Lecture 5: Dealing with Data

Bálint Joó
Jefferson Lab

INT Summer School – Lattice QCD and its Applications

http://www.jlab.org/~bjoo/Lecture5.pdf

Introduction

- I will discuss the following topics:
  - Equilibration / Thermalization / Setup cuts
  - Getting at errors with resampling methods
    - Jackknife
    - Bootstrap
  - Autocorrelations
    - Blocking
  - Basic Minimum χ^2 Fitting
    - Use fitting code as a 'black box'

Let's get some data

- You should check out `seattle_tut/example5` via CVS as usual:
  - `export CVSROOT=:pserver:anonymous@cvs.jlab.org:/group/lattice/cvsroot`
  - `cvs checkout seattle_tut/example5`
- In this example, I have packaged up some real data for you from our current production on the ORNL Cray.
  - Anisotropic Lattice, Tadpole improved Luescher-Weisz gauge action, 3 flavours of Wilson-Clover Fermions, generated with an RHMC algorithm.
    - Plaquette data
    - Data for the lowest/highest eigenvalues of the Preconditioned Fermion matrix used in the production.
    - Some spectroscopy data from a 3 flavor clover run on a small lattice (12^3x128)

Let's look at the plaquette first

- in the example5 directory look in Data/Raw
  - copy the plaquette data to a temporary work directory

```
$ mkdir work
$ cd work
$ cp ../Data/Raw/sztcl3_b2p00_x3p500_um0p054673_n1p0_plaquette_11-1160.tar.gz .
```

  - unzip the tarfile

```
gunzip sztcl3_b2p00_x3p500_um0p054673_n1p0_plaquette_11-1160.tar.gz
tar xvf sztcl3_b2p00_x3p500_um0p054673_n1p0_plaquette_11-1160.tar
```

  - you should end up with a bunch of small files- one per RHMC traj.

```
$ ls
plaquette.meas.xml.100
plaquette.meas.xml.1000
plaquette.meas.xml.1001
...
```

traj. number

Looking at one file

- Look at one plaquette file: eg: plaquette.meas.xml.102

/Plaquette/w_plaq

/Plaquette/s_plaq

```
<?xml version="1.0"?>
<Plaquette>
  <update_no>102</update_no>
  <w_plaq>0.584091158756O3</w_plaq>
  <s_plaq>0.400594009096578</s_plaq>
  <t_plaq>0.767588308415482</t_plaq>
  <plane_01_plaq>0.400598437664356</plane_01_plaq>
  <plane_02_plaq>0.400789580107107</plane_02_plaq>
  <plane_12_plaq>0.40039400951827</plane_12_plaq>
  <plane_03_plaq>0.76756753732247</plane_03_plaq>
  <plane_13_plaq>0.767583397034504</plane_13_plaq>
  <plane_23_plaq>0.767613990889472</plane_23_plaq>
  <link>0.000129365860688846</link>
</Plaquette>
```

average plaquette

average spatial plaquette

average temporal plaquette

plaquettes in the 6 individual planes

The average link trace

Xpath expressions: /Plaquette/w_plaq identify nodes.

Data Extraction

- We need to get our desired measurement out of all the files and ordered by update number. One way, is to just use bash and some UNIX tools:

```
ls -1 plaquette.meas.xml.* | cut -f4 -d'.' | sort -n > trajs
```

list files

select 4th field delimited by '.' ie the traj. number

sort result numerically

  - At this point the file 'trajs' should contain the list of trajectories in sorted numerical order.
  - Next step: extracting the plaquettes

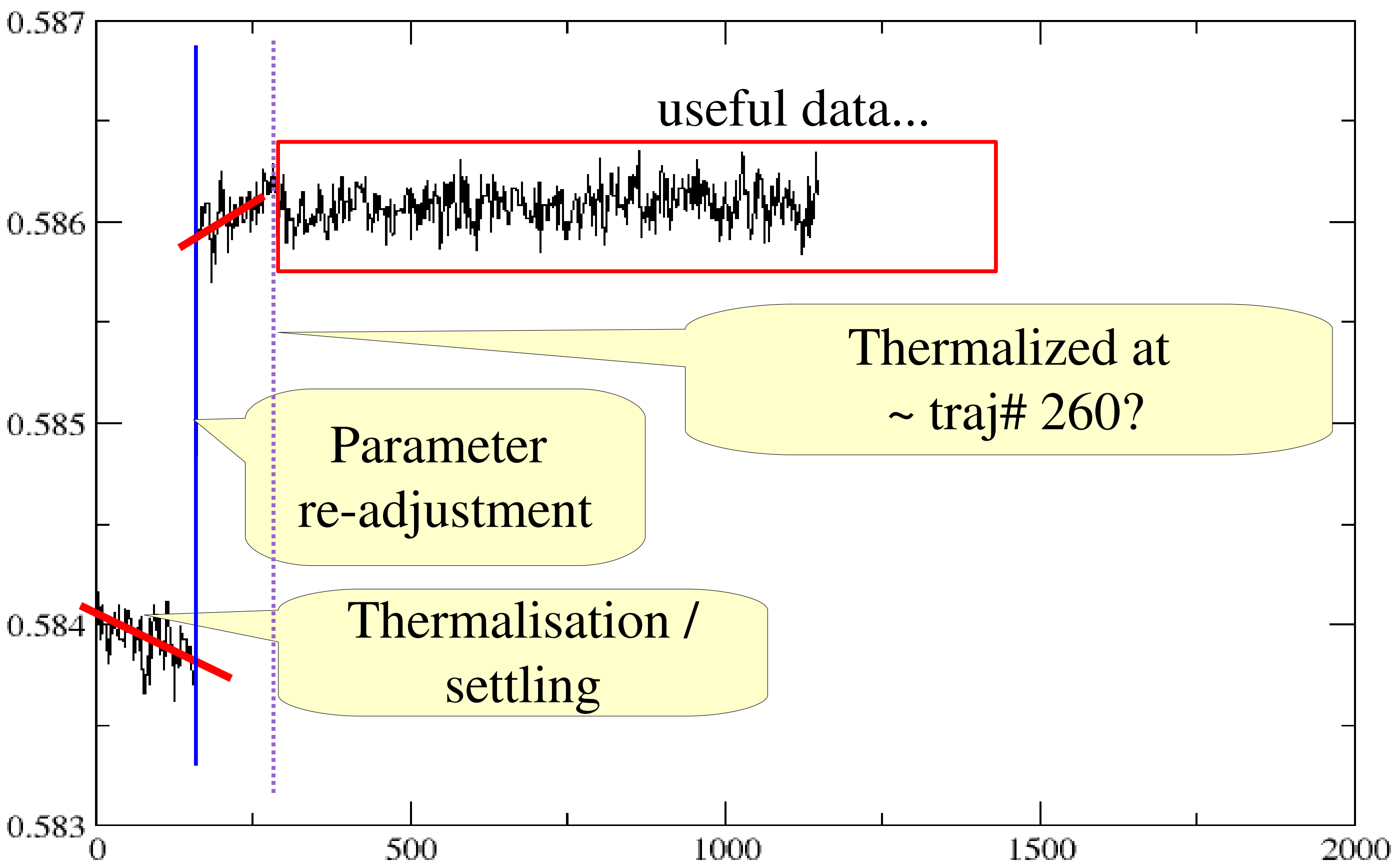
Using print_xpath

- With QDP++ we bundle a utility called `print_xpath`
  - `$HOME/install/qdp++/bin/print_xpath`
  - Make sure this bin/ directory is on your $PATH
- which can be used to extract data from XML files using Xpath expressions.
- let us extract the w_plaq measurement
  - Xpath: /Plaquette/w_plaq
  - Using a bash 'for' loop (foreach for tcsh I think)

```
for x in `cat trajs`; do \
  plaq=`print_xpath plaquette.meas.xml.$x /Plaquette/w_plaq` ; \
  echo $x $plaq ; \
done > w_plaq.dat
```

- The file 'plaquettes' now contains <traj#> <plaquette> pairs

Let's look at the time history...



