SANEPIC for HERSCHEL

Alexandre Beelen

(alexandre.beelen@ias.u-psud.fr)

ESA Map Making Workshop 2013

Outline

1. Maximum Likelihood Mapmaking
   - Definition
   - Hands on !
2. The SANEPIC approach
   - Inverting the Noise Covariance Matrix
   - Caveats and Other Little Stories
3. Application to HERSCHEL data
   - SPIRE
   - PACS
4. A Peak Into SANEPIC programs

Maximum Likelihood Mapmaking in a nutshell

A general data model

- Let the sky be a vector s ,
- Let observe it in a certain way $A\,s$,
- Add some noise n and ...
- The data d is $d = A\,s + n$.

A Maximum Likelihood approach

- The noise n is a Gaussian, i.e.
  - the probability distribution wrt. the values is Gaussian...
  - but can have any power spectrum and/or correlation...
  - $N = \langle n\,n^t \rangle$
- The log-likelihood of the data is
  - $\log \mathcal{L}(d|s) = -\frac{1}{2}\,(d - A\,s)\,N^{-1}\,(d - A\,s)$
- Maximizing the log-likelihood wrt s leads to
  - $\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$

Maximum Likelihood Mapmaking in a nutshell

A general data model

- Let the sky be a vector s ,
- Let observe it in a certain way $A\,s$,
- Add some noise n and ...
- The data d is $d = A\,s + n$.

A Maximum Likelihood approach

- The noise n is a Gaussian, i.e.
  - the probability distribution wrt. the values is Gaussian...
  - but can have any power spectrum and/or correlation...
  - $N = \langle n\,n^t \rangle$
- The log-likelihood of the data is
  - $\log \mathcal{L}(d|s) = -\frac{1}{2}\,(d - A\,s)\,N^{-1}\,(d - A\,s)$
- Maximizing the log-likelihood wrt s leads to
  - $\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$

Example 2 sky pixels, 4 data points

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{array}{|c|}\hline d_1 \\ \hline d_2 \\ \hline d_3 \\ \hline d_4 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array} \begin{array}{|c|}\hline s_1 \\ \hline s_2 \\ \hline \end{array} + \begin{array}{|c|}\hline n_1 \\ \hline n_2 \\ \hline n_3 \\ \hline n_4 \\ \hline \end{array}$$

$$d = \begin{pmatrix} 1.2 \\ 2.1 \\ 1.8 \\ 1.0 \end{pmatrix} \; A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \; N = \langle n\, n^t \rangle = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{pmatrix}$$

so...

$$\hat{s} = \left(A^t N^{-1} A \right)^{-1} A^t N^{-1} d$$

Example 2 sky pixels, 4 data points

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix}$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} 1.2 \\ 2.1 \\ 1.8 \\ 1.0 \end{pmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 12 \\ 21 \\ 9.0 \\ 5 \end{pmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix}$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$$

$$\hat{s} = \left(\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} 17 \\ 30 \end{pmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix}$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$$

$$\hat{s} = \begin{pmatrix} 15 & 0 \\ 0 & 15 \end{pmatrix}^{-1} \begin{pmatrix} 17 \\ 30 \end{pmatrix}$$

Example 2 sky pixels, 4 data points

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix}$$

$$\hat{s} = \left(A^t N^{-1} A\right)^{-1} A^t N^{-1} d$$

$$\hat{s} = \begin{pmatrix} 1.13 \\ 2.0 \end{pmatrix}$$

$$d = \begin{pmatrix} 1.2 \\ 2.1 \\ 1.8 \\ 1.0 \end{pmatrix} \; A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \; N = \langle n\, n^t \rangle = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{pmatrix}$$

In practice...

