How to Recognize Garbage

Michael Shirts
Department of Chemical & Biological Engineering
University of Colorado Boulder

Hands-On with Molecular Simulation
October 29th, AIChE 2017, Minneapolis, MN

Locations of files

- If you want to run on your own computer:

- git clone https://github.com/shirtsgroup/physical-validation
- python setup.py install
- git clone https://github.com/shirtsgroup/AICHE_workshop_2017.git

What I won't cover

- Am I asking the right scientific question?
- What level of theory do I need to use?
- Am I looking at the correct variable?
- Is what I am planning feasible to do?

- If there was a recipe for this, then everybody would only do fantastic research.
- We are **instead** focusing on generating data assuming you are **trying** to do the right thing.
- Just one slide on this, though:

All models are wrong; some are useful

Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary, following William of Occam he should seek an economical description of natural phenomena.

Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.

George Box, 1978

My suggestion: Develop a hypothesis, and use molecular simulation to answer that hypothesis.

It's very easy to screw up computer model

"All happy families are alike; all unhappy families are unhappy in their own way"

"All correct simulations are alike; all bad simulations are bad in their own way"

"I cannot tell you all the things whereby ye may commit sin; for there are divers ways and means, even so many that I cannot number them."

Five principles to recognize garbage

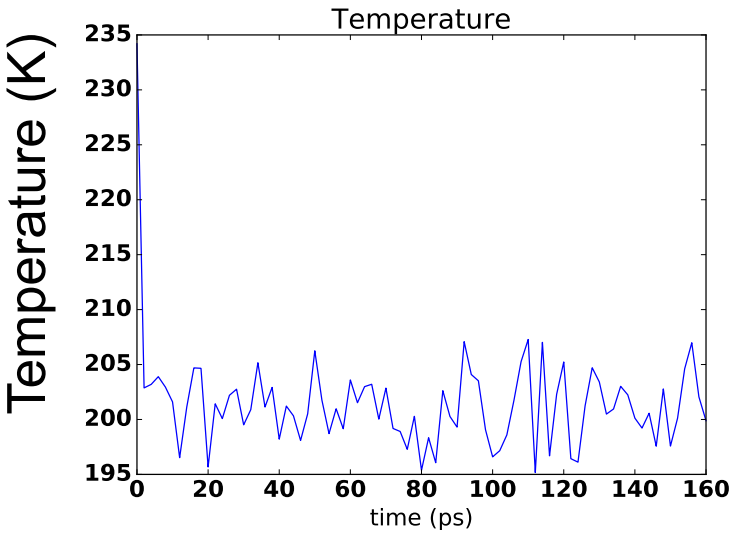
1. You need to look at both pictures and numbers
2. Always calculate everything two different ways
3. Physics is your friend
4. Statistics is your friend
5. Never trust a computer

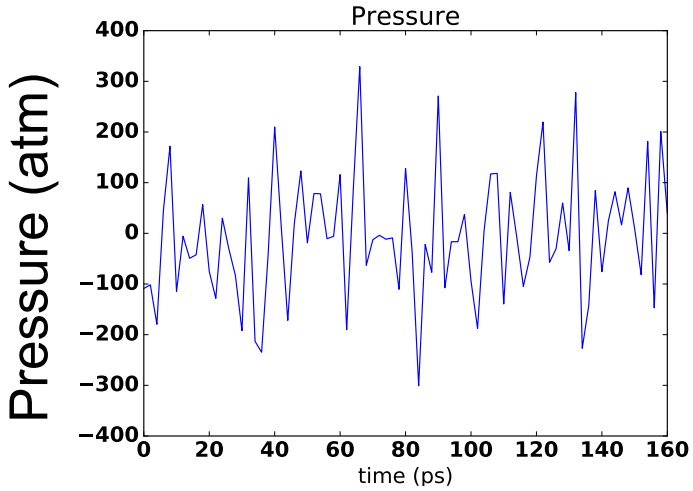
Principle 1: Look at both pictures and numbers

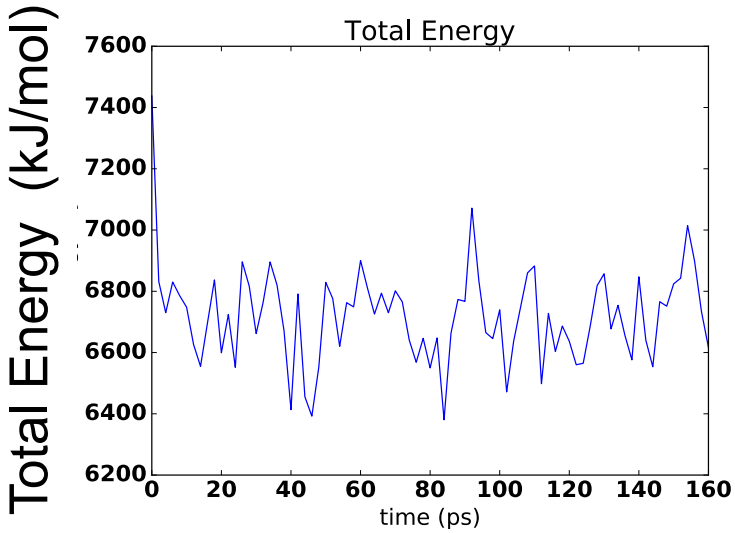
- Molecular simulations involve a huge amount of data
- We can only understand it by looking at reduced dimensions of the data.
- Each way of reducing the dimension only looks at part of the data: you need to look at this in multiple ways
  - Pictures: we are good at visual processing.
  - Numbers: things like energy, pressure