Key points:

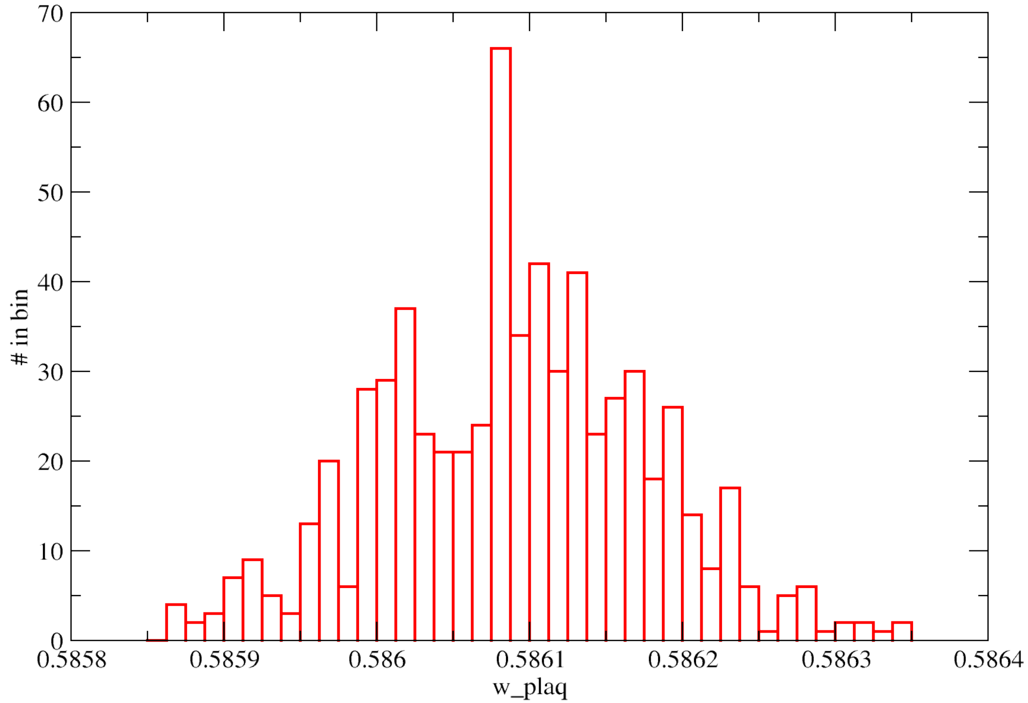
- HMC has a 'settling' (equilibration) time
  - also one frequently 'tunes' the run at the outset
  - data from this phase ought to be discarded
- How much to discard?
  - formally: 1 or 2 x the exponential autocorrelation time
    - defined as the longest autocorrelation time in the system
  - in practice, one looks at time histories for some observables
    - preferably long range ones (ie not plaquette)
      - lowest eigenvalue of fermion matrix
      - large timeslice value of a meson

Exercise: the lowest eigenvalue

- In the Data/Raw directory we have some data about the lowest and highest eigenvalues of the squared preconditioned operator used in our RHMC in the file:
  - `sztcl3_b2p00_x3p500_um0p054673_n1p0_eigen_mdagm_15-1160.tar.gz`
- extract the data from this file and plot the time history.
- NOTE: We measure this only every 5th trajectory.
- When does it look like it has thermalized?

Error Estimation

- We'll consider error estimation
  - For now, we'll assume that our data is 'independent'
    - We'll worry about autocorrelations a little later
  - Let us look at a histogram of the plaquette from traj# 500 :



- Does it look Gaussian distributed to you?

Jackknife Erros

Original Sample

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \xrightarrow{\text{mean, N data}} \bar{y}$$

Jackknife Samples

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \xrightarrow{\text{mean, N-1 data}} \bar{y}_0 \qquad \begin{bmatrix} y_0 \\ y_2 \\ y_3 \end{bmatrix} \xrightarrow{\text{mean, N-1 data}} \bar{y}_1 \qquad \begin{bmatrix} y_0 \\ y_1 \\ y_3 \end{bmatrix} \xrightarrow{\text{mean, N-1 data}} \bar{y}_2 \qquad \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \xrightarrow{\text{mean, N-1 data}} \bar{y}_3$$

$$\sigma_J = \sqrt{\frac{N-1}{N} \sum (\bar{y}_i - \bar{y})^2}$$

Distribution of Jackknife means “simulates” distribution of actual means.

Exercise: Quick and Dirty Jackknife

- Write a program to compute the jackknife error for a set of real numbers
  - You don't need QDP++ for this exercise
  - You may consider using the std::vector<> class from the C++ Standard Template Library – this is like multi1d<> in QDP++
    - See http://www.cplusplus.com/reference/stl/vector/
  - You may consider using C++ style I/O
  - An model answer is in
    - seattle_tut/example5/src/jack.cc

Computing the mean

- This is really easy, especially using the std::vector class to hold your data

```
// 'import' the vector class
#include <vector>
using namespace std;

// Compute the arithmetic mean of a vector of doubles
double mean(const vector<double>& data) {
  double sum=0;
  for(int i=0; i < data.size(); i++) {
    sum += data[i];
  }
  return sum / (double)(data.size());
}
```

Creating a Jackknife Dataset.

```
// Compute the jack_sample-th jackknife sample
// from input vector data into output vector jack_data
void jackSet(const vector<double>& data,
             vector<double>& jack_data,
             int jack_sample)
{
  if ( jack_sample < 0 || jack_sample >= data.size() ) {
    cerr << "jack_sample=" << jack_sample
         << " outside allowable range [0, " << data.size()-1 << "]"
         << endl;
    exit(1);
  }

  jack_data.resize(0);
  for(int i=0; i < data.size(); i++){
    if( i != jack_sample ) {
      jack_data.push_back(data[i]);
    }
  }
}
```

Error Checking

Append

Computing the Jackknife error

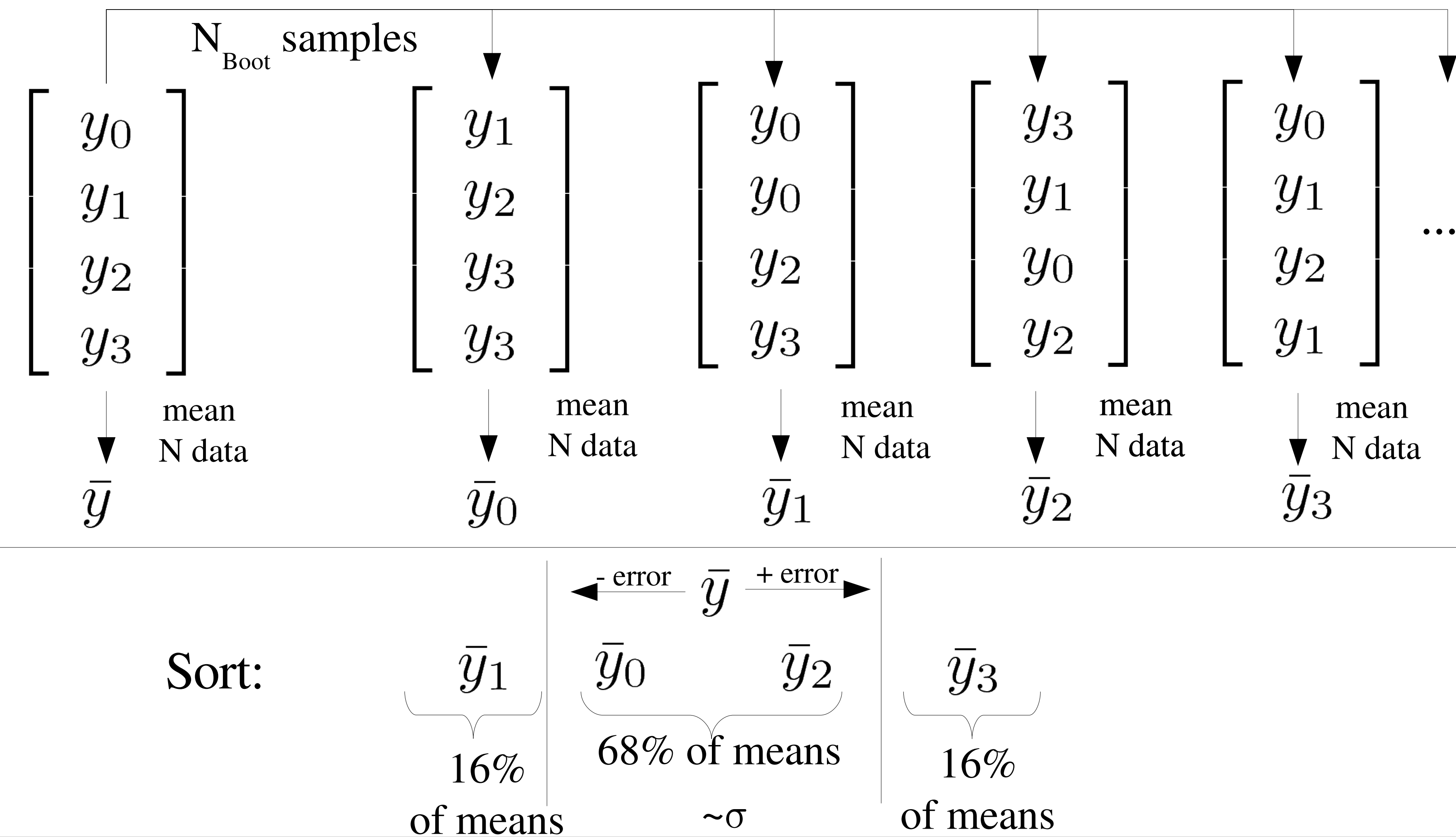
```
// Compute the jackknife error
double jackErr(const vector<double>& data)
{
  double m = mean(data); // Get original mean
  double sumsq = 0;      // Use this for variance: Sum ( jackMean - m)^2

  // Compute mean on each jackknife sample
  for(int i=0; i < data.size(); i++) {
    vector<double> jack_sample;
    jackSet(data, jack_sample, i);  // Get i-th jackknife sample
    double jackMean = mean(jack_sample); // Compute ith jackknife mean
    sumsq += (jackMean - m)*(jackMean - m); // accumulate variance term
  }

  // Normalize variance
  sumsq *= (double)(data.size()-1) / (double)data.size();
  return sqrt(sumsq); // return square root of variance ie error
}
```