Large numbers

- $\mathrm{n_{samples}} = \sum_{\mathrm{obs.}} \mathrm{n_{det}} \times \mathrm{f_{samp}} \times \mathrm{T}$
- $\mathrm{n_{sky}} \gg 1000$

Pointing Matrix – A

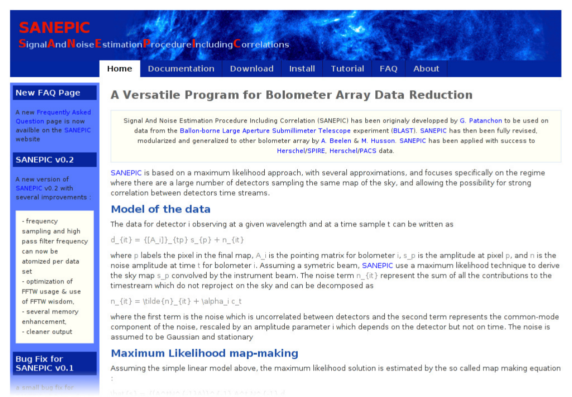
- large $\mathrm{n_{sky}} \times \mathrm{n_{samples}}$
- sparse (only 0 or 1)
- $\left(A^t N^{-1} A\right)$ has no obvious symmetry

Noise Covariance Matrix – N

- very large $\mathrm{n_{samples}} \times \mathrm{n_{samples}}$
- could be dense ...
- ... can not easily be inverted ...

Signal And Noise Estimation Procedure Including Correlation

- First developed for BLAST experiment data. (Patanchon, G. et al 2008, ApJ, 681, 708)
- Extended for the general case at IAS, Orsay
  - LABOCA, SPIRE, PACS, NIKA
  - `http://www.ias.u-psud.fr/sanepic`



The SANEPIC approach

The magic of Fourier Transform

- IF the data segment is circulant,
- and there is no gap in the data
- then $N_{tt'} = C(|t - t'|)$ and $N = F^{\dagger}\Lambda F$ where
  - F is the Fourier transform ($\dagger$ transpose conjugate)
  - Λ is a diagonal matrix
  - $\Lambda_{\omega,\omega} = P(\omega)$ the power spectrum of the data segment
- and... $N^{-1} = F^{\dagger}\Lambda^{-1}F$, is also a circulant matrix

Is the data circulant ?

- In general no, but...
- can be made circulant by
  - polynomial baseline, linear fit on the edges and/or apodization
  - high pass pre-filtering (f_{cut})

The SANEPIC approach

The magic of Fourier Transform

- IF the data segment is circulant,
- and there is no gap in the data
- then $N_{tt'} = C\left(|t - t'|\right)$ and $N = F^{\dagger}\Lambda F$ where
  - F is the Fourier transform ($\dagger$ transpose conjugate)
  - Λ is a diagonal matrix
  - $\Lambda_{\omega,\omega} = P(\omega)$ the power spectrum of the data segment
- and... $N^{-1} = F^{\dagger}\Lambda^{-1}F$, is also a circulant matrix

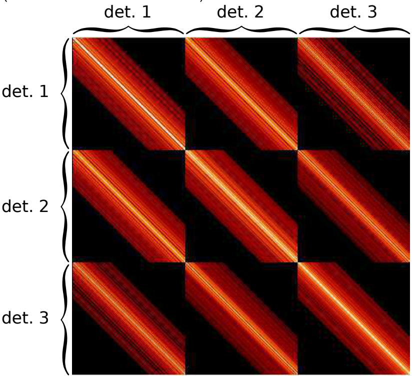
Is the data circulant ?

- In general no, but...
- can be made circulant by
  - polynomial baseline, linear fit on the edges and/or apodization
  - high pass pre-filtering ($\mathrm{f}_{\mathrm{cut}}$)

Inverse Noise Covariance Matrix

- 10 min. of BLAST data (60,000 samples)
- each submatrix is described by $P_{ij}(\omega)^{-1}$
- $\left[N^{-1}\right]_{ijtt'} = 0$ for $|t - t'| > min(\lambda_{\rm cut}, {\rm n_s}/2)$
- auto AND cross spectra (cf. MADCAP)

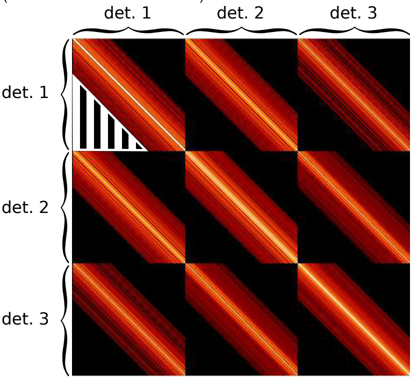
(from Patanchon et al. 2008)



