Post-processing

<u>Uses as main inputs:</u>

- ntuple ‚markovChain' from the fit (whatever the initial constraints were)
- a Fittino input file with the desired experimental constraints:

The input file is turn into a simple text file with just the names of the observables and their values

<u>[1] Cleaning step:</u>

- Removal of points with negative LHC, AF, HS chi2
- Removal of multiple points:

for each point „i", loop from „i+1" to „i+100", if an identical point is found (same M0 and M12 and TanBeta and A0) the point „i" is removed, so only the last multiple point remains
*The cleaned input ntuple is called *_cleaned.root and is used as ntuple input for the step [2]*

<u>[2] Calculation of the chi2 for each point</u>

Chi2 calculated using

- external codes (HiggsSignal, LHC, AF)
- experimental constraints & uncertainties given in the input file:

look at the function (postprocessing.h>assignLEO()): for B->tau nu a theoretical uncertainty is added, waiting that the „scanning uncertainty" in the input file be replaced.

<u>[3] Removal of buggy points:</u>

Use of splines (not yet implemented)

<u>[4] Best fit point</u>

- identify the point with lowest chi2
- save its information in a text file (coordinates, chi2, observables)

Post-processing

The main source is postProcessing.cc, where the overview is clear but not much detailled, all the details are contained in the headers

Uses as main inputs:

- ntuple ‚markovChain' from the fit (whatever the initial constraints were)
- a Fittino input file with the desired experimental constraints:

The input file is turn into a simple text file with just the names of the observables and their values

preparFittinoInput.sh ➔ script creates fittinoNewObservables.txt used as input for step [1]

[1] Cleaning step:

- Removal of points with negative LHC, AF, HS chi2
- Removal of multiple points:

for each point „i", loop from „i+1" to „i+100", if an identical point is found (same M0 and M12 and TanBeta and A0) the point „i" is removed, so only the last multiple point remains
*The cleaned input ntuple is called *_cleaned.root and is used as ntuple input for the step [2]*

Header Cleaning.h (function cleaningInputFile) called in postProcessing.cc

[2] Calculation of the chi2 for each point

Chi2 calculated using

- external codes (HiggsSignal->check with Tim the uncertainties and analysis, LHC, AF)
- experimental constraints & uncertainties given in the input file:

look at the function (postprocessing.h>assignLEO()): for B->tau nu a theoretical uncertainty is added, waiting that the „scanning uncertainty" in the input file be replaced.

Reading of exp. Constraints: Header postProcessing.h (function assignLEObs) called in postProcessing.cc
Processing function is processData(X) in postProcessing.h, where X=0 for toys and 1 for data: Header postProcessing.h (function calculateChi2) called in postProcessing.cc

[3] Removal of buggy points:

Use of splines (not yet implemented)

[4] Best fit point

- identify the point with lowest chi2
- save its information in a text file (coordinates, chi2, observables)

Header postProcessing.h (function writeBestFitPoint()) called in postProcessing.h > processData()

Toys

The main source is postProcessing.cc, where the overview is clear but not much detailled, all the details are contained in the headers.
It uses almost the same code as the post-processing

Uses as main inputs:
- ntuple ‚markovChain' from the fit (whatever the initial constraints were)
- a text file with the coordinates of the best fit point (step 4 of the post-processing)
- a Fittino input file with the desired experimental constraints, but fro which only the uncertainties will be taken.

[1] Read the input text files
- the best fit point
- the experimental uncertainties in fittinoNewObservables.txt (obtained from the Fittino input file)

[2] Smearing of all observables
- Low energy observables are smeared according to a Gaussian
- LHC: smearing of the number of observed events
- Astrofit: smearing of the exclusion contour by XENON100
- HiggsSignal: smearing of the masses and ratio accounting for the correlations

[3] Calculate the chi2 (idem as step[2] of post-processing)

[4] Removal of buggy points (idem as step[3] of post-processing)

[5] Best fit point
- identify the point with lowest chi2
- save its information in an ntuple

Toys

The main source is postProcessing.cc, where the overview is clear but not much detailled, all the details are contained in the headers.
It uses almost the same code as the post-processing

Uses as main inputs:
- ntuple ‚markovChain' from the fit (whatever the initial constraints were)
- a text file with the coordinates of the best fit point (step 4 of the post-processing)
- a Fittino input file with the desired experimental constraints, but fro which only the uncertainties will be taken.

preparFittinoInput.sh ➔ script creates fittinoNewObservables.txt used as input for step [1]

[1] Read the input text files
- the best fit point
- the experimental uncertainties in fittinoNewObservables.txt (obtained from the Fittino input file)

Header postProcessing.h (function assignLEObs) called in postProcessing.cc
Header postProcessing.h (function readBestFitPoint()) called in postProcessing.h > processData(X)

[2] Smearing of all observables
- Low energy observables are smeared according to a Gaussian
- LHC: smearing of the number of observed events
- Astrofit: smearing of the exclusion contour by XENON100
- HiggsSignal: smearing of the masses and ratio accounting for the correlations