Bootstrap Errors

Original Sample Bootstrap Samples: Random picks w. repetition



Autocorrelations

- Data from Markov Chain Monte Carlo methods may well be affected by autocorrelations.
- Typically successive configurations are correlated

$$\sigma^2(\mathcal{O}) = (2\mathcal{A}_{\mathcal{O}} + 1)\,\sigma_n^2(\mathcal{O})$$

True Variance

Integrated autocorrelation time for the observable (=0 for independent data)

naïve variance (ie the one we find if we assume are samples are independent)

Autocorrelations

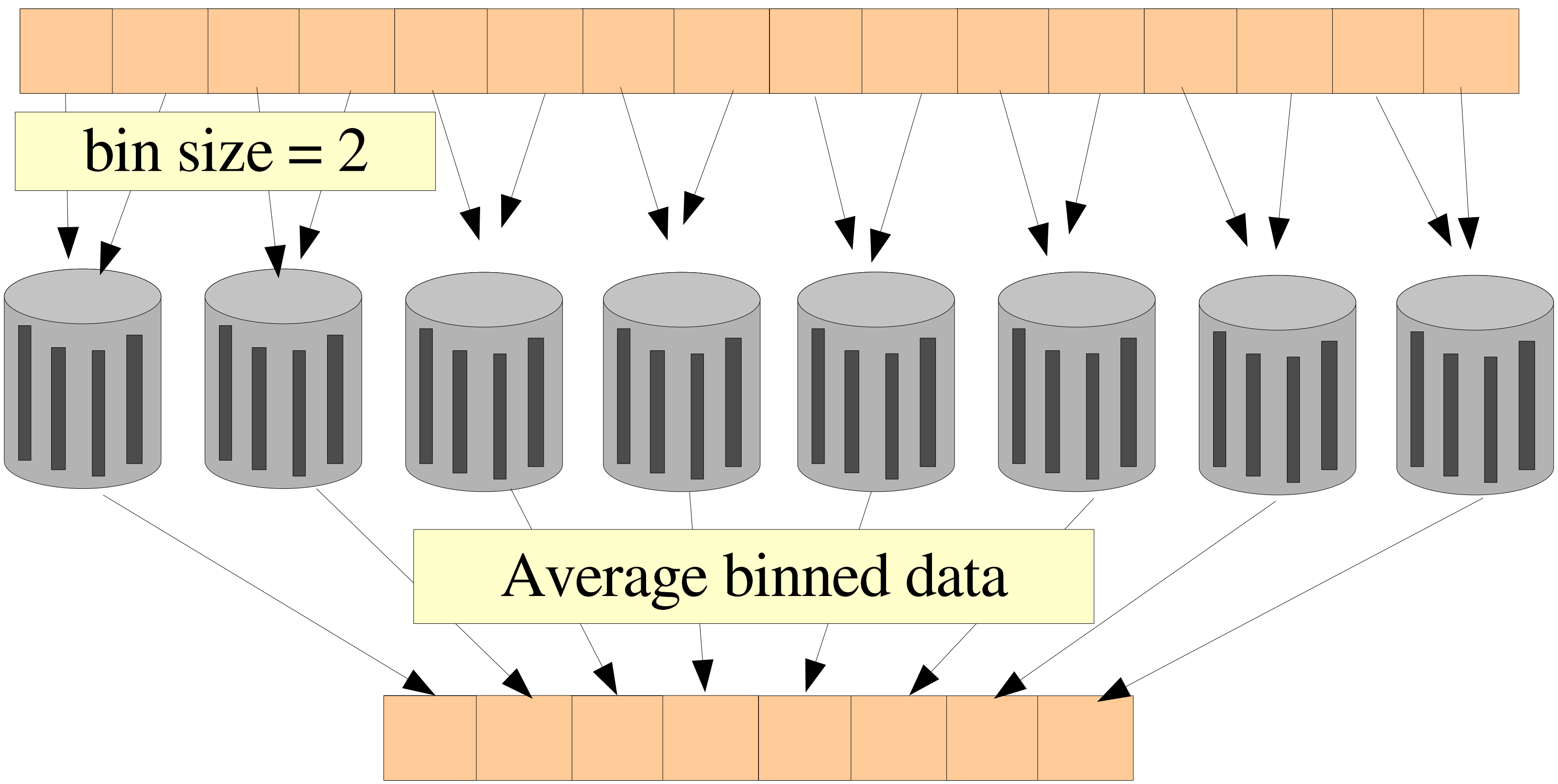
$$\mathcal{A} = \sum_{t=1}^{\infty} C(t)$$

$$C(t) = \frac{1}{\sigma^2} \langle \left(\mathcal{O}(|t| + t_0) - \langle \mathcal{O} \rangle \right) \left(\mathcal{O}(t_0) - \langle \mathcal{O} \rangle \right) \rangle_{t_0}$$

- Measuring Autocorrelations is hard because
  - The quantity C(t) is very noisy (an error on an error)
  - The convergence of the sum for $\mathcal{A}$ depends on delicate cancellations in C(t).
- A pragmatic approach is to 'make' our data independent
  - Measure sufficiently infrequently AND/OR
  - Block (rebin) data

Binning Data

Original correlated data:

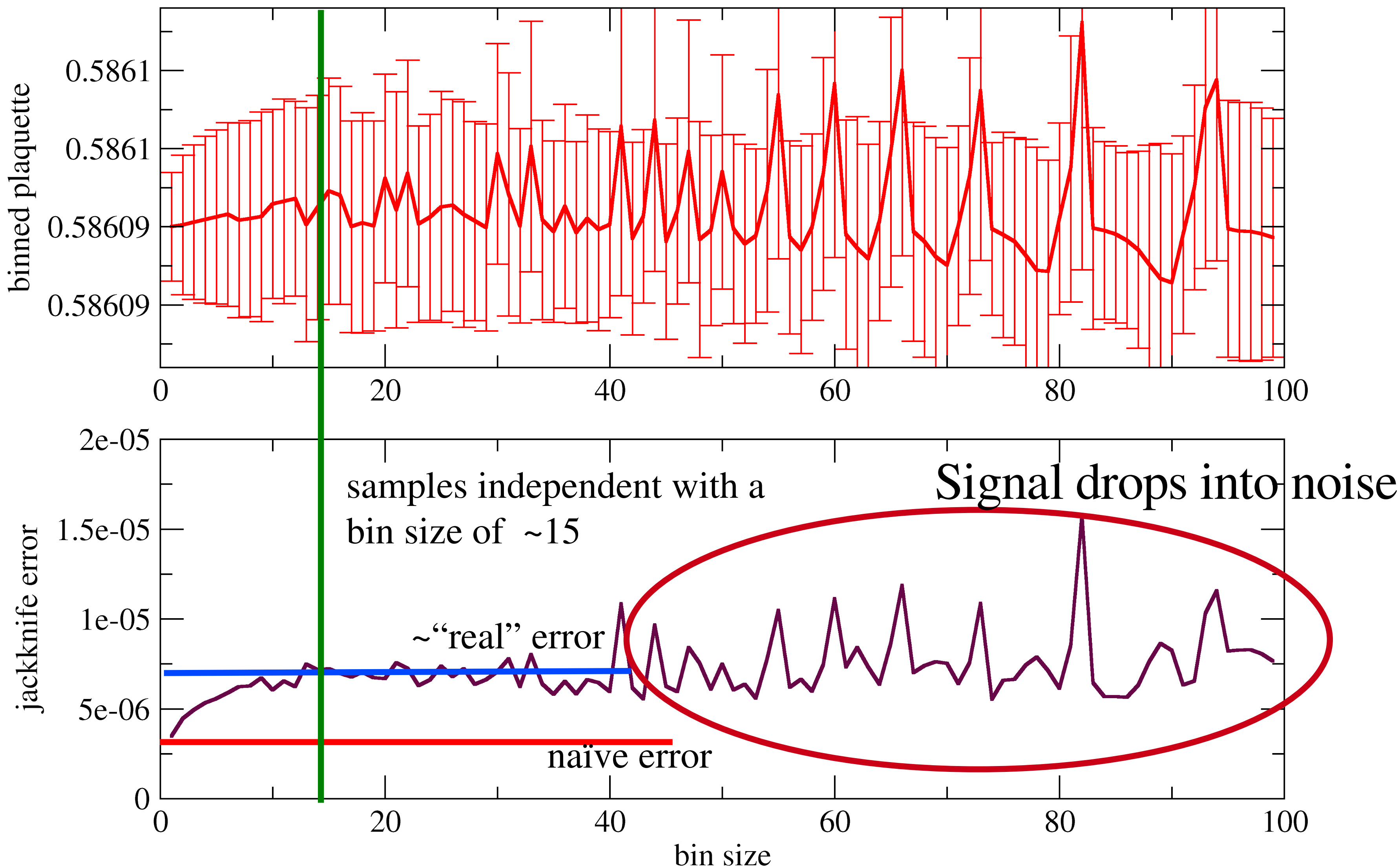


Less correlated data

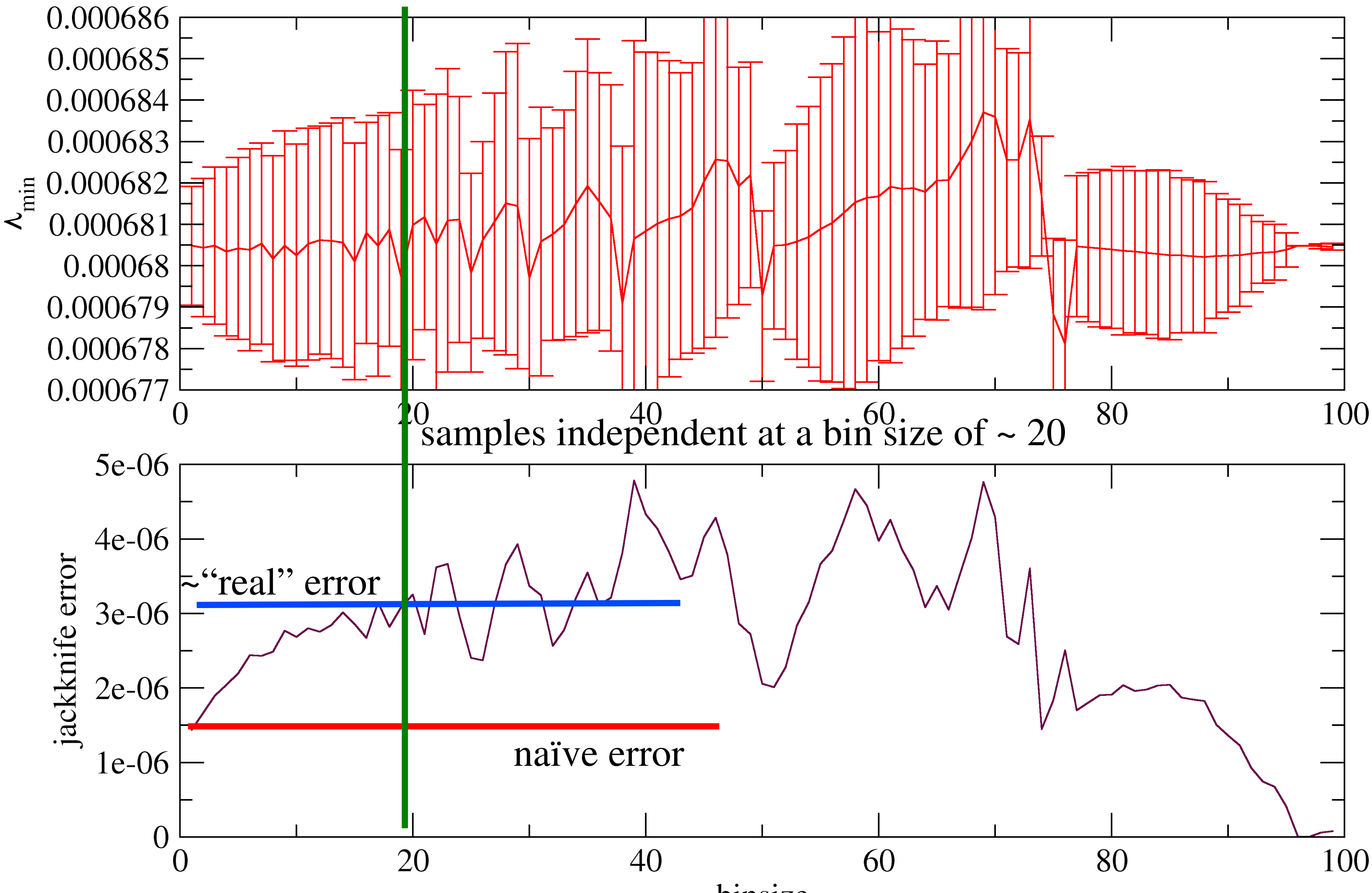
Exercise

- Modify your jackknife program to:
  - Bin the data with a given bin width of 2
  - Compute the jack-knife error on the rebinned data
  - Compute the jack-knife error as a function of bin size for bin sizes ranging from 1 to 100.
  - Plot the mean and jackknife error as a function of bin size using your favourite plotting program

Binning the plaquette



Binning the low EVs



Autocorrelation time

- Plaquette
  - samples independent with a bin size of about 15
  - we measure on every trajectory so

$$2\mathcal{A}_{\mathrm{Plaq}} + 1 \approx 15 \Longrightarrow \boxed{\mathcal{A}_{\mathrm{Plaq}} \approx 7}$$

- Lowest eigenvalue of $\tilde{M}^\dagger \tilde{M}$
  - samples independent with a bin size of about 20
  - we measure on 5$^{\mathrm{th}}$ trajectory so

$$2\mathcal{A}_{\lambda} + 1 \approx 20 \times 5 = 100 \Longrightarrow \boxed{\mathcal{A}_{\lambda} \approx 50}$$

Fitting Correlation functions

- We extract physics from our simulation data, by fitting correlation functions to models

$$C_\pi(t) = \sum_{i=0}^{\infty} A_i e^{-E_i t} \xrightarrow{t \to \infty} A_0 e^{-E_0 t}$$

Fit model or Fit function

- In a fit, the things that vary are the parameters (ie A0 and E0)
  - The data is fixed by the simulation, and the fit function is fixed by our choice

$\{y_i\}$	The computed correlation fn at timeslice t_i
$C(i, A_0, E_0)$	The chosen fit function at timeslice t_i

Minimising the χ^2

- One popular way of fitting to the data is maximum likelyhood estimation, it involves minimising χ^2

$$\chi^2(A_0,E_0) = \sum_{i,j} [y_i - C(i,A_0,E_0)]\, M(i,j)^{-1}\, [y_j - C(j,A_0,E_0)]$$

- $M(i,j)$ is the data covariance matrix:

$$M(i,j) = \langle (y_i - \langle y_i \rangle)\,(y_j - \langle y_j \rangle) \rangle$$