Inverse Noise Covariance Matrix

- 10 min. of BLAST data (60,000 samples)
- each submatrix is described by $P_{ij}(\omega)^{-1}$
- $\left[N^{-1}\right]_{ijtt'} = 0$ for $|t - t'| > min(\lambda_{\rm cut}, \mathrm{n_s}/2)$
- auto AND cross spectra (cf. MADCAP)

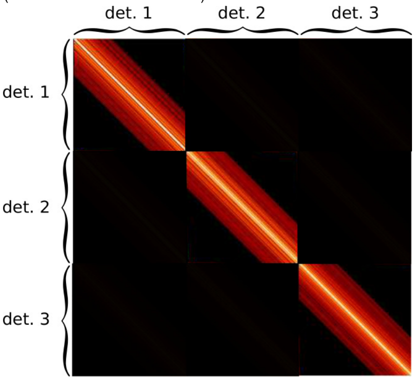
(from Patanchon et al. 2008)



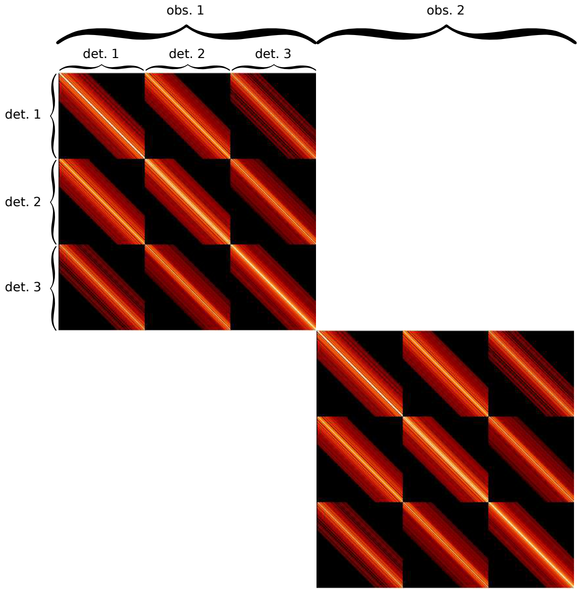
Inverse Noise Covariance Matrix

- 10 min. of BLAST data (60,000 samples)
- each submatrix is described by $P_{ij}(\omega)^{-1}$
- $\left[N^{-1}\right]_{ijtt'} = 0$ for $|t - t'| > min(\lambda_{\rm cut}, {\rm n_s}/2)$
- auto AND cross spectra (cf. MADCAP)

(from Patanchon et al. 2008)



Inverse Noise Covariance Matrix



Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

In practice

- P_{ij} is assumed to be perfectly known for each data set ...
- d_i should be pure noise

- Iterative approach on *blank* field
- Bootstrap $P(w)$ for other fields ...
- $n_i = \tilde{n}_i + \sum_k \alpha_{i,k} c_k$

Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

In practice

- P_{ij} is assumed to be perfectly known for each data set ...
- d_i should be pure noise
- Iterative approach on *blank* field
  1. Estimate $P_0(w)$ using *blank* field
  2. Compute $\hat{s}$ using $P_0(w)$
  3. Estimate $P(w)$ from $d - A\hat{s}$
  4. Re-compute $\hat{s}$ using $P(w)$ and iterate 3-4 until convergence
- Bootstrap $P(w)$ for other fields ...
- $n_i = \tilde{n}_i + \sum_k \alpha_{i,k} c_k$

Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

In practice

- P_{ij} is assumed to be perfectly known for each data set ...
- d_i should be pure noise

- Iterative approach on *blank* field
- Bootstrap $P(w)$ for other fields ...
  ... and iterate again
- $n_i = \tilde{n}_i + \sum_k \alpha_{i,k} c_k$

Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

In practice

- P_{ij} is assumed to be perfectly known for each data set ...
- d_i should be pure noise

- Iterative approach on *blank* field
- Bootstrap $P(w)$ for other fields ...
- $n_i = \tilde{n}_i + \sum_k \alpha_{i,k} c_k$
  - blind component separation
  - $P_{ii} = \langle \tilde{n}_i^* \tilde{n}_i \rangle + \sum_k \alpha_{i,k}{}^2 \langle \tilde{c}_k^* \tilde{c}_k \rangle$ $\qquad P_{ij} = \sum_k \alpha_{i,k} \alpha_{j,k} \langle \tilde{c}_k^* \tilde{c}_k \rangle$

