Documentation of the SOHO/EPHIN Level3varbins data product

Patrick Kuehl

April 11, 2019

Contacts

- Patrick Kühl, kuehl@physik.uni-kiel.de
- Bernd Heber, heber@physik.uni-kiel.de

Contents

1 Introduction

The SOHO/EPHIN Leve3 varbins data product is basically only a wrapper to produce any given energy channels from 5-50 MeV/nuc for protons and helium based on the SOHO/EPHIN Level3 data product (see `http://www.ieap.uni-kiel.de/et/people/kuehl/ephin_level3/DOCUMENTATION-COSTEP-EPHIN-L3-20181002.pdf`). The code given in section 8 uses Input files as given in Section 2.1 and calculates all necessary quantities such as energy loss thresholds and response factors for each channel based on simulation results. The resulting macro file (section 2.2) is than used by the code given in section 9 to calulate channel fluxes as well as their uncertainties based on measured PHA data.

2 Input macros

2.1 Input file

The following exemplaric input file will create 8 proton and 8 helium channels:

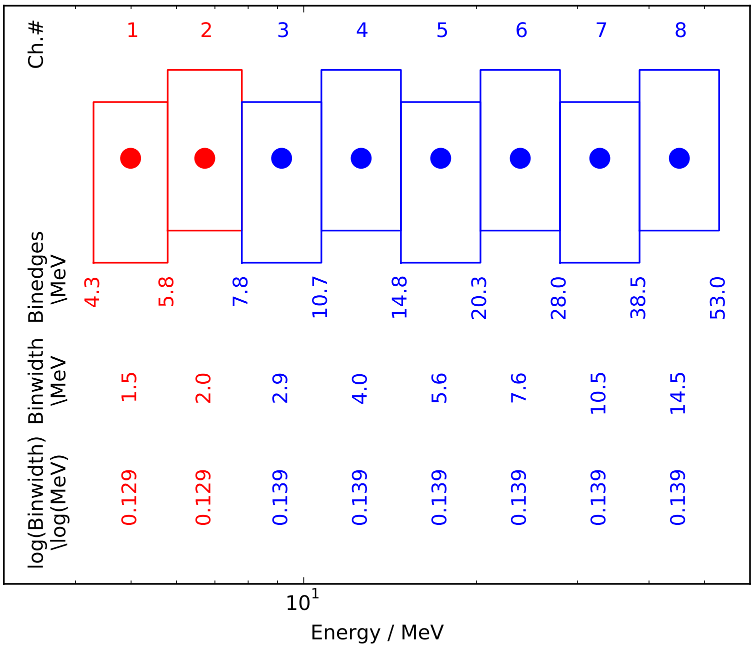
```
# inout in order to create l3imacro file
# channelname particletype minenergy maxenergy
#
# NOTE!!!: energies channels must be seperated at 7.8 MeV/nuc!
# NOTE!!!: Channels above 7.8 are not allowed to have gabs in-between them
# NOTE!!!: Highest channel must have 53 MeV/nuc as upper limit
#
P02+06:01 p 4.30 5.79
P02+06:02 p 5.79 7.80
P02+06:03 p 7.80 10.73
P02+06:04 p 10.73 14.77
P02+06:05 p 14.77 20.33
P02+06:06 p 20.33 27.98
P02+06:07 p 27.98 38.51
P02+06:08 p 38.51 53.00
He02+06:01 he 4.30 5.79
He02+06:02 he 5.79 7.80
He02+06:03 he 7.80 10.73
He02+06:04 he 10.73 14.77
He02+06:05 he 14.77 20.33
He02+06:06 he 20.33 27.98
He02+06:07 he 27.98 38.51
He02+06:08 he 38.51 53.00
```

2.2 Macrofile derived based on simulation

The derived macro file based on the input given in section 2.1:

```
# chname type range geom geom_syserr geom_roff geom_roff_syserr th1 th2 emin emax
#
# chname: string - channel name
# type: string - particle type ('p': proton, 'he': helium)
# range: string - particle range ('AB': detector A and B, 'ABC': detector A, B and C)
#     Note: range should be 'AB' below ~8 MeV/nucleon and 'ABC' above ~8MeV/nucleon
# geom: float - geom factor in units of 'cm**2 sr MeV' if ring is on
# geom_syserr: float - systematic uncertainty of geom factor in units of 'cm**2 sr MeV' if ring is on
# geom_roff: float - geom factor in units of 'cm**2 sr MeV' if ring is off
# geom_roff_syserr: float - systematic uncertainty of geom factor in units of 'cm**2 sr MeV' if ring is off
# th1: float - lower threshold for E_A+E_B in units of 'MeV', i.e.: E_A+E_B>=th1
# th2: float - upper threshold for E_A+E_B in units of 'MeV', i.e.: E_A+E_B<th2
#     Note: for range='AB' these thresholds correspond to the total energy of the particles
#     Note: for range='ABC' these thresholds correspond to the differential energy losses of these particles
# emin: float - lower energy limit in MeV/nuc
# emax: float - upper energy limit in MeV/nuc
#
#
P02+06:01 p AB 6.6245 0.2621 0.2353 0.0100 4.3000 5.7900 4.3 5.79
P02+06:02 p AB 9.7642 0.3729 0.2884 0.0178 5.7900 7.8000 5.79 7.8
P02+06:03 p ABC 12.2522 0.6845 0.4969 0.0061 4.3662 9.9734 7.8 10.73
P02+06:04 p ABC 19.7197 0.7016 0.8127 0.0446 3.0552 4.3662 10.73 14.77
P02+06:05 p ABC 27.0977 1.0887 0.8337 0.0282 2.2790 3.0552 14.77 20.33
P02+06:06 p ABC 36.2607 1.0259 1.2283 0.0906 1.7741 2.2790 20.33 27.98
P02+06:07 p ABC 51.3567 0.7673 1.8857 0.1278 1.3591 1.7741 27.98 38.51
P02+06:08 p ABC 60.2701 1.7503 2.3273 0.2298 1.0523 1.3591 38.51 53.0
He02+06:01 he AB 6.6647 0.3069 0.1843 0.0069 4.3000 5.7900 4.3 5.79
He02+06:02 he AB 9.8107 0.3422 0.3476 0.0145 5.7900 7.8000 5.79 7.8
He02+06:03 he ABC 12.7092 0.7048 0.4671 0.0110 17.1879 39.8936 7.8 10.73
He02+06:04 he ABC 19.9220 0.8230 0.7650 0.0288 12.0269 17.1879 10.73 14.77
He02+06:05 he ABC 25.5887 1.0170 0.8106 0.0312 9.0676 12.0269 14.77 20.33
He02+06:06 he ABC 37.6467 0.9873 1.3476 0.0918 6.9837 9.0676 20.33 27.98
He02+06:07 he ABC 51.7863 1.2413 1.7902 0.0858 5.3217 6.9837 27.98 38.51
He02+06:08 he ABC 59.9005 0.5849 2.2939 0.1944 4.1647 5.3217 38.51 53.0
```

In order to visualize the energy coverage of different channel binning, figure 1 shows 8 (top) and 16 (bottom) protons bins for geometrically spaced channels (i.e. the channels are evenly spaced on a logarithmic scale).



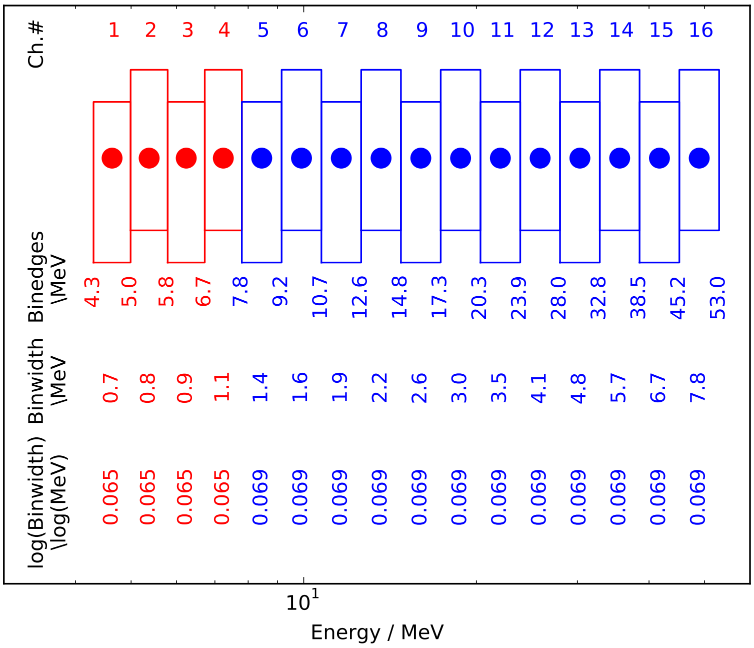


Figure 1: Example of bin spacing for 8 (top) and 16 (bottom) proton bins. AB coincidences are marked in red, ABC in blue.

3 Comparison between different Binning

The nominal energy bins have been validated in the Level3 documentation (`http://www.ieap.uni-kiel.de/et/people/kuehl/ephin_level3/DOCUMENTATION-COSTEP-EPHIN-L3-20181002.pdf`). Hence, a comparison beteen these nominal channels and newly created channels is a reasonable indicator for the data quality. Figure 2 shows the daily proton and helium spectra on January, 11th, 2000 for the nominal energy bins (blue and green for protons and helium, respectively) as well as for 8 (red and cyan) and 16 bins (purple and yellow). All different binnings are in good agreement.
Figure 3 shows the annual fluence spectra for these bins (same color coding as in figure 2). All spectra are once again in good agreement.