Principle 1: Look at both pictures and numbers

Box of liquid isobutane simulated at NVT

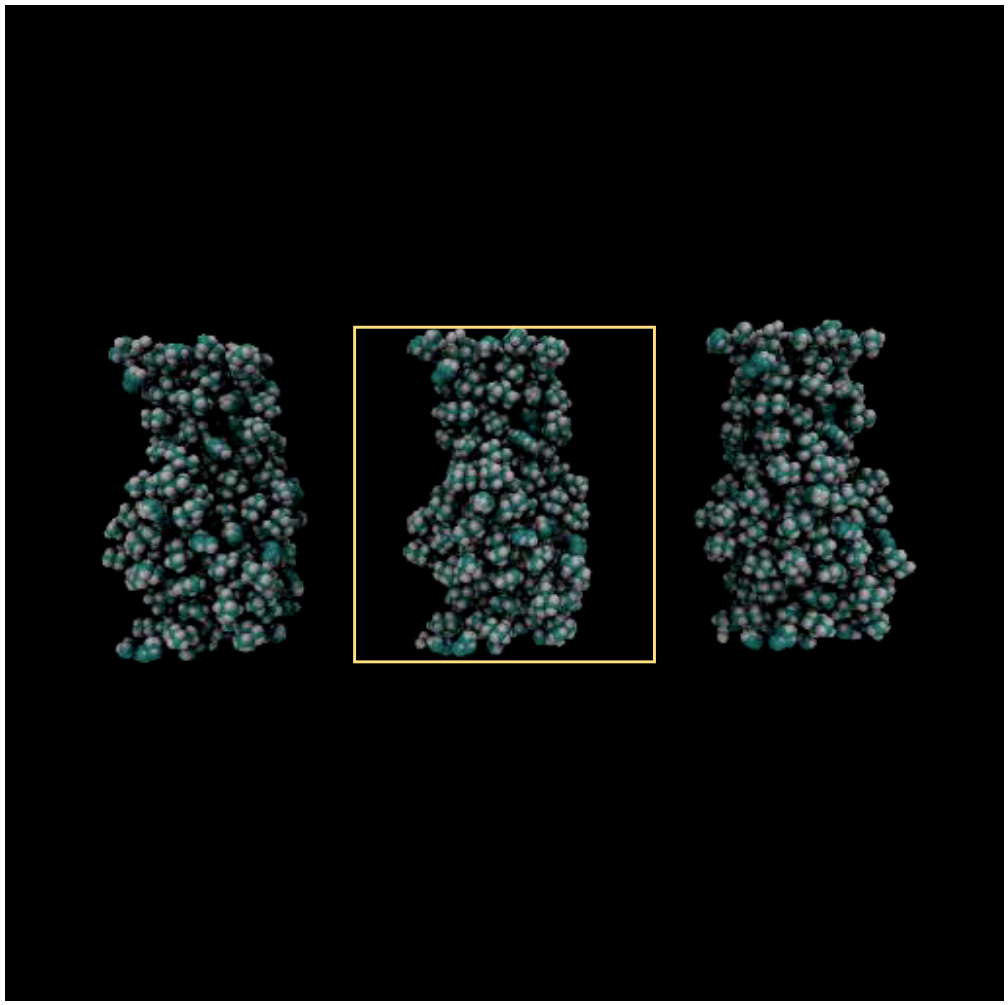






Principle 1: Look at both pictures and numbers

see pillars.mpg



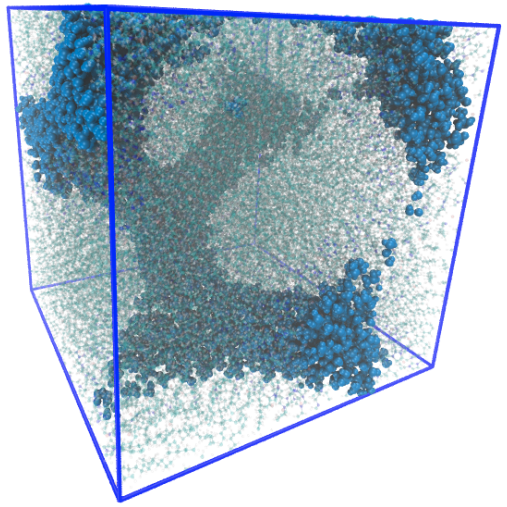
Principle 1: Look at both pictures and numbers

Example: Building a bicontinuous cubic polymer matrix

Q1_structures/Q1_pictures

Look at Q1_data.xvg and Q1_structure.gro

Does it seem like it's reasonable?



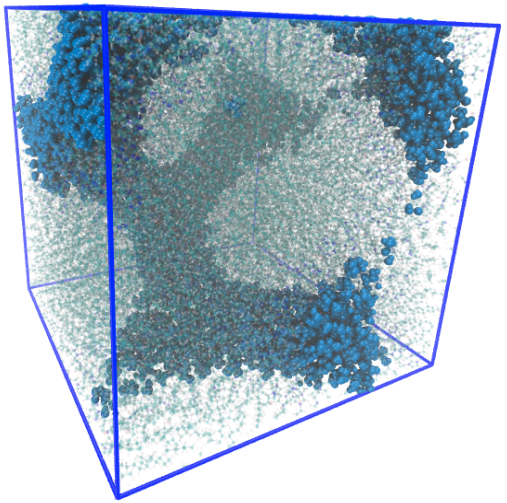
Principle 1: Look at both pictures and numbers

Example: Building a bicontinuous cubic polymer matrix

Q1_structures/Q1_numbers

Look at Q1_bad_structure.gro

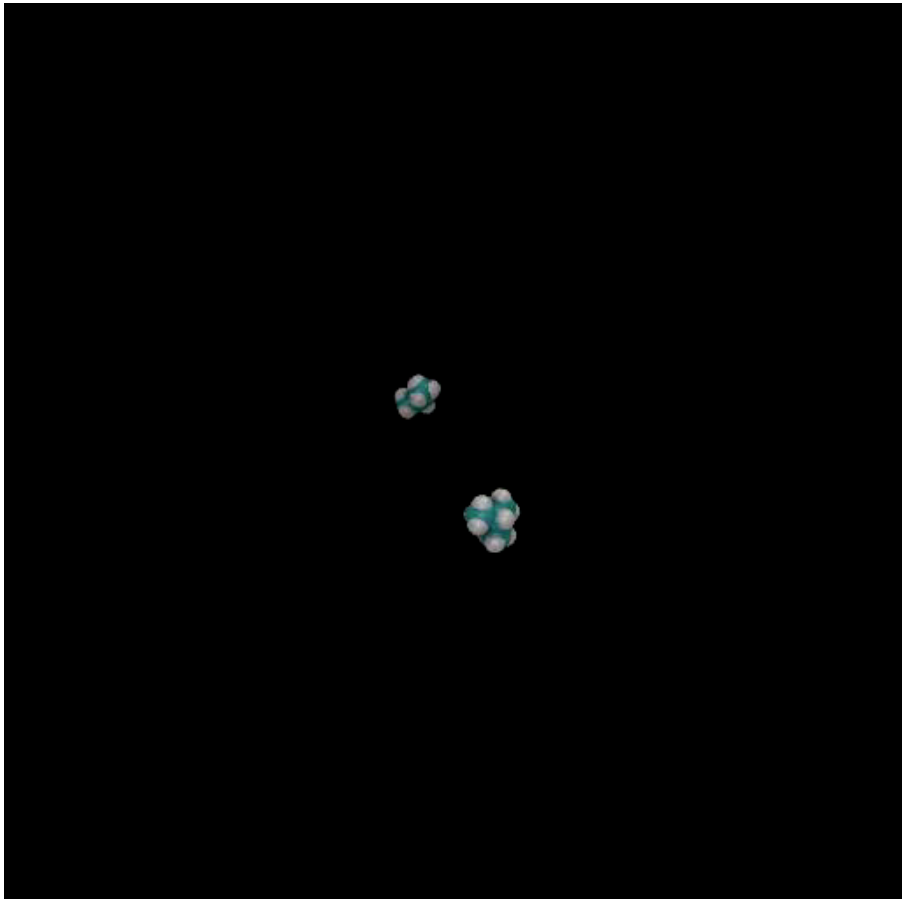
This structure crashes immediately on the first step of energy minimization. What might be wrong?