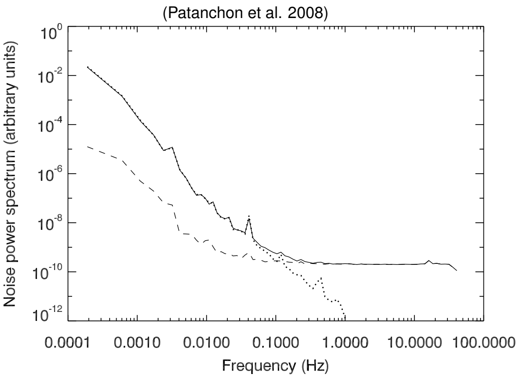
Noise Power Spectra

Definition – Binned Power spectrum

$$P_{ij}(q) = \frac{1}{n_q} \sum_{w_{min}(q)}^{w_{max}(q)} \tilde{d}^*_{iw} \tilde{d}_{jw}$$

BLAST data

- 3h
- ELAIS-N1
- 250μm



Gaps in the data

Gaps ?

- calibration block, turnaround, missing data...
- processing flags (glitches ...)

Power Spectra

- data stream needs to be continuous
- linear interpolation

Map Making

- all data needs to be mapped (pointing matrix)
- two strategies:
  - one extra pixel
  - extra map for flagged data

Hidden Hypothesis

Pixels describe a constant sky — problem for strong sources

- variables objects $\rightarrow$ artifacts
- unflagged glitches $\rightarrow$ artifacts
- strong gradient within a pixel $\rightarrow$ artifacts
- pointing errors within a pixel $\rightarrow$ artifacts

Data segments are circulant

- depends on the scanning strategy ...
- ... all very large scales might be lost
- ... usually not measurable – $P(k)$
- ... can be controlled by f_{cut}

Redundancy is the key

- different scanning angle

Hidden Hypothesis

Pixels describe a constant sky   problem for strong sources

- variables objects $\rightarrow$ artifacts
- unflagged glitches $\rightarrow$ artifacts
- strong gradient within a pixel $\rightarrow$ artifacts
- pointing errors within a pixel $\rightarrow$ artifacts

Data segments are circulant

- depends on the scanning strategy ...
- ... all very large scales might be lost
- ... usually not measurable – $P(k)$
- ... can be controlled by $\mathrm{f_{cut}}$

Redundancy is the key

- different scanning angle

Hidden Hypothesis

Pixels describe a constant sky — problem for strong sources

- variables objects $\rightarrow$ artifacts
- unflagged glitches $\rightarrow$ artifacts
- strong gradient within a pixel $\rightarrow$ artifacts
- pointing errors within a pixel $\rightarrow$ artifacts

Data segments are circulant

- depends on the scanning strategy ...
- ... all very large scales might be lost
- ... usually not measurable – $P(k)$
- ... can be controlled by $\mathrm{f_{cut}}$

Redundancy is the key

- different scanning angle

SANEPIC Requirements & Other Features

Parallel by essence $P/S \approx 0.94$ – MPI

- by observing block <u>and/or</u> receivers
- IO intensive so use local disk

SANEPIC Requirements & Other Features

Parallel by essence $P/S \approx 0.94$ – MPI

- by observing block <u>and/or</u> receivers
- IO intensive so use local disk

Memory friendly

- keep only the filled pixels in memory
- $\times 9$ per processes

SANEPIC Requirements & Other Features

Parallel by essence $P/S \approx 0.94$ – MPI

- by observing block and/or receivers
- IO intensive so use local disk

Memory friendly

- keep only the filled pixels in memory
- $\times 9$ per processes

Projection WCSLIB

- any known projection ...
- or any valid fits header
- can convert to/from galactic from/to equatorial

SANEPIC Requirements & Other Features

SANEPIC input

- fits_filelist : list of file to process
- noise_dir : corresponding noise powerspectra
- pixsize... or mask_file : to control the projection