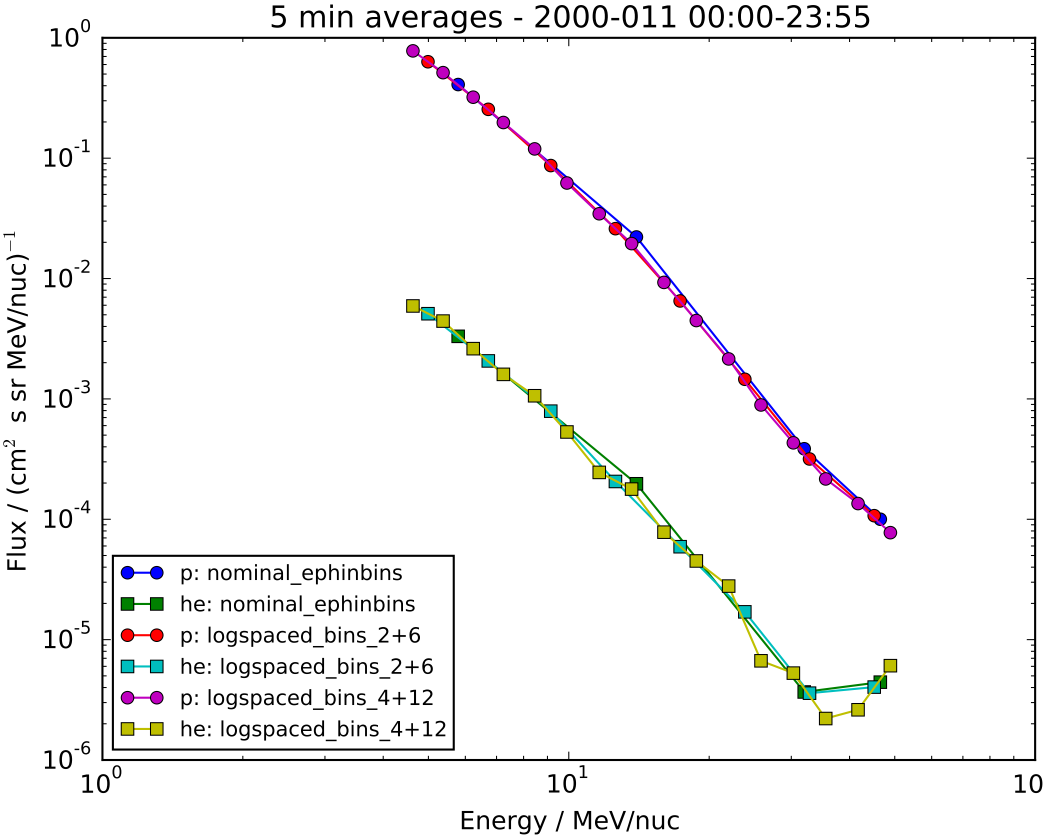


Figure 2: Daily spectra of proton and helium for different channel binning.

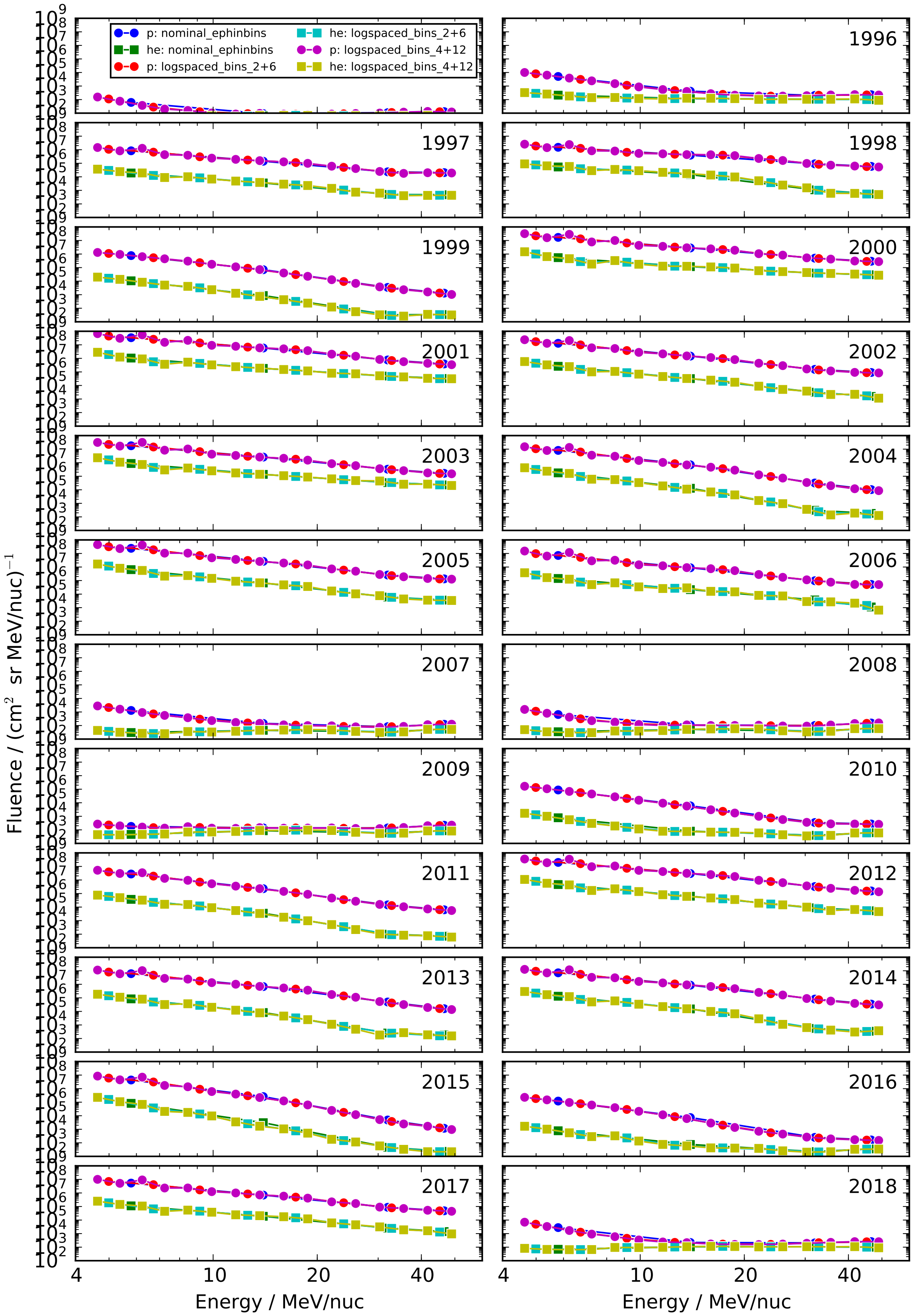


Figure 3: Annual fluence spectra of proton and helium for different channel binning.

4 Comparison with different Missions

The possibility to use any energy ranges for the different channels allows detailed comparisons with different mission. As an example, Figures 4 and 5 show comparisons between SOHO/ERNE SL2 data (`https://srl.utu.fi/export/`) in six different energy channels for protons and helium, respectively. The figures present data from 1996 until 2018 based on one minute time resolution. Please note the logarithmic color scale.

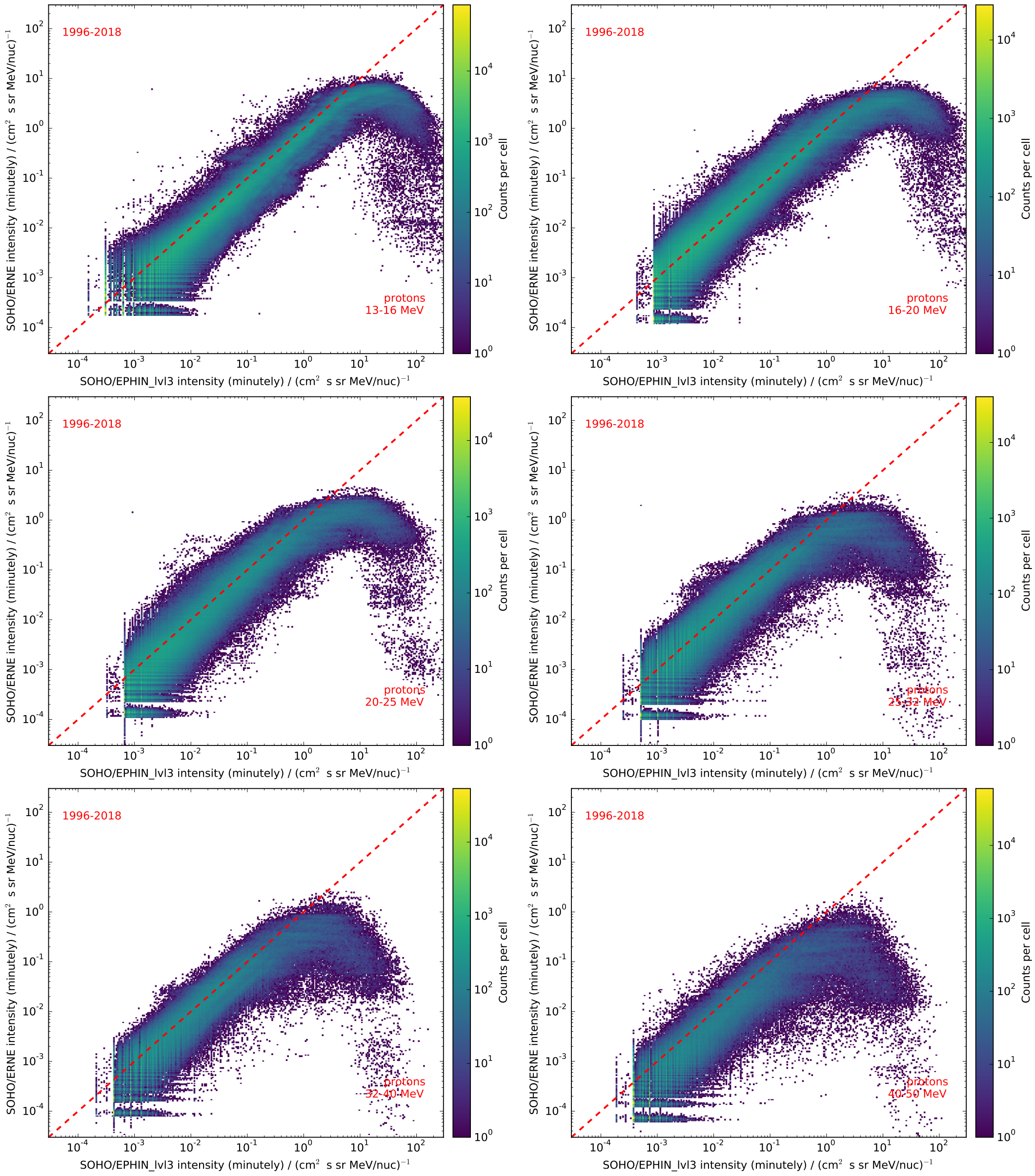


Figure 4: Comparison between SOHO/ERNE and SOHO/EPHIN for 6 different proton channels

Except for dead time issues of SOHO/ERNE during high fluxes, the proton intensities are in good agreements (especially regarding the non-ideal response function of the SOHO/EPHIN Level3 data, see `http://www.ieap.uni-kiel.de/et/people/kuehl/ephin_level3/DOCUMENTATION-COSTEP-EPHIN-L3-20181002.pdf`). In addition to the dead time effect and more limited statistics, systematic differences between SOHO/EPHIN and SOHO/ERNE can be observed. In detail, EPHIN seems to over-, and/or ERNE seems to underestimate the fluxes. It has to be noted, that a similar behaviour was observed in the validation/comparison with the nominal EPHIN energy ranges.