What information does the dump.txt of positions, velocities and forces give?

Pictures AND numbers: Other examples of garbage

- Protein wrapped around the box
- Any type there are decoupled degrees of freedom



- **Many** other ways to screw things up: I cannot tell you all the things whereby ye may commit sin

see bad_gas.mpg

Principle 2: Always calculate everything two ways

The man with one clock always knows what time it is.
The man with two clocks is never sure.

- The chance of bugs in any code tends to 100% as the length increases
- If you have two different ways of performing the same calculation then:
  - Those bugs are likely to be different bugs
  - You will find different violations of assumptions about the data.
- Takes longer, but faster than retracting paper 2 years later!

Principle 2: Always calculate everything two ways

- **Example**: Heat capacity
  - Two formulas:

$$C_v = \frac{dU}{dT} \approx \frac{U(T+\Delta T) - U(T-\Delta T)}{2\Delta T}$$

$$C_v = \frac{\langle (U - \langle U \rangle)^2 \rangle}{k_B T^2}$$

Principle 2: Always calculate everything two ways

$$C_v = \frac{dU}{dT} \approx \frac{U(T+\Delta T) - U(T-\Delta T)}{2\Delta T} \qquad C_v = \frac{\langle (U - \langle U \rangle)^2 \rangle}{k_B T^2}$$

go to **`heat_capacity/`**

run **`calc_cv.py`**

Two experiments with liquid isobutane
Heat capacity is calculated using the fluctuation formula?

Can you calculate the heat capacity with the finite difference approximation?

What differences do you observe?

$$C_v \approx \frac{U(T+\Delta T) - U(T)}{\Delta T}$$

3. Physics is your friend

- Physical laws that must be satisfied.
- If they are not satisfied, then you are simulating in some alternative universe that your experimental colleagues are not interested in

- Energy and momentum must be conserved
- Kinetic energy must equipartition at equilibrium
- Energies must follow the Boltzmann distribution

Is the kinetic energy Maxwell-Boltzmann distributed?

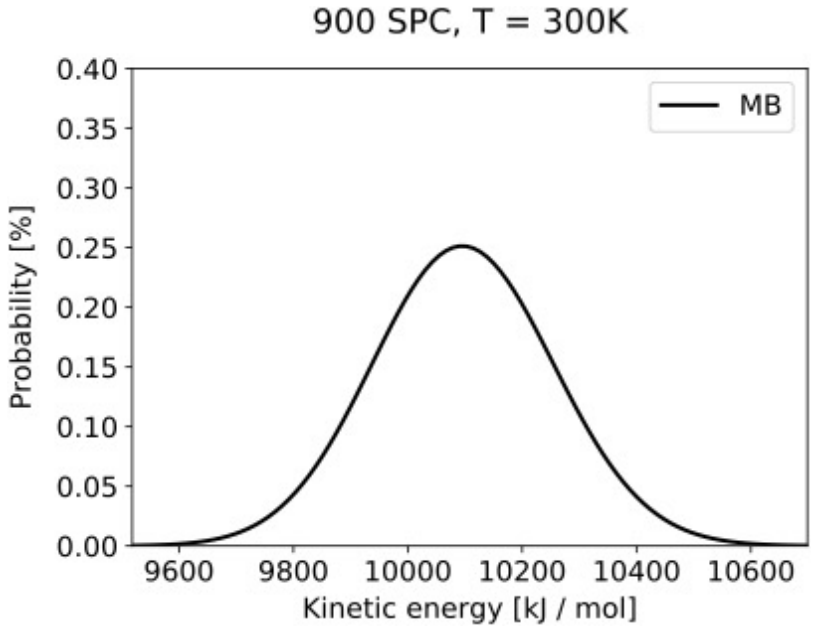
- *Expectation:* Kinetic energy K is Maxwell-Boltzmann (MB) distributed:

$$P(K) \sim K^{\frac{N-2}{2}} e^{-\beta K}$$

MB distribution is a chi-squared distribution → statistical tests available (*e.g.* Kolmogorov-Smirnov test)

- K trajectory / distribution
- Hypothesis: K is MB distributed
- α confidence interval

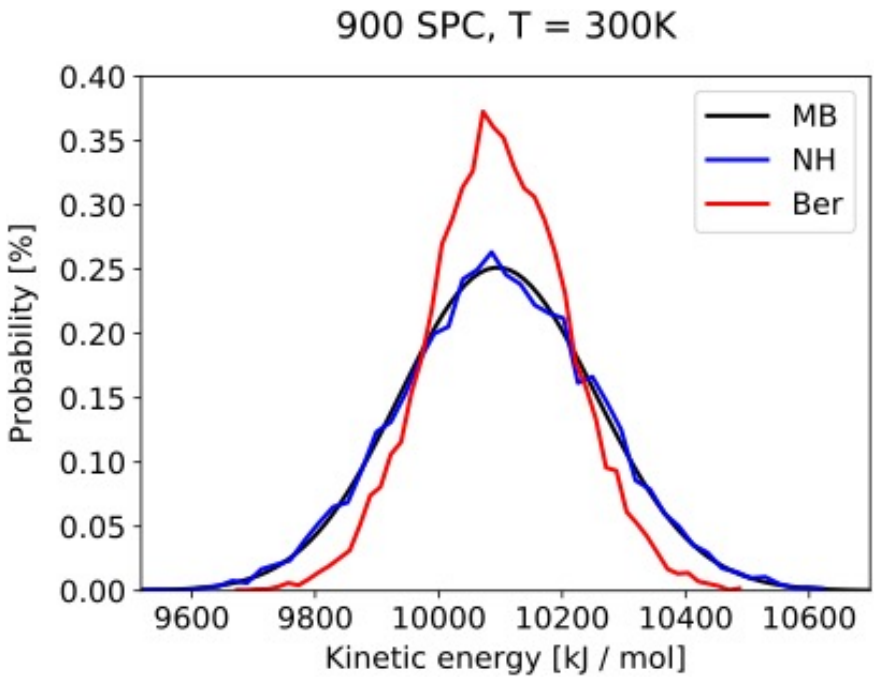
→ fast & robust assessment



Example: Water system, 2 different thermostats – which yields a MB distribution?