SANEPIC Requirements & Other Features

SANEPIC input

- fits_filelist : list of file to process
- noise_dir : corresponding noise powerspectra
- pixsize... or mask_file : to control the projection

SANEPIC pre-processing

- f_{hp} : for high-pass filtering of the timeline
- f_{cut} : for cutting the noise powerspectra
- poly_order : polynomial baseline subtraction
- linear_baseline : simple linear baseline on edge

Application to HERSCHEL/SPIRE

Going out of HIPE

- best flags available
- all data must be calibrated in flux
- positions must be corrected of any offsets
- no `temperatureDriftCorrection` task in the pipeline
- use `export_SpireToSanepic.py` (P. Panuzzo)
  - export all three bands into one fits file

Have a look a the data

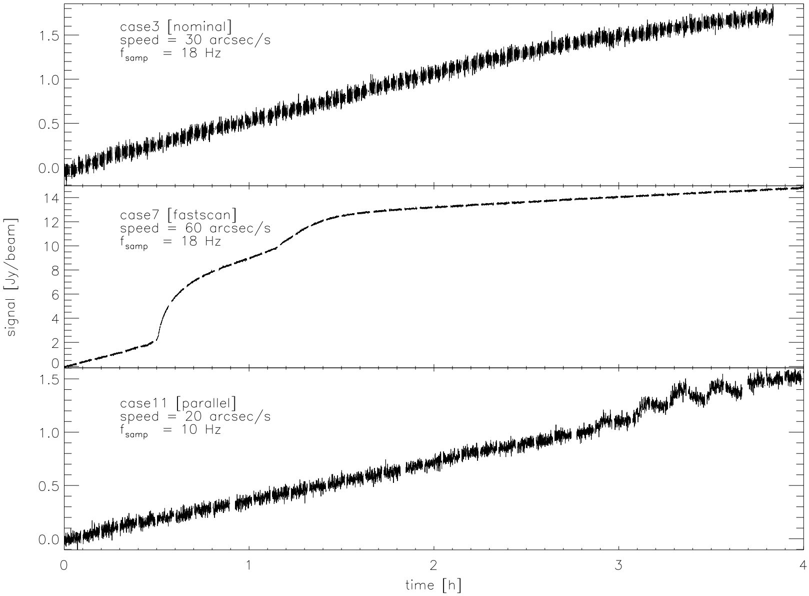
Application to HERSCHEL/SPIRE

Going out of HIPE

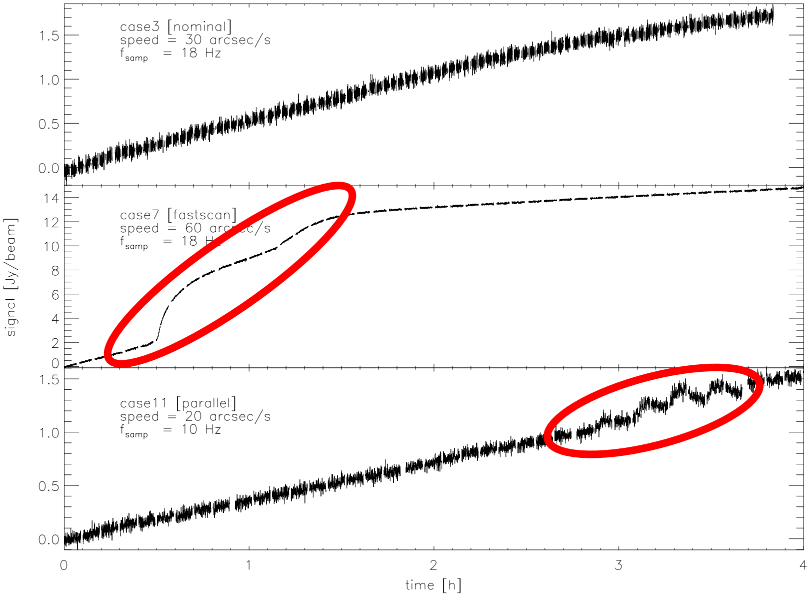
- best flags available
- all data must be calibrated in flux
- positions must be corrected of any offsets
- no `temperatureDriftCorrection` task in the pipeline
- use `export_SpireToSanepic.py` (P. Panuzzo)
  - export all three bands into one fits file