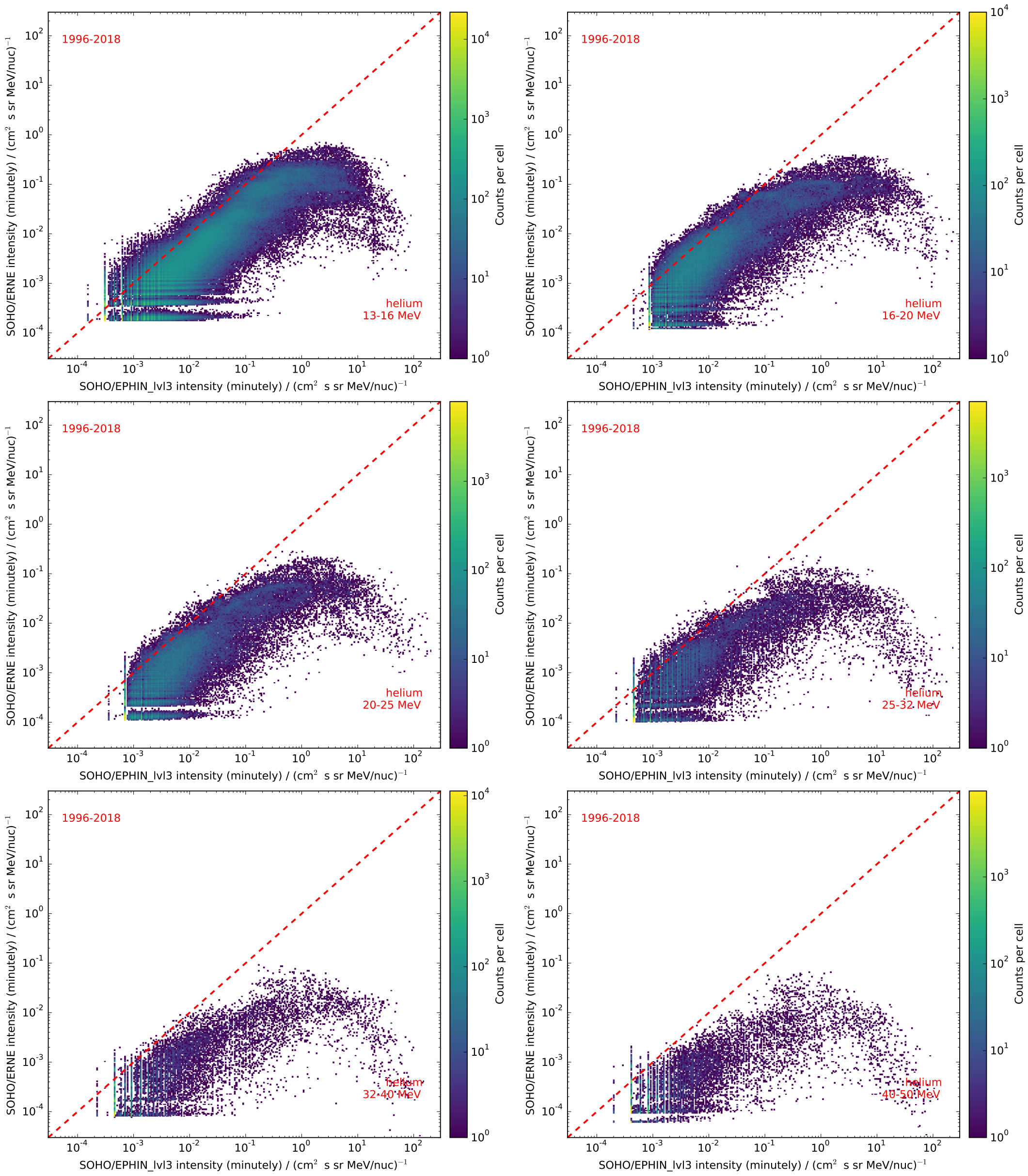


Figure 5: Comparison between SOHO/ERNE and SOHO/EPHIN for 6 different helium channels

In order to analyze the deviation between the SOHO/ERNE and SOHO/EPHIN Level3 varbin data, figure 6 presents the difference of the fluxes between the two missions divided by the uncertainty of the EPHIN data (including both, the systematic and statistical uncertainty). The shaded faces represent gaussian distribution with 1 σ, 2 σ and 3σ to guide the eye. Figure 7 presents the same analysis for fluxes above the background level (i.e. EPHIN flux above 10^{-2} (cm^2 sr s MeV)$^{-1}$). Both figures confirm the systematic shift (especially for helium) that was found in figures 4 and 5. However, the deviations are 1) in the order of the uncertainty derived for the data, 2) similar to what was found for the nominal Level3 channels and 3) understandable due to the non-ideal response functions of the Level3 channels (compare `http://www.ieap.uni-kiel.de/et/people/kuehl/ephin_level3/DOCUMENTATION-COSTEP-EPHIN-L3-20181002.pdf`).

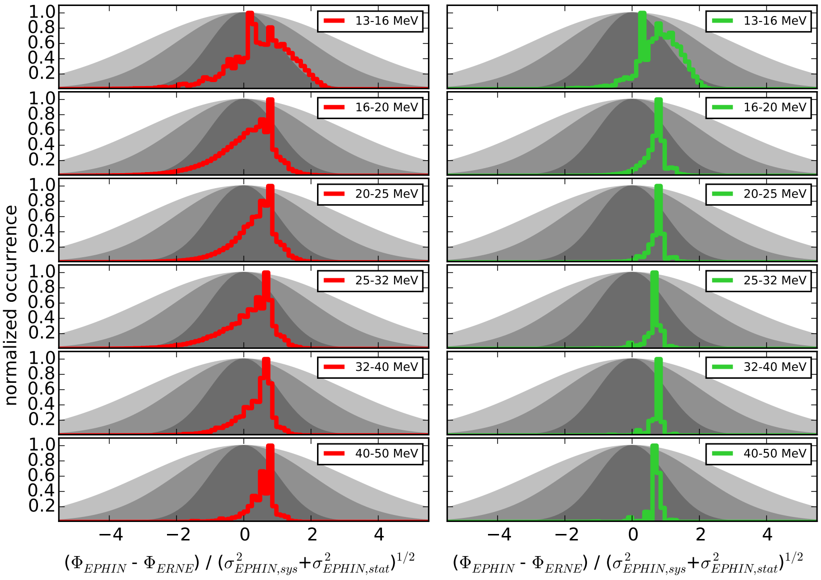


Figure 6: Comparison between SOHO/ERNE and SOHO/EPHIN with respect to the EPHIN uncertainties. The data set excludes timeperiods during which either of the two missions measured a flux of zero.

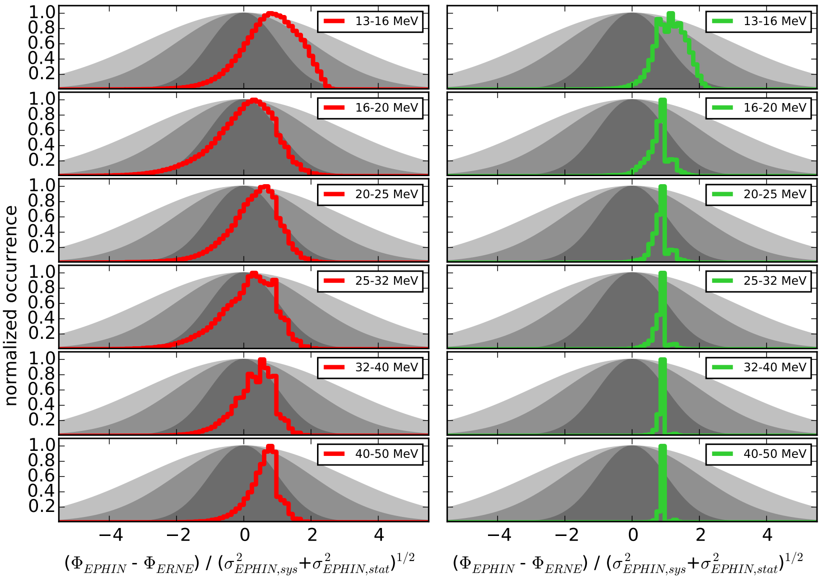


Figure 7: Comparison between SOHO/ERNE and SOHO/EPHIN with respect to the EPHIN uncertainties. The data set excludes timeperiods during which EPHIN measured fluxes below 10^{-2} (cm^2 sr s MeV)$^{-1}$.