Nosé-Hoover:

```
Kolmogorov-Smirnov test
  result: p = 0.58
Null hypothesis: Kinetic
  energy is MB distributed
Confidence alpha = 0.05
Result: Hypothesis stands
```



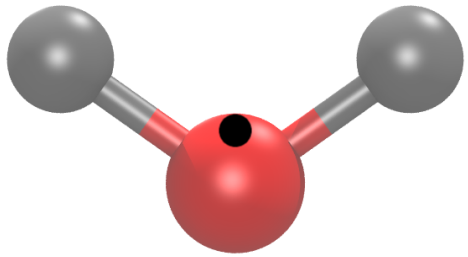
Berendsen:

```
Kolmogorov-Smirnov test
  result: p = 3.87751e-63
Null hypothesis: Kinetic
  energy is MB distributed
Confidence alpha = 0.05
Result: Hypothesis rejected
```

Does equipartition hold for the kinetic energy?

Equipartition expected – homogeneous temperature!

- Parts of the system: Functional / arbitrary divisions
  - Randomly divide molecules in groups
  - Compare solute / solvent temperatures
  - Compare temperatures of components of liquid mixture
- Sets of degrees of freedom (dof)
  Rigid body / internal dof



Jellinek & Li, Phys Rev Lett (1989)

How do I know if we're sampling from the correct distribution?

Run the same system, same options, *except* at two different temperatures

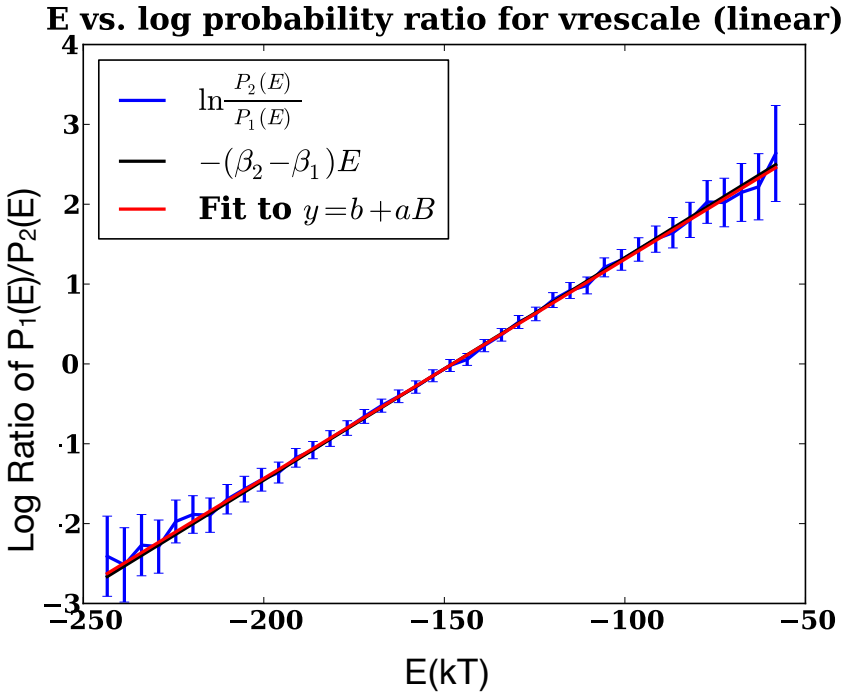
$$P_1(E) = Q_1^{-1}\Omega(E)e^{-\beta_1 E}$$

$$P_2(E) = Q_2^{-1}\Omega(E)e^{-\beta_2 E}$$

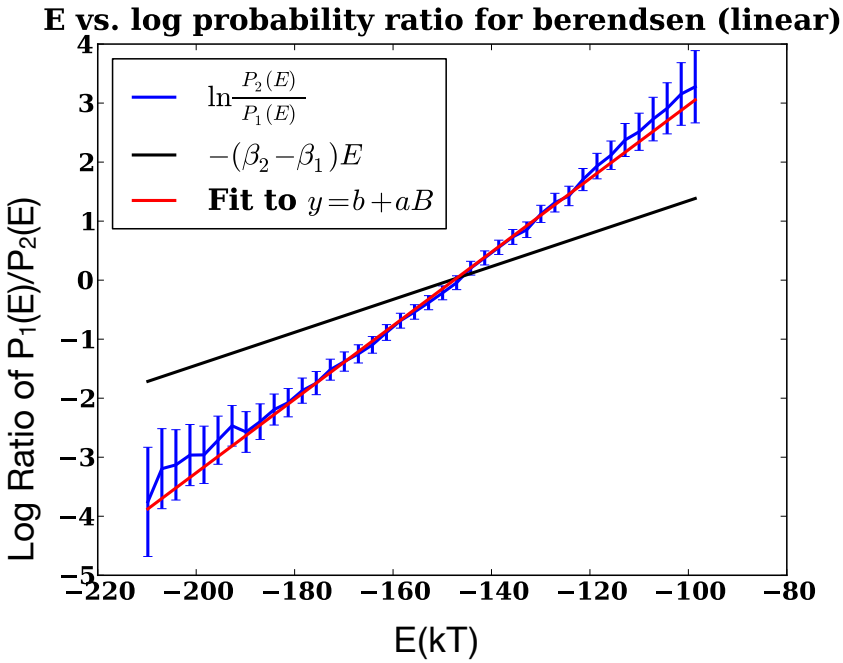
$$\frac{P_1(E)}{P_2(E)} = \frac{Q_2}{Q_1}e^{(\beta_2-\beta_1)E}$$