- If our data are independent as a function of t:

$$M(i,j) = \sigma^2(y_i)\,\delta_{i,j}$$

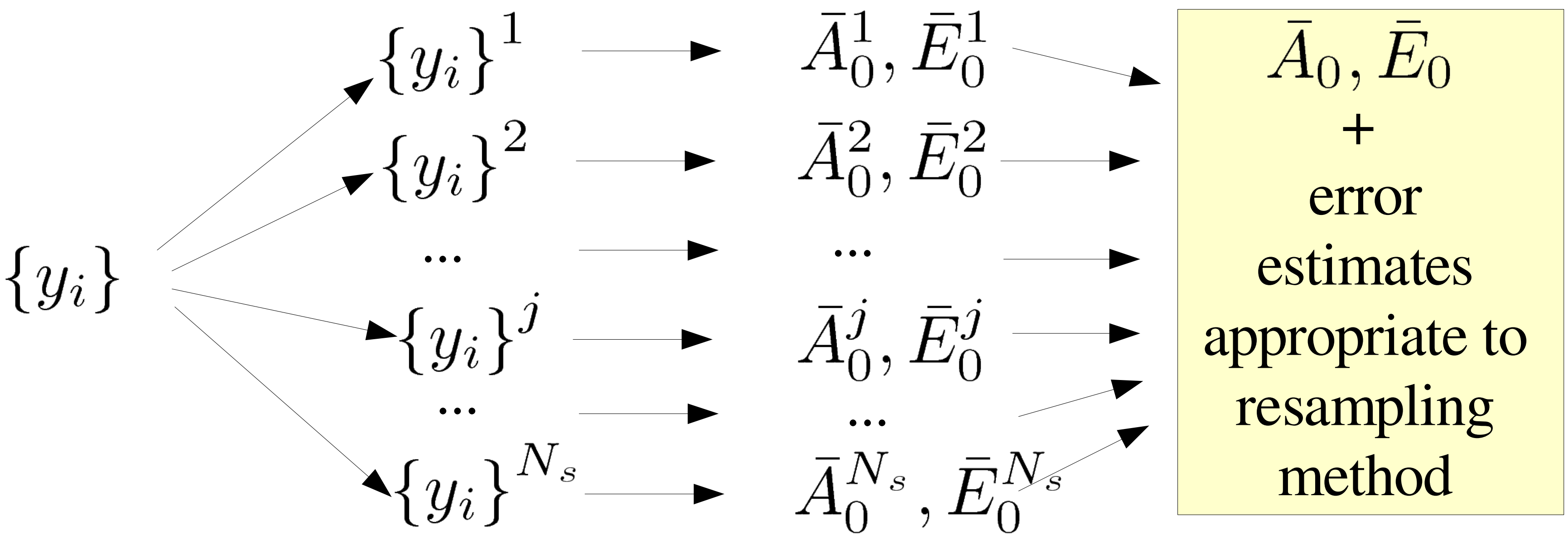
$$\chi^2 = \sum_i \frac{[y_i - C(i)]^2}{\sigma^2(y_i)}$$

What about errors?

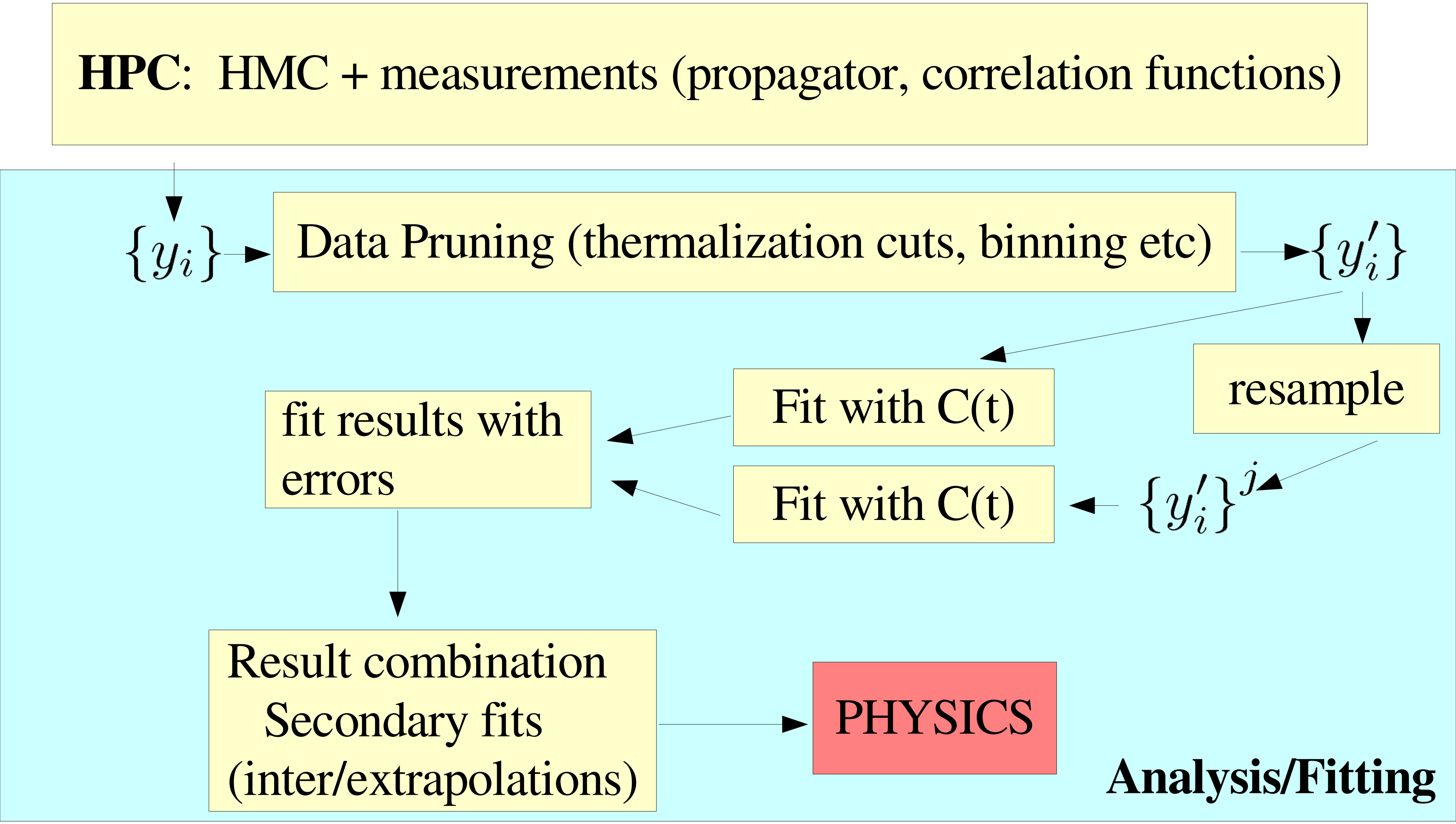
- Essentially a fit is a function of our data and our fit model:

$$\text{fit} \;:\; \{y_i\}\,, C(i, A, m) \longrightarrow \bar{A}_0, \bar{E}_0$$

- In a re-sampling technique such as the jackknife or the bootstrap, we can carry out the fit on each of the N_s (re)samples and analyze the distribution of the means.



The QCD workflow



Details of the fitting

- This is beyond the scope of this lecture
- and tends to be somewhat of a religious topic.
- People tend to write their own as a “right of passage”
- For the demonstrations and exercises I will use an old code called the 4H fitting code, which is still used in UKQCD
  - I repackaged it for 'simplicity'
  - I won't go through the details but at a high level
    - It can do correlated or uncorrelated fits
    - It uses an implementation of the Marquardt-Levenberg algorithm for its minimization
    - We will use the 'single_exp_fit' program which has been pre-written to work with chroma data

Compiling the fitting code

- Go to `seattle_tut/example5/src`
- Make a build directory and cd into it

```
mkdir build
cd build
```

- Configure the code

```
../hhhh/configure -prefix=$HOME/install/hhhh
make
```

  - The code should now build. Let's install it

```
make install
```

- Add the installation bin/ directory to the path

```
export PATH=$HOME/install/hhhh/bin:$PATH
```

- Check it works: run `single_exp_fit` (`single_exp_fit.exe`)

Look at a few mesons

- Go to seattle_tut/example5/work
- Look at the effective mass of a zero momentum pion:

An output prefix

```
av_chroma_corr_and_effmass \
     ../Data/Raw/mesons/pion.D-546.P_1.P_1.PP pion1
```