5 Applications

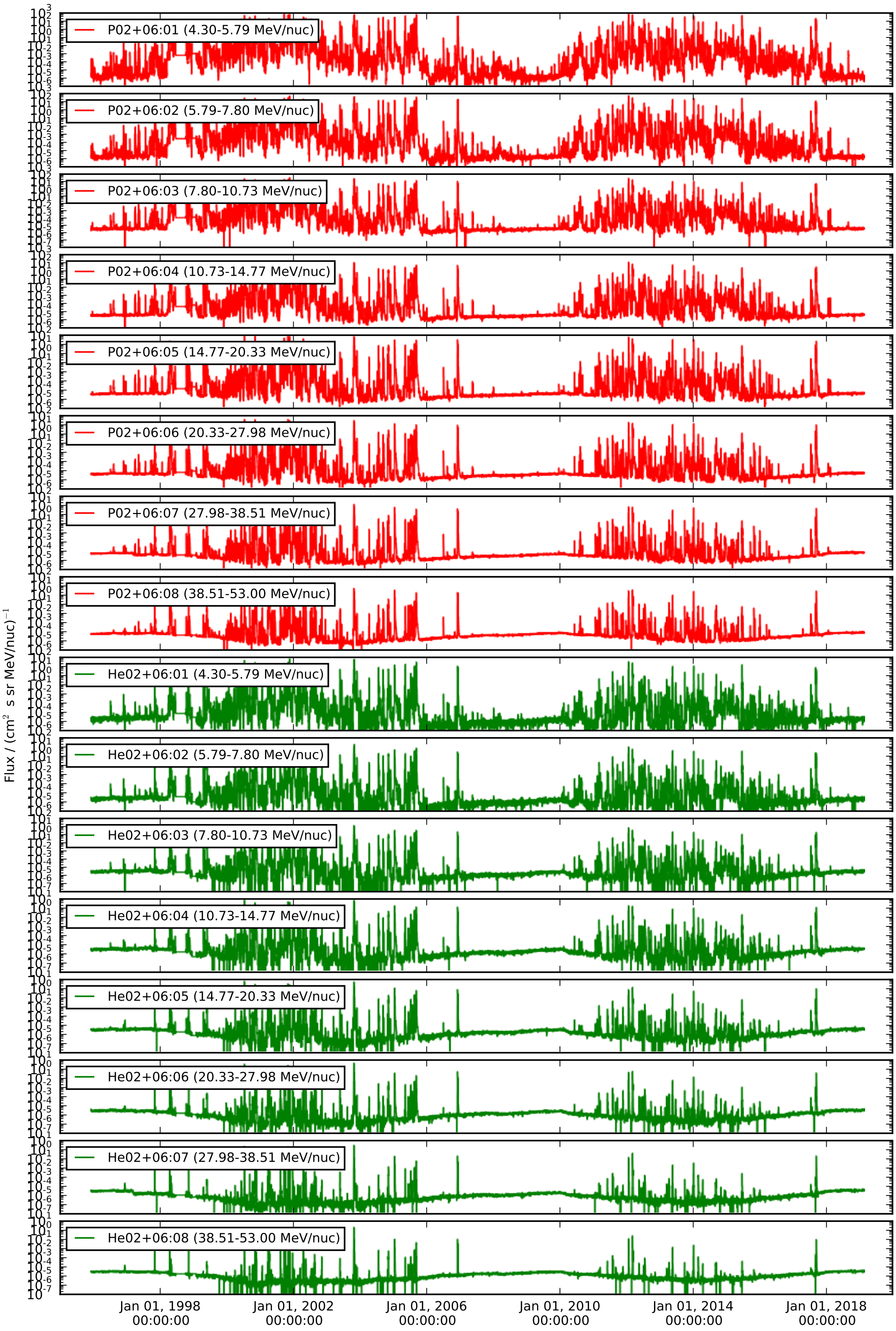


Figure 8: cap

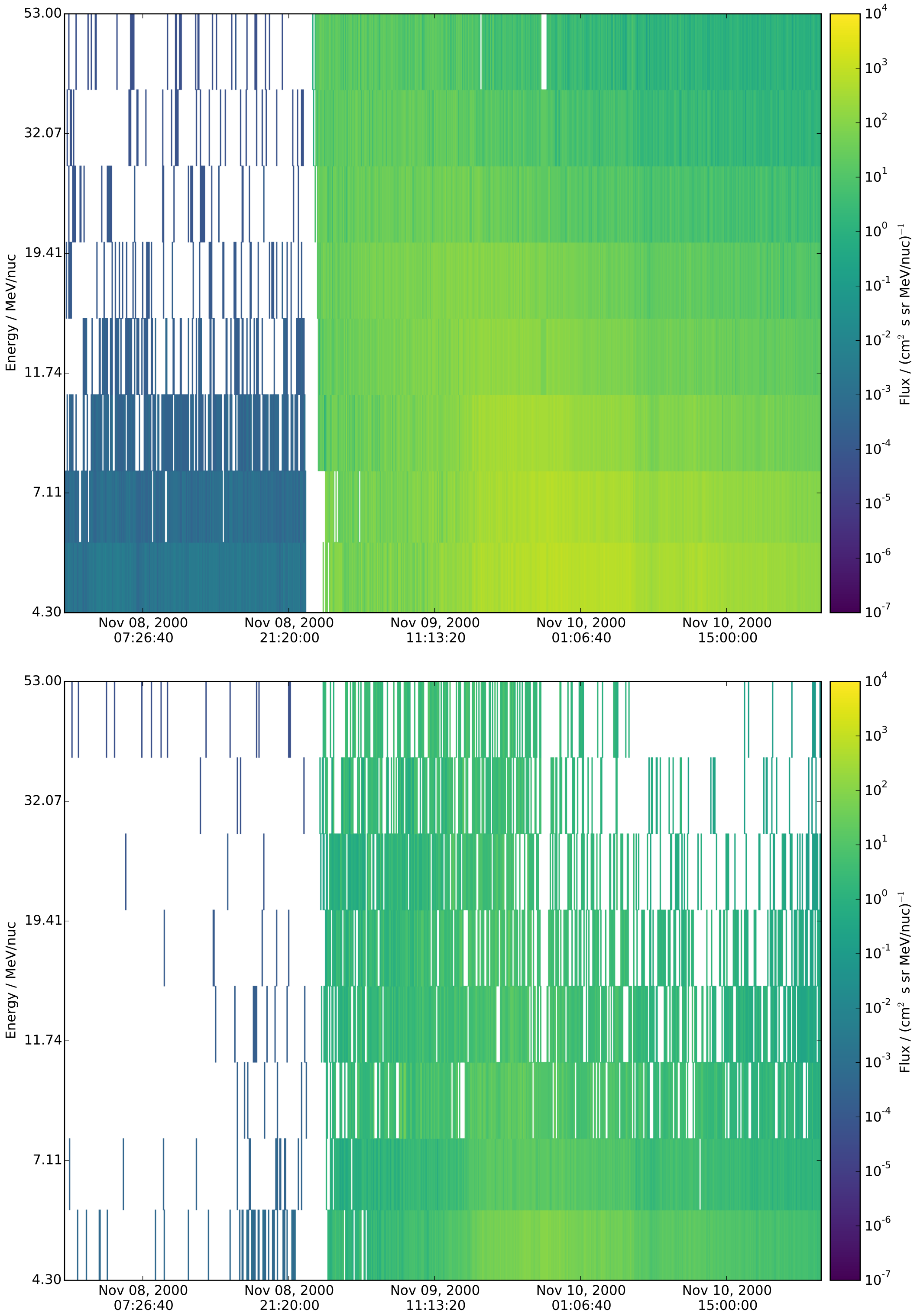

53.00
32.07
19.41
11.74
7.11
4.30
Energy / MeV/nuc
Nov 08, 2000 07:26:40
Nov 08, 2000 21:20:00
Nov 09, 2000 11:13:20
Nov 10, 2000 01:06:40
Nov 10, 2000 15:00:00
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
Flux / (cm^2 s sr MeV/nuc)$^{-1}$
53.00
32.07
19.41
11.74
7.11
4.30
Energy / MeV/nuc
Nov 08, 2000 07:26:40
Nov 08, 2000 21:20:00
Nov 09, 2000 11:13:20
Nov 10, 2000 01:06:40
Nov 10, 2000 15:00:00
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
Flux / (cm^2 s sr MeV/nuc)$^{-1}$

Figure 9: cap

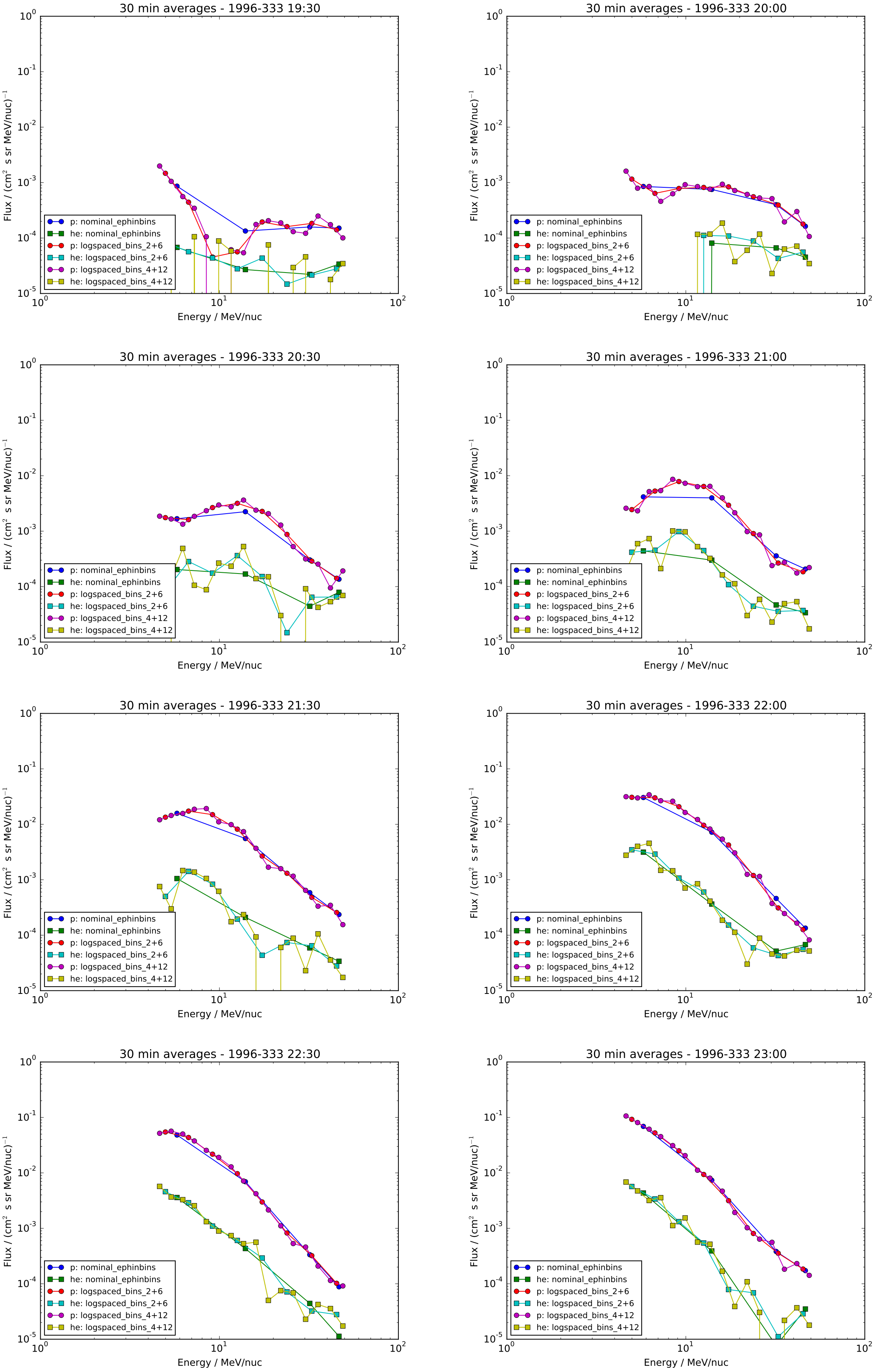


Figure 10: cap

6 Data Product

The created Level3 intensity files will be provided in different time resolutions as ASCII text files: 1, 5, 10, 30, 60 and 1440 minutes. The format of the data product is given in table 1. Note that

