a quick guide for getting started

Tutorial session @ Tools 2020

https://smodels.github.io/

Gaël Alguero, Jan Heisig, Charanjit K. Khosa, Sabine Kraml, Suchita Kulkarni, Andre Lessa, Philipp Neuhuber, Humberto Reyes-Gonzalez, Wolfgang Waltenberger, Alicia Wongel

Everything you need is explained on **smodels.github.io**

GitHub | pypi package 1.2.4.post1 | launch binder | docs master

and in the talk “New developments in SModelS”
by G. Alguero in this tutorial session.

- A detailed documentation is available in the **online manual**
- For instructions on how to install SModelS, check the **installation** section in the manual.
- You may also want to check the release notes and known issues
- Here are the list of analyses in the latest database version, the respective validation plots and an SMS dictionary explaining the Tx names used by SModelS.
- A discussion of re-interpretation methods and tools, and recommendations about the presentation of results can be found in this report by the LHC Reinterpretation Forum.

Mailing lists:

- For questions and comments, send an e-mail to: smodels-users@lists.oeaw.ac.at.
- To receive updates and announcements, subscribe to smodels-info.

… and lots more useful infos ….

Requirements

SModelS is a Python package that requires **Python 2.6 or later; default is Python 3**.
It depends on the following ***external* Python libraries**:

- unum>=4.0.0
- numpy>=1.13.0
- argparse
- requests>=2.0.0
- docutils>=0.3
- scipy>=1.0.0
- pyslha>=3.1.0
- pyhf>=0.4.3 (>=0.5.2 recommended!)
- jsonpatch>=1.25
- jsonschema>=3.2.0

(+ recommended for pyhf: pytorch)

The cross section computer provided by smodelsTools.py requires:

- Pythia 8.2 (requires a C++ compiler) or Pythia 6.4.27 (requires fortran)
- NLL-fast 1.2, 2.1, and 3.1 (requires a fortran compiler)

These tools need not be installed separately, as the SModelS build system takes care of that.

The database browser provided by smodelsTools.py requires IPython, while the interactive plotter requires plotly and pandas.

https://smodels.readthedocs.io/en/latest/Installation.html

Standard installation

Download the code from https://github.com/SModelS/smodels/releases and extract it in a source directory, e.g.:

```
> tar -zxvf smodels-1.2.4.tar.gz
> cd smodels-1.2.4
```

then run

```
> make smodels   (or: make FC=<path_to_fortran> smodels)
```

in the top-level directory. This will install the required dependencies (using pip install) and compile Pythia and NLL-fast.

If the (MSSM) cross section computer is not needed, run instead

```
> make smodels_noexternaltools
```

In case the Python libraries cannot be successfully installed, the user can install them separately using his/her preferred method. Pythia and NLL-fast can also be compiled separately running `make externaltools`.

Alternatively:

- using python setuptools:

```
setup.py install [--user]
```

- using pip:

```
pip3 install [--user] smodels
```

https://smodels.readthedocs.io/en/latest/Installation.html

Using SModelS: runSModelS.py

SModelS can take **SLHA files** (with masses, decay tables and *cross sections* !) **or LHE files** as input.
It ships with a command-line tool `runSModelS.py`, which reports on the SMS decomposition and theory predictions in several output formats.

The usage of runSModelS is:

```
> ./runSModelS.py [-h] -f FILENAME [-p PARAMETERFILE] [-o OUTPUTDIR]
```

- -h : show help message and exit
- -f : SLHA or LHE input file(s); can be a directory
- -p : name of parameters file (default is ./smodels/etc/parameters_default.ini)
- -o : output directory
- other optional arguments: verbosity level, colored output, timeout, etc.

https://smodels.readthedocs.io/en/latest/RunningSModelS.html

MSSM example

```
> ./runSModelS.py -f inputFiles/slha/gluino_squarks.slha -p parameters.ini -o results/.
```