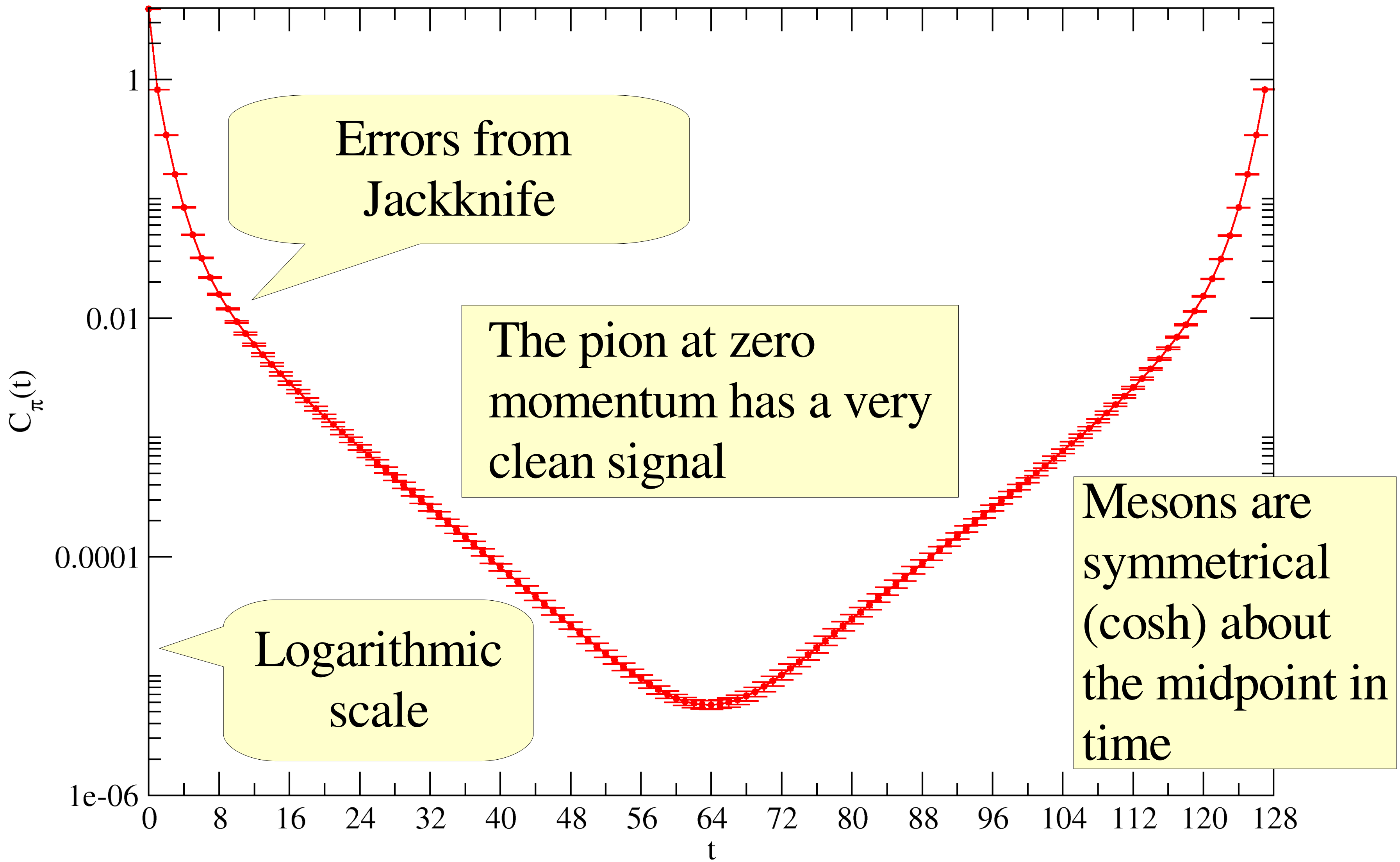
The correlator file

- This should produce the following files:

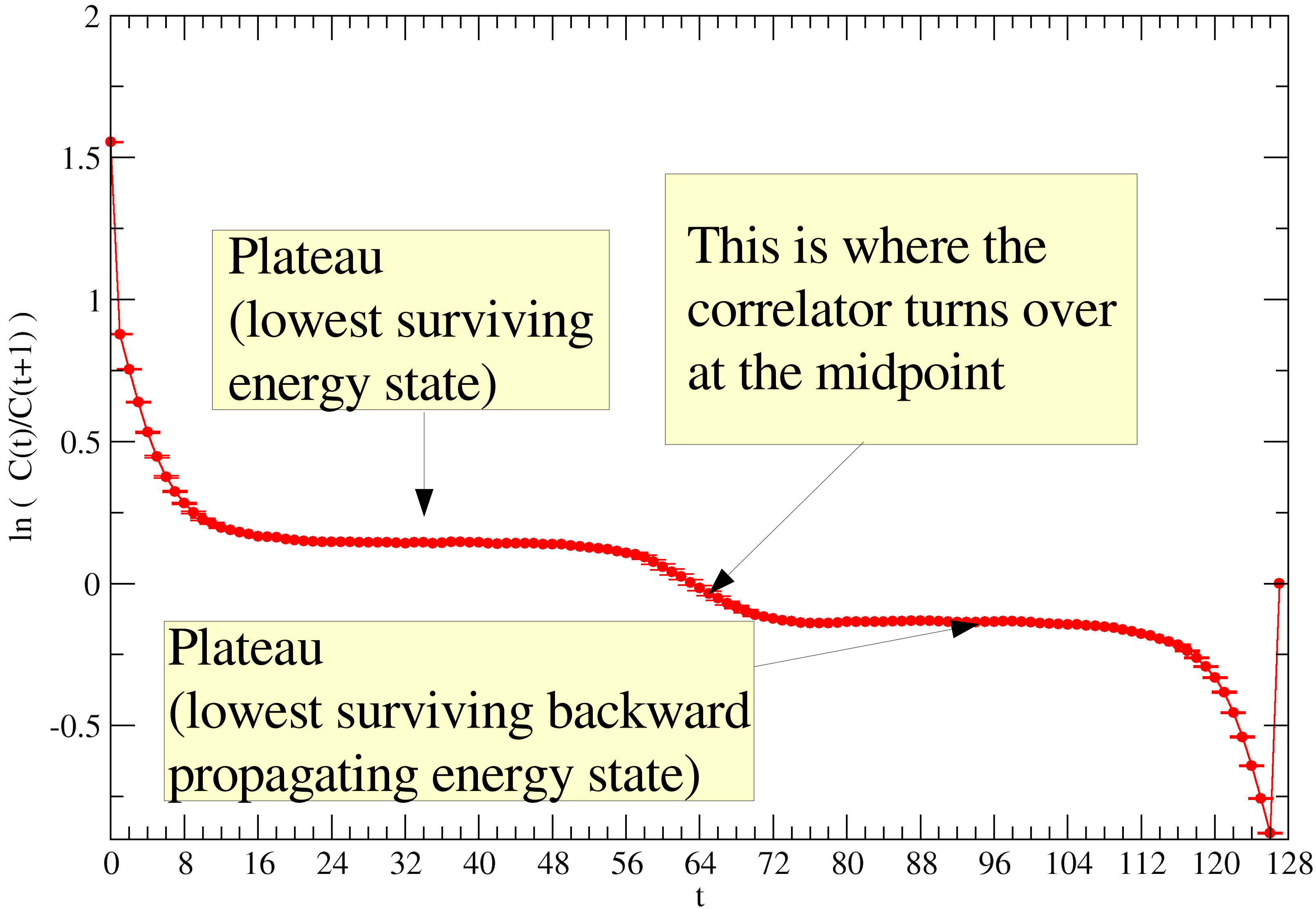
```
pion1_av_corr.dat    pion1_fold_av_corr.dat
pion1_eff_mass.dat   pion1_fold_eff_mass.dat
```

- Let us look at the correlator and effective mass files (I use xmgrace for plotting):

