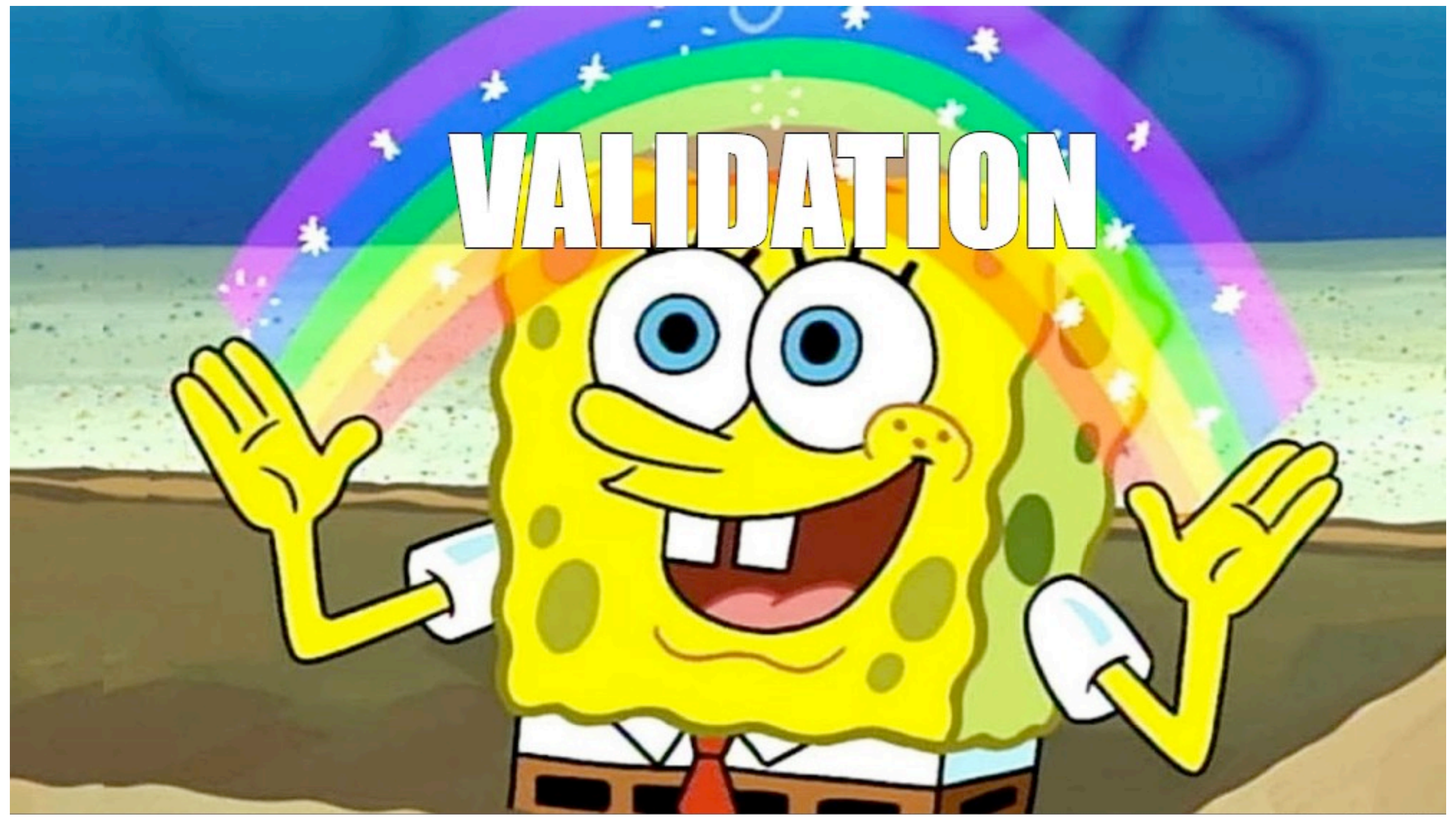
Documentation

- Short outline:
  - A general introduction
  - Installation guide
  - How to set up the configuration files with examples
  - Meaning of the entries of a configuration file
  - How to use the different applications
- Work in progress (almost by definition)



