

sntools: A Supernova Event Generator

Dr. Jost Migenda (they/them)

KING'S
College
LONDON



sntools: An event generator for supernova burst neutrinos

Jost Migenda^{1,2}, Susan Cartwright², Liz Kneale², Matthew Malek², Yan-Jie Schnellbach³, and Owen Stone²

1 King's College London 2 University of Sheffield 3 University of Liverpool

DOI: [10.21105/joss.02877](https://doi.org/10.21105/joss.02877)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Juanjo Bazán](#) ↗

Reviewers:

- [@bradkav](#)
- [@benjaminrose](#)

Submitted: 23 November 2020

Published: 15 April 2021

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

In partnership with



This article and software are linked with research article DOI [10.3847/1538-4357/abf7c4](https://doi.org/10.3847/1538-4357/abf7c4), published in the *The Astrophysical Journal*.

Summary

Supernovae are stellar explosions that produce many of the chemical elements necessary for life to exist and form neutron stars and black holes. However, despite millennia of observations and almost a century of in-depth study, the explosion mechanism of supernovae is not yet well understood.

Observing the neutrino signal from the next galactic supernova can help solve this puzzle. To prepare for this once-in-a-lifetime event, it is essential to study how various detectors respond to the neutrino fluxes from different supernovae. sntools helps with this by providing a link between computer simulations of supernovae and those of neutrino detectors: From the neutrino fluxes predicted by a given supernova model, it generates data sets of neutrino interactions, taking into account detailed, energy-dependent cross-sections for all relevant interaction channels. These data sets can then be used as input for a full detector simulation and event reconstruction toolchain.

Statement of Need

sntools is an event generator for supernova burst neutrinos which is written in Python and makes extensive use of numpy ([Walt et al., 2011](#)) and scipy ([Virtanen et al., 2020](#)). It currently supports multiple detector configurations using either water, liquid scintillator or water-based liquid scintillator as a detection material. It also supports several different input formats for neutrino fluxes from various computer simulations. New detector configurations, materials or input formats can be added easily.

sntools was initially developed to study supernova model discrimination with Hyper-Kamiokande ([Migenda, 2019](#)) and is also used to develop a supernova DAQ system for the same experiment. More recently, sntools was adapted by the WATCHMAN ([Askins et al., 2015](#)) experiment and for early studies for the THEIA detector concept ([Askins et al., 2020](#)).

Paper:

[DOI:10.21105/joss.02877](https://doi.org/10.21105/joss.02877)