- the time given in the data set marks the beginning of the time interval
- the statistical and systematic uncertainties of a given channel are set to '-999' if the channel has zero counts in a time interval (the intensity will be '0' though)
- the 'type' column in the table describes the format of the data product with 'int', '4.4f' and '4.4e' refering to integer, float and scientific (float and exponent), respectively
- the status flag is a decimal code which results from the summation of the flag bit values given in table 2

item	label	data content	units	type
1	year	year	years	int
2	month	month	months	int
3	day	day	days	int
4	doy	day of year	days of year	int
5	hour	hour	hours	int
6	minute	minute	minutes	int
7	status	status flag	binary status word	int
8	accum.time	accumulation time	seconds	4.4f
9	int_ch1	channel 1 intensity	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e
10	sys_ch1	channel 1 systematic uncertainty	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e
11	stat_ch1	channel 1 statistical uncertainty	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e
:	:	:	:	:
6+3*n.	int_chn	channel n intensity	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e
7+3*n	sys_chn	channel n systematic uncertainty	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e
8+3*n	stat_chn	channel n statistical uncertainty	(cm^2 s sr Mev/nuc)$^{-1}$	4.4e

Table 1: Explaination of the data product of Level3varbins intensities.

Flag Bit Value	Remarks
0	Nominal observation, i.e. High Voltage ON, no failure mode, ring segment switching disabled
1	Failure Mode E
2	Ring A/B OFF
4	E patch uploaded
8	Commissioning
16	Standby or maintenance, i.e. High Voltage OFF
32	Calibration, i.e. test mode
64	TBD

Table 2: EPHIN status flag description (source: 'ephispec.doc')

The data structure of the Level3 varbins files is as follows::

- the main folder contains sub-directories for all time resolutions (e.g. 1, 5, 10, 30, 60 and 1440 minutes)
- all time resolution sub-directories have further sub-directories for each year which contain daily files
  - e.g. for 1 minute time resolution, year 2017 and day of year 1:
    main_directory/1min/2017/2017_001.l3i
- all time resolution sub-directories contain also annual files
  - e.g. for 10 minute time resolution, year 2001:
    main_directory/10min/2001.l3i
- the time resolution sub-directories for 60 and 1440 minutes also contain files for the entire mission
  - e.g. for 60 minute time resolution:
    main_directory/60min/entire_mission_60min.l3i

7 Code

All functions to create a the macro file for the Level3 varbin calculation are defined in the file

level3_varbins_macro_creator.py

The entire code can be found in section 8. In the following, an explanation of all defined functions is given:

energy_ranges Calculates the energy thresholds in $E_A + E_B$ for the ABC coincidences. A plot presenting the results will be also created and stored.

calc_response Calculates the response factors as well as their uncertainties for the different channels. A plot presenting the results will be also created and stored.

write_header Writes a standard header to the macro file

derive_macro The overall function that derives the macro file from the input file using the other functions defined above.

Furthermore, the code requires a set of paths (input and output) that have to be defined either in 'level3_funcs.py' or in a given executable script:

channelinput name of the input file (compare section 2.1) (e.g. 'logspaced_bins_2+6.l3iinput')

macrofile name of the file as which the derived macrofile should be saved (compare section 2.2) (e.g. 'logspaced_bins_2+6.l3imacro')

sim_pha_proton Location of the proton simulation file (e.g. '/home/pacifix/kuehl/work/simulations/G4ET_2015/build_level3_stopping/data/proton_fw.sim_pha')

sim_pha_helium Location of the helium simulation file (e.g. '/home/pacifix/kuehl/work/simulations/G4ET_2015/build_level3_stopping/data/helium_fw.sim_pha')

All function required to create the Level3varbins intensity files based on a given macro are defined in the file

level3_funcs_varbins.py

The entire code can be found in section 9. In the following, an explanation of all defined functions is given:

load_level2_pha Loads Level2 PHA data

load_level1_sci Loads Level1 SCI data

check_coinc Checks for and deletes wrong coincidences in the PHA

add_wfact_to_pha Synchronizes Level1 SCI and Level2 PHA data. Adds ratio of total counts and number of PHA words as well as the status word to PHA files (so-called PHAWS files)

phaws_from_year_doy Creates the PHAWS file for a given year and doy

int_in_lvl3_ch_from_ea_eb Calculates counts/(cm^2 sr MeV) for AB

int_in_lvl3_ch_from_ea_eb_ec Calculates counts/(cm^2 sr MeV) for ABC

load_macro Loads the macro file and sorts its information

calc_lvl3_intensities_timeresolution_varbins Calculates complete Level3 varbins intensity files for a given timeresolution (in minutes)

merge_level3_daily_to_annual Merges daily files of a given time resolution to annual files

Furthermore, the code requires a set of paths (input and output) that have to be defined either in 'level3_funcs.py' or in a given executable script:

macrofile name of the macro file that should be used (compare section 2.2) (e.g. '/data/etph/kuehl/ephin_lvl3_varbins/logspaced_bins_2+6.l3imacro')

lvl2_pha_path Location of the Level2 PHA data set (e.g. '/data/missions/soho/costep/level2/pha/')

lvl1_sci_path Location of the Level1 SCI data set (e.g. '/data/missions/soho/costep/level1/sci/')

phaws_path Storage location for PHAWS files (can be temporarly). Note: PHAWS is a combined dataset of PHA and SCI information that is created during the calculation of the Level3 intensities.