Have a look a the data

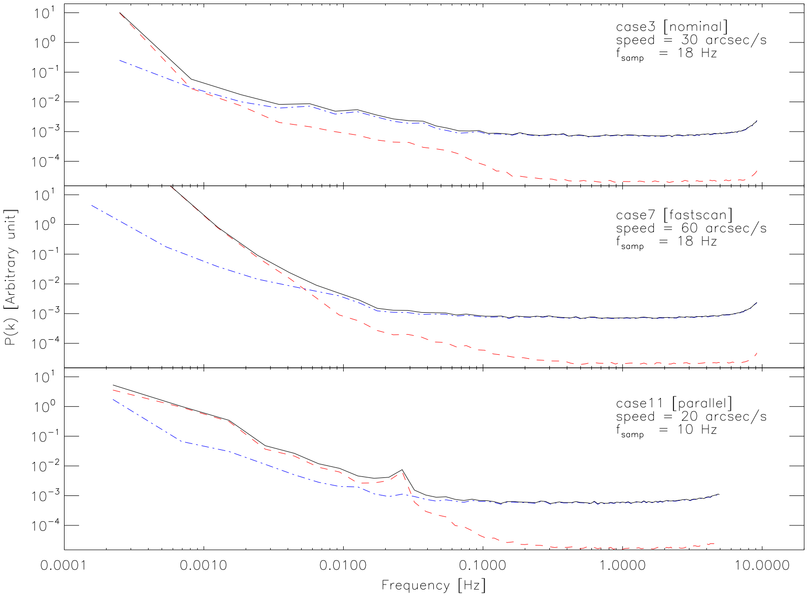
SPIRE data timeline



SPIRE data timeline



SPIRE power spectra



SPIRE rule of thumb

Have a look at your data

- Non-stationarity within an observation
  - on what scale ? where does it affect the data ?
  - filters-it (f_{cut}),
  - or cut the data in chunks,
  - or down-weight the entire dataset
- Unflagged glitches
  - second level deglitching
  - recompute the noise power spectra (?)

Mapping Transfer function $P_{out}(k)/P_{in}(k)$

- Monte-Carlo simulation
- Mandatory as filtering depends on the observing strategy
- Compares very well with PLANCK maps on large scale

SPIRE rule of thumb

Have a look at your data

- Non-stationarity within an observation
  - on what scale ? where does it affect the data ?
  - filters-it (f_{cut}),
  - or cut the data in chunks,
  - or down-weight the entire dataset
- Unflagged glitches
  - second level deglitching
  - recompute the noise power spectra (?)

Mapping Transfer function $P_{out}(k)/P_{in}(k)$

- Monte-Carlo simulation
- Mandatory as filtering depends on the observing strategy
- Compares very well with PLANCK maps on large scale

Application to HERSCHEL/PACS

Going out of HIPE

- best flags available
- all data must be calibrated in flux
- positions must be corrected of any offsets / errors
- no `highpassFilter` / `photGlobalDriftCorrection` / ... task in the pipeline
- use `export_PacsToSanepic.py` (M. Sauvage)
  - make sure that you ran a second level deglitcher