The Pion Correlator



The Pion Effective Mass

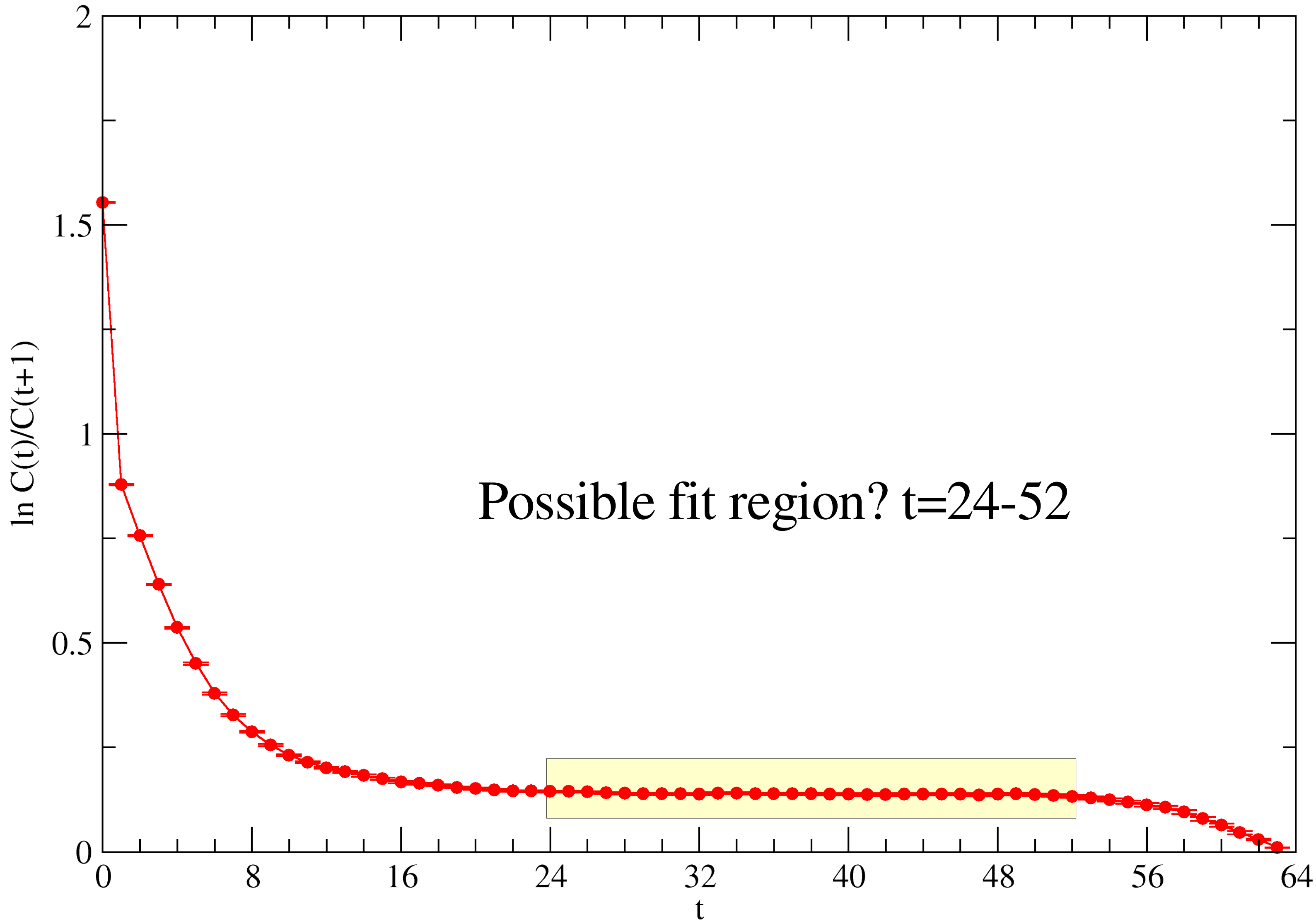


Folding the propagator

- Mesons are symmetric about the midpoint in time
- We can use this fact to 'double' our statistics by folding the meson correlator about the midpoint

$$C_F = \frac{1}{2}\left(C(t) + C(L_T - t)\right)$$

Folded Pion Effective Mass



Exercise

- In `seattle_tut/example5/Data/Raw/baryons` is the correlation file for a zero momentum proton
  - `proton.D-546.P_1.P_1.PP`
- Have a look at its average correlation function and effective mass
- Is the correlation function symmetric?
- The program `av_chroma_corr_and_effmass` automatically folds the correlator about the midpoint.
  - Does it make sense to fold proton or other baryons?

Fitting the folded pion

- Let us fit the folded pion. Again in your work directory, run the (installed) program:

```
single_exp_fit  -f \
        -m 24 -M 52 \
        -P pion1   \
        ../Data/Raw/mesons/pion.D-546.P_1.P_1.PP
```

Fold correlation function

Min. & Max. timeslices

Output prefix

some encoding of the mass

Chroma correlation function file

Pion Channel 1 uses $\Gamma = \gamma_5$ we also have channel 2 using $\Gamma = \gamma_4\gamma_5$

Point source Point sink

Fitting the folded Pion

- The Output from the program should look like:

```
Attempting to read correlators from: ../Data/Raw/mesons/pion.D-546.P_1.P_1.PP
File contains 115 Correlators
Spatial Extent is 12
Temporal Extent is 128
Read Correlator: n_tslice = 128, n_corrs = 115
Will fold data around midpoint
Fit: t_min = 24
Fit: t_max = 52
Nboot = 460
Initial guesses: A : 0.0251883
Initial guesses: m : 0.144016
param[0] =   2.34438839e-02  +  7.93389305e-04 -  7.70332663e-04
param[1] =   1.41590680e-01  +  1.30904127e-03 -  1.28498398e-03
Chisq =   7.89051653e+01  Chisq / d.o.f =   2.92241353e+00  Q =   5.52726403e-07
```

param[0]<->A_0

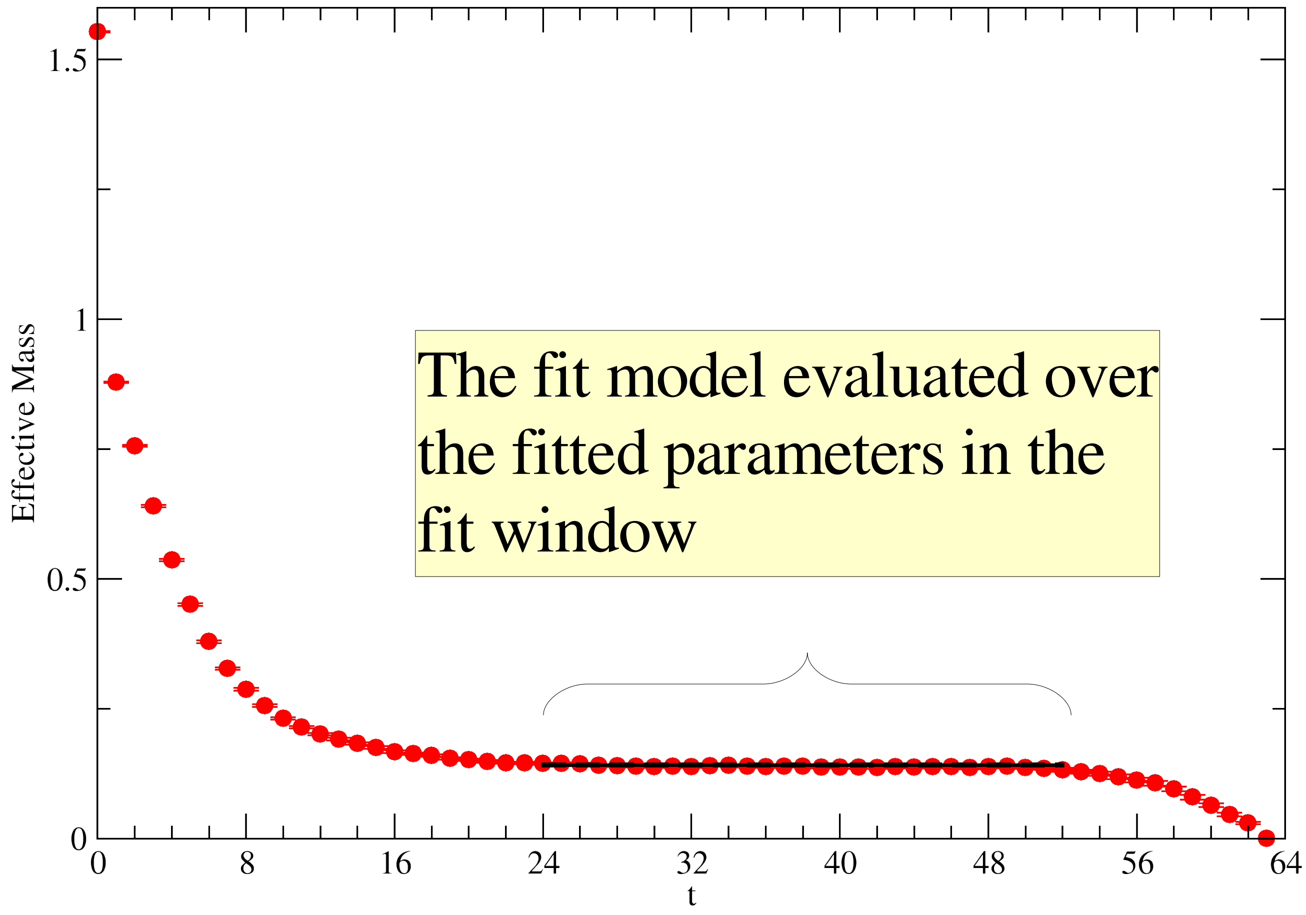
param[1]<->E_0

assymetric errors from bootstrap

d.o.f=# fit timeslices – # params

The fitter also outputs a plot

- The file is: `pion1_fit_results_24_52.agr` – a file for xmgrace



A couple of files

- The fitter also output a few other files:
  - `pion1_t_24_52_param_00.boot00` ← A_0
  - `pion1_t_24_52_param_01.boot00` ← E_0
- These are files containing the bootstrapped means of the parameters ie:
  - the best value from the original data
  - the N values computed on the N re-sampled datasets
  - These files are needed if we want to do some secondary fitting.