lvl3_out_path Output location for the Level3 varbin intensity files

8 Appendix I: level3_varbins_macro_creator.py

```python
# This scripts includes all function necessary in order to create macros for the EPHIN lvl3 ion varbins intensities
# Patrick Kuehl, June 7 2018 kuehl@physik.uni-kiel.de
#from pylab import *
import numpy as np
import numpy.ma as ma
import time as time
import datetime as dt
import os
np.seterr(divide=’ignore’, invalid=’ignore’)
import subprocess
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
from matplotlib.backends.backend_pdf import PdfPages


# sections and defined functions
"""
functions to calculate energy ranges
  energy_ranges(macro,verbosity)
functions to calculate response factors
  calc_response(macro,verbosity)
functions to derive complete macro
  write_header(file)
  derive_macro(verbosity=1)
"""

# paths (shall be defined in actual processing code)
"""
channelinput="nominal_ephinbins.l3iinput"
macrofile="nominal_ephinbins.l3imacro"
sim_pha_proton="/home/pacifix/kuehl/work/simulations/G4ET_2015/build_level3_stopping/data/proton_fw.sim_pha"
sim_pha_helium="/home/pacifix/kuehl/work/simulations/G4ET_2015/build_level3_stopping/data/helium_fw.sim_pha"
"""



""" functions to calculate energy ranges """
# calcs energy ranges
def energy_ranges(macro,verbosity):
  pidxs,heidxs=[],[]
  for idx in range(len(macro["chnames"])):
    macro["chabmin"].append("edmin")
    macro["chabmax"].append("edmax")
    if macro["chranges"][idx]=="AB":
      macro["chabmin"][idx]=macro["chemin"][idx]
      macro["chabmax"][idx]=macro["chemax"][idx]
    else:
      if macro["chtypes"][idx]=="p": pidxs.append(idx)
      if macro["chtypes"][idx]=="he": heidxs.append(idx)
  # now actually calc energy ranges for p and he
  plt.rcParams.update({’font.size’: 18})
  th_q2_lower=7.3 #*4
  th_q2_upper=27.5 #*4
  th_q2_lower_he=29.5 #*4
  th_q2_upper_he=110 #*4
  th_q1_lower=-0.35
  th_q1_upper=0.15
  pp= PdfPages('%s_energy_ths.pdf'%macrofile.split(".l3imacro")[0])
  for type in ["p","he"]:
      if verbosity==1:  print "  ---> Calculating ranges for %s channels..."%type
      hists=[]
      fig=plt.figure(figsize=(8,3))
      ax1=plt.gca()
      plt.subplots_adjust(left=0.1,top=0.95,bottom=0.23,right=0.97,hspace=0.1,wspace=0.27)
      if type=="p":
        simfile=sim_pha_proton
        idxs=pidxs
      if type=="he":
        simfile=sim_pha_helium
        idxs=heidxs
      minx,maxx=0.7,10  #0.03,30
      if type=="he": minx,maxx=minx*4,maxx*4
      miny,maxy=5,100
      bins=500
      binx=np.logspace(np.log10(minx),np.log10(maxx),bins)
      biny=np.logspace(np.log10(miny),np.log10(maxy),bins)
      msdoy,coinc,aseg,bseg,dummy1,ea,eb,ec,ed,ee,etot,wfacts,pha_status,dummy2,dummy3=np.loadtxt(simfile,unpack=True)
      y=ea+eb+ec
      x=(ea*2-eb)/(ea*2+eb)
      lower_ths=[]
      upper_ths=[]
      labels=[]
      for tidx in idxs:
        lower_ths.append(macro["chemin"][tidx])
        upper_ths.append(macro["chemax"][tidx])
        labels.append(macro["chnames"][tidx])
      lower_ths.append(53)
      upper_ths.append(100)
      labels.append("INT")
      for ch in range(len(labels)):
        this_label='%s %s-%s MeV'%(labels[ch],lower_ths[ch],upper_ths[ch])
        if type=="p":
          mask=[(x>=th_q1_lower)&(x<=th_q1_upper)&(y>=th_q2_lower)&(y<=th_q2_upper) &(ec>0.37) &(msdoy>lower_ths[ch])&(msdoy<=upper_ths[ch])]   #ec>0.37 gets rid of ab coincidences
        if type=="he":
          mask=[(x>=th_q1_lower)&(x<=th_q1_upper)&(y>=th_q2_lower_he)&(y<=th_q2_upper_he) &(ec>0.37) &(msdoy/4.>lower_ths[ch])&(msdoy/4.<=upper_ths[ch])]   #ec>0.37 gets rid of ab coincidences
        ab=ea[mask]+eb[mask]
        hist,xedges=np.histogram(ab,bins=binx)
        xpos=10**(  (np.log10(xedges[:-1])+np.log10(xedges[1:]))/2. )
        p=plt.step(xpos,hist/float(np.max(hist)),lw=2.,rasterized=True,label=this_label)
        hists.append(hist/float(np.max(hist)))
        plt.xlim(minx,maxx*2)
        plt.xscale("log")
        plt.xlabel("E$_A$+E$_B$ / MeV")
        plt.ylabel(r"normalized counts")
      plotths=[]
      if type=="p":
        myindex=pidxs
      if type=="he":
        myindex=heidxs
      prev=xpos[-1]
      for tidx in range(len(labels)-1):
        diff=hists[tidx]-hists[tidx+1]
```

```python
        abth=xpos[np.where(diff>0.01)[0][0]]
        macro["chabmin"][myindex[tidx]]=abth
        macro["chabmax"][myindex[tidx]]=prev
        prev=abth
        plotths.append(abth)
      for pth in plotths:
        plt.plot([pth,pth],[0,1],"k-",lw=3,zorder=-1)
        plt.plot([pth,pth],[0,1],"w--",lw=3,zorder=-1)
      plt.legend(fontsize=10)
      pp.savefig()
      plt.close()
  pp.close()
  return macro


""" functions to calculate response factors """
# calcs response factor
def calc_response(macro,verbosity):
  for idx in range(len(macro["chnames"])):
    macro["chgeoms"].append("0")
    macro["chgeomssys"].append("0")
    macro["chgeomsoff"].append("0")
    macro["chgeomsoffsys"].append("0")
  def kugelkalotte(r,theta):
    h=r-r*np.cos((theta/2.)/360.*2.*np.pi)
    A= np.pi*r*2*h
    return A
  sim_particles=2000000000.
  sim_energy_range=60.
  coinces_min_energy=np.array(macro["chemin"])
  coinces_max_energy=np.array(macro["chemax"])
  coinces_lin_mean_energy=(np.array(coinces_min_energy)+np.array(coinces_max_energy))/2.
  coinces_log_mean_energy=np.sqrt(coinces_min_energy*coinces_max_energy)
  norm=sim_particles/sim_energy_range/ (kugelkalotte(12,180)  *np.pi)
  pp= PdfPages("%s_responses.pdf"%macrofile.split(".l3imacro")[0])
  for tchannel in range(len(macro["chnames"])):
    if verbosity==1:  print "   ---> Calculating response for channel %s..."%macro["chnames"][tchannel]
    if macro["chtypes"][tchannel]=="p":
      simfile=sim_pha_proton
      nuc=1.
    if macro["chtypes"][tchannel]=="he":
      simfile=sim_pha_helium
      nuc=4.
    fig=plt.figure(figsize=(8,3))
    ax=plt.subplots_adjust(left=0.16,top=0.95,bottom=0.23,right=0.84,hspace=0.1,wspace=0.27)
    resp_factors_ron=[]
    resp_factors_roff=[]
    gammas=np.arange(-3,1,0.5)
    energy,simcoinc,aseg,bseg,ea,eb,ec=np.loadtxt(simfile,usecols=(0,1,2,3,5,6,7),unpack=True)
    mask_roff=(aseg==0)&(bseg==0)
    if macro["chranges"][tchannel]=="AB":
      kappa=eb
      lam=(ea+eb)*ea
      mu=(ea+eb)/ea
      mask_kappa=(kappa>0.13)
      mask_lam_proton=(lam>10)&(lam<25)
      mask_lam_helium=(lam>120)&(lam<350)
      mask_coinc=(simcoinc==0)+(simcoinc==4)+(simcoinc==8)
      if macro["chtypes"][tchannel]=="p":
        mask_type=mask_lam_proton
      if macro["chtypes"][tchannel]=="he":
        mask_type=mask_lam_helium
      mask_mu=(mu>1.0)&(mu<5.3)
      if macro["chtypes"][tchannel]=="p":
        mask_ab=(ea+eb>=macro["chabmin"][tchannel])&(ea+eb<macro["chabmax"][tchannel])
      if macro["chtypes"][tchannel]=="he":
        mask_ab=(ea+eb>=4*macro["chabmin"][tchannel])&(ea+eb<4*macro["chabmax"][tchannel])
      tenergy_ron=energy[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_ab)&(mask_coinc) ]
      tenergy_roff=energy[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_ab)&(mask_roff)&(mask_coinc) ]
    if macro["chranges"][tchannel]=="ABC":
      kappa=(2*ea-eb)/(2*ea+eb)
      lam=ea+eb+ec
      mu=ea+eb
      mask_kappa=(kappa>-0.35)&(kappa<0.15)
      mask_lam_proton=(lam>7.8)&(lam<27.5)
      mask_lam_helium=(lam>29.5)&(lam<110)
      mask_coinc=(simcoinc!=0)&(simcoinc!=4)&(simcoinc!=8)&(simcoinc!=12)
      if macro["chtypes"][tchannel]=="p":
        mask_type=mask_lam_proton
      if macro["chtypes"][tchannel]=="he":
        mask_type=mask_lam_helium
      mask_mu=(mu>=macro["chabmin"][tchannel])&(mu<macro["chabmax"][tchannel])
      tenergy_ron=energy[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_coinc)]
      tenergy_roff=energy[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_roff)&(mask_coinc)]
    for spectral_gamma in gammas:
      weigths_ron=(tenergy_ron/nuc)**(spectral_gamma)
      intensity_ron=np.sum(weigths_ron)
      weigths_roff=(tenergy_roff/nuc)**(spectral_gamma)
      intensity_roff=np.sum(weigths_roff)
      should_log=norm*coinces_log_mean_energy[tchannel]**spectral_gamma
      should_lin=norm*coinces_lin_mean_energy[tchannel]**spectral_gamma
      tgeom_ron=intensity_ron/should_log  # should_lin is not used here, see lvl3 documentation
      tgeom_roff=intensity_roff/should_log  # should_lin is not used here, see lvl3 documentation
      resp_factors_ron.append(tgeom_ron)
      resp_factors_roff.append(tgeom_roff)
    macro["chgeoms"][tchannel]=np.mean(resp_factors_ron)
    macro["chgeomssys"][tchannel]=np.std(resp_factors_ron)
    macro["chgeomsoff"][tchannel]=np.mean(resp_factors_roff)
    macro["chgeomsoffsys"][tchannel]=np.std(resp_factors_roff)
    plt.plot(gammas,resp_factors_ron,’bo-’,label="%s (ring on)"%macro["chnames"][tchannel])
    plt.ylabel("Response factor \n / (cm$^2$ sr MeV)")
    plt.legend(ncol=3,fontsize=10,loc=2)
    plt.xlabel("power-law index $\gamma$")
    ax2=plt.twinx()
    ax2.plot(gammas,resp_factors_roff,’ro-’,label="%s (ring off)"%macro["chnames"][tchannel])
    ax2.legend(ncol=3,fontsize=10,loc=1)
    plt.ylabel("Response factor \n / (cm$^2$ sr MeV)")
    pp.savefig()
    plt.close()
  pp.close()
  return macro

""" functions to derive complete macro """
```

```
# write header to macro
def write_header(file):
  header_lines=["# chname type range geom geom_syserr geom_roff geom_roff_syserr th1 th2 emin emax","#","# chname: string - channel name","# type: string - particle type ('p': proton, 'he': helium)","# range: string - particle range ('AB': detector A and B, 'ABC': detector A, B and C)","#     Note: range should be 'AB' below ~8 MeV/nucleon and 'ABC' above ~8MeV/nucleon","# geom: float - geom factor in units of 'cm**2 sr MeV' if ring is on","# geom_syserr: float - systematic uncertainty of geom factor in units of 'cm**2 sr MeV' if ring is on","# geom_roff: float - geom factor in units of 'cm**2 sr MeV' if ring is off","# geom_roff_syserr: float - systematic uncertainty of geom factor in units of 'cm**2 sr MeV' if ring is off","# th1: float - lower threshold for E_A+E_B in units of 'MeV', i.e.: E_A+E_B>=th1","# th2: float - upper threshold for E_A+E_B in units of 'MeV', i.e.: E_A+E_B<th2","#     Note: for range='AB' these thresholds correspond to the total energy of the particles","#     Note: for range='ABC' these thresholds correspond to the differential energy losses of these particles","# emin: float - lower energy limit in MeV/nuc","# emax: float - upper energy limit in MeV/nuc","#","#"]
  for line in header_lines:
    file.write(line+"\n")

# derive macro from input file (includes energy ths and response calculation based on simulation file)
def derive_macro(verbosity=1):
  f=open(macrofile,"w")
  write_header(f)
  g=open(channelinput,"r")
  macro={"chnames":[],"chtypes":[],"chranges":[],"chgeoms":[],"chgeomssys":[],"chgeomsoff":[],"chgeomsoffsys":[],"chabmin":[],"chabmax":[],"chemin":[],"chemax":[]}
  for line in g:
    lists=["chnames","chtypes","chemin","chemax"]
    if line[0]!="#":
      line=line.replace("\n","")
      for i in range(30): line=line.replace("  "," ")
      sline=line.split(" ")
      for idx,val in enumerate(sline):
        if idx>1: val=float(val)
        macro[lists[idx]].append(val)
  g.close()


  for idx in range(len(macro["chnames"])):
    if macro["chemin"][idx]<7.8 and macro["chemax"][idx]>7.8:
      print "Channels are not allowed to surpass 7.8 MeV - terminating macro creation! (violation in channel %s)"%macro["chnames"][idx]
      break
    if macro["chemax"][idx]<=7.8:
      macro["chranges"].append("AB")
    else:
      macro["chranges"].append("ABC")

  if verbosity==1:
    print "finished loading input"
    print "creating macro for channels: ",macro["chnames"]

  # calc energy ranges
  if verbosity==1:    print "starting calculation of ranges..."
  macro=energy_ranges(macro,verbosity)
  if verbosity==1:    print "finished calculation of ranges"