$$\ln\frac{P_1(E)}{P_2(E)} = \ln\frac{Q_2}{Q_1} + (\beta_2 - \beta_1)E$$

We can visually observe deviations from the correct energy distribution



β_2-β_1
0.66 σ from true value



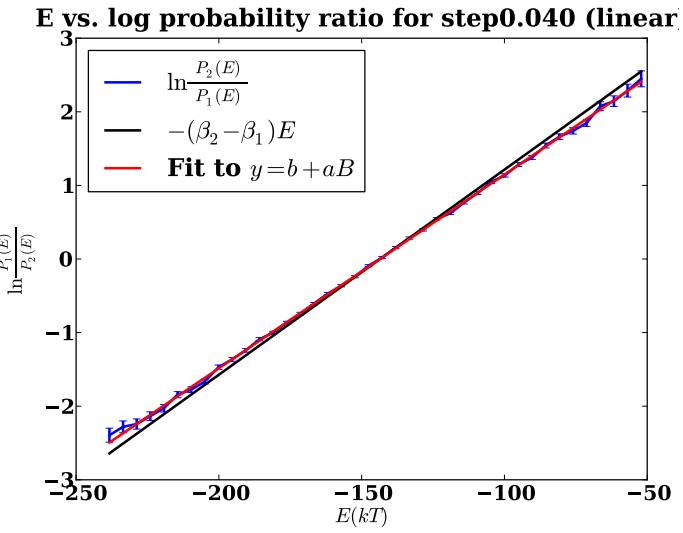
β_2-β_1
24 σ from true value

Quantiles calculated with maximum likelihood

These tests can validate simulation parameters in an automated, quantitative way

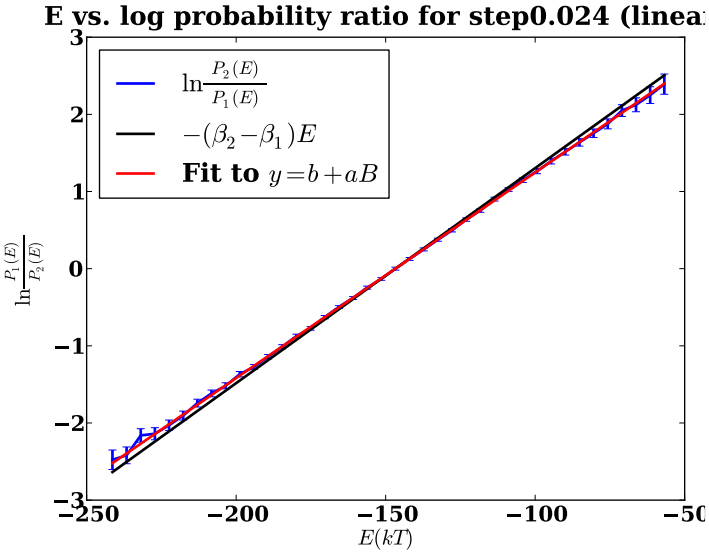
- Example: Validating the molecular dynamics time steps for argon