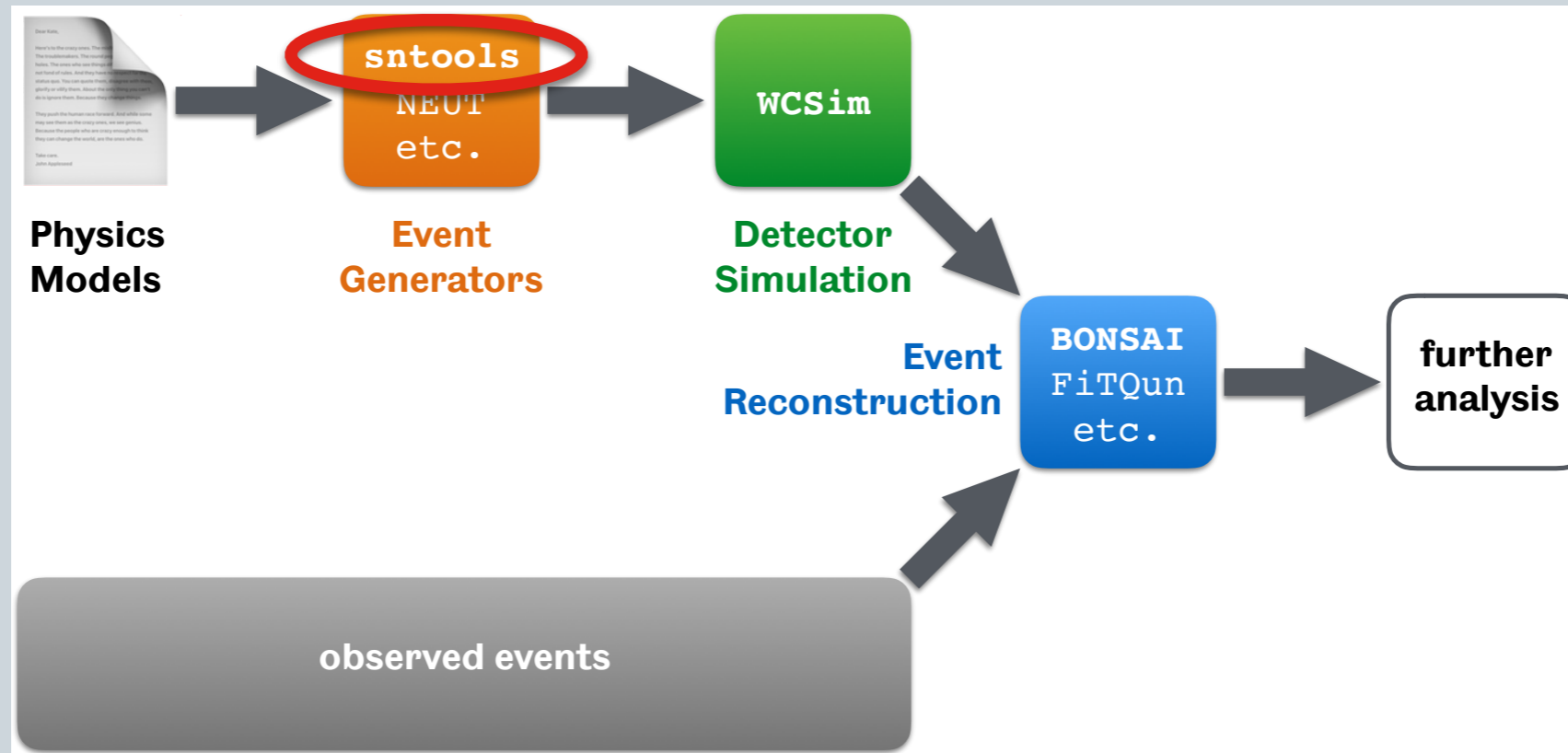
Code:

github.com/JostMigenda/sntools



(Very) Brief History of sntools

- ♦ 2017–2019: developed as SN event generator for Hyper-Kamiokande (with support for Super-K)



- ♦ Early 2020: worked for WATCHMAN (a.k.a. AIT/NEO)
 - ♦ Added support for LS & WbLS (including additional interaction channels on ^{12}C)
 - ♦ Added WATCHMAN and THEIA geometries

Architecture of sntools

- ♦ Written in Python, based on Numpy, SciPy
- ♦ Goal: modular & easy to extend

- ♦ New detector:
5 LOC

```
elif name == "WATCHMAN-WbLS":  
    self.shape = "cylinder"  
    self.height = 1280  
    self.radius = 1280 / 2  
    self.material = wbls(0.03) # 3% LS, 97% water
```