Output file: `gluino_squarks.slha.smodels`

Input status: 1
Decomposition output status: 1 #decomposition was successful
Input File: inputFiles/slha/gluino_squarks.slha
[...]
Database version: 1.2.4

$$r = \frac{\sigma(\text{signal})}{\sigma(95\%\text{CL})} \quad , \quad \text{r} \geq 1 \text{ considered excluded}$$

```
==========================================================================
#Analysis  Sqrts  Cond_Violation  Theory_Value(fb)  Exp_limit(fb)  r  r_expected

ATLAS-SUSY-2016-07  1.30E+01    0.0  9.791E+00  1.270E+00  7.710E+00  9.151E+00
Signal Region:  5j_Meff_1600
Txnames:  T1, T2, T5GQ, T5WW, T5ZZ, T6WW, TGQ
Chi2, Likelihood =  2.577E+01  2.469E-09
--------------------------------------------------------------------------
CMS-SUS-19-006  1.30E+01    0.0  1.169E+01  5.138E+00  2.274E+00  1.898E+00
Signal Region:  (UL)
Txnames:  T2
--------------------------------------------------------------------------
```

Efficiency map (EM) result, sum over several topologies, Chi2 and likelihood evaluated

Cross section upper limit (UL) result, only one topology

MSSM example (cont.)

```
> ./runSModelS.py -f inputFiles/slha/gluino_squarks.slha -p parameters.ini -o results/.
```

Output file: `gluino_squarks.slha.smodels`

```
[....]
================================================================
The highest r value is = 7.709501308305477

Total cross section considered (fb):  3.635E+03
Total missing topology cross section (fb):  2.838E+03
Total cross section where we are outside the mass grid (fb):  1.948E+02
Total cross section in long cascade decays (fb):  1.317E+03
Total cross section in decays with asymmetric branches (fb):  1.504E+03

Full information on unconstrained cross sections
================================================================
Missing topologies with the highest cross sections (up to 10):
Sqrts (TeV)   Weight (fb)        Element description
 13.0  1.482E+02    #        [[[jet],[W]],[[jet,jet],[W]]](MET,MET)
 13.0  1.445E+02    # [[[jet,jet],[W]],[[jet],[jet,jet],[W]]](MET,MET)
 13.0  1.047E+02    #        [[[b,t],[W]],[[jet,jet],[W]]](MET,MET)
 [.....]
```

Summary information

Topologies not covered by the results in the database (SModelS bracket notation)

The `parameters.ini` file

The basic options and parameters used by `runSModelS.py` are defined in the parameters file. A commented example, including all available parameters together with short descriptions, is given in `parameters.ini` in the top-level directory.

```
[options]
checkInput = True ; Set True to check the input file for possible errors
doInvisible = True ; Set True if invisible compression should be performed, False elsewise
doCompress = True ; Set True if mass compression should be performed, False elsewise
computeStatistics = True ; Set True to compute likelihood and chi2 for EM results, False elsewise
testCoverage = True ; Set True if topologies not covered by experiments (missing topologies)
should be identified, False otherwise
combineSRs = False ; Set True to combine signal regions when covariance matrix or pyhf JSON
likelihood is available. False uses only best SR.
```

v1.2.4 onwards,
see presentation by G. Alguero

https://smodels.readthedocs.io/en/latest/RunningSModelS.html#the-parameters-file

The `parameters.ini` file

The basic options and parameters used by `runSModelS.py` are defined in the parameters file. A commented example, including all available parameters together with short descriptions, is given in `parameters.ini` in the top-level directory.

```
[particles]
model=share.models.mssm ; path to particles module that defines the list of R-even and R-odd
particles. If omitted, we search in the current working directory, as well as “smodels/share/models”;
default is MSSM is the default.
# model=share.models.nmssm ; path to the particles module for the NMSSM.
....

[database]
path=official ; URL to the database pickle file (it will be downloaded)
#path=./smodels-database/ ; path to your local database (it will parsed and pickled)
```

https://smodels.readthedocs.io/en/latest/RunningSModelS.html#the-parameters-file

The `parameters.ini` file

The basic options and parameters used by `runSModelS.py` are defined in the parameters file. A commented example, including all available parameters together with short descriptions, is given in `parameters.ini` in the top-level directory.