40 fs time step



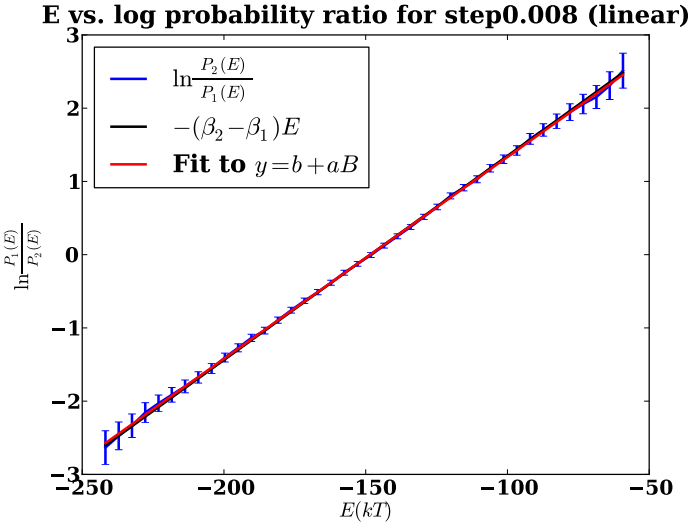
β_2-β_1: 14 σ from true

24 fs time step



β_2-β_1: 8 σ from true

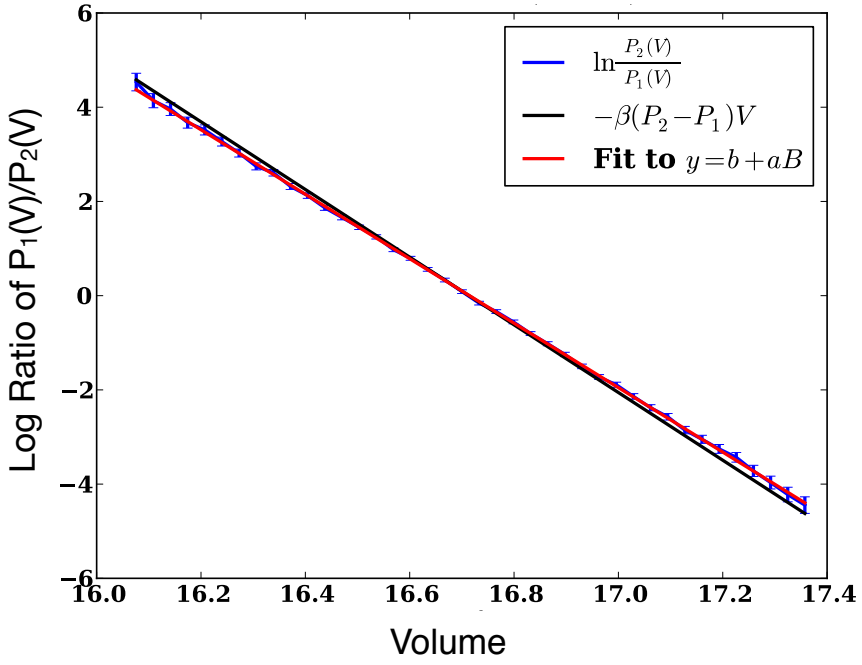
8 fs time step



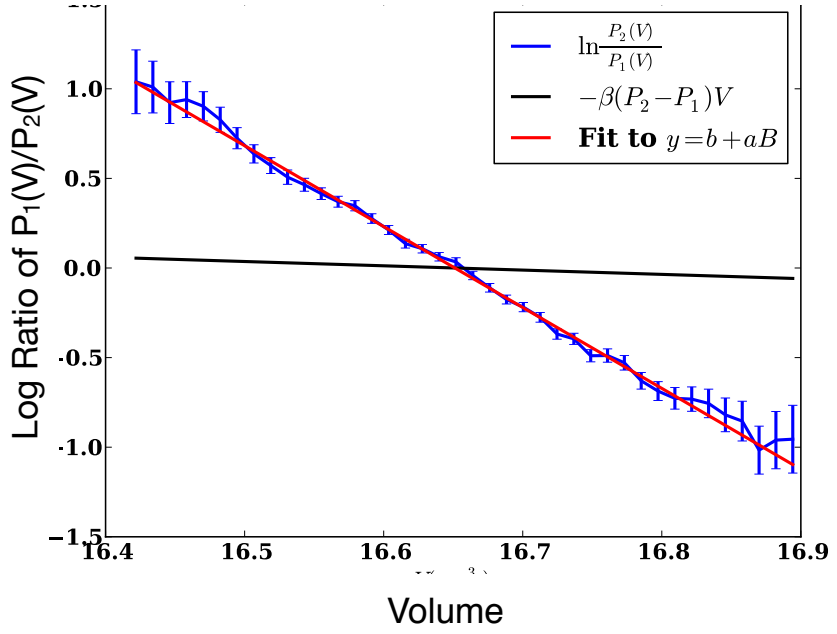
β_2-β_1: 1.0 σ from true

Validation of Volume Fluctuations in NPT

Parrinello-Rahman Pressure Control Algorithm

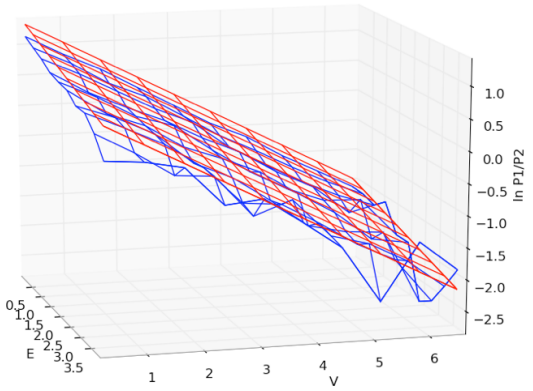


Berendsen Pressure Control Algorithm



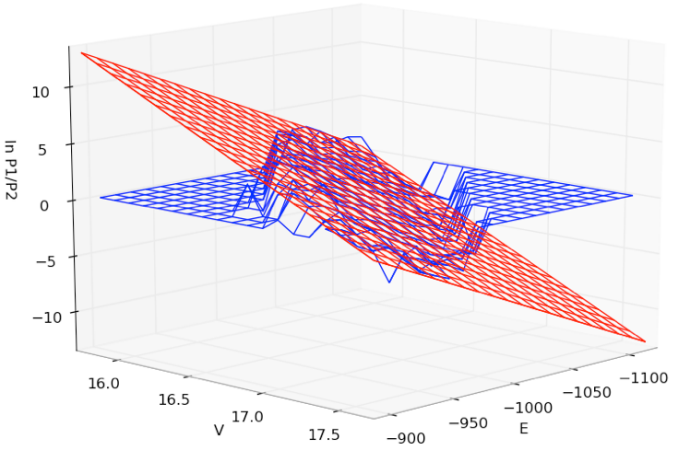
Other variations on a theme

- Applies separately to kinetic and potential energy
  - Can use for Monte Carlo algorithms as well
- NPT simulations
  - Can look at distribution of H
  - Can look at distribution of V alone
  - Can look at joint distribution of E and V
- Grand canonical simulations



Model with K(V)

Blue = Data
Red = Fit



Lennard-Jones fluid

Python module: **`physical-validation`**

shirtsgroup.github.io/physical-validation (beta version!)

Two major parts:

Tests

```
kinetic_energy.py
  mb_ensemble()
  equipartition()

ensemble.py
  check()

integrator.py
  convergence()
```

Class representing simulation results, communicates with tests

`SimulationData`

Parsers:

- Flat-text
- GROMACS
- GROMOS
- More to come!

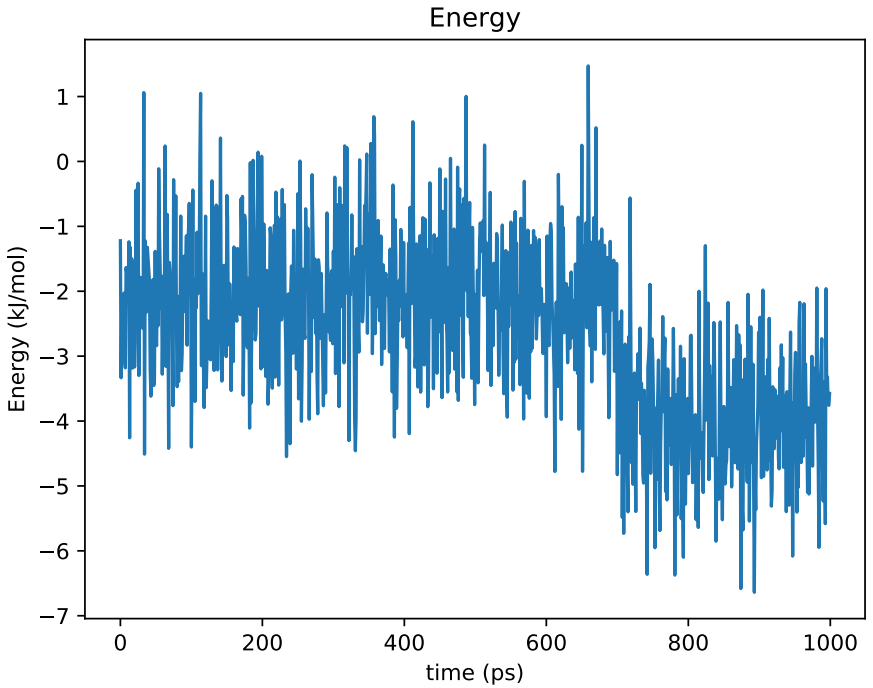
Example

- Go to **`physical_validation/flat_water`**
- Run **`ana_water.py`**
- Let's look at the output.