```

```
# calc response
if verbosity==1:    print "starting calculation of responses..."
macro=calc_response(macro,verbosity)
if verbosity==1:    print "finished calculation of responses"


for idx in range(len(macro["chnames"])):
  f.write("%s %s %s %4.4f %4.4f %4.4f %4.4f %4.4f %4.4f %s %s\n"%(macro["chnames"][idx],macro["chtypes"][idx],macro["chranges"][idx],macro["chgeoms"][idx],macro["chgeomssys"][idx],macro["chgeomsoff"][idx],macro["chgeomsoffsys"][idx],macro["chabmin"][idx],macro["chabmax"][idx],macro["chemin"][idx],macro["chemax"][idx]))
f.close()
if verbosity==1:    print "finished compilation of macro"
```

9 Appendix II: level3_funcs_varbins

```python
# This scripts includes all function necessary in order to derive EPHIN lvl3 ion intensities
# Patrick Kuehl, June 7 2018 kuehl@physik.uni-kiel.de
#from pylab import *
import numpy as np
import numpy.ma as ma
import time as time
import datetime as dt
import os
np.seterr(divide='ignore', invalid='ignore')
import subprocess
import warnings
warnings.filterwarnings("ignore")

# sections and defined functions
"""
functions for the PHAWS data processing
    load_level2_pha(year,doy,unpack=False)
    load_level1_sci(year,doy)
    check_coinc(co,a,b,c,d,e)
    add_wfact_to_pha ( year , doy )
    phaws_from_year_doy ( year , doy , save = True )

level3 AB-coincidence functions
    int_in_lvl3_ch_from_ea_eb(ea,eb, mywfact,myringoff,myaseg,mybseg, i_p_ron,s_p_ron,i_h_ron,s_h_ron, i_p_roff,s_p_roff,i_h_roff,s_h_roff)

level3 ABC-coincidence functions
    int_in_lvl3_ch_from_ea_eb_ec(ea,eb,ec, mywfact,myringoff,myaseg,mybseg, i_p_ron,s_p_ron,i_h_ron,s_h_ron, i_p_roff,s_p_roff,i_h_roff,s_h_roff)

functions for the actual level3 data processing
    calc_lvl3_intensities_timeresolution(year,doy,tres,create_phaws=True, delete_phaws=True)
    merge_level3_daily_to_annual(year,timeres,header_lines=3)
"""

# paths (shall be defined in actual processing code)
"""
macrofile="/data/etph/kuehl/ephin_lvl3_varbins/nominal_ephinbins.l3imacro"
lvl2_pha_path="/data/missions/soho/costep/level2/pha/"
lvl1_sci_path="/data/missions/soho/costep/level1/sci/"
phaws_path="/data/missions/soho/python/l3i/tmp/"
lvl3_out_path="/data/missions/soho/costep/level3/l3i/"
"""

""" functions for the PHAWS data processing """
# load level2 pha file for given year and doy
def load_level2_pha(year,doy,unpack=False):
  pha_path=lvl2_pha_path  # /data/missions/soho/costep/level2/pha/
  thisyear=year
  thisdoy=doy
  if thisyear <2000:
    thisyear2d=thisyear-1900
    prefix='eph'
  else:
    thisyear2d=thisyear-2000
    prefix='epi'
  data=np.loadtxt("%s%s/%s%02d%03d.pl2" %(pha_path,thisyear,prefix, thisyear2d,thisdoy))
  if (year>=2017 and doy>276) or year>2017: fmd=True
  else: fmd=False
  if True:  # remove wrong coincidences
    cc=[]
    for q in range(len(data[:,1])):
      if check_coinc(data[q,1],data[q,5],data[q,6],data[q,7],data[q,8],data[q,9],fmd=fmd):
        cc.append(q)
    data=data[cc]
  if unpack==False:
    return data
  else:
    time=data[:,0] #  ms since year 0
    coinc=data[:,1]
    aseg=data[:,2]
    bseg=data[:,3]
    ea=data[:,5]
    eb=data[:,6]
    ec=data[:,7]
    ed=data[:,8]
    ee=data[:,9]
    etot=data[:,10]
    return time,coinc,aseg,bseg,ea,eb,ec,ed,ee,etot

# load level1 sci file for given year and doy
def load_level1_sci(year,doy):
  if year <2000:
    thisyear2d=year-1900
    prefix='eph'
  else:
    thisyear2d=year-2000
    prefix='epi'
  year,doy,msdoy,e1,e2,e3,e4,p1_1,p1_2,p1_3,p2_1,p2_2,p2_3,p3_1,p3_2,p3_3,p4_1,p4_2,p4_3, h1_1,h1_2,h1_3,h1_4,h2_1,h2_2,h2_3,h2_4,h3_1,h3_2,h3_3,h3_4,h4_1,h4_2,h4_3,h4_4, total_int_counts,status=np.loadtxt("%s%s/%s%02d%03i.sci"%(lvl1_sci_path,year,prefix,thisyear2d,doy),usecols=(0,1,2,36,37,38,39, 22,23,24, 25,26,27, 41,42,43, 44,45,46, 28,29,30,31, 32,33,34,35, 47,48,49,50, 51,52,53,54, 40,-1),unpack=True)
  p1=p1_1+p1_2+p1_3
  p2=p2_1+p2_2+p2_3
  p3=p3_1+p3_2+p3_3
  p4=p4_1+p4_2+p4_3
  h1=h1_1+h1_2+h1_3+h1_4
  h2=h2_1+h2_2+h2_3+h2_4
  h3=h3_1+h3_2+h3_3+h3_4
  h4=h4_1+h4_2+h4_3+h4_4
  lvl1_counts=[year,doy,msdoy, e1,e2,e3,e4, p1,p2,p3,p4, h1,h2,h3,h4, total_int_counts,status]
  return lvl1_counts

# checks for wrong coincidences
def check_coinc(co,a,b,c,d,e, fmd=False):
  t=0
  # def ths:
  a0,a1,a2,a3,a4=0.03,0.27,0.97,2.1,5.3
  b0,c0,d0,e0=0.06,0.37,0.58,0.58
  # electrons
  if co<4 and a>a0 and a<a1 and b>b0:
    if co==0 and c<c0 and d<d0 and e<e0: t=1
    if co==1 and c>c0 and d<d0 and e<e0: t=1
    if co==2 and c>c0 and d>d0 and e<e0: t=1
    if co==3 and c>c0 and d>d0 and e>e0: t=1
  # protons
  if 3<co<8 and a>a1 and b>b0:
    if fmd==False:
      if co==4 and a<a4 and c<c0 and d<d0 and e<e0: t=1
      if co==5 and a<a3 and c>c0 and d<d0 and e<e0: t=1
      if co==6 and a<a2 and c>c0 and d>d0 and e<e0: t=1
      if co==7 and a<a2 and c>c0 and d>d0 and e>e0: t=1
    else:   # if failure mode d: threshold in a changes
      if co==4 and a<a4 and c<c0 and d<d0 and e<e0: t=1
      elif a<a3: t=1

  # helium
  if 7<co and b>b0:
    if fmd==False:
      if co==8 and a>a4 and c<c0 and d<d0 and e<e0: t=1
      if co==9 and a>a3 and c>c0 and d<d0 and e<e0: t=1
      if co==10 and a>a2 and c>c0 and d>d0 and e<e0: t=1
      if co==11 and a>a2 and c>c0 and d>d0 and e>e0: t=1
    else:   # if failure mode d: threshold in a changes
      if co==8 and a>a4 and c<c0 and d<d0 and e<e0: t=1
      elif a>a2: t=1

  # returner
  if t==0: return False
  if t==1:  return True

# returns a pha like data product that includes wfacts (ratio counts/num_of_pha) and status bit
def add_wfact_to_pha(year,doy):
  scidata= load_level1_sci(year,doy)
  sci_msdoy=scidata[2]
  sci_status=scidata[-1]
  phadata= load_level2_pha(year,doy,unpack=False)
  phadata=phadata[phadata[:,1]!=12] # remove penetrating
  pha_time=phadata[:,0] #  ms since year 0
  coinc=phadata[:,1]
  msoffset=(dt.datetime(year,1,1)+dt.timedelta(doy-1))-dt.datetime(1,1,1)+dt.timedelta(366)
  pha_msdoy= pha_time-msoffset.total_seconds()*1e3
  wfacts=np.zeros(len(pha_msdoy))
  pha_status=np.ones(len(pha_msdoy))*-1
  for thismsec in sci_msdoy:
    # add status to pha
    pha_status[(pha_msdoy==thismsec)]=sci_status[sci_msdoy==thismsec][0]
    # get coinc counts in this minute
    coinccounters=[]
    for q in range(13): coinccounters.append(scidata[3+q][sci_msdoy==thismsec][0])
    # calc wfact for each coinc in this minute
    thiswfacts=[]
    for thiscoinc in range(13):
      numphas= len(pha_msdoy[(pha_msdoy==thismsec)&(coinc==thiscoinc)])
      ### care for failure modes!
      if (year>=1997 and doy>50) or year>1997:  #failure mode e as well as failure mode d (fmE: pha: 0,1,3  ,rl2: 0,1,2   fmDE: pha 0,3, rl2: 0,2)
            if thiscoinc in [3,7,11]:
              thiswfacts.append(coinccounters[thiscoinc-1]/numphas)
            else:
              thiswfacts.append(coinccounters[thiscoinc]/numphas)
      else:
        thiswfacts.append(coinccounters[thiscoinc]/numphas)
    # dump wfacts in wfacts-array
    for thiscoinc in range(13):
      wfacts[(pha_msdoy==thismsec)&(coinc==thiscoinc)]=thiswfacts[thiscoinc]
  aseg=phadata[:,2]
  bseg=phadata[:,3]
  ea=phadata[:,5]
  eb=phadata[:,6]
  ec=phadata[:,7]
  ed=phadata[:,8]
  ee=phadata[:,9]
  etot=phadata[:,10]
  return pha_msdoy,coinc,aseg,bseg,ea,eb,ec,ed,ee,etot,wfacts,pha_status

#  makes a phaws from year and doy
def phaws_from_year_doy(year,doy,save=True):
  os.system("mkdir %s%i -p" %(phaws_path,year))
  msdoy,coinc,aseg,bseg,ea,eb,ec,ed,ee,etot,wfacts,pha_status=add_wfact_to_pha(year,doy)
  list_of_arrays=[msdoy.astype(int),coinc.astype(int),aseg.astype(int),bseg.astype(int),ea,eb,ec,ed,ee,etot,wfacts,pha_status.astype(int)]
  shape = list(list_of_arrays[0].shape)
  shape[:0] = [len(list_of_arrays)]
  arr = np.concatenate(list_of_arrays).reshape(shape).T
  if save==True:
    np.savetxt("%s%i/%i-%03d.phaws"%(phaws_path,year,year,doy),arr,fmt="%i %i %i %i %3.2f %3.2f %3.2f %3.2f %3.2f %3.2f %4.4f %i")
  else:
    return arr

""" level3 AB-coincidence functions """

# calc intensity in ab coinc masks
def int_in_lvl3_ch_from_ea_eb(ea,eb, mywfact,myringoff,myaseg,mybseg, tgeomon,tgeomonsys,tgeomoff,tgeomoffsys,tabmin,tabmax,chtype):
  if 1 in myringoff:
    mask_center=(myaseg==0)&(mybseg==0)
    ea=ea[mask_center]
    eb=eb[mask_center]
    mywfact=mywfact[mask_center]
  kappa=eb
  lam=(ea+eb)*ea
  mu=(ea+eb)/ea
  mask_kappa=(kappa>0.13)
  mask_lam_proton=(lam>10)&(lam<25)
  mask_lam_helium=(lam>120)&(lam<350)
  mask_mu=(mu>1.0)&(mu<5.3)
  if chtype=="p":
    mask_type=mask_lam_proton
    mask_ab=(ea+eb>=tabmin)&(ea+eb<tabmax)
  if chtype=="he":
    mask_type=mask_lam_helium
    mask_ab=(ea+eb>=4*tabmin)&(ea+eb<4*tabmax)
  acctime=59.953
  inte=sum( mywfact[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_ab) ] )
  counts=len( mywfact[ (mask_kappa)&(mask_type)&(mask_mu)&(mask_ab) ] )
  if 1 in myringoff:
    geom=tgeomoff
    geomsys=tgeomoffsys
  else:
    geom=tgeomon
    geomsys=tgeomonsys
  chinte=inte/geom
  chsys=chinte*geomsys/geom
  chstat=chinte/np.sqrt(counts)
  return chinte,chsys,chstat