The smeared values are saved in a separate ntuple for later study

Functions setLHCchi2Tools, setAstrofit, smearLEObs, smearHiggs, called in postProcessing.h > calculateChi2

[3] Calculate the chi2 (idem as step[2] of post-processing)

Reading of exp. Constraints: Header postProcessing.h (function assignLEObs) called in postProcessing.cc
Processing function is processData(X) in postProcessing.h, where X=0 for toys and 1 for data: Header postProcessing.h (function calculateChi2) called in postProcessing.cc

[4] Removal of buggy points (idem as step[3] of post-processing)

[5] Best fit point
- identify the point with lowest chi2
- save its information in an ntuple

Function saveToyResult in postProcessing.h > processData

How to run the post-processing / toys

Script to run the post-processing/toys on the NAF batch: /fittino/postProcessing/job_batch

```
#!/usr/bin/env zsh
#$ -j y
##$ -l os=sld5
#$ -cwd
#$ -P atlas
#$ -l h_cpu=06:00:00
#$ -w e
#$ -m ae -M prudent@physik.tu-dresden.de
```

Various info for the batch (CPU, email, account,…)

```
#----------------------------------
# Name of the fit to post-process or to study with toys (Attention, no .root at the end)
fit=fittino.out.point1.CMSSM.allObs.Summer2012_merged

# Fittino input file with the desired constraints to be applied
input_file=/afs/naf.desy.de/user/p/prudent/fittino/inputs/summer2012/fittino.in.point1.CMSSM.allObs.summer2012

# Where does the input fit ntuple lies
input_directory=/scratch/hh/current/atlas/users/bechtle/fittino/fittino.out.summer2012_01_probablyWrongGlobalHiggsC
hi2/

# Output of post-processing and toys
output_destination=/scratch/hh/current/atlas/users/prudent/fittino/postProcessing/postProcessing_2012/

# Where the processing programs lie
working_directory=/afs/naf.desy.de/user/p/prudent/fittino/postProcessing

# Where to retrieve the LHC chi2 tools from
tools_directory=/afs/naf.desy.de/user/p/prudent/fittino/tools/GetToyLHCChi2

# Number of toys to run
export NUMBERTOYS=0

# Do you want a 500MB file with all information (1=yes,0=no)?
export VERBOSE=0

# Post-processing (1) or toy study (0)
export DATATOYS=1
#----------------------------------
```

NB: of course it does not make sense to set NUMBERTOYS > 0 if you run on data

How to run the post-processing / toys

```
export ROOTSYS=/afs/naf.desy.de/products/root/amd64_rhel50/5.26.00
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/afs/naf.desy.de/user/p/prudent/fittino/tools/GetToyLHCChi2

date
echo " --- Script started"

cd $TMP
ini ROOT526

echo " --- Copying input files and macros"

# General program
cp $working_directory/postProcessing.cc .
cp $working_directory/postProcessing.h .
cp $working_directory/compil .

# Cleaning
cp $working_directory/cleaning.h .

# Higgs
cp $working_directory/Higgs.h .

# LHC
cp $working_directory/LHC.h .
cp $tools_directory/ToyLHCChi2Provider.h .
histograms=$tools_directory/histograms.root
signalGrids=$tools_directory/signalGrids.root

# Astrofit
cp $working_directory/astrofit.h .
cp $working_directory/dd_xenon100_2012.dat .

# To extract the observables from the Fittino file
cp $working_directory/preparFittinoInput.sh .

echo " --- Extracting observables from the Fittino input file"
chmod u+x preparFittinoInput.sh
cp $input_file .
./preparFittinoInput.sh $input_file
```

Setting of librairies
Initialization of ROOT

Copying every necessary programs to the temporary directory

Extracting the observables from the Fittino input file

How to run the post-processing / toys

```
echo " --- Compiling"
chmod u+x compil
./compil
```

Compiling

```
echo " --- Running the macro"
./postProcessing $fit $input_directory $output_destination $histograms $signalGrids

echo ""
echo " --- Copying output files"
cp $fit.root  $output_destination
```

Running

```
echo " --- Script finished"
date
```

Copying the output root file

Content of the output ntuples

<u>For toys:</u>

- **toyNtuple**: values at the best fit point for each toy
- **smearedObsNtuple**: values of the smeared observables

If only 1 toy is processed, all his points are saved in a ‚**markovChain**' ntuple, as for data, in order to compare the nominal post-processing with a pseudo-data

<u>For post-processing</u>

- **markovChain**: all info contained in the fit ntuple, plus the values of the chi2 for each individual observables

Check: markovChain->Print(„*chi2*")

Calculation of the p-value

ROOT macro /fittino/postProcessing/pValue.C

Input: the chi2 distribution for all the best fit points of toys

Calculate the fraction of toys whose chi2 is at least larger than the observed chi2 with the post-processing

This macro can in principle also calculate the contours from the integration of the toys distribution. To be further discussed…