Utility of **`physical-validation`**

- Can identify problems with decoupled degrees of freedom.
- Can identify some problems in lack of ergodicity

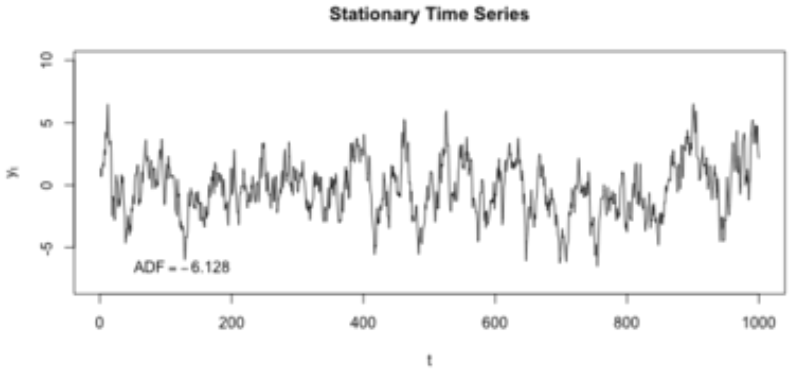
- Example: This data set will cause a failure, because the two different states have not reached equilibrium

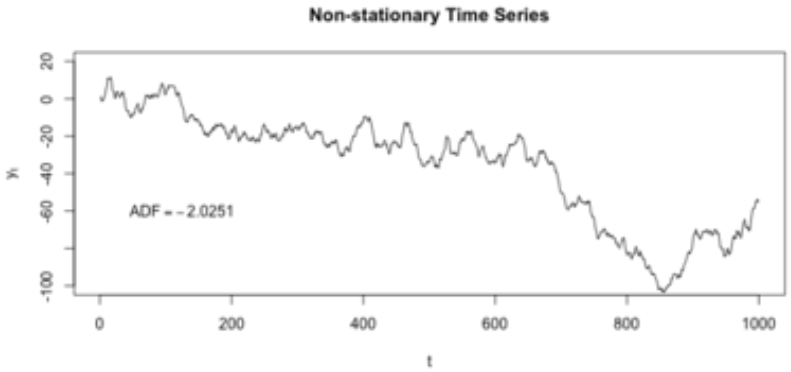


4. Statistics is your friend

- Most things we calculate are ensemble averages.
- Does your data satisfy the conditions for taking an ensemble average?

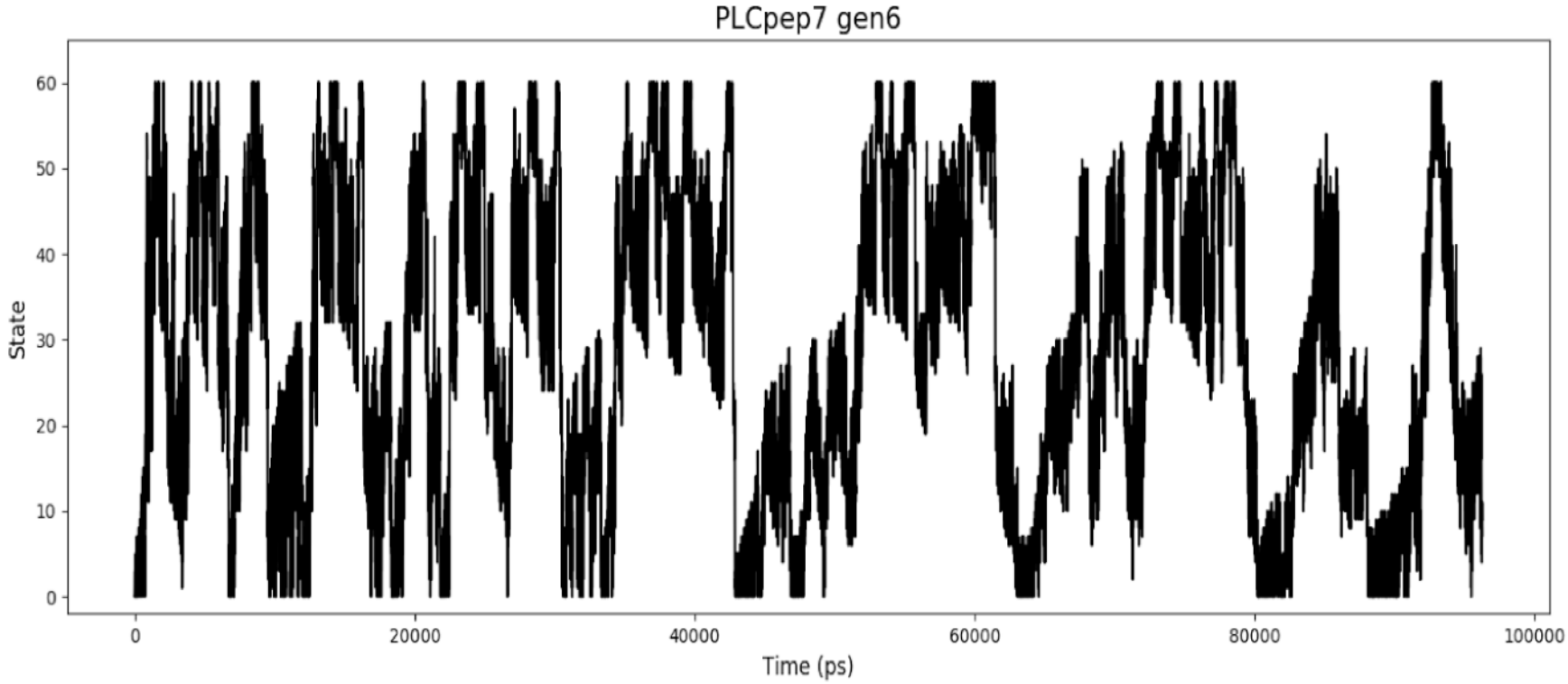
- Ergodic hypotheses is that ensemble averages are equal to time averages
- For an ensemble average to be defined, you must have a stationary time series.





What is the correlation time?