Testing and Validation

- Tests workflow implemented for a continuous validation
- We have two type of tests running: unit tests with GoogleTest and executable testing with specialized script (running mock analysis and checking expectations)
- Executable testing with specialized script can be divided in:
  - Fast: always run and used during continuous integration (eg. few-parameter fit)
  - Regular: quick tests that are not used for CI, but should be run locally before any PR (eg. response functions interpolation)
  - Extended: slower tests (eg. MCMC)
  - Slow: long validation tests (eg. more complex fit)
- Always looking for missing tests!!!



GUNDAM Oscillation Analysis Tools

- Oscillation probability calculation
  - Already using Prob3++ but other libraries can be used
- OA functionality to be implemented:
  - Frequentist analysis, ie profiling:
    - Constant $\Delta\chi^2$ (fast fit), $\Delta\chi^2$ with FC, etc.
  - Bayesian analysis
    - Marginalization with MCMC
- Would like to test what has developed so far and continue
  - Wei Shi (SBU) pointed us toward inputs at this link used for DUNE PRISM
  - Guidance on the latest inputs DUNE-LBL WG is using would help us setting up our config files and add features to the code base if needed

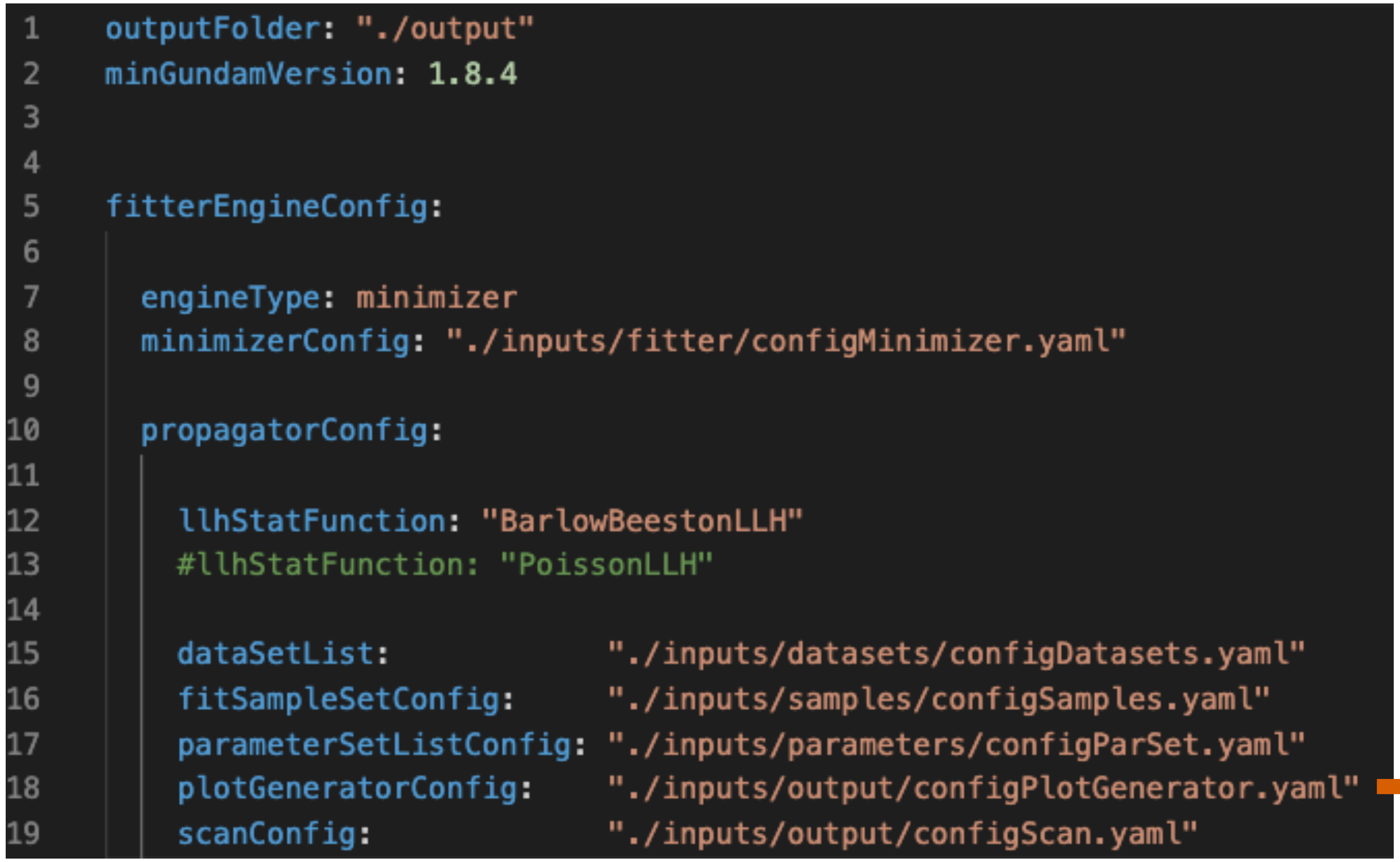
Conclusions

- GUNDAM is a versatile, fully configurable and high-performance tool that can analyze high-dimensional models and binning
- Used for T2K OA-ND and cross-section analyses, but also ICARUS Collaborators are using it for a cross-section analysis
- Can contribute to DUNE LBL in many ways:
  - LBL and LBL+atm sensitivities
    - DUNE PRISM and “classical” ND sensitivities
  - Cross-section analyses (future)
  - Many other analyses
- GUNDAM Oscillation Analysis Tools work ongoing and we exploring the interests of other groups and individuals