Have a look a the data

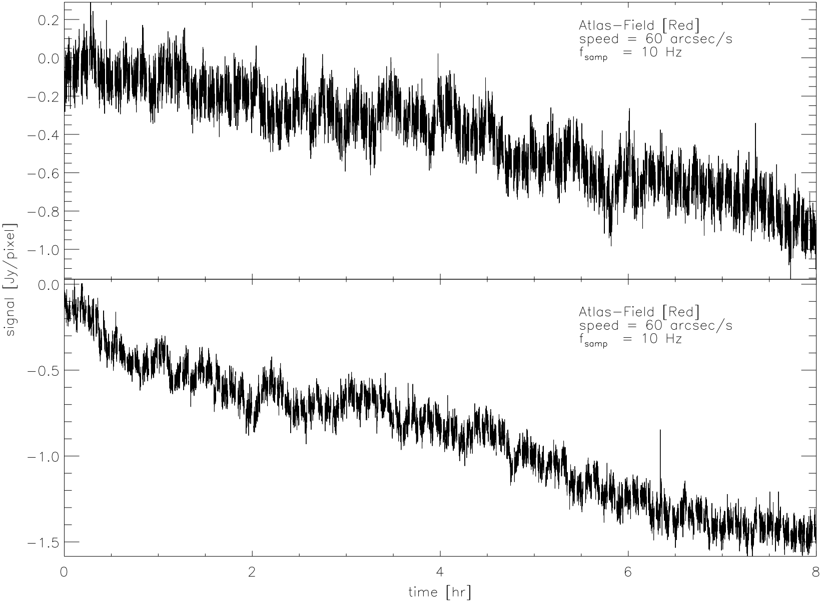
Application to HERSCHEL/PACS

Going out of HIPE

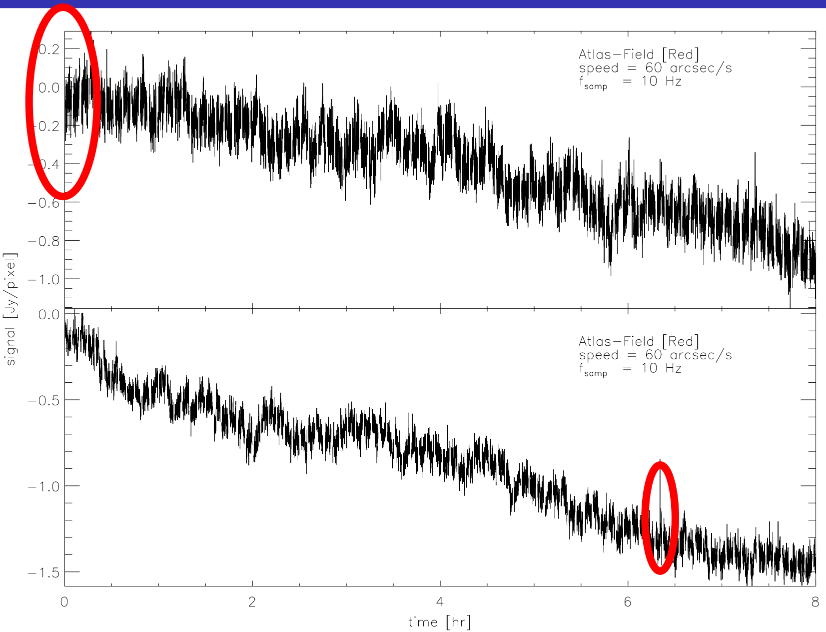
- best flags available
- all data must be calibrated in flux
- positions must be corrected of any offsets / errors
- no `highpassFilter` / `photGlobalDriftCorrection` / ... task in the pipeline
- use `export_PacsToSanepic.py` (M. Sauvage)
  - make sure that you ran a second level deglitcher

Have a look a the data

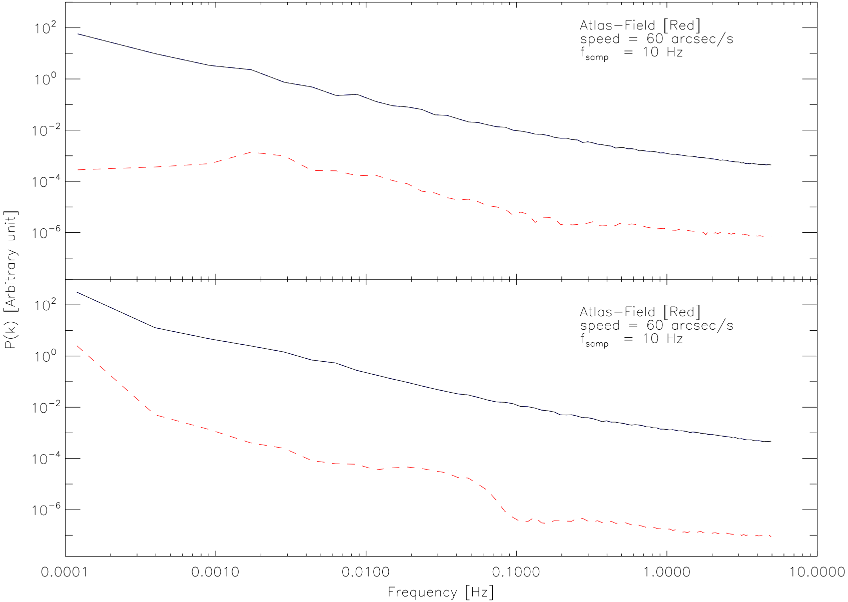
PACS data timeline



PACS data timeline



PACS power spectra



PACS rule of thumb

Have a look at your data

- Noise is often stationary, but check
- Unflagged calibration block
  - can affect whole map power distribution (FFT)
- Unflagged glichtes
  - second level deglitching (after SANEPIC)
  - recompute the noise power spectra (?)