- ♦ New material:
5 LOC

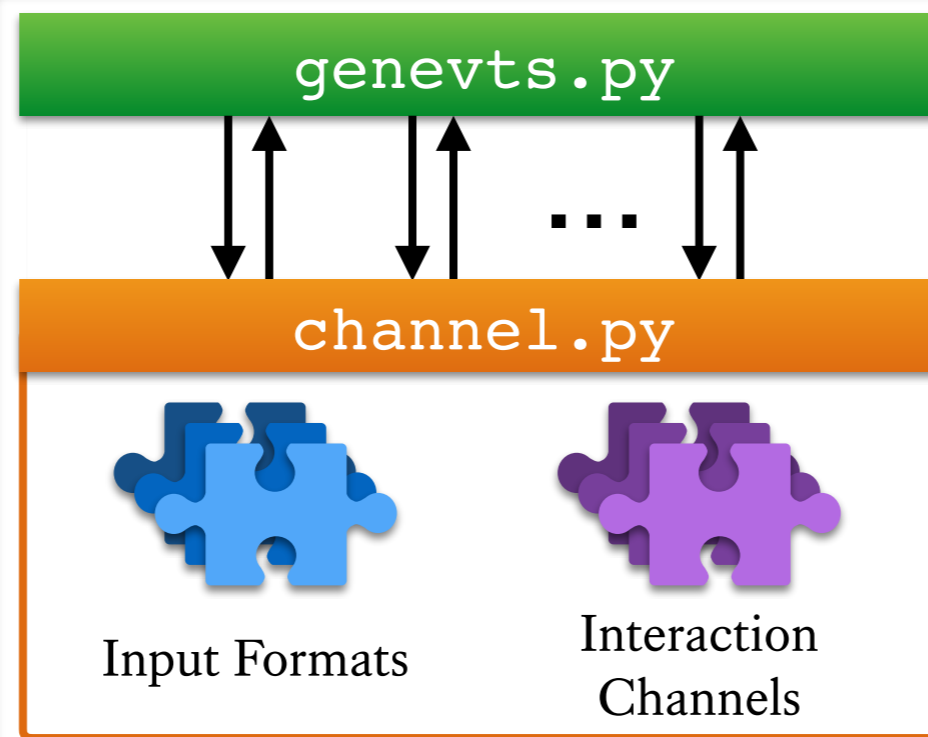
```
water = {  
    "molecular_weight": 18.0153, # g/mol  
    "density": 1.0, # g/cm^3  
    "channel_weights": {"ibd": 2, "es": 10, "o16e": 1, "o16eb": 1}, # targets per molecule  
}
```

- ♦ New interaction channel or input format:
typically <100 LOC

Architecture of sntools

genevts.py

- Provide command line interface
- Call channel.py for each channel/flavour to generate events
- Write output file for detector simulation



channel.py

- Uses modules for various input file formats & interaction channels
- Read in ν flux
- Calculate event count in each 1 ms bin
- Generate event times, energies, directions

Typical usage:

(see `sntools --help` for more options)

```
$ sntools flux.dat --format=gamma --ordering=noosc --detector=HyperK --channel=ibd
                               warren2020      normal      SuperK      es
                               nakazato         inverted    THEIA-25    o16e
                               totani           THEIA-100   o16eb
                               princeton       WATCHMAN    c12e
                                               WATCHMAN-LS c12eb
                                               WATCHMAN-WbLS c12nc
```

Comparison at a Glance

	SNEUPy (+SNOWGLOBES)	sntools
Programming language	Python (+C, Perl)	Python
Input formats for neutrino fluxes	multiple, easily extensible	multiple, easily extensible
Flux transformations	none, adiabatic MSW, neutrino decay, ...	none, adiabatic MSW
Cross sections	tabulated	calculated at runtime
Handling of detector effects	approximated by smearing matrix	left to experiments' own sim & reco toolchain
Output	event rates, binned in time & energy	list of true events, distributed in detector
	Event rate calculator	Event generator*

sntools as a Cross Section Library

- ♦ sntools includes code to calculate cross sections for all implemented channels
- ♦ Can treat it as a library & use it from other code:

```
from sntools.interaction_channels import ibd # or nu-e scattering, 016-CC, ...

ibd.bounds_eE(15) # kinematically allowed e+ energies for a 15 MeV neutrino
>> [13.281429092279229, 13.70577544781821]

ibd.dSigma_dE(eNu, eE) # differential cross section as function of e+ energy
ibd.dSigma_dCosT(eNu, cosT) # differential cross section as function of e+ angle
# ... and more
```

- ♦ If you're interested, please talk to me about it!