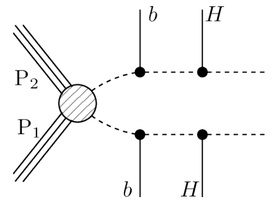
```
analyses = all  ; Set all to use all analyses included in the database
# to use only specific analyses, give a list of the names separated by comma
# analyses = CMS-PAS-SUS-13-008, CMS-SUS-13-013, ATLAS-SUSY-2013-04
# Wildcards are understood as in shell-expansion of file names: * ? [<list of letters>]
# Filter centre-of-mass energy with suffix beginning with a colon, in unum-style, like :13*TeV
# Note that the asterisk in the suffix is not a wildcard.
....

[printer]
outputType = summary  ; Define the output format(s)
# available output formats: summary, stdout, log, python, xml, slha
....
```

For detailed description of [each type of] output, see OutputDescription section in the online manual.

https://smodels.readthedocs.io/en/latest/RunningSModelS.html#the-parameters-file

Exercise

- Run SModelS for the three SLHA files (simplified MSSM spectra) provided in the “testPoints” folder on Indico:
  - T6bbHH_1400_600_60_1400_600_60.slha
  - TChiWH_750_100_750_100.slha
  - TStauStau_380_61_380_61.slha
- Switch combineSRs = False → True in the parameters file.
- What differences do you observe?



T6bbHH_1400_600_60_1400_600_60.slha

```
#Analysis  Sqrts  Cond_Violation  Theory_Value(fb)  Exp_limit(fb)  r  r_expected

combineSRs = False
--------------------------------------------------------------------------------
ATLAS-SUSY-2018-31  1.30E+01    0.0  2.693E-02  3.680E-02  7.319E-01  6.053E-01
Signal Region:  SRA_H
Txnames:  T6bbHH
Chi2, Likelihood =  3.621E+00  1.167E-02

combineSRs = True
--------------------------------------------------------------------------------
ATLAS-SUSY-2018-31  1.30E+01    0.0  6.176E-02  6.269E-02  9.852E-01  7.641E-01
Signal Region:  (combined)
Txnames:  T6bbHH
Chi2, Likelihood =  3.314E-01  8.239E-22
```

Non-SUSY example: Inert Doublet Model (IDM)

- set `model=share.models.idm` in parameters.ini file
- `./runSModelS.py -f inputFiles/slha/idm_example.slha -p parameters.ini -o results/.`

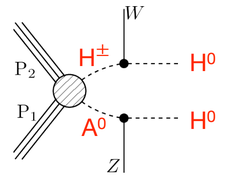
```
Decomposition output status: 1 #decomposition was successful
# Input File: inputFiles/slha/idm_example.slha
[…]
================================================================================
#Analysis  Sqrts  Cond_Violation  Theory_Value(fb)  Exp_limit(fb)  r  r_expected

 CMS-SUS-17-004  1.30E+01    0.0  8.062E+00  3.103E+02  2.598E-02  N/A
 Signal Region:  (UL)
 Txnames:  TChiWZ
[…]
--------------------------------------------------------------------------------
 ATLAS-SUSY-2018-06  1.30E+01    0.0  8.062E+00  4.418E+02  1.825E-02  2.709E-02
 Signal Region:  (UL)
 Txnames:  TChiWZ
[…]
--------------------------------------------------------------------------------
 ATLAS-SUSY-2016-24  1.30E+01    0.0  1.999E-03  1.300E-01  1.538E-02  1.234E-02
 Signal Region:  SR2-int
 Txnames:  TChiWZ
 Chi2, Likelihood =  4.904E-01  3.208E-02
[…]
```