```

```python

""" level3 ABC-coincidence functions """
# calc intensity in abc coinc masks
def int_in_lvl3_ch_from_ea_eb_ec(ea,eb,ec, mywfact,myringoff,myaseg,mybseg,tgeomon,tgeomonsys,tgeomoff,tgeomoffsys,tabmin,tabmax,chtype):
  if 1 in myringoff:
    mask_center=(myaseg==0)&(mybseg==0)
    ea=ea[mask_center]
    eb=eb[mask_center]
    ec=ec[mask_center]
    mywfact=mywfact[mask_center]
  kappa=(2*ea-eb)/(2*ea+eb)
  lam=ea+eb+ec
  mu=ea+eb
  mask_kappa=(kappa>-0.35)&(kappa<0.15)
  mask_lam_proton=(lam>7.8)&(lam<27.5)
  mask_lam_helium=(lam>29.5)&(lam<110)
  mask_mu=(mu>=tabmin)&(mu<tabmax)
  if chtype=="p":
    mask_type=mask_lam_proton
  if chtype=="he":
    mask_type=mask_lam_helium
  acctime=59.953
  inte=sum( mywfact[ (mask_kappa)&(mask_type)&(mask_mu) ] )
  counts=len( mywfact[ (mask_kappa)&(mask_type)&(mask_mu) ] )
  if 1 in myringoff:
    geom=tgeomoff
    geomsys=tgeomoffsys
  else:
    geom=tgeomon
    geomsys=tgeomonsys
  chinte=inte/geom
  chsys=chinte*geomsys/geom
  chstat=chinte/np.sqrt(counts)
  return chinte,chsys,chstat

""" functions for the actual level3 data processing """
# loads macro file
def load_macro(macrofile):
  f=open(macrofile,"r")
  macro={"chnames":[],"chtypes":[],"chranges":[],"chgeoms":[],"chgeomssys":[],"chgeomsoff":[],"chgeomsoffsys":[],"chabmin":[],"chabmax":[],"chemin":[],"chemax":[]}
  lists=["chnames","chtypes","chranges","chgeoms","chgeomssys","chgeomsoff","chgeomsoffsys","chabmin","chabmax","chemin","chemax"]
  for line in f:
    if line[0]!="#":
      line=line.replace("\n","")
      for i in range(30): line=line.replace("  "," ")
      sline=line.split(" ")
      for idx,val in enumerate(sline):
        if idx>2: val=float(val)
        macro[lists[idx]].append(val)
  f.close()
  return  macro


# calcs intensity all coinces for given time resolution
def calc_lvl3_intensities_timeresolution_varbins(year,doy,tres,create_phaws=True,delete_phaws=True):
  macro=load_macro(macrofile)
  os.system("mkdir %s%imin/ -p"%(lvl3_out_path,tres))
  os.system("mkdir %s%imin/%i -p"%(lvl3_out_path,tres,year))
  if True:
    if True:   ######################################## try:
      if create_phaws==True:
        phaws_from_year_doy(year,doy,save=True)
      msdoy,coinc,aseg,bseg,ea,eb,ec,ed,ee,etot,wfacts,pha_status=np.loadtxt("%s%i/%i-%03d.phaws"%(phaws_path,year,year,doy),unpack=True)
      ringoff=np.zeros(len(pha_status))
      for q in range(len(pha_status)):
        binaries='{0:08b}'.format(int(pha_status[q]))
        if int(binaries[-2]): ringoff[q]=1
      f=open("%s%imin/%i/%i_%03d.l3i"%(lvl3_out_path,tres,year,year,doy) ,"w")
      f.write("# year month day doy hour minute status accum.time ")
      for chname in macro["chnames"]:
        f.write("int_%s sys_%s stat_%s "%(chname,chname, chname))
      for idx,chname in enumerate(macro["chnames"]):
        f.write("\n# Channel description: %s - particletype: %s - energy: %2.2f-%2.2f MeV/nuc"%(chname,macro["chtypes"][idx],macro["chemin"][idx],macro["chemax"][idx]))
      f.write("\n# all values except for time and status are intensities in units of (cm^2 s sr mev/nuc)^-1\n# zero counts in given channel are indicated by a '-999' in the intensity, stat and sys uncertainty\n")
      tinter=[0]
      while tinter[-1]<1440:
        tinter.append(tinter[-1]+tres)
      #for mytime in np.unique(msdoy):
      for tidx in range(len(tinter)-1):
        smin,emin=tinter[tidx],tinter[tidx+1]
        tmask=(msdoy>=smin*60000)&(msdoy<emin*60000)
        # write time and status
        if not any(tmask):
          continue
        hour,minutes=divmod(smin,60)
        mydate=dt.datetime(int(year),1,1)+dt.timedelta(int(doy)-1)
        month,day=mydate.month,mydate.day
        mystatus=np.max(pha_status[tmask])  #pha_status[tmask][0]
        f.write("%i %i %i %i %i %i %i " %(year,month,day,doy,hour,minutes, mystatus ) )
        f.write("  ")
        tnorm=len(np.unique(msdoy[tmask]))*59.953  # timeinterval in seconds
        f.write("%4.4f   "%tnorm)


        for chidx in range(len(macro["chnames"])):
          chrange=macro["chranges"][chidx]
          chtype=macro["chtypes"][chidx]
          tgeomon,tgeomonsys,tgeomoff,tgeomoffsys=macro["chgeoms"][chidx],macro["chgeomssys"][chidx],macro["chgeomsoff"][chidx],macro["chgeomsoffsys"][chidx]
          tabmin,tabmax=macro["chabmin"][chidx],macro["chabmax"][chidx]

          if chrange=="AB":
            lvl3_coinc_mask=((coinc==0)+(coinc==4)+(coinc==8))
          if chrange=="ABC":
            lvl3_coinc_mask=((coinc!=0)&(coinc!=4)&(coinc!=8)&(coinc!=12))
          myea=ea[(tmask)&(lvl3_coinc_mask)]
          myeb=eb[(tmask)&(lvl3_coinc_mask)]
          myec=ec[(tmask)&(lvl3_coinc_mask)]
          myaseg=aseg[(tmask)&(lvl3_coinc_mask)]
          mybseg=bseg[(tmask)&(lvl3_coinc_mask)]
          mywfact=wfacts[(tmask)&(lvl3_coinc_mask)]
          myringoff=ringoff[(tmask)&(lvl3_coinc_mask)]

          if chrange=="AB":
            chinte,chsys,chstat=int_in_lvl3_ch_from_ea_eb(myea,myeb,mywfact,myringoff,myaseg,mybseg, tgeomon,tgeomonsys,tgeomoff,tgeomoffsys,tabmin,tabmax,chtype)
          if chrange=="ABC":
            chinte,chsys,chstat=int_in_lvl3_ch_from_ea_eb_ec(myea,myeb,myec,mywfact,myringoff,myaseg,mybseg, tgeomon,tgeomonsys,tgeomoff,tgeomoffsys,tabmin,tabmax,chtype)

          chinte,chsys,chstat=chinte/tnorm,chsys/tnorm,chstat/tnorm

          # if int=0 => set sys,stat uncertainties = -999
          set_zeros_invalid=True
          if set_zeros_invalid:
            keyword=-999
            if chinte==0: chsys,chstat=keyword,keyword
          # write lvl3 intensities
          f.write("%2.4e %2.4e %2.4e "%(chinte,chsys,chstat))
        f.write("\n")
      f.close()
    #except:
    #  d=1
    if delete_phaws:  os.system("rm -f %s%i/%i-%03d.phaws"%(phaws_path,year,year,doy))

# merge level3 daily files to annual
def merge_level3_daily_to_annual(year,timeres,header_lines=3):
  init=1
  g=open("%s%imin/%i.l3i"%(lvl3_out_path,timeres,year),"w")
  for doy in range(1,370):
    try:
      f=open("%s%imin/%i/%i_%03d.l3i"%(lvl3_out_path,timeres,year,year,doy),"r")
      if init==0: #skip header
        for i in range(header_lines): f.readline()
      for line in f:
        g.write(line)
      init=0
      f.close()
    except:
      continue  #print "no file", year, doy
  g.close()
```