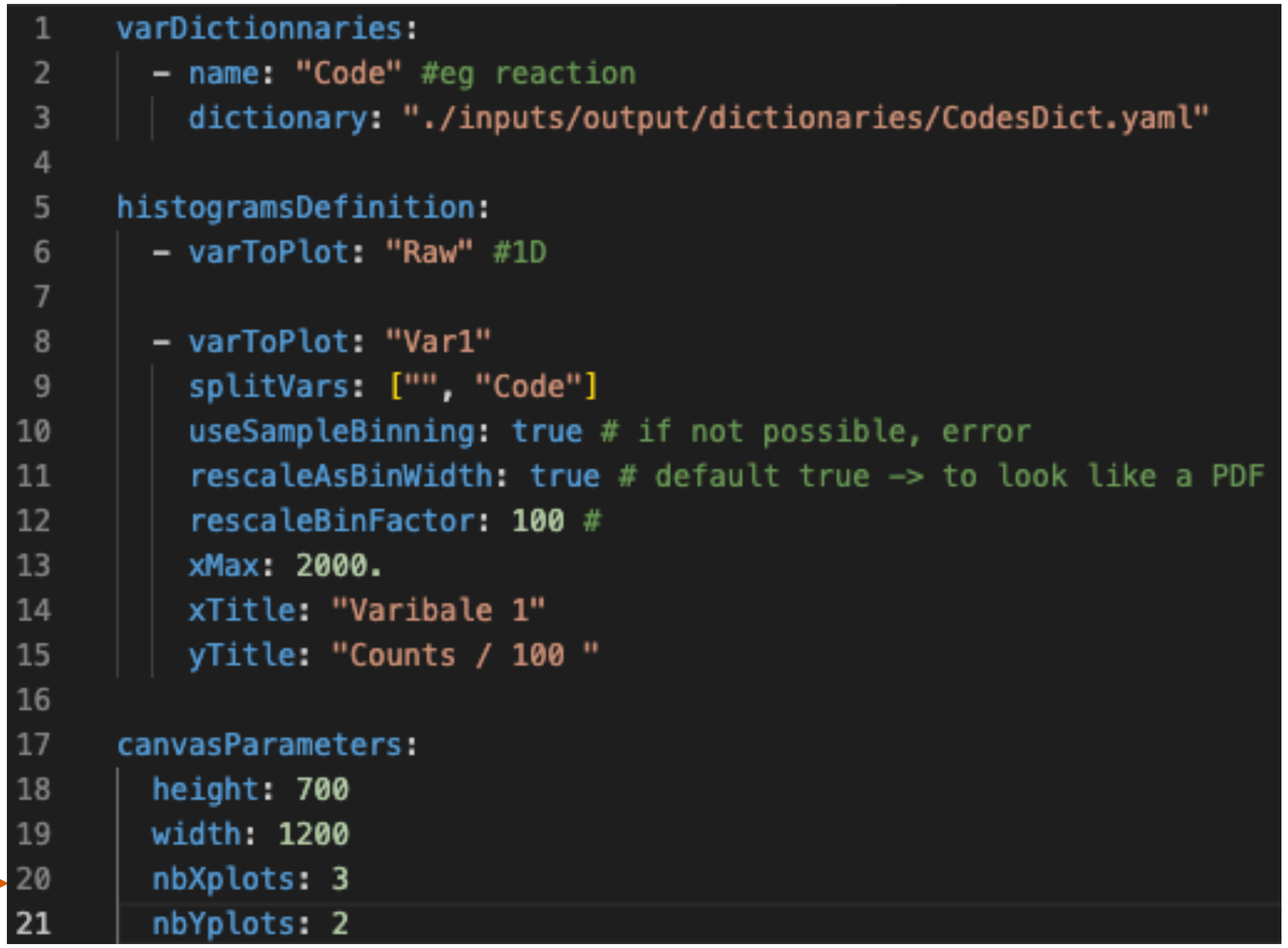
Backup

Plot generator

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:              "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:       "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:      "./inputs/output/configPlotGenerator.yaml"
    scanConfig:               "./inputs/output/configScan.yaml"
```

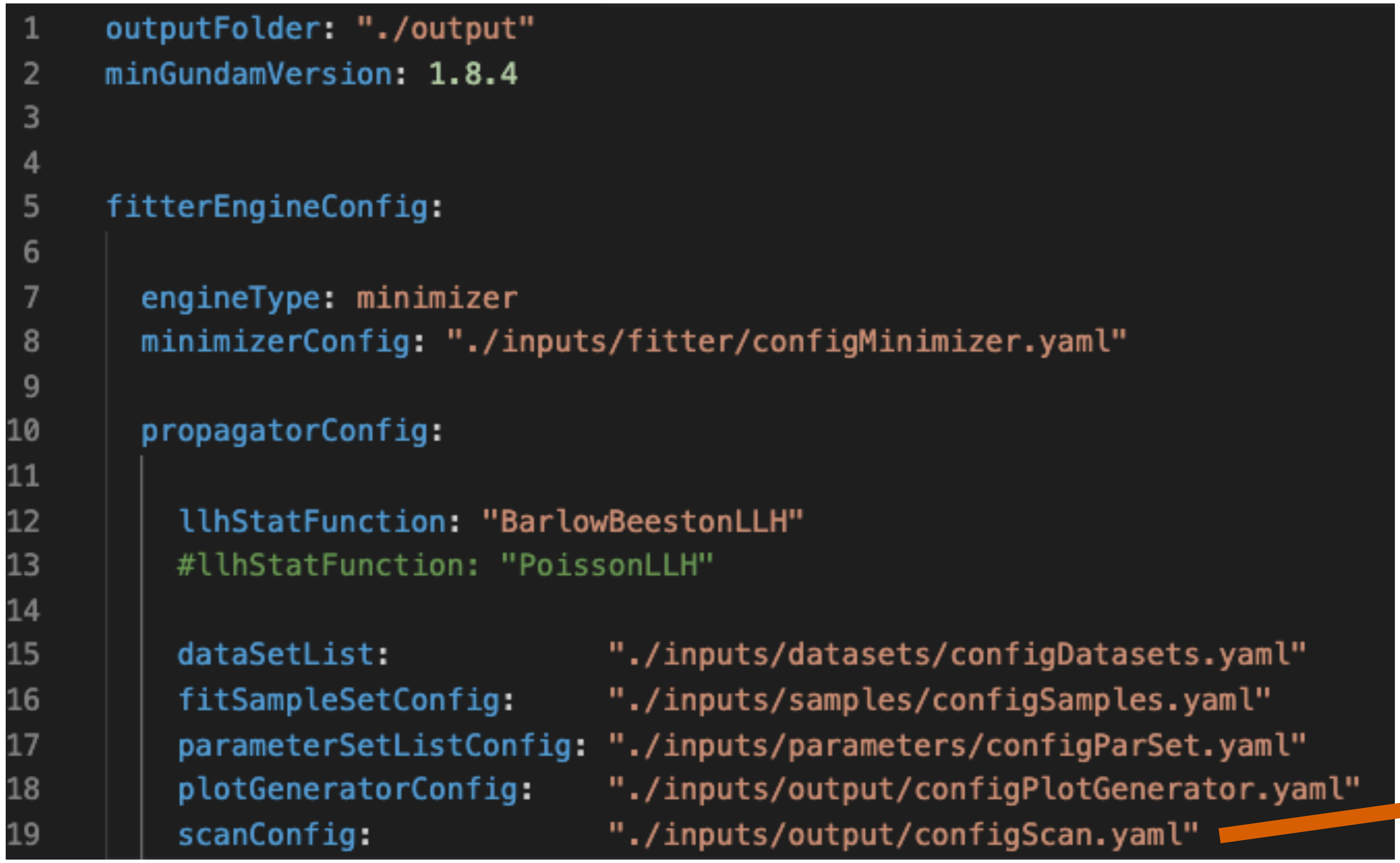