$r = \sigma_{signal}/\sigma_{excluded}$

expected limits are available

efficiency map result

```
BLOCK MASS
  25 1.250000e+02 # mh
  35 92.7974884679 # MH0
  36 258.378192106 # MA0
  37 258.276962119 # MHch
```




```
================================================================================
The highest r value is = 0.02598365353065823

Total cross section considered (fb):  6.104E+01
Total missing topology cross section (fb):  4.795E+01
Total cross section where we are outside the mass grid (fb):  0.000E+00
Total cross section in long cascade decays (fb):  0.000E+00
Total cross section in decays with asymmetric branches (fb):  4.795E+01

Full information on unconstrained cross sections
================================================================================
Missing topologies with the highest cross sections (up to 10):
Sqrts (TeV)   Weight (fb)       Element description
 13.0  2.807E+01    #                  [[],[[W]]](MET,MET)
 13.0  1.988E+01    #                  [[],[[Z]]](MET,MET)
```

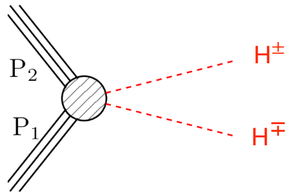
- output: ./results/idm_example.slha.smodels

Non-SUSY example: Inert Doublet Model (IDM)

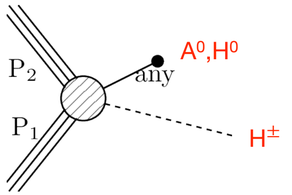
- set `model=share.models.idm` in parameters.ini file
- `./runSModelS.py -f inputFiles/slha/idm_longLived.slha -p parameters.ini -o results/.`

```
Decomposition output status: 1 #decomposition was successful
# Input File: inputFiles/slha/idm_longLived.slha
[....]
================================================================================
#Analysis  Sqrts  Cond_Violation  Theory_Value(fb)  Exp_limit(fb)  r  r_expected

 CMS-PAS-EXO-16-036  1.30E+01    0.0  3.092E+01  6.240E-01  4.955E+01  7.091E+01
 Signal Region:  c000
 Txnames:  THSCPM1b, THSCPM2b
 Chi2, Likelihood =  6.602E+02 4.190E-145
--------------------------------------------------------------------------------
    CMS-EXO-13-006  8.00E+00    0.0  2.053E+01  1.020E+00  2.013E+01  1.884E+01
 Signal Region:  c000
 Txnames:  THSCPM1b, THSCPM2b
 Chi2, Likelihood =  2.905E+01  1.337E-09
[....]
================================================================================
The highest r value is = 49.54788699262764

Total cross section considered (fb):  1.128E+02
[....]
```

```
BLOCK MASS
   25 1.250000e+02 # mh
   35 150.441511906 # MH0
   36 152.128899365 # MA0
   37 150.451750755 # MHch

DECAY  37   1.705000e-23
#  BR             NDA   ID1     ID2    ...
   1.000000e+00   3     -11   12   35 # 1.705e-23
```



[[],[]]
(HSCP,HSCP)



[[*],[]]
(MET,HSCP)

- output: ./results/idm_longLived.slha.smodels

Including other models — the particles module

- SModelS includes predefined model files for the MSSM, NMSSM, MDGSSM and the IDM (stored in `smodels/share/models/`).
- To include a new model, it suffices to specify the particle content in an analogous model file, defining which states are Z_2-even (`rEven` block) and which are Z_2-odd (`rOdd` block).
- For the odd states, the quantum numbers need to be specified as well (`qNumbers` block). Format:

  PDG: [spin*2, electrical charge*3, color dimension]

- micrOMEGAs automatically writes the respective `particles.py` file for any new model to be used with SModelS (and also writes the decay blocks and cross sections, see slide 15)

```
rEven = {
      25 : "higgs",
      23 : "Z",
      22 : “photon",
      24 : “W+",
      […]
}

rOdd = {
       35 : "H0",
      -35 : "H0",
       36 : "A0",
      -36 : “A0",
       37 : "H+",
      -37 : “H-"
}

qNumbers = {
   35:[0,0,1],
   36:[0,0,1],
   37:[0,3,1],
}
```

`idm.py`

SModelS as a python library — your own code

- Users more familiar with Python and the SModelS language may prefer to write their own main program. A simple example code for this purpose is provided in Example.py on the top-level directory.

```
from smodels import particlesLoader
from smodels.theory import slhaDecomposer,lheDecomposer
from smodels.tools.physicsUnits import fb, GeV, TeV
from smodels.theory.theoryPrediction import theoryPredictionsFor
from smodels.experiment.databaseObj import Database
from smodels.tools import coverage
from smodels.tools.smodelsLogging import setLogLevel
setLogLevel("info")

[…]
```

- Step-by-step explanations: RunningSModelS.html#example-py

- Online tutorial from PyHEP2020 conference: jupyter notebook and youtube video !