Blue/Green data

- n_{chan} prevent full power spectra estimation
- can be split into sub-arrays ...
- ... and then recombined before full inversion with SANEPIC

Mapping Transfer function $P_{out}(k)/P_{in}(k)$

- Monte-Carlo simulation
- Mandatory as filtering depends on the observing strategy

PACS rule of thumb

Have a look at your data

- Noise is often stationary, but check
- Unflagged calibration block
  - can affect whole map power distribution (FFT)
- Unflagged glichtes
  - second level deglitching (after SANEPIC)
  - recompute the noise power spectra (?)

Blue/Green data

- n_{chan} prevent full power spectra estimation
- can be split into sub-arrays ...
- ... and then recombined before full inversion with SANEPIC

Mapping Transfer function $P_{out}(k)/P_{in}(k)$

- Monte-Carlo simulation
- Mandatory as filtering depends on the observing strategy

PACS rule of thumb

Have a look at your data

- Noise is often stationary, but check
- Unflagged calibration block
  - can affect whole map power distribution (FFT)
- Unflagged glichtes
  - second level deglitching (after SANEPIC)
  - recompute the noise power spectra (?)

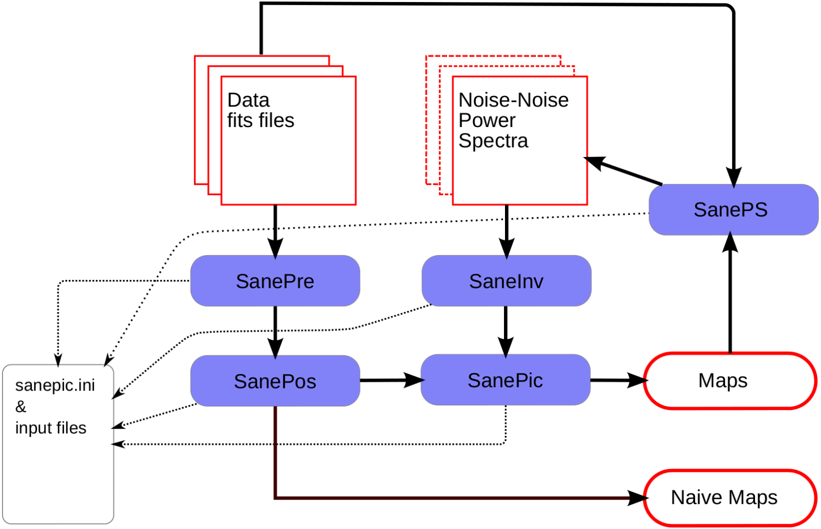
Blue/Green data

- n_{chan} prevent full power spectra estimation
- can be split into sub-arrays ...
- ... and then recombined before full inversion with SANEPIC

Mapping Transfer function $P_{out}(k)/P_{in}(k)$

- Monte-Carlo simulation
- Mandatory as filtering depends on the observing strategy

Programs structure



Output Map

```
    optimMap_sanePic.fits has 7 extensions
Primary header: 21 records
No data
Extension 1 -- Image
               Header : 40 records
               IMAGE ( 1750 2104 )
Extension 2 -- Error
               Header : 40 records
               IMAGE ( 1750 2104 )
Extension 3 -- Coverage
               Header : 40 records
               IMAGE ( 1750 2104 )
Extension 4 -- Findchart
               Header : 40 records
               IMAGE ( 1750 2104 )
Extension 5 -- mask
               Header : 22 records
               IMAGE ( 1750 2104 )
Extension 6 -- IniFile
               Header : 21 records
               Binary Table ( 193 55 )
Extension 7 -- InputFiles
               Header : 33 records
               Binary Table ( 189 2 )
```