```yaml
varDictionnaries:
  - name: "Code" #eg reaction
    dictionary: "./inputs/output/dictionaries/CodesDict.yaml"

histogramsDefinition:
  - varToPlot: "Raw" #1D

  - varToPlot: "Var1"
    splitVars: ["", "Code"]
    useSampleBinning: true # if not possible, error
    rescaleAsBinWidth: true # default true -> to look like a PDF
    rescaleBinFactor: 100 #
    xMax: 2000.
    xTitle: "Varibale 1"
    yTitle: "Counts / 100 "

canvasParameters:
  height: 700
  width: 1200
  nbXplots: 3
  nbYplots: 2
```

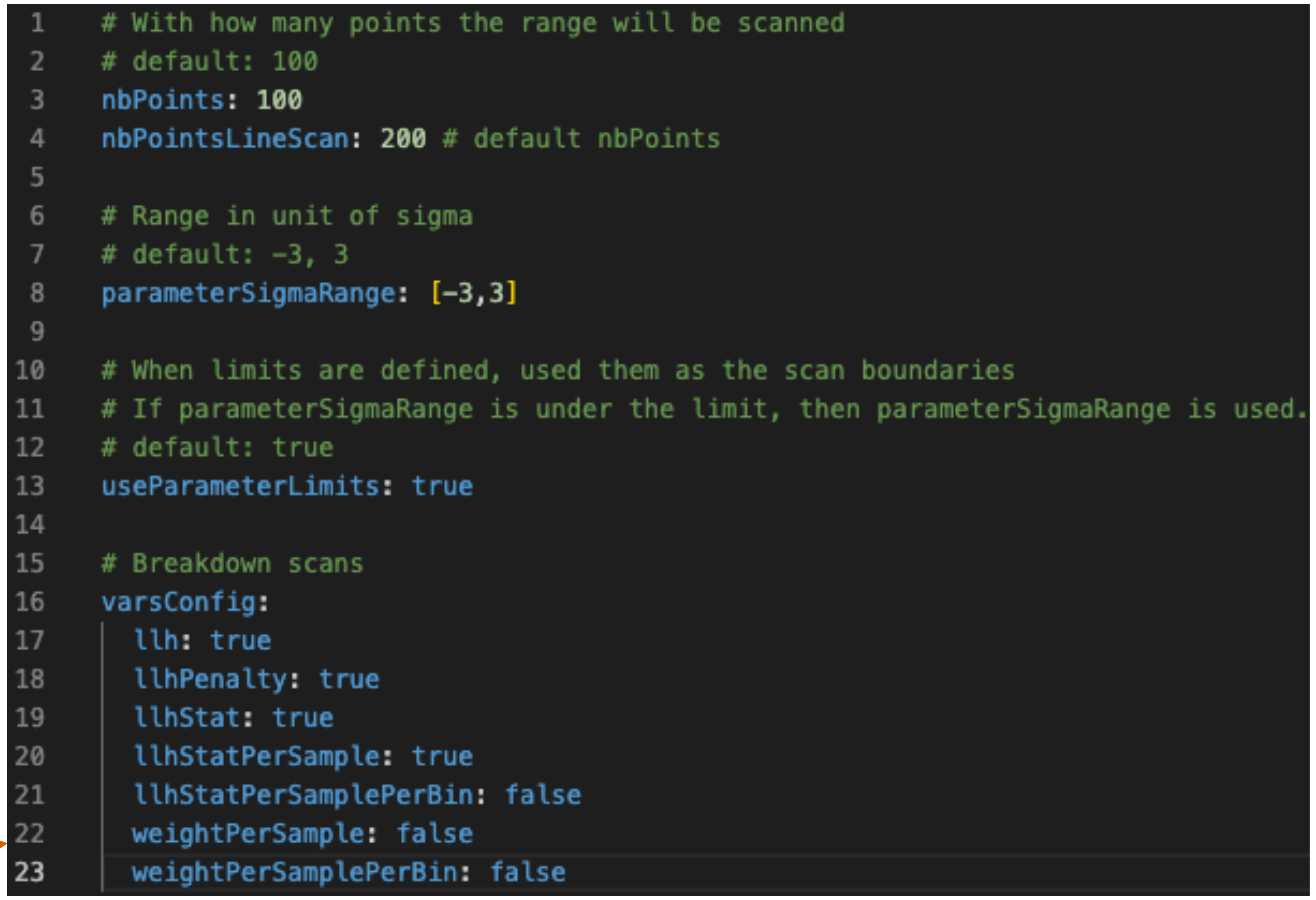
Note: illustrative only examples

Likelihood scanning

mainConfig.yaml



```yaml
outputFolder: "./output"
minGundamVersion: 1.8.4


fitterEngineConfig:

  engineType: minimizer
  minimizerConfig: "./inputs/fitter/configMinimizer.yaml"

  propagatorConfig:

    llhStatFunction: "BarlowBeestonLLH"
    #llhStatFunction: "PoissonLLH"

    dataSetList:            "./inputs/datasets/configDatasets.yaml"
    fitSampleSetConfig:     "./inputs/samples/configSamples.yaml"
    parameterSetListConfig: "./inputs/parameters/configParSet.yaml"
    plotGeneratorConfig:    "./inputs/output/configPlotGenerator.yaml"
    scanConfig:             "./inputs/output/configScan.yaml"
```



```yaml
# With how many points the range will be scanned
# default: 100
nbPoints: 100
nbPointsLineScan: 200 # default nbPoints

# Range in unit of sigma
# default: -3, 3
parameterSigmaRange: [-3,3]

# When limits are defined, used them as the scan boundaries
# If parameterSigmaRange is under the limit, then parameterSigmaRange is used.
# default: true
useParameterLimits: true

# Breakdown scans
varsConfig:
  llh: true
  llhPenalty: true
  llhStat: true
  llhStatPerSample: true
  llhStatPerSamplePerBin: false
  weightPerSample: false
  weightPerSamplePerBin: false
```

Note: illustrative only examples

Development policy

- For internal development an issue should be opened and it should happen in a dedicated branch with a descriptive name of the feature. We recommend the following tags:
  - fix/myFix: to address specific issues with the code
  - feature/myFeature: to add specific feature
  - doc/myDoc: for documentation additions
  - experimental/myBranch: for idea developments
- Change description (commit messages and PR) must be explicit and limited to a single issue
- Create a PR and merge after review and CI is successfully completed
- External developers are encouraged following the same procedure as above (might fork the code)