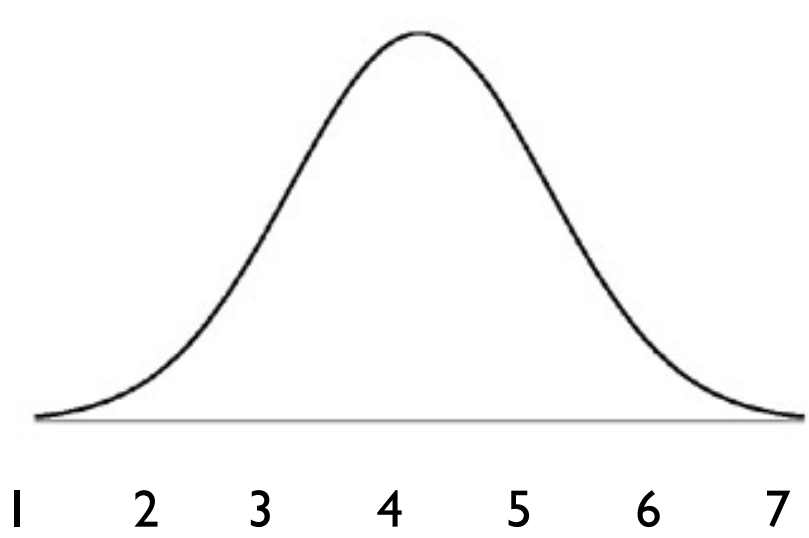
- This is a 100 ns simulation
- How many independent samples does it have?



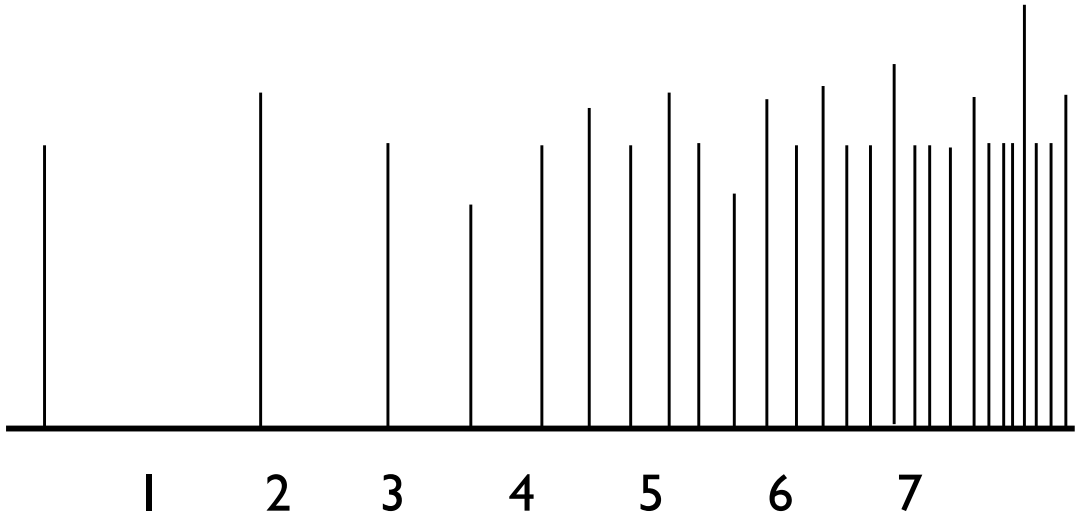
Bootstrapping a useful tool for uncertainties

- Bootstrap uncertainty is as good alternative to than running 5 times in many circumstances
- go to directory **`bootstrap/`**
- A crazy function you can't do error propagation with
- Can you compute the bootstrap standard deviation?

What's the best estimate of the distribution sampled so far?



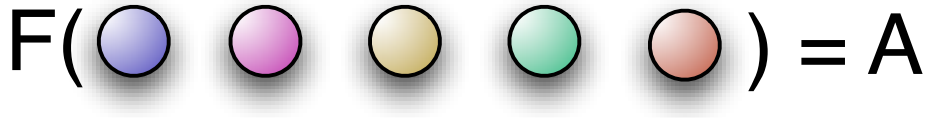
Distribution in theory

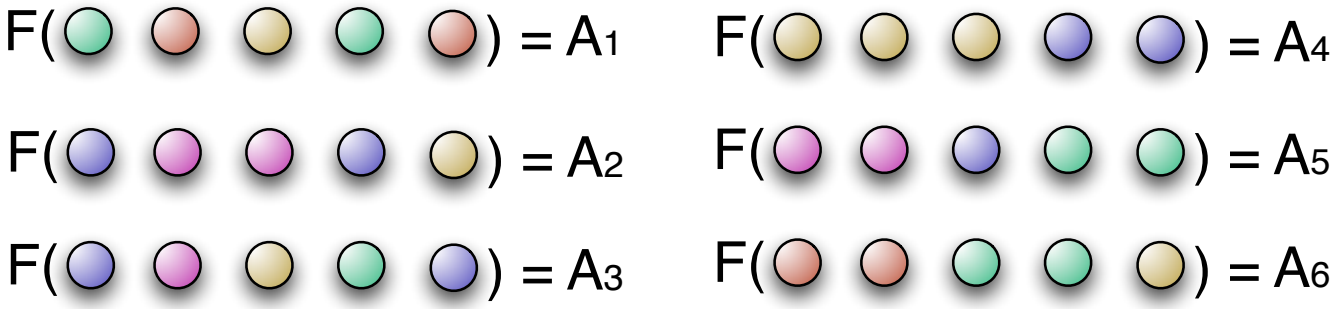


Empirical distribution
What we actually sample

The empirical distribution is our best guess as to what the real distribution is.
So let's gather independent samples from the empirical distribution

Computing the bootstrap error estimate





$Var(A)_{Bootstrap}$ = Sample variance over A_i

Generally, 200 bootstrap samples gives good results

5. Never trust a computer

- What you think you tell the computer is usually not what you tell the computer.
- **Step through the program** and make sure that it's what you think it's doing
- Especially important if automating your process.
- Learn to use your debugger and an IDE!
- Example: PyCharm

Five principles

1. You need to look at both pictures and numbers
2. Always calculate everything two different ways
3. Physics is your friend
4. Statistics is your friend
5. Never trust a computer

How to AVOID garbage

- Automate your process if possible:
  - We are very bad at paying attention to the fiddly bits.
  - Computers are good at paying attention to the fiddly bits!
  - If you automate your process, the computer will at least avoid this problems.
- HOWEVER, you run the risk of not recognizing when you make a mistake, but this time for the ENTIRE workflow.
- Lots of other ideas beyond the scope of this presentation.

Conclusions

- Apply the five principles!
- You want to make sure that your mistakes are science errors.
- Those mistakes are the interesting ones to be making, since you actually learn things.
- It is very hard to be too paranoid about the results you get out of a simulation.