Exercises

- Re-fit the pion but vary the range of the fit window
  - How does the χ^2 change?
  - Can you find a better fit window?
  - Is the answer for the mass (param1) stable?
- Have a look at the other pion channel:
  - `pion.D-546.P_2.P_2.PP` (using $\Gamma=\gamma_4\gamma_5$)
- Have a look at pions at non zero momenta:
  - `pion_pxA_pyB_pzC.D-546.P_1.P_1.PP`
    - eg (A,B,C)=(1,0,0) <=> (px,py,pz) = (1,0,0)
  - Do they get noisier?
- Try fitting the proton. Remember about baryons and folding?

Secondary Fitting: Dispersion Relation

- You have the data, for pions at zero momentum and other momenta, for several channels.
- This data is from a simulation with an anisotropic action (the temporal and spatial lattice spacings are different)
- Here we need to tune parameters so that the speed of light is

$$c = 1$$

- We can find the speed of light from the dispersion relation

The momentum ($\vec{n} = (p_x, p_y, p_z)$)

What we fitted before - units of a_t

$$E^2 = m^2 + c^2 \left(\frac{2\pi|\vec{n}|}{L_s} \frac{1}{\xi} \right)^2$$

Anisotropy: a_s/a_t convert to units of at

Exercise: Doing it

- We need to fit the lowest energy on our pion correlators at zero and finite momenta:
  - Use the `pion_pxA_pyB_pzC.D-546.DG4_2.P_2.SP` files (smeared a the source, point sink)
    - cleaner signal than most
  - Use 500 bootstrap samples for all (`-b 500` option to `single_exp_fit`) the fits.
  - We have data for $|n^2| = \{0,1,2,3,4,5\}$
  - We'll keep the resulting `*_param_01.boot00` files.
  - The program `c2_check` will perform a secondary fit.
  - We need to tell it which `.boot00` file corresponds to which momentum.
  -

The input file

- Create a file looking called params which looks like this

Col 1
$| n |^2$

```
0.0   pion2_p2_0.0_t_20_35_param_01.boot00
1.0   pion2_p2_1.0_t_25_40_param_01.boot00
2.0   pion2_p2_2.0_t_25_40_param_01.boot00
3.0   pion2_p2_3.0_t_10_20_param_01.boot00
4.0   pion2_p2_4.0_t_05_15_param_01.boot00
5.0   pion2_p2_5.0_t_05_15_param_01.boot00
```

Col. 2
file for
value of
$|n|^2$

Control the name of the output file using the
-P <prefix> option of single_exp_fit

Now run c2_check

- Now run the c2_check program:

File described on previous slide

```
$ c2_check params 500
```

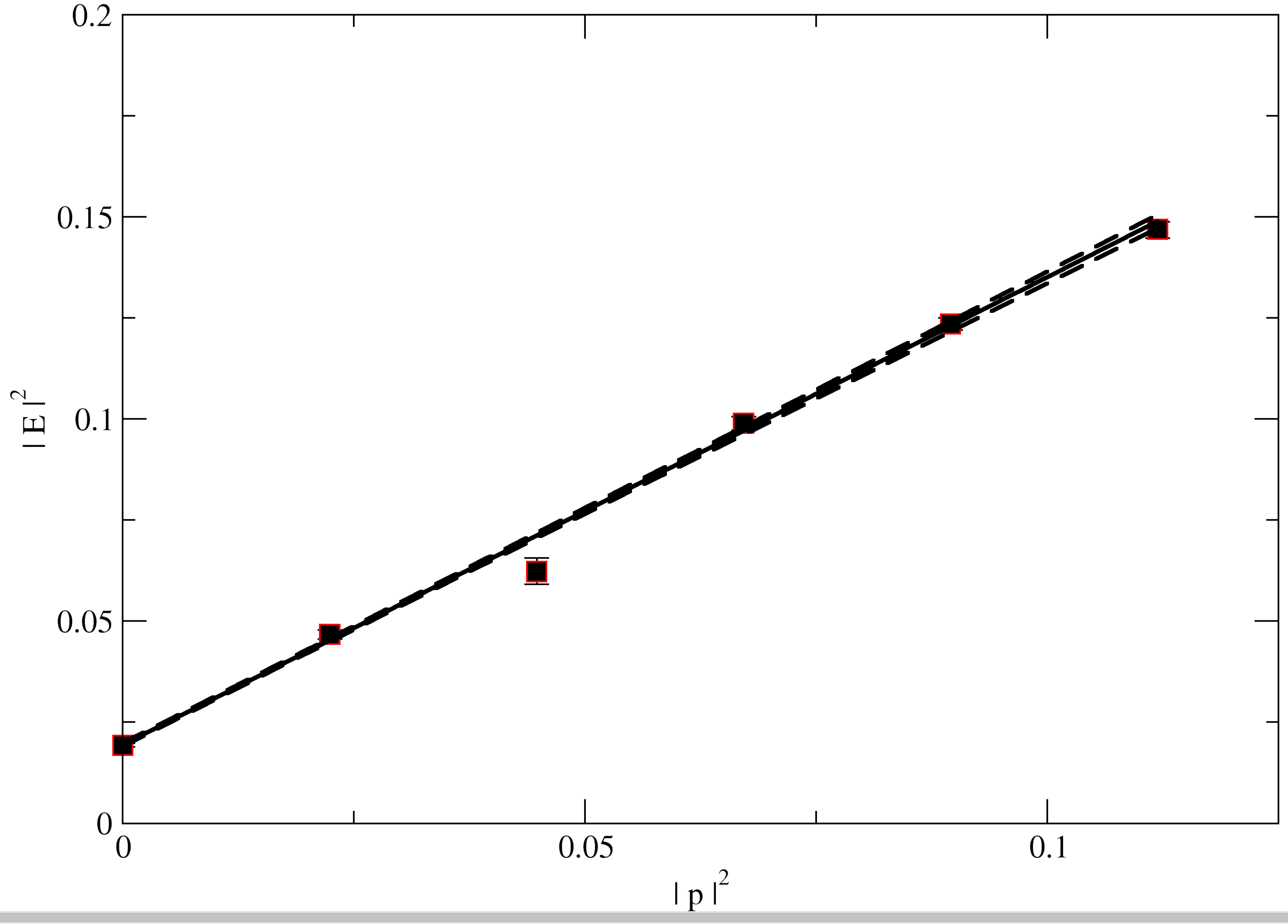
Needs to be on your PATH

No of bootstrap samples

- I get as output:

```
m^2: 0.019448 ( + 0.000438, - 0.000399 )
c^2: 1.156078 ( + 0.015552, - 0.015632 )
c: 1.075187 ( + 0.007232, - 0.007270 )
Chisq / d.o.f: 3.580948 ( + 1.593725, - 1.638315 )
```

c2_check also produces a graph



Summary

- We have dealt with
  - Pruning HMC Data
  - Resampling methods for Error estimation
    - In particular the Jackknife
  - Looking at Correlation Function data
    - Looking at Effective Masses
    - Fitting single exponentials using a 'black box' fitter
    - Performing a secondary fit using a 'black box' fitter
    - All this with real data.
  - Armed with Mike's Lectures, and the sample code, you should be well on the way to performing your own fitting in the future.