SModelS Tools

`smodelsTools.py` provides a number of convenient applications for the user:

- **`xseccomputer`** : a cross section calculator (for MSSM particles) based on Pythia and NLLfast,
- **`slhachecker`** and **`lhechecker`** : to check SLHA or LHE input files for completeness and sanity,
- **`database-browser`** : for easy direct access to the database of experimental results,
- **`interactive-plots`** : a plotting tool to make interactive plots based on plotly.

Other useful functionalities provided through smodelsTools.py:

- **`fixpermissions`** : to fix a problem with file permissions for the cross section computers in system-wide installs,
- **`toolbox`** : to quickly show the state of the external tools.

https://smodels.readthedocs.io/en/stable/SModelSTools.html

Interface from micrOMEGAs

micrOMEGAs has, since its version 4.3, an interface to SModelS which **automatically writes** for any model:

- an **SLHA-type input file for SModelS**, `smodels.in`, containing the *mass spectrum, decay tables and production cross sections* (!) for the parameter point under investigation
- the **particles.py** file defining the `even' and `odd' particle content of the model for SModelS

The **output** is written in SLHA format to `smodels.in.smodelsslha`, (or an alternative name selected by the user).

SModelS parameters are set in `micromegasInterfaceParams.ini` in micromegas' `include` folder.

```
#define SMODELS

[…]

#ifdef SMODELS
{   int result=0;
    double Rvalue=0;
    char analysis[30]={},topology[30]={};
    int LHCrun=LHC8|LHC13;
#include "../include/SMODELS.inc"
}
#endif
```

main.c

Detailed description in arXiv:1606.03834
(see also the manual in micrOMEGAs' `man/` folder)

When you use SModelS v1.2.4, please cite:

- ***A SModelS interface for pyhf likelihoods***, ← **SModelS v1.2.4**
  Gaël Alguero, Sabine Kraml, Wolfgang Waltenberger, arXiv:2009.01809
- ***SModelS database update v1.2.3***,
  Charanjit K. Khosa, Sabine Kraml, Andre Lessa, Philipp Neuhuber, Wolfgang Waltenberger, arXiv:2005.00555, LHEP 158 2020
- ***SModelS v1.2: long-lived particles, combination of signal regions, and other novelties***,
  Federico Ambrogi et al., arXiv:1811.10624, CPC 251 (2020) 106848
- ***Constraining new physics with searches for long-lived particles: Implementation into SModelS***,
  Jan Heisig, Sabine Kraml, Andre Lessa, arXiv:1808.05229, Phys.Lett. B788 (2019) 87-95.
- ***SModelS extension with the CMS supersymmetry search results from Run 2***,
  Juhi Dutta, Sabine Kraml, Andre Lessa, Wolfgang Waltenberger, arXiv:1803.02204, LHEP 1 (2018) no.1,5-12
- ***SModelS v1.1 user manual: improving simplified model constraints with efficiency maps***,
  Federico Ambrogi, Sabine Kraml, Suchita Kulkarni, Ursula Laa, Andre Lessa, Veronika Magerl, Jory Sonneveld, Michael Traub, Wolfgang Waltenberger, arXiv:1701.06586, CPC 227 (2018) 72-98
- ***SModelS: a tool for interpreting simplified-model results from the LHC and its application to supersymmetry***,
  Sabine Kraml, Suchita Kulkarni, Ursula Laa, Andre Lessa, Wolfgang Magerl, Doris Proschofsky, Wolfgang Waltenberger, arXiv:1312.4175, EPJC (2014) 74:2868