Markov-based analysis of biomolecular systems pt. I

Toni Giorgino
National Research Council of Italy
toni.giorgino@cnr.it
www.giorginolab.it

@giorginolab

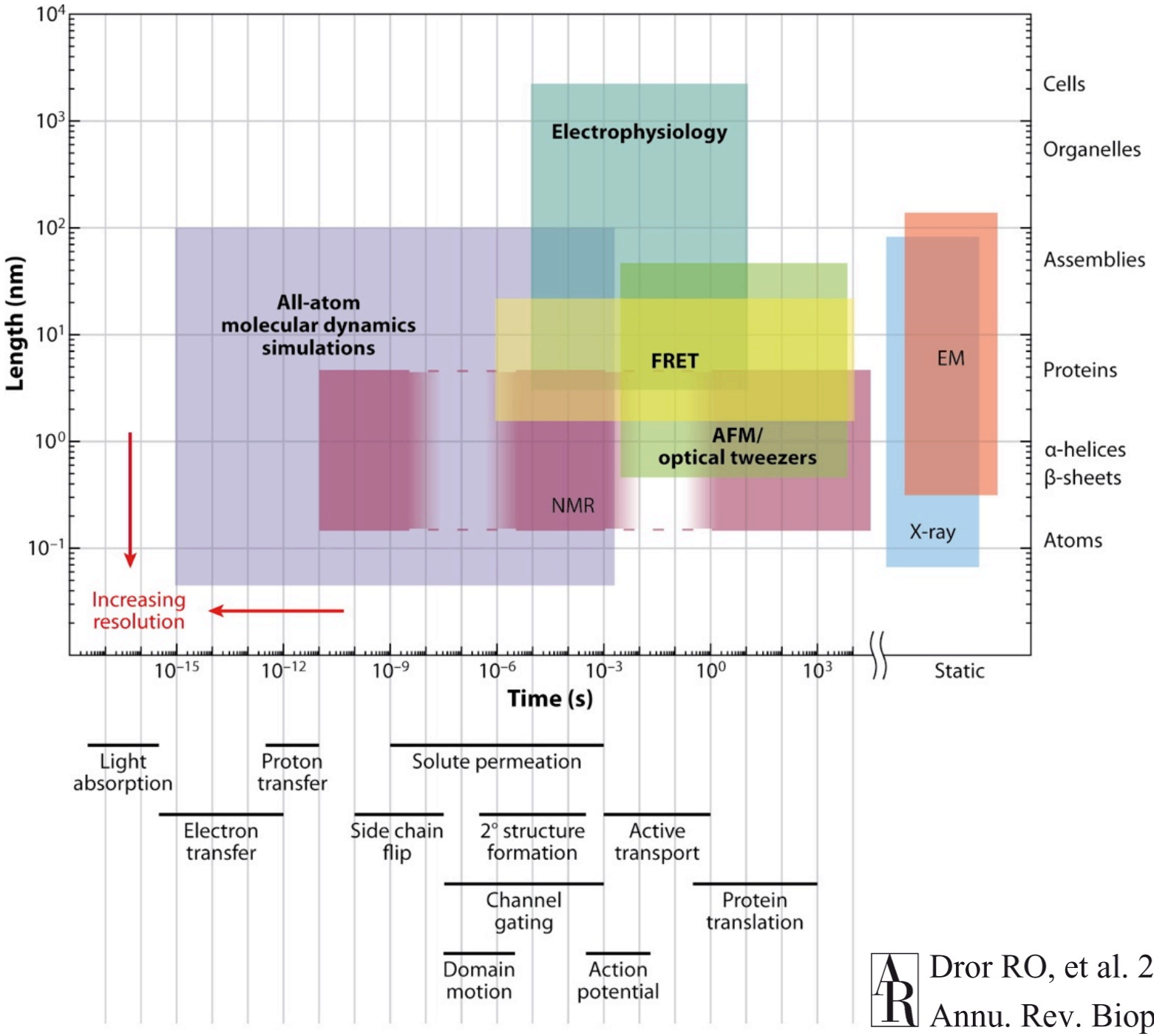
Projects available!

Master in Bioinformatics for Health Sciences
Barcelona, 26 Apr 2019

Introduction

- The aim of this class is to provide a practical *overview* of MD-based state models in computational structural biology
- Markov-state models (MSM) emerging because:
  - reconstruct *kinetic* information*, including state transition *networks*, from simulated trajectories
  - start from ***unbiased simulations*** (no *a priori* reaction coordinate hypothesis necessary)
  - microsecond-scale (high-throughput) trajectory data are becoming accessible (e.g., with GPUs)
- Success cases: *ab initio* folding, drug binding, peptide binding, …

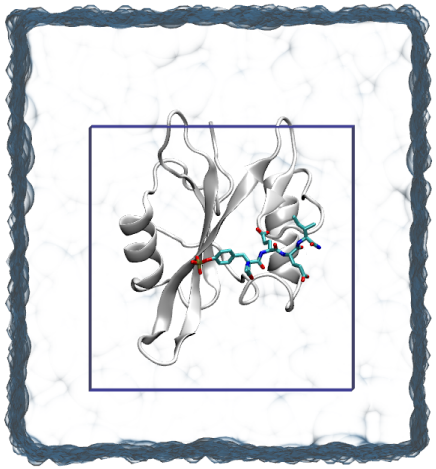
* obviously with the corresponding structures



Dror RO, et al. 2012.
Annu. Rev. Biophys. 41:429–52

Trajectory analysis and collective variables

Collective variables



$\mathbf{x}_i(t)$

C.V.

$f(t)$

- Distance
- Angle
- Energy
- Contacts
- Solvation
- …

CVs are more *meaningful* than raw coordinates. To compute them:

- VMD (mouse)
- VMD (TCL scripting)
- PLUMED (scripting+driver)
- Other libraries (Python, R)

These tools "understand" a trajectory + structure files (C α, helix…)

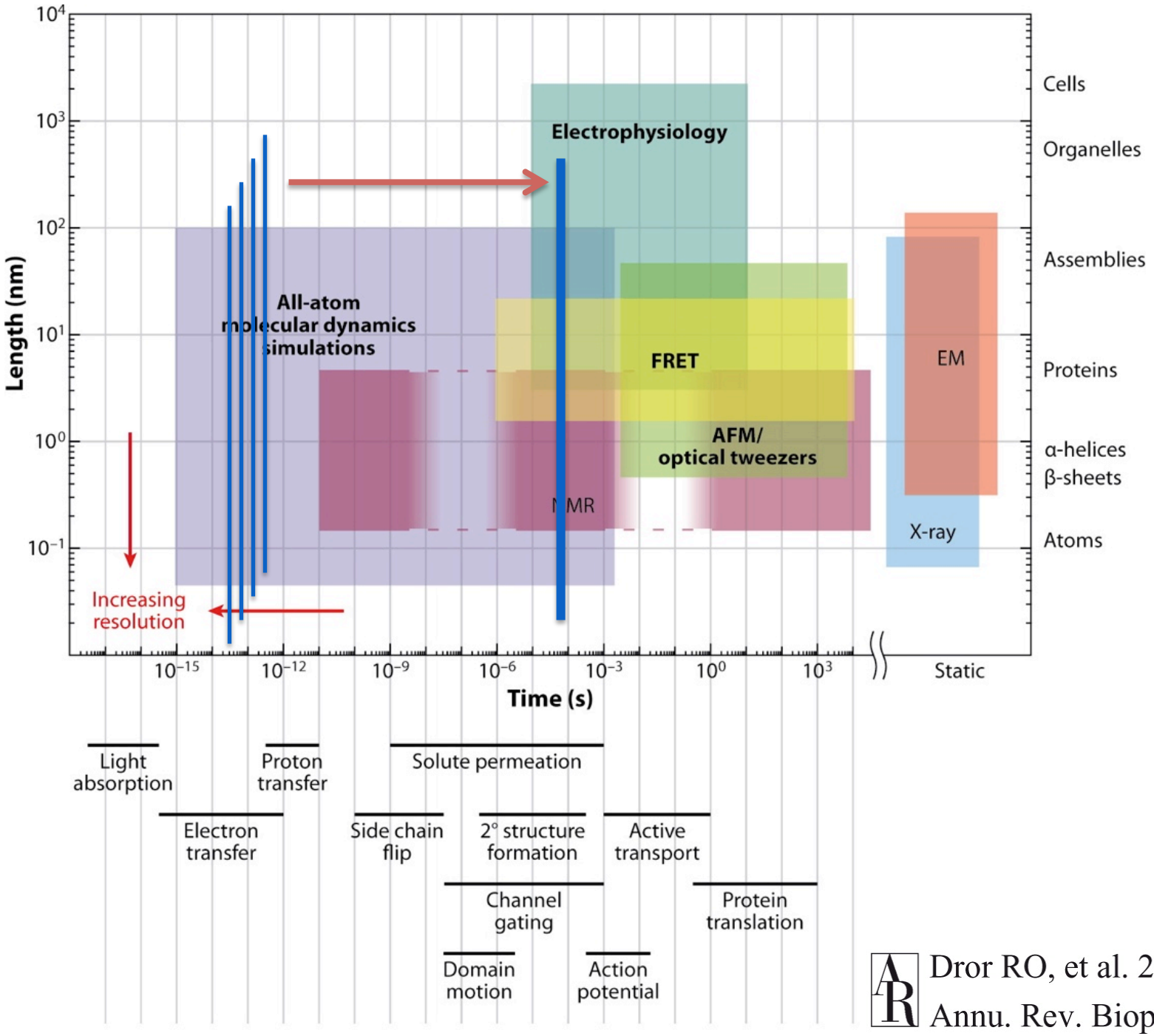
Time-series analysis of CVs

- Useful when I have a system *at equilibrium*
- I.e., it is exploring states *around* the biologically relevant structure

- Problem: cannot make deductions beyond the observed time-scales.
- TS of the phenomena of interest are often very long (barriers).
- E.g.: drug binding, conf. changes, folding, permeation, …

Motivation

- Can we use an *ensemble* of simulations to estimate dynamic behaviours which happen on timescales longer than each of the observed trajectories?
- In other words, can we leverage several "short-sighted" (non-equilibrium) observations to *extrapolate* long-time behaviour?



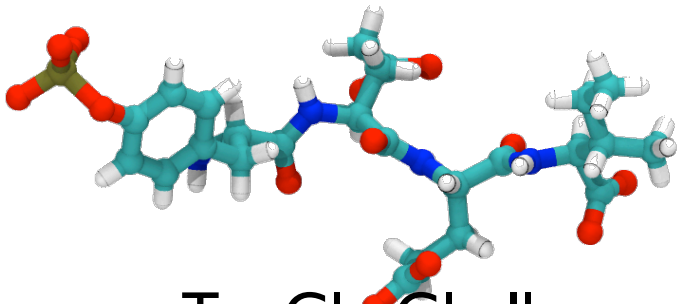
Dror RO, et al. 2012.
Annu. Rev. Biophys. 41:429–52

Basic example:
two-state model (binding)

Unbiased simulations:
SH2 – pYEEI binding example

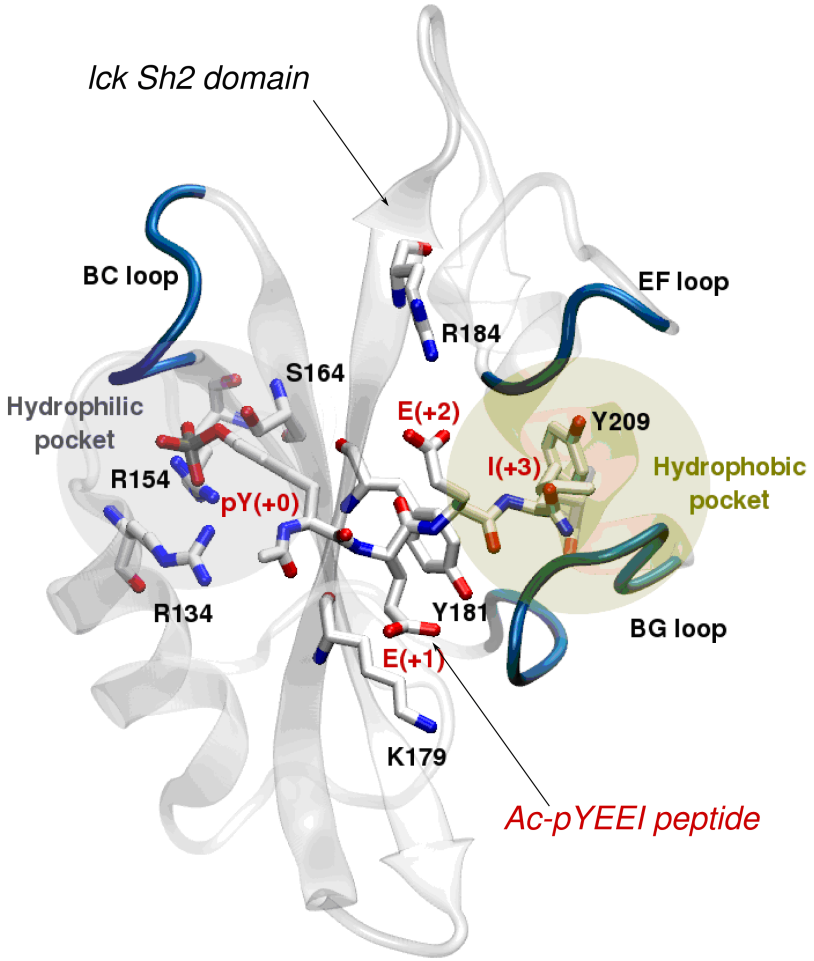
Giorgino et al., JCTC 2012

SH2 Domains

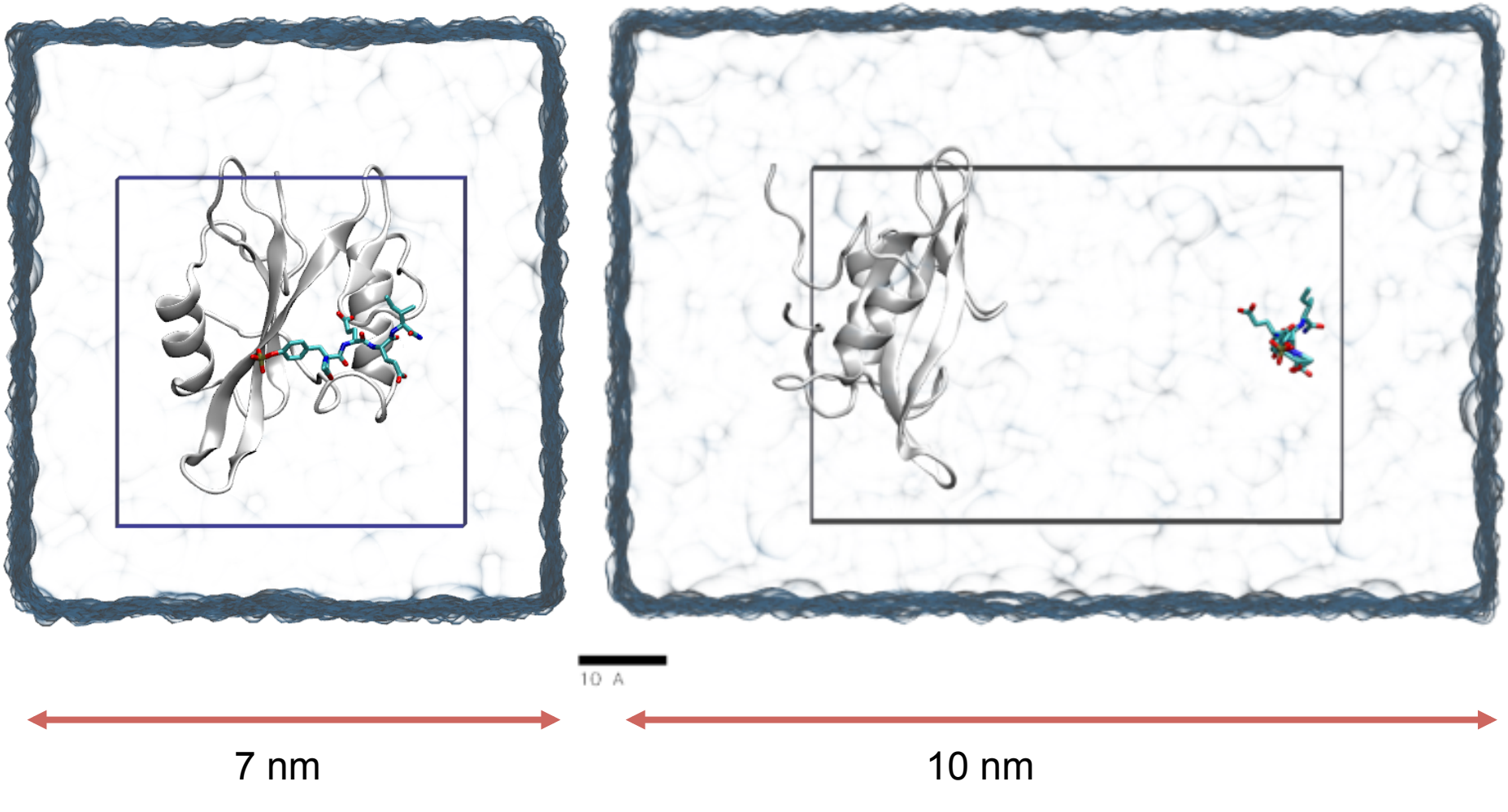


pTyr Glu Glu Ile

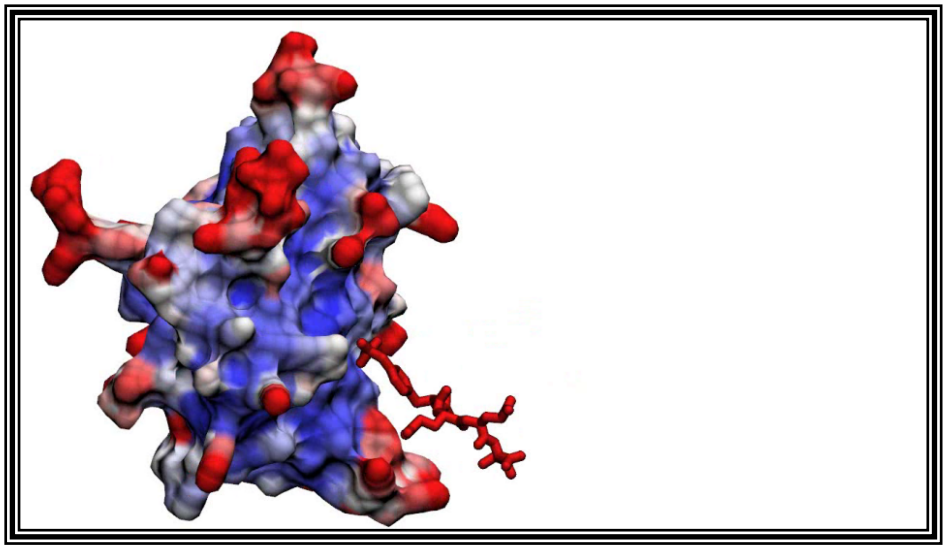
- Small domain (100 AA), well-conserved
- Recognize short pY-* sequences with high specificity
- Prominent role in signaling
- Found in ~110 human proteins
- "Socket with two holes"



In silico set-up



Teaser (1:36)



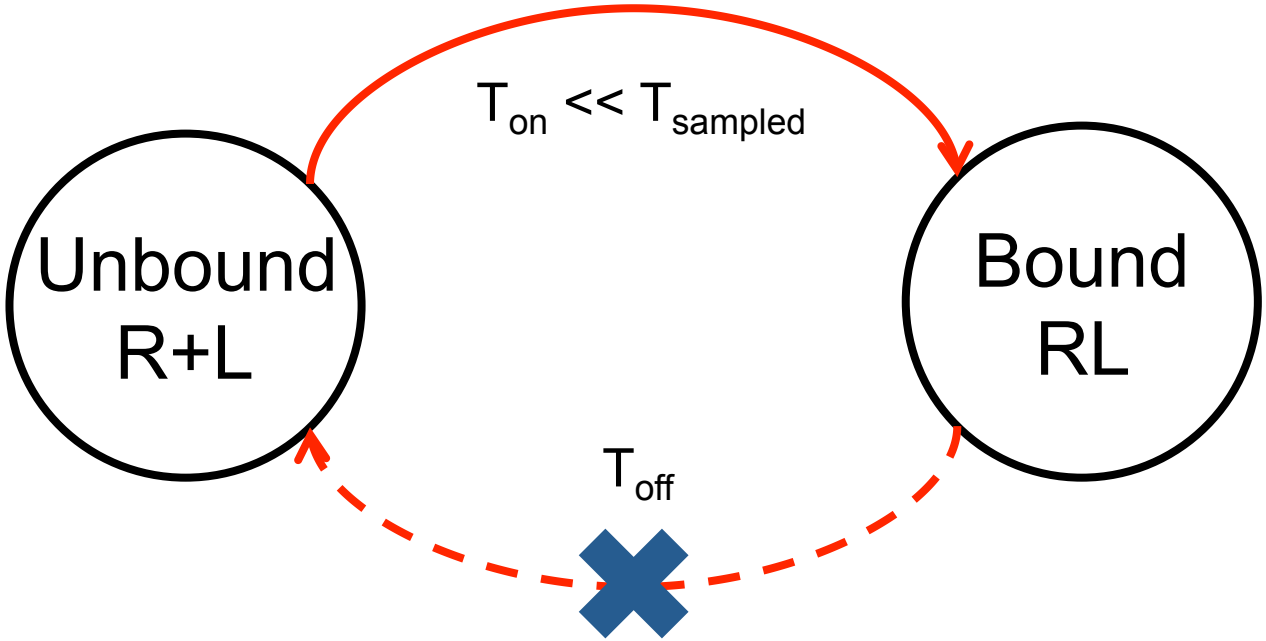
R + L → RL

http://goo.gl/ZLI42
also https://youtu.be/Xhof-pf_Eo8

- *Video: unbiased binding simulation of pYEEI to Lck SH2 domain.*
- *Protein surface and ligand's atoms color-coded by local flexibility, from blue to red (lowest to highest RMSF).*
- *As the peptide binds into its native conformation (RMSD < 2 Å from crystal structure), it loses flexibility; the same happens to the protein.*
- ***By construction, we are not at equilibrium.***

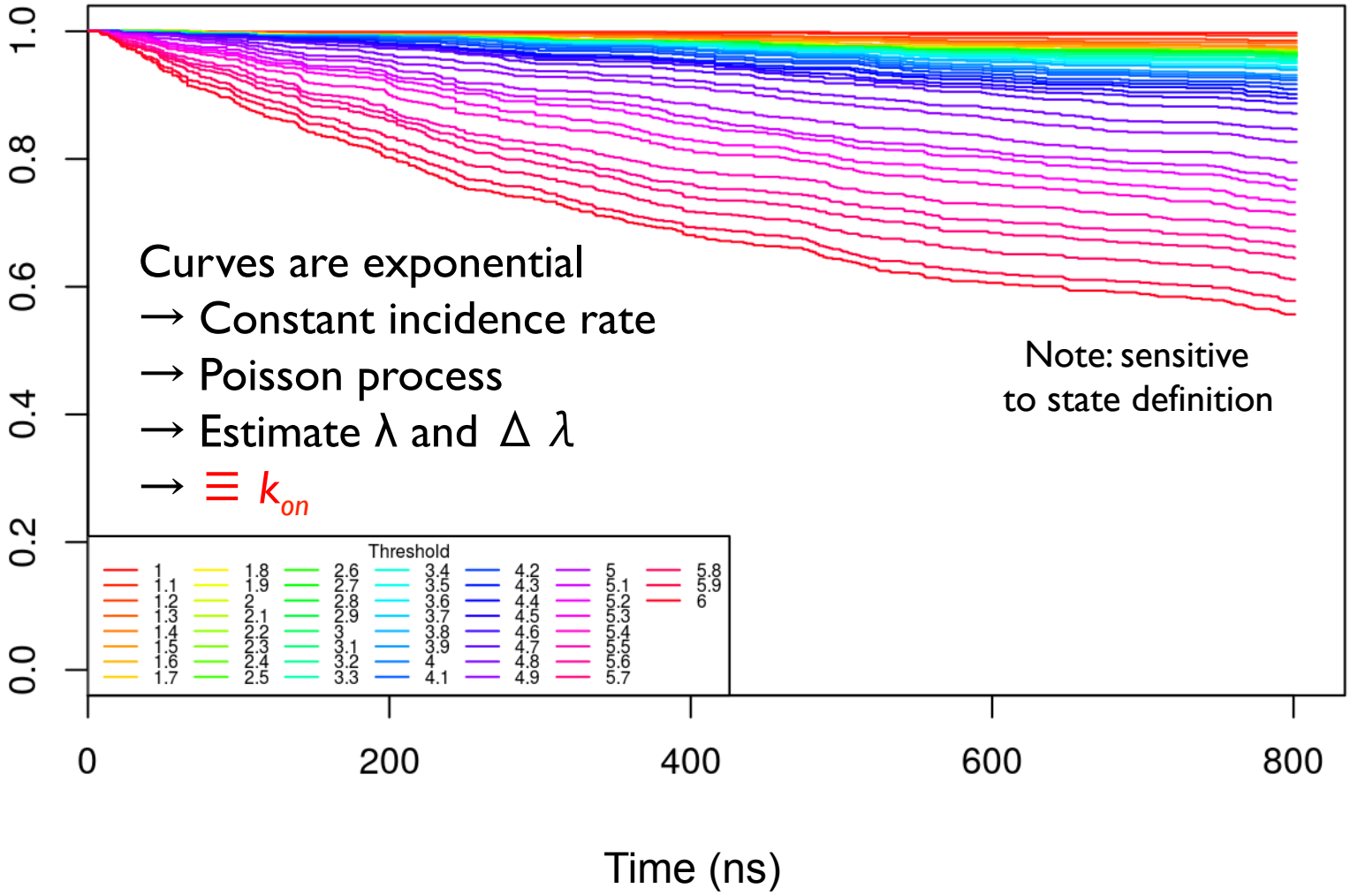
Binding as an *irreversible* two-state model

On the sampled timescales…



$$T_{event} << T_{on} << T_{sampled} << T_{off}$$

"Out of ~1000 equivalent trajectories, how many reached the bound state, and when?"



Let's look for a more general solution

This 2-state method is not sufficient:

1. One needs to define the *bound state*
   – which is not available if we lack
   a crystallographic or NMR structure.
2. We need to observe a sizeable number of *full* binding events.
   – Association processes are slow.
   They may be (far) outside our sampling power.
3. We get no information about *off* rates.
4. We get no information about *metastable states*.

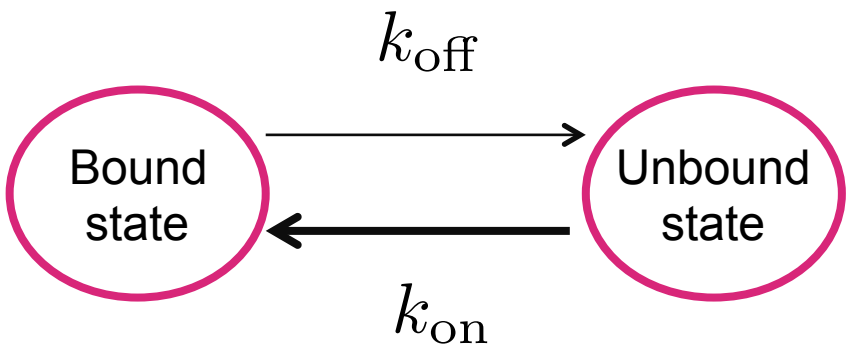
Going multi-state

Kinetics

We can use the *dynamic* information, computing the *kinetics* as well. This is why we are doing *unbiased dynamics* in the first place!

Binding kinetics is important for rational drug design *as well* (e.g. related to the time spent in complex with receptors).

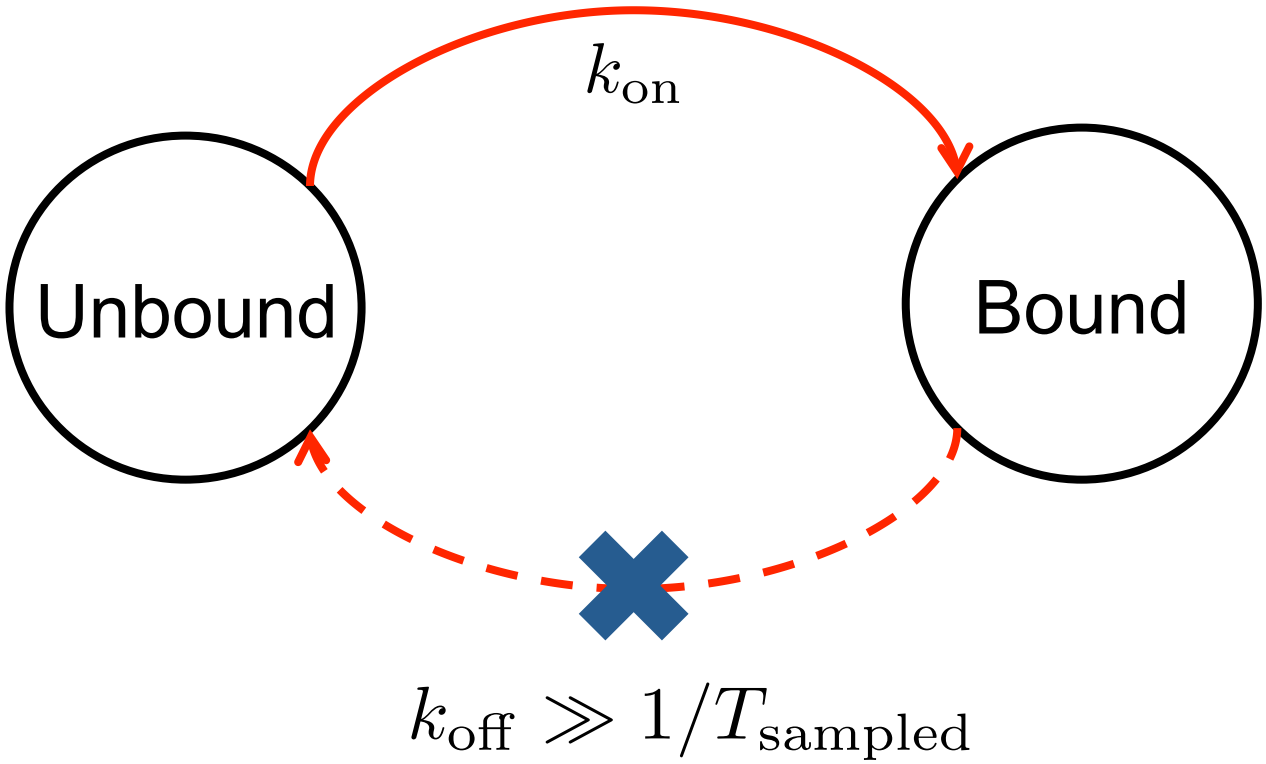
$$K_D = k_{\text{off}}/k_{\text{on}}$$
$$\propto \tau_{\text{on}}/\tau_{\text{off}}$$
$$\sim t_U^{(B)}/t_B^{(U)}$$



Swinney, D. C. Curr. Opin. Drug Discov. Devel. 12, 31–39 (2009).

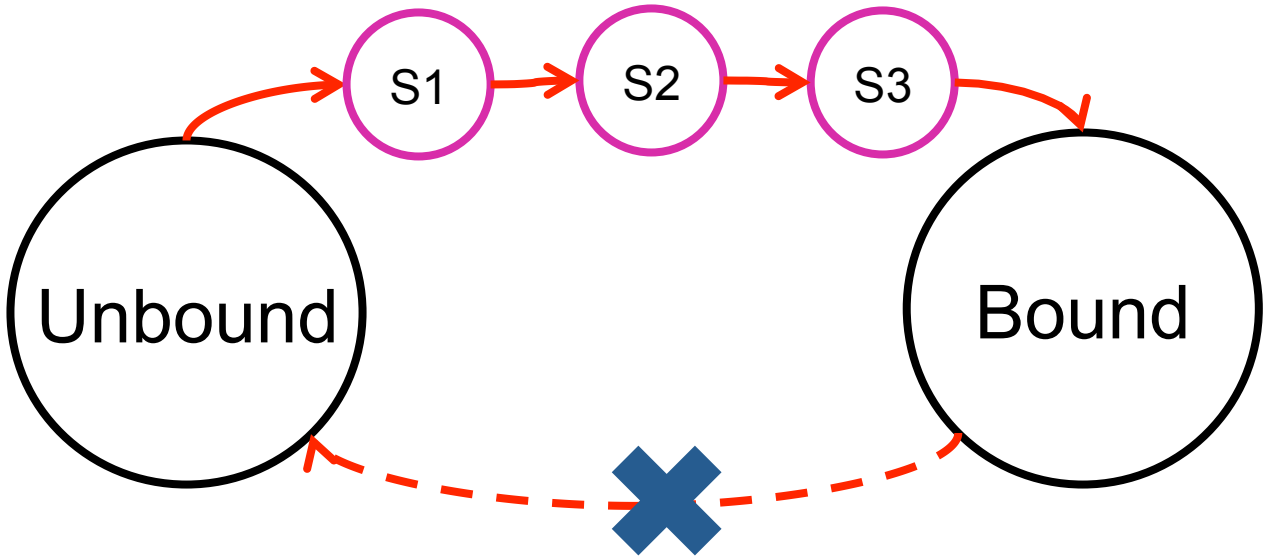
Let's move beyond this model…

On the sampled timescales…



Idea

- We introduce several *intermediate steps*

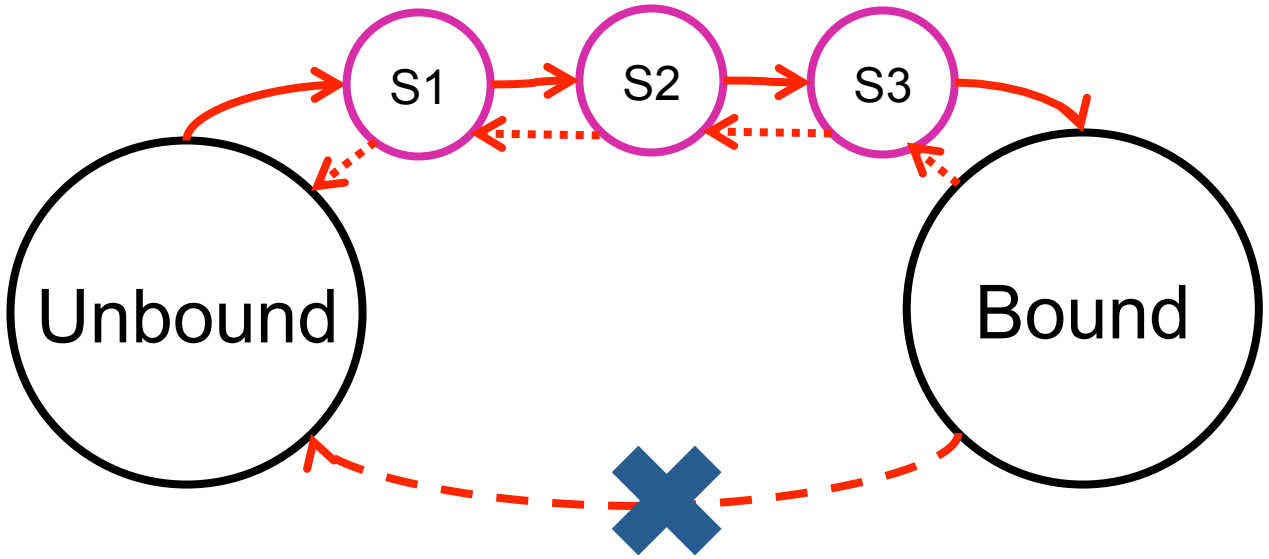


k_{on} reconstructed combining rates of different intermediate steps

Buch et al., PNAS 2012

Idea

- We introduce several *intermediate steps*
- Compute MFPT



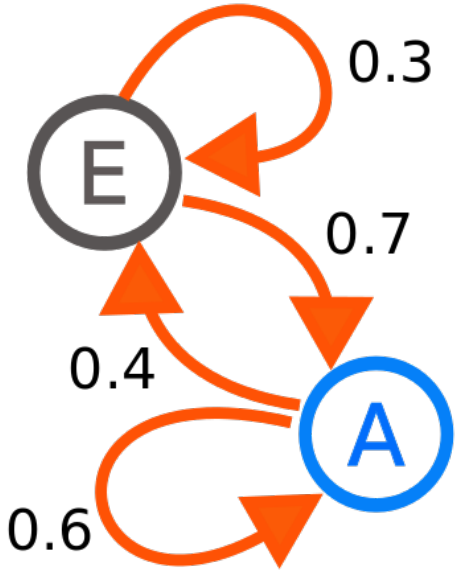
k_{on} reconstructed combining rates of different intermediate steps

k_{off} may also be reconstructed, if states are fine-grained!

Discrete-time Markov chains

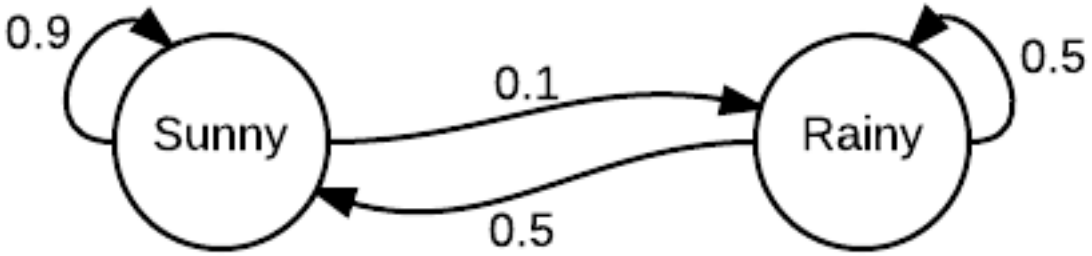
Andrei Markov
1856-1922

Discrete Time Markov Chains



- A **random** process.
- The system's state is a **discrete** variable.
- It undergoes transitions between states at uniformly-spaced (**discrete**) time points.
- Transition probabilities do not depend on the previous history of states (**memorylessness**).

First example



$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$

Assume a deterministic initial condition:

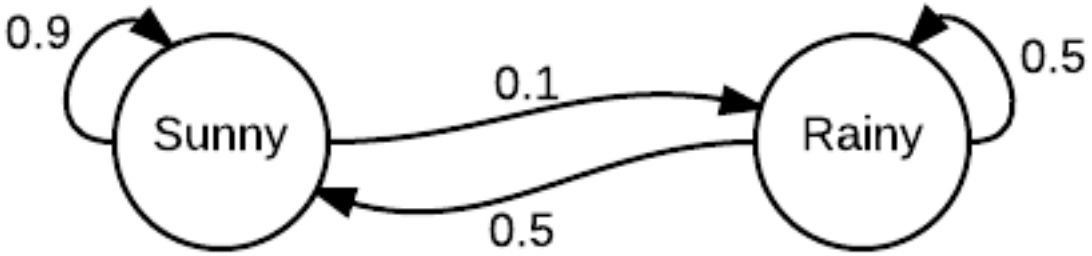
X_0 = Sunny with certainty; *i.e.*,

p(Sunny | t=0)=1 and
p(Rainy | t=0)=0 *i.e.*

$s_0 = [1, 0]$

…now, what is s_1?

First example



$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$

What is s_1?

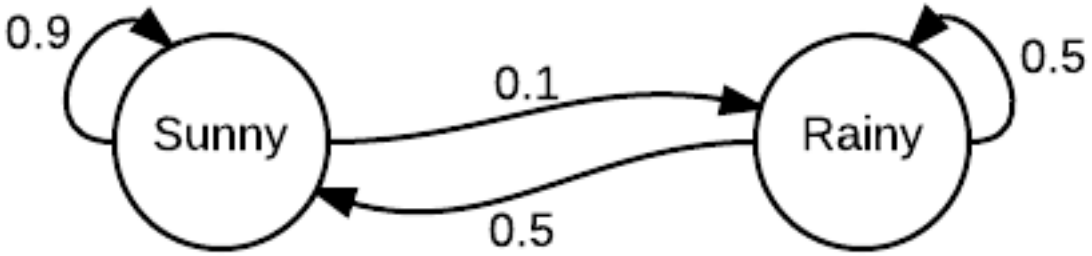
s_1 { p(Sunny | t=1) = 0.9
p(Rain | t=1) = 0.1

In matrix form…

$$\mathbf{s}_1 = \mathbf{s}_0 P$$

…now, what is s_2?

First example



$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$

What is s_2?

s_2 { p(Sunny | t=2) = 0.9 p(Sunny | t=1) + 0.5 p(Rainy | t=1) = 0.86
p(Rainy | t=2) = 0.1 p(Sunny | t=1) + 0.5 p(Rainy | t=1) = 0.14

In matrix form…

$$\mathbf{s}_2 = \mathbf{s}_1 P$$

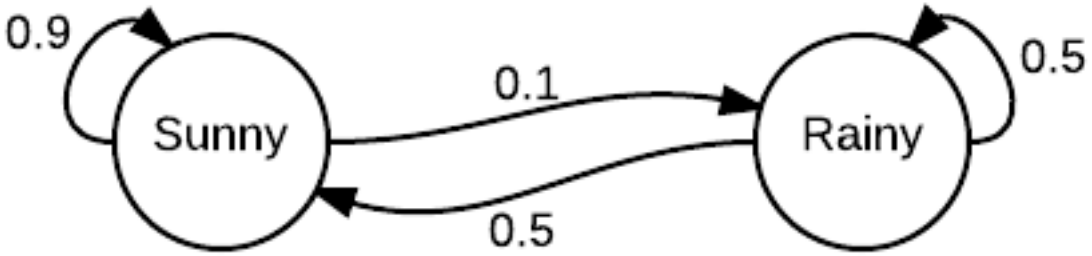
And in general…

$$\mathbf{s}_{t+1} = \mathbf{s}_t P$$

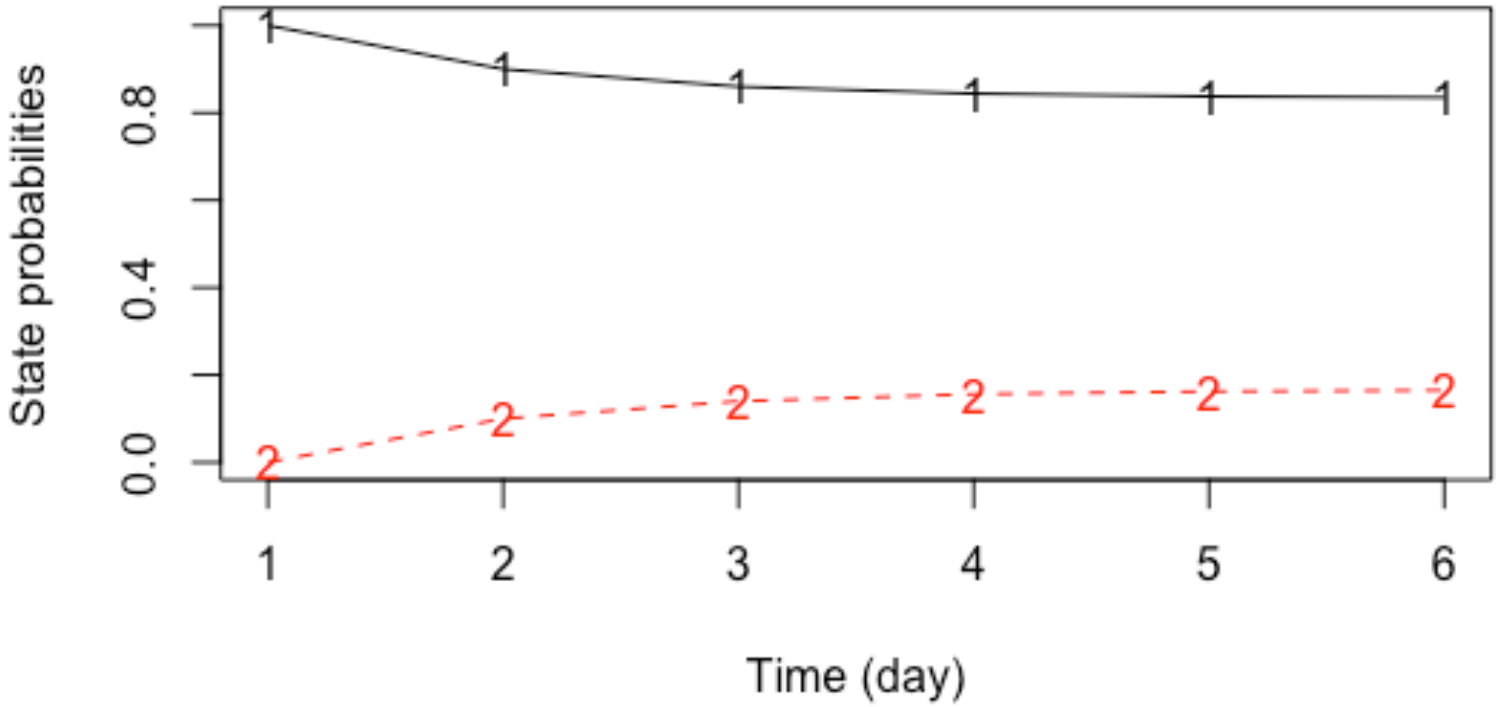
Meaning…

$$\mathbf{s}_t = \mathbf{s}_0 P^t$$

Let's do a numerical test…

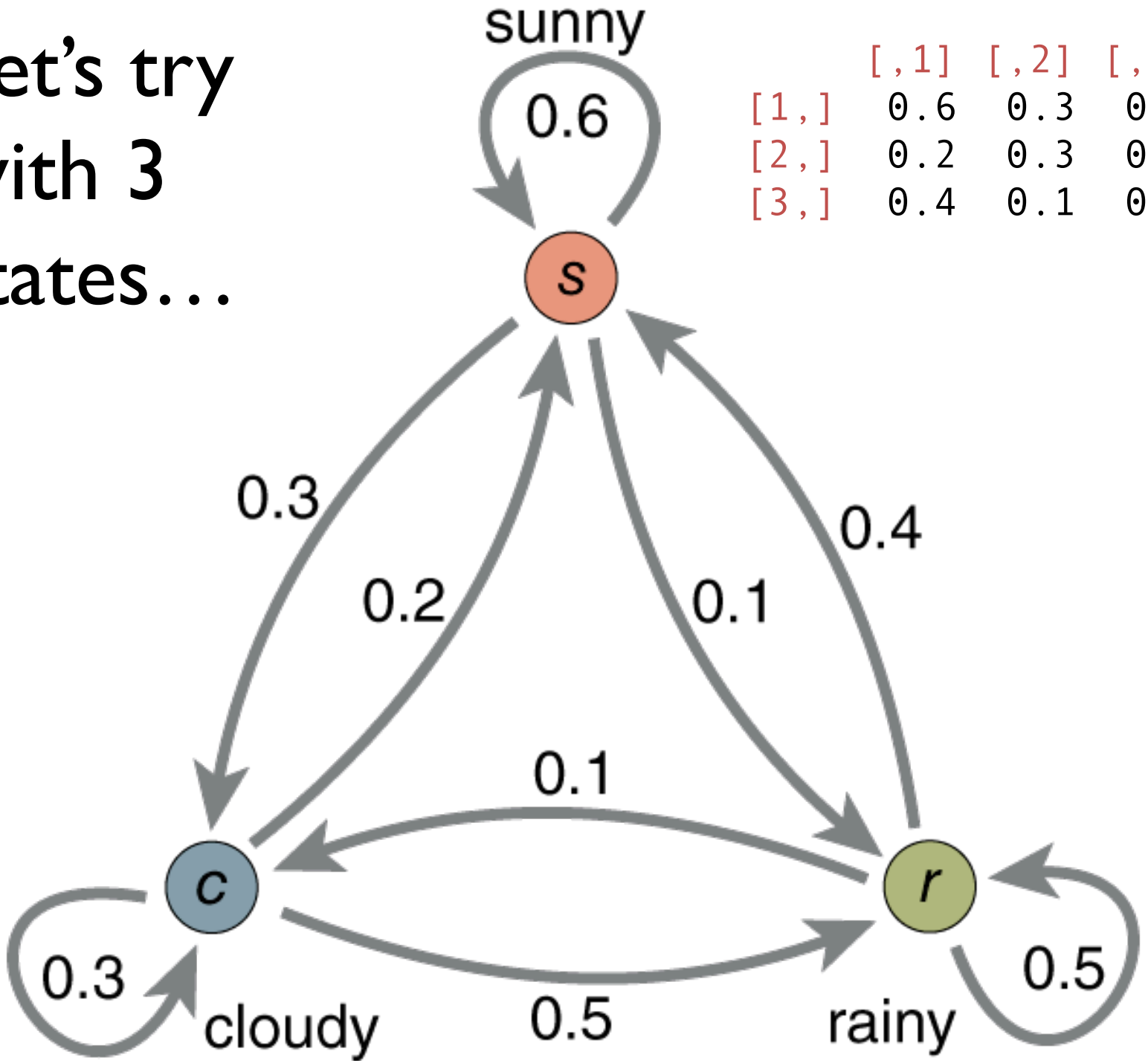


$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$



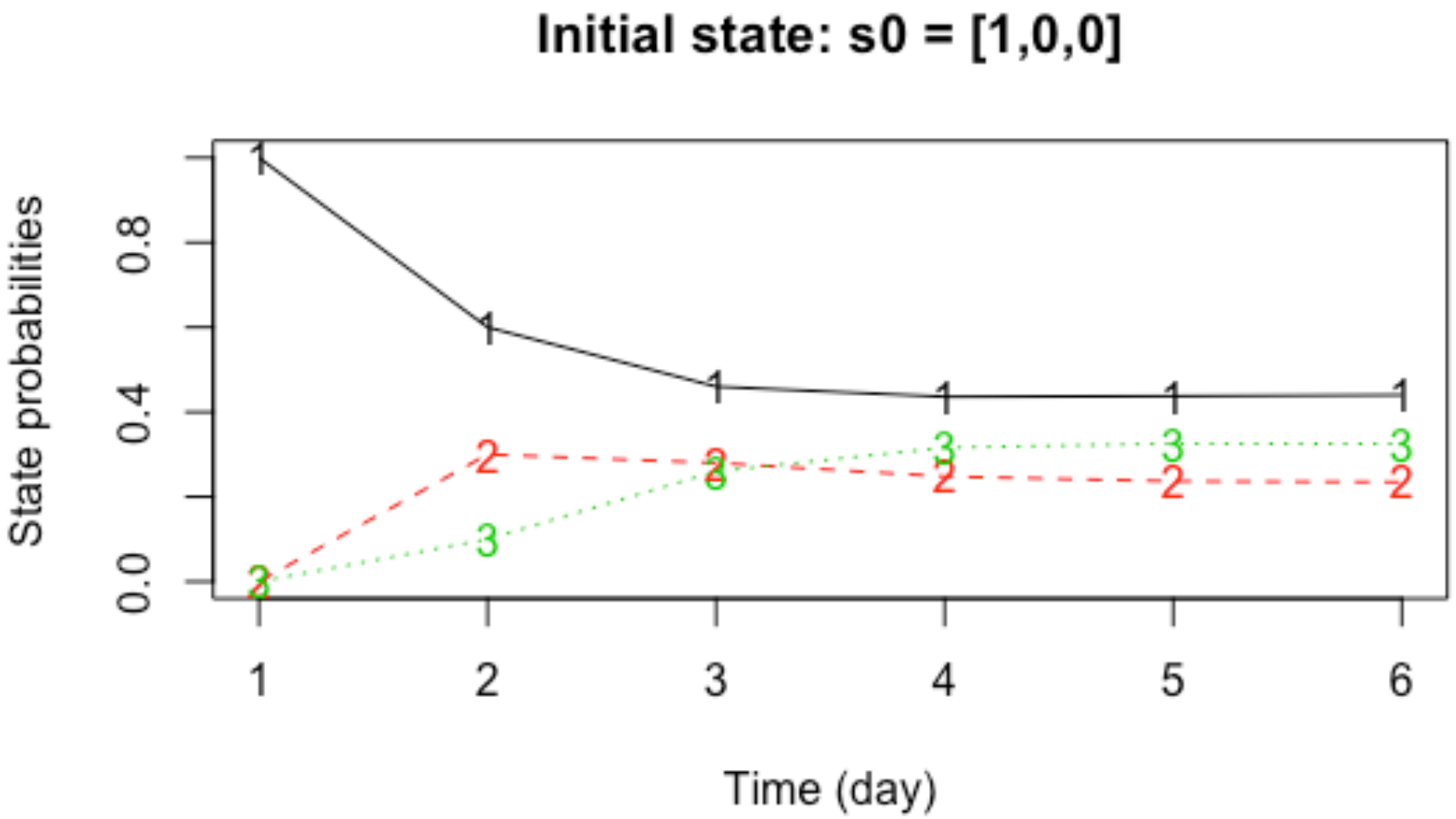
Let's try with 3 states…

```
     [,1] [,2] [,3]
[1,]  0.6  0.3  0.1
[2,]  0.2  0.3  0.5
[3,]  0.4  0.1  0.5
```



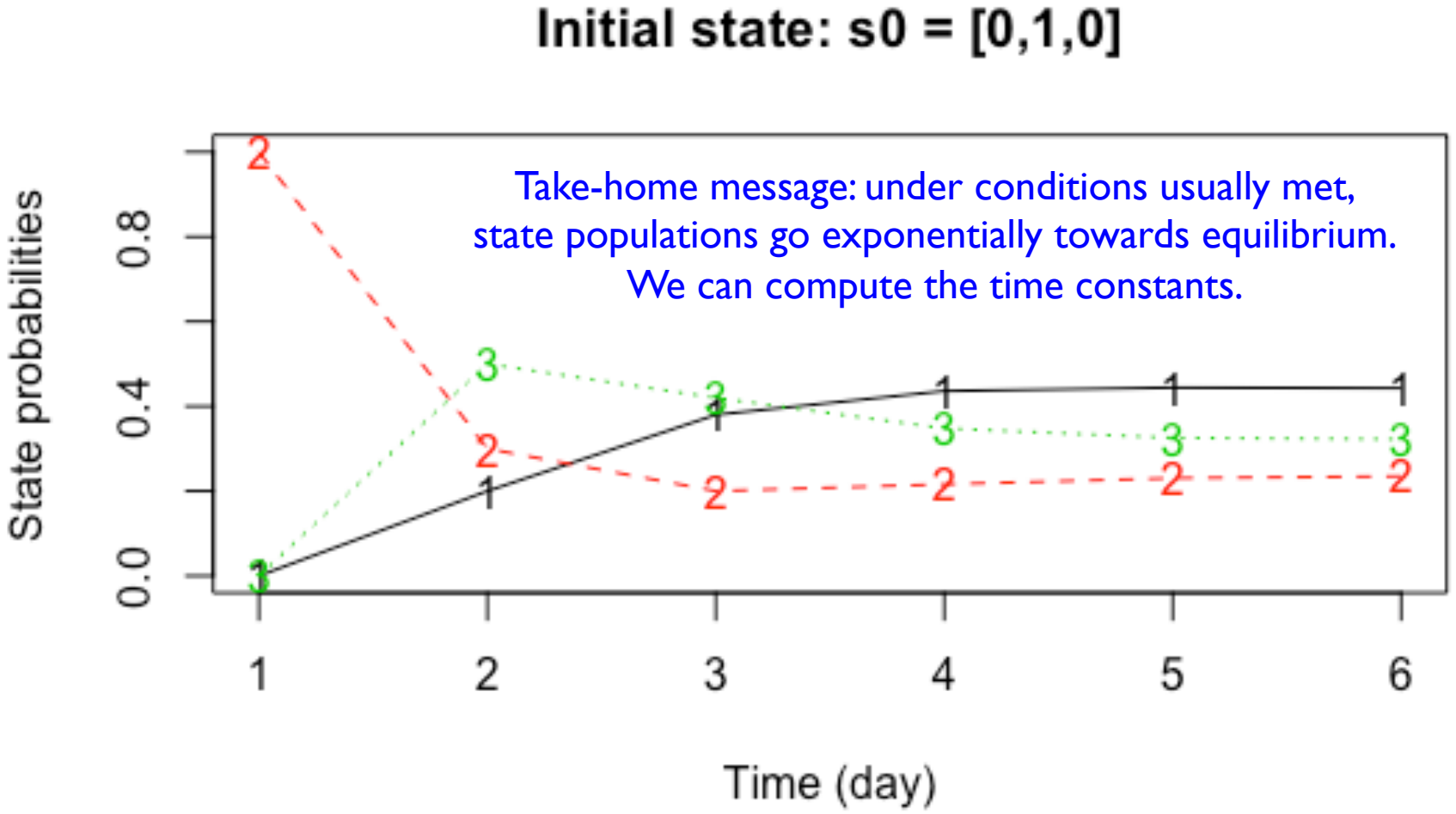
Starting from *Sunny*…


```
     [,1] [,2] [,3]
[1,]  0.6  0.3  0.1
[2,]  0.2  0.3  0.5
[3,]  0.4  0.1  0.5
```



Starting from *Cloudy*…


```
     [,1] [,2] [,3]
[1,]  0.6  0.3  0.1
[2,]  0.2  0.3  0.5
[3,]  0.4  0.1  0.5
```

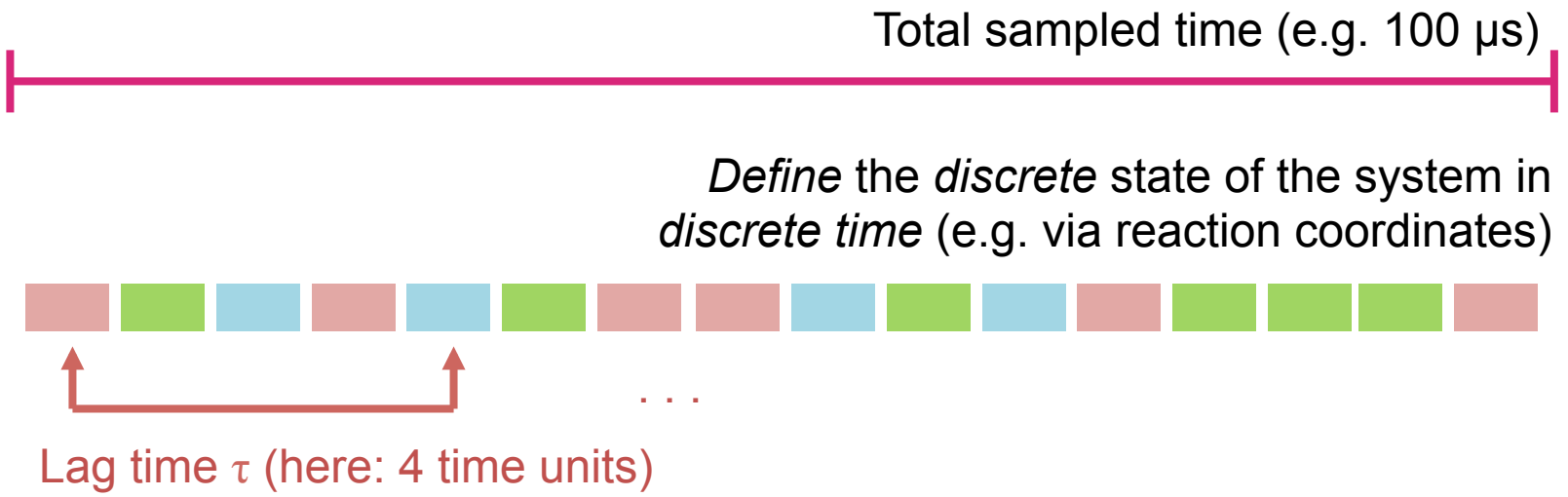


Important quantities we can compute

- Stationary distribution (→ eq. probabilities)
- Relaxation times
- Mean-first passage times (→ kinetic rates)
- And others we won't discuss:
  - Committor probabilities
  - Fluxes
  - …

Learning matrices from trajectories (of discrete states)

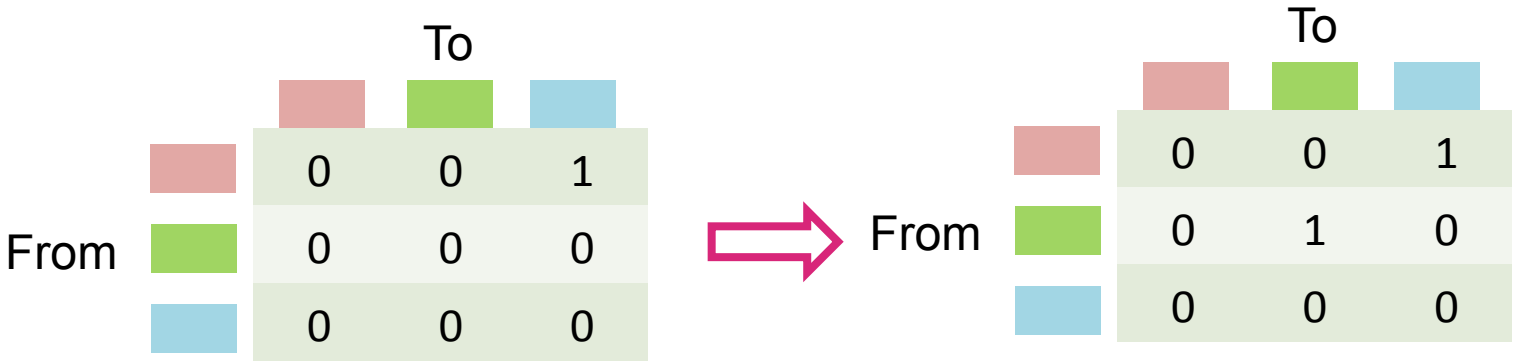
Markov models



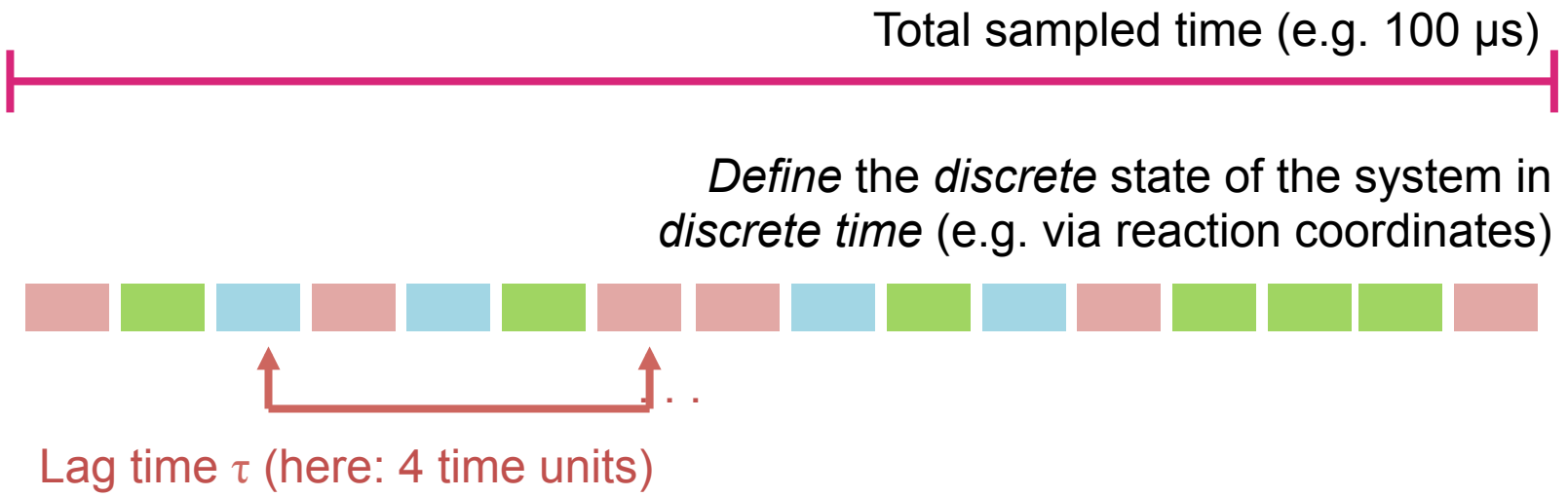
Markov models

Define the *discrete* state of the system in *discrete time* (e.g. via reaction coordinates)

Compute the transition probability matrix sliding a window of lag time τ



Markov models



Compute the transition probability matrix sliding a window of lag time τ

From \ To			
	0	0	1
	0	1	0
	0	0	0

From \ To			
	0	0	1
	0	1	0
	1	0	0

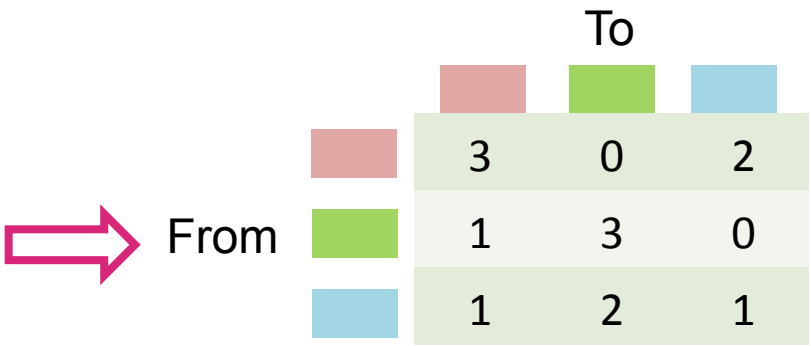
Markov models

Total sampled time (e.g. 100 μs)

Define the *discrete* state of the system in *discrete time* (e.g. via reaction coordinates)

Lag time τ (here: 4 time units)

Compute the transition probability matrix sliding a window of lag time τ



Repeat until the end of the trajectory.

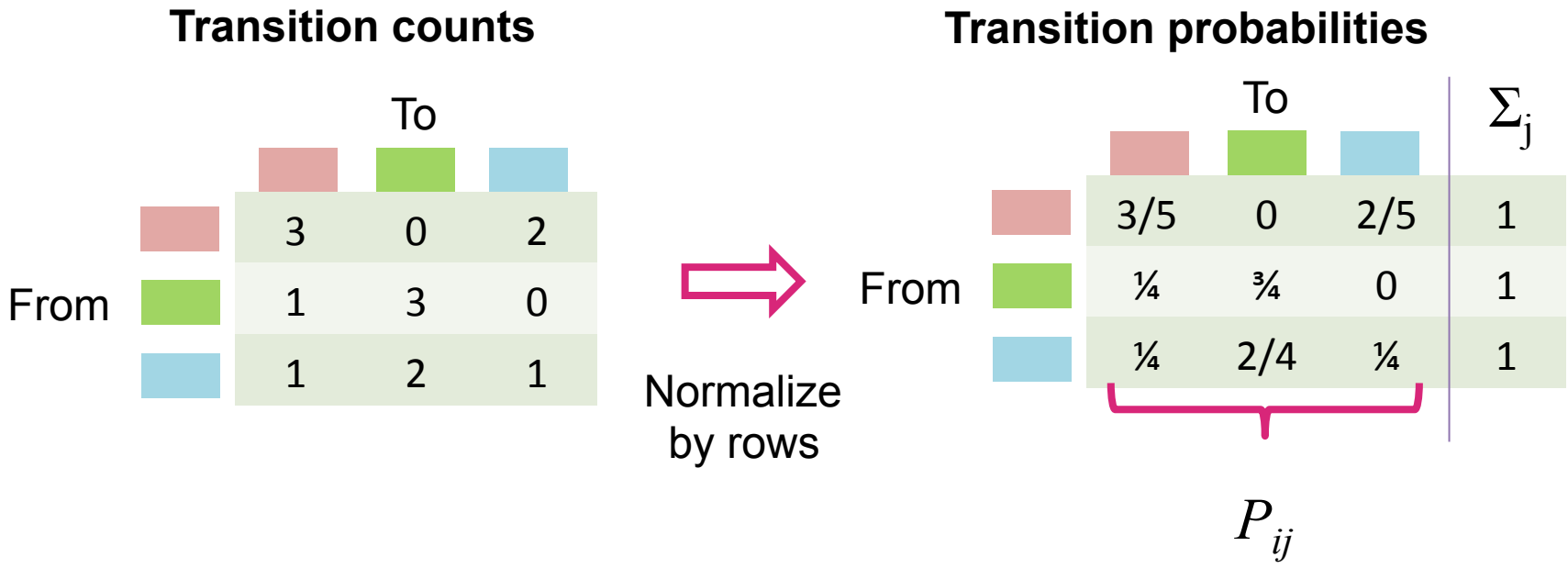
Note the Markovian assumption:
transition probabilities **do not** depend
on history (neither, for us, on time).

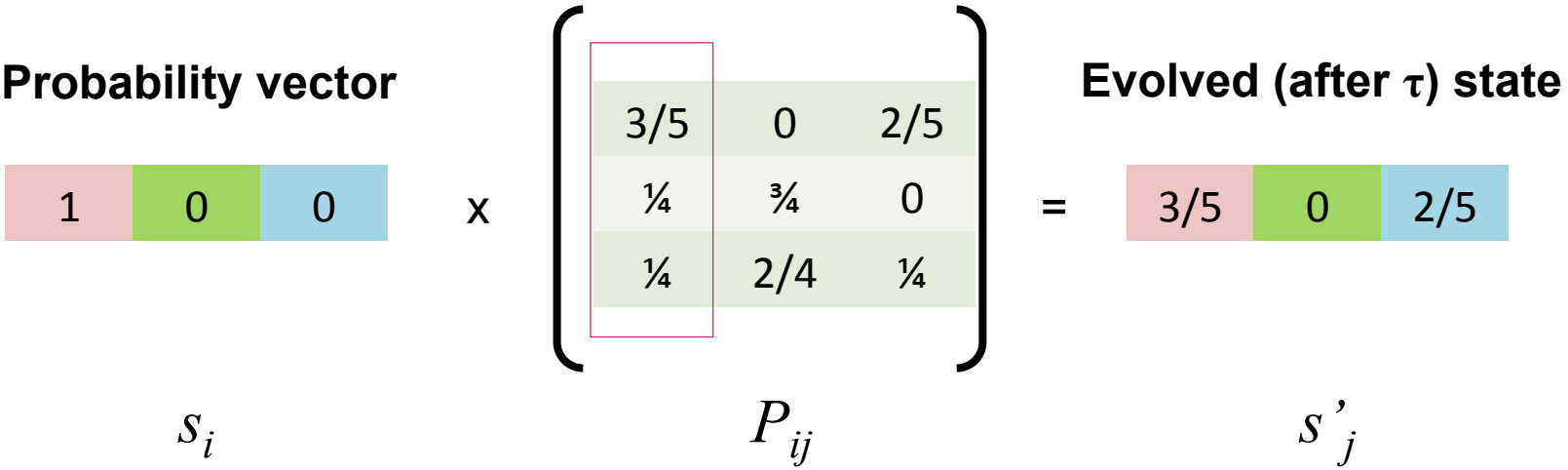
(Note how "Early" and "late" events
are squashed in the same matrix.)

Important: you can accumulate counts
from **different** trajectories in the **same** matrix!

It's all valid, independent sampling
of the same underlying system.

Example: multiple weather stations in the same city.
They may be active at different times!


Transition counts
To
From
3 0 2
1 3 0
1 2 1
Normalize by rows
Transition probabilities
To
Σ_j
From
3/5 0 2/5 1
¼ ¾ 0 1
¼ 2/4 ¼ 1
P_{ij}


Probability vector
1 0 0
x
3/5 0 2/5
¼ ¾ 0
¼ 2/4 ¼
=
Evolved (after τ) state
3/5 0 2/5
s_i
P_{ij}
s'_j

Probability vector

1	0	0

s_i

x

$$\begin{pmatrix} 3/5 & 0 & 2/5 \\ ¼ & ¾ & 0 \\ ¼ & 2/4 & ¼ \end{pmatrix}$$

P_{ij}

=

Evolved state

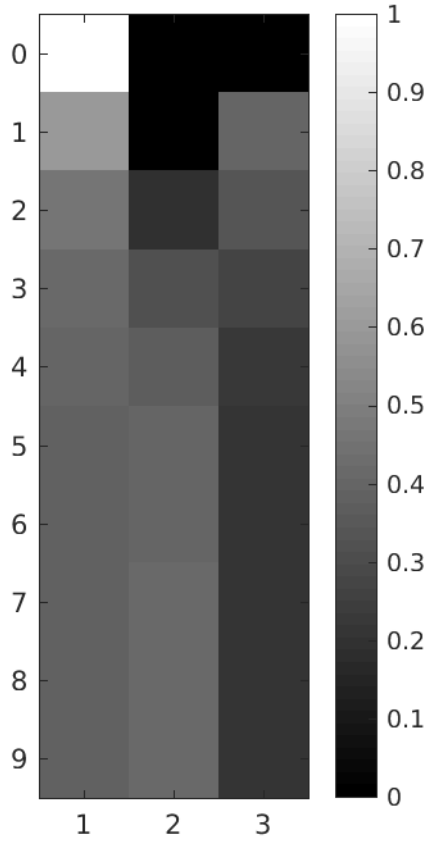
3/5	0	2/5

s'_j

$$s' = sP$$
$$s'' = (sP)P = sP^2$$
$$s^{(n)} = sP^n$$

1	0	0
.60	0	.40
.46	.20	.34
.41	.32	.27
.39	.37	.23
.39	.40	.22
...	...	...

time



Probability vector

1	0	0

s_i

x

3/5	0	2/5
¼	¾	0
¼	2/4	¼

P_{ij}

=

Evolved state (after tau)

3/5	0	2/5

s'_j

$$s^{\infty} P = s^{\infty}$$

$$s' = sP$$
$$s'' = (sP)P = sP^2$$
$$s^{(n)} = sP^n$$
$$\ldots$$
$$s^{\infty} = \quad ?$$

Left eigenvector of P (eigenvalue 1)

Is the stationary state
= equilibrium probabilities
= the free energy surface

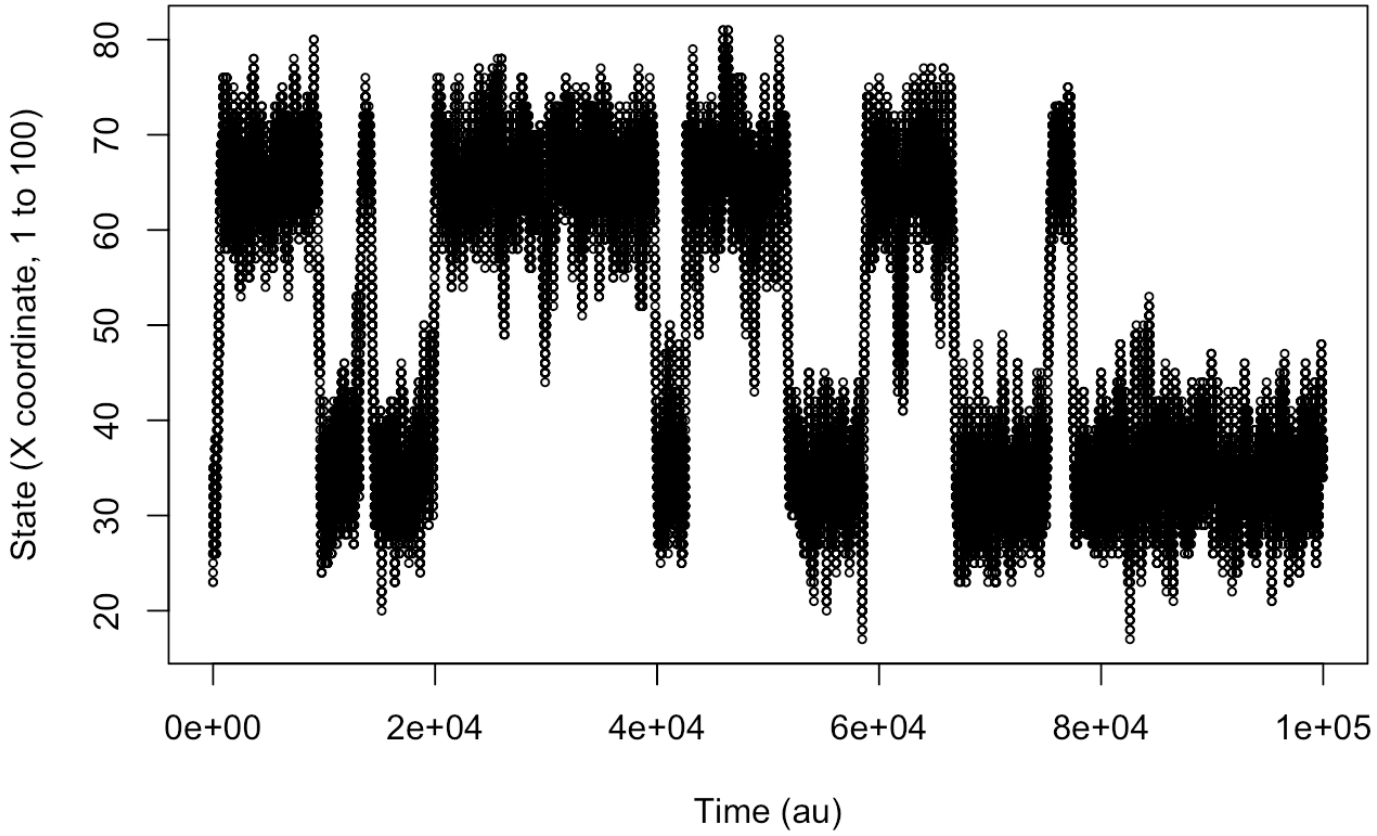
```
[a,b]=eig(P')
a(:,3)/sum(a(:,3))
= 0.385 0.410 0.205 = [5/13 16/39 8/39]
```

Markov modeling a 1D trajectory

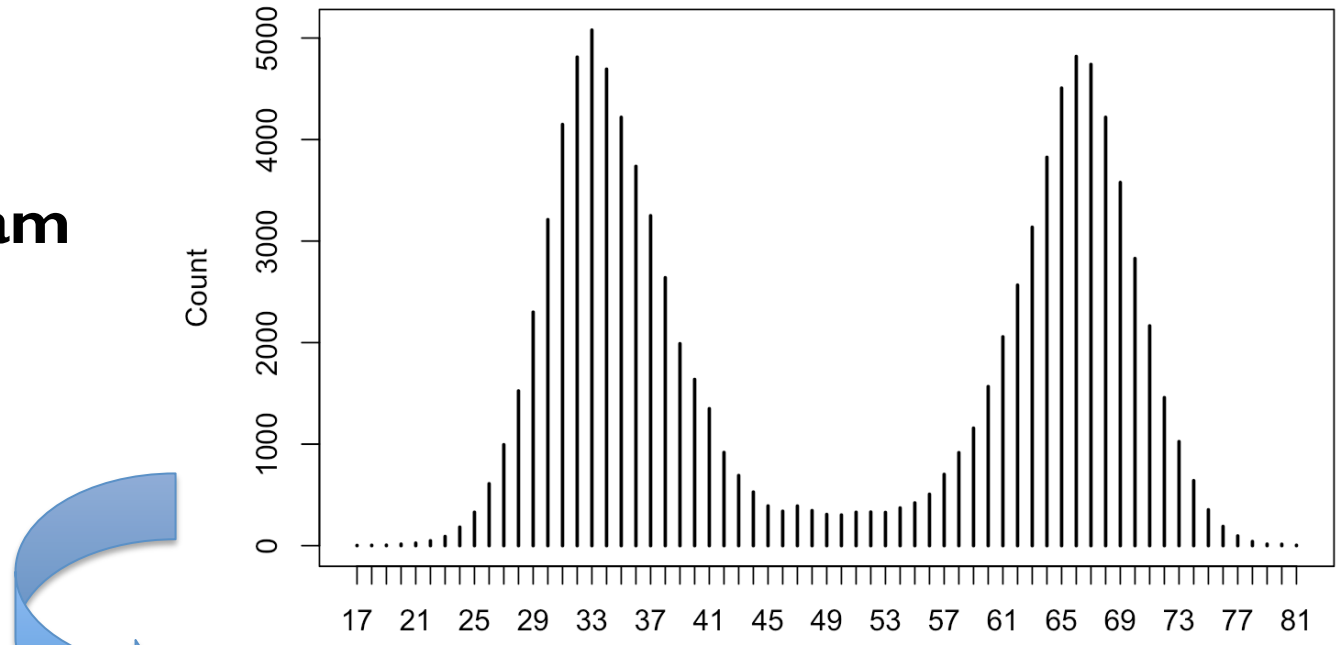
(Please find the extended version online,
"Markov state models of a 1D trajectory",
with R code)

Start with a 1-D trajectory

- Already discretized in 100 bins

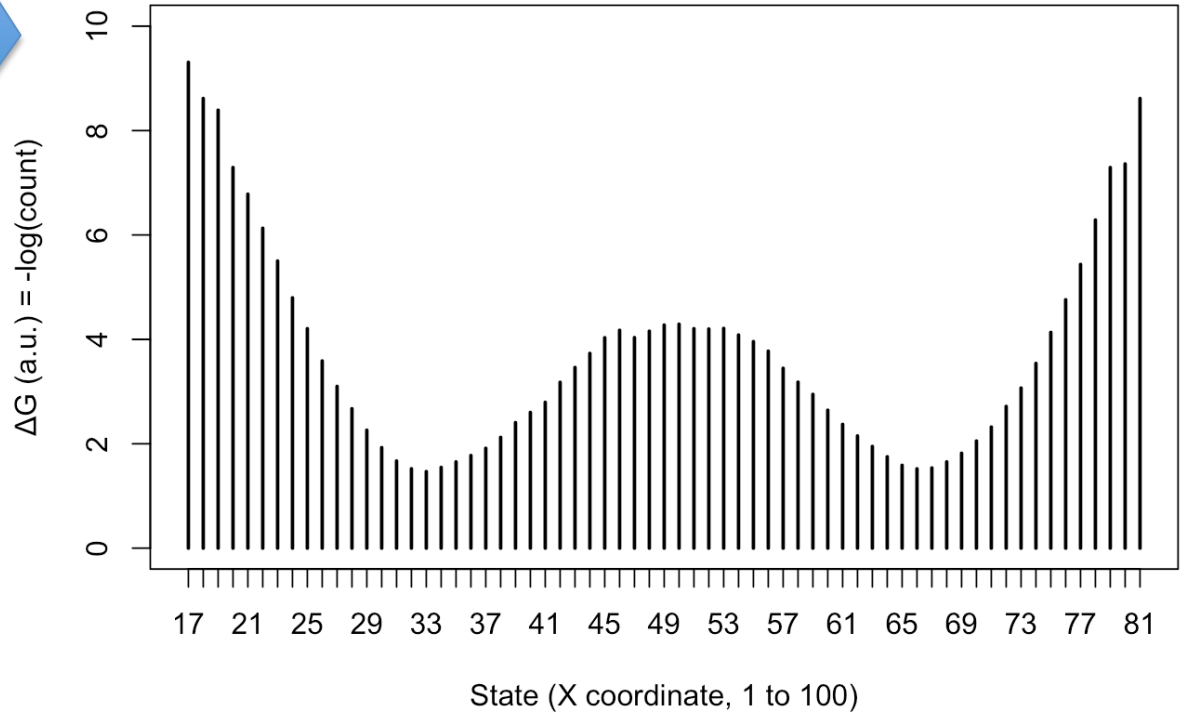


Histogram



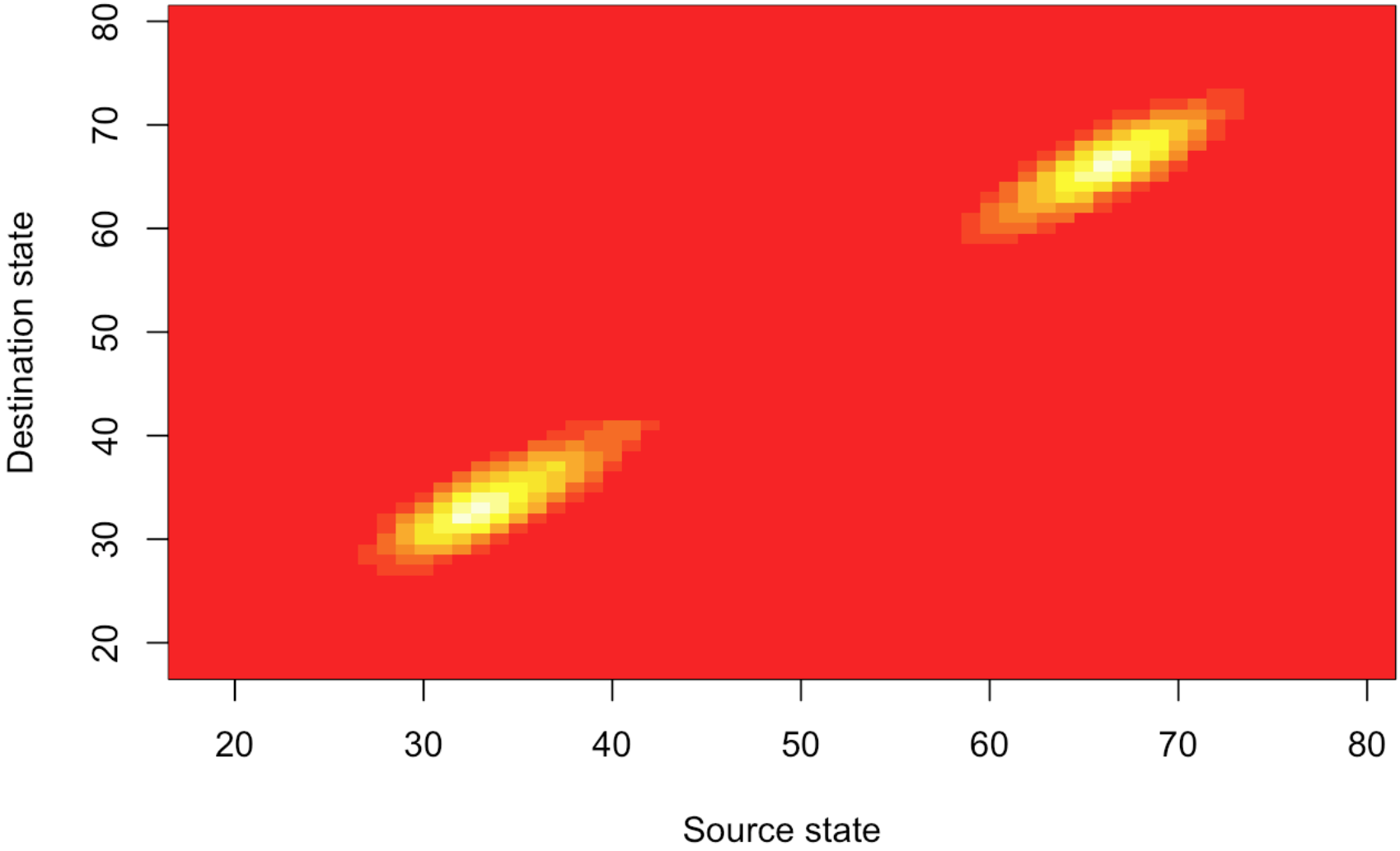
Boltzmann inversion

-log(count)



The transition *count* matrix*

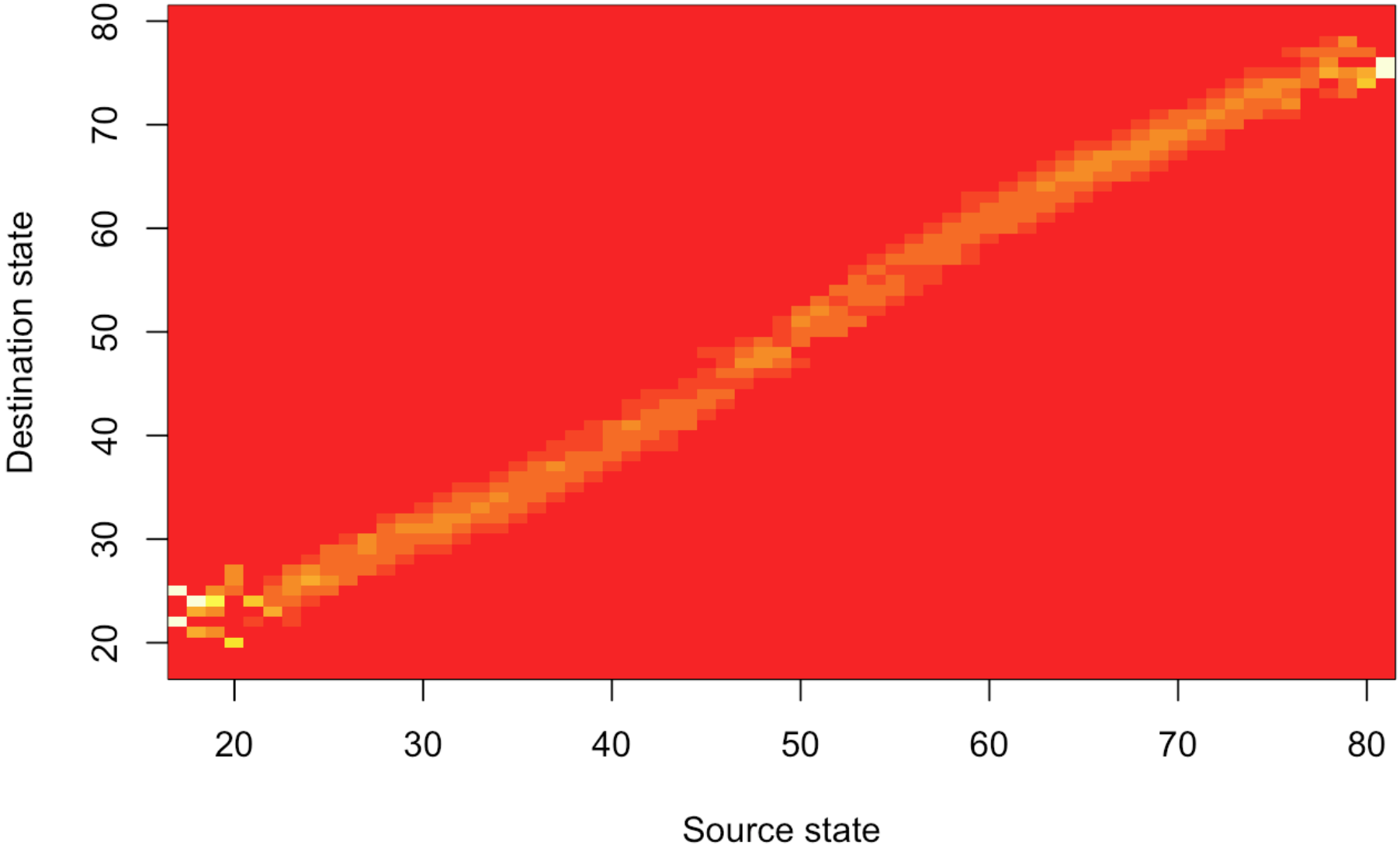
How many times we have seen state i going to j after $\tau = 10$ *time units*



* shown as an image for compactness

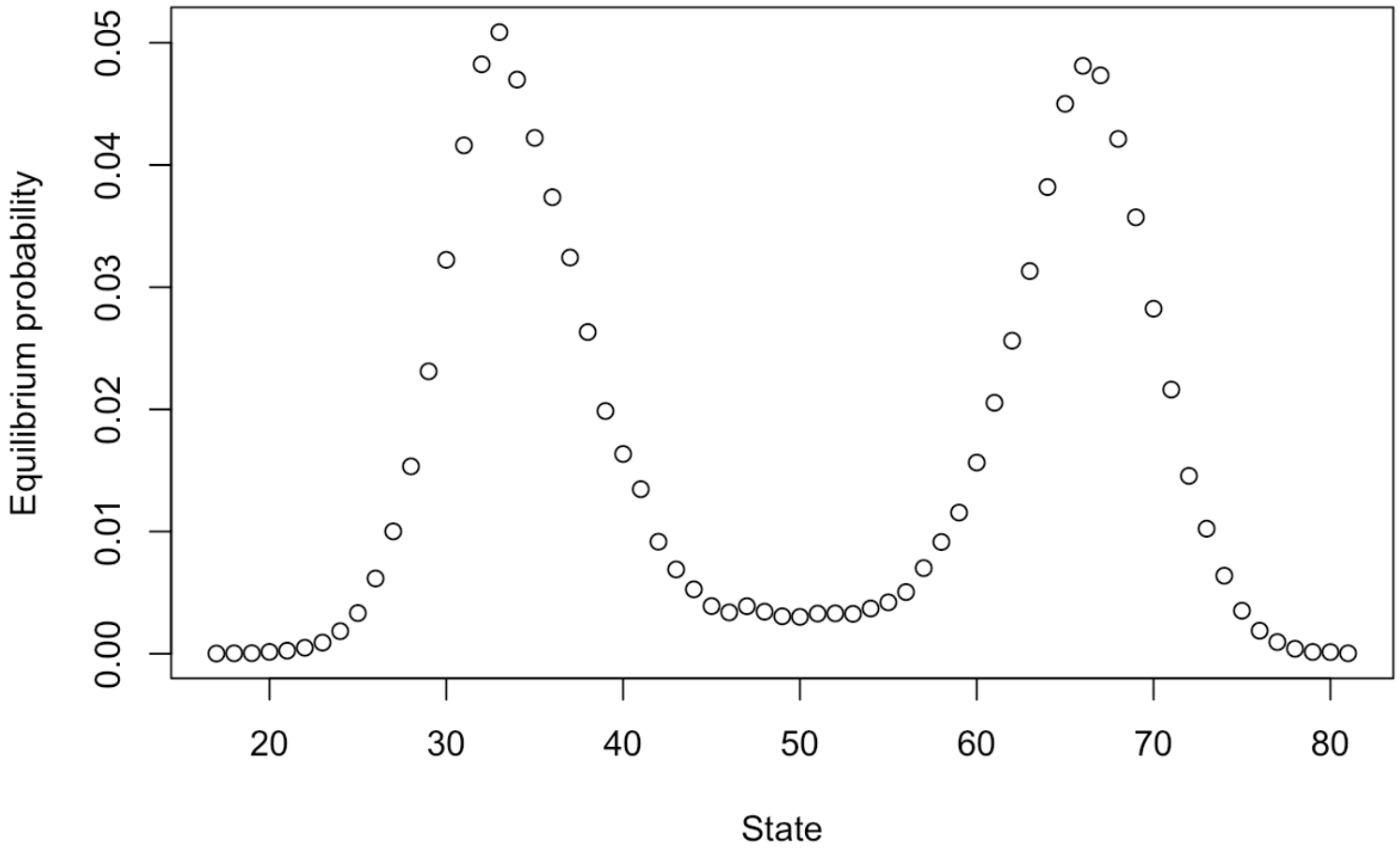
The transition *probability* matrix

Rows normalized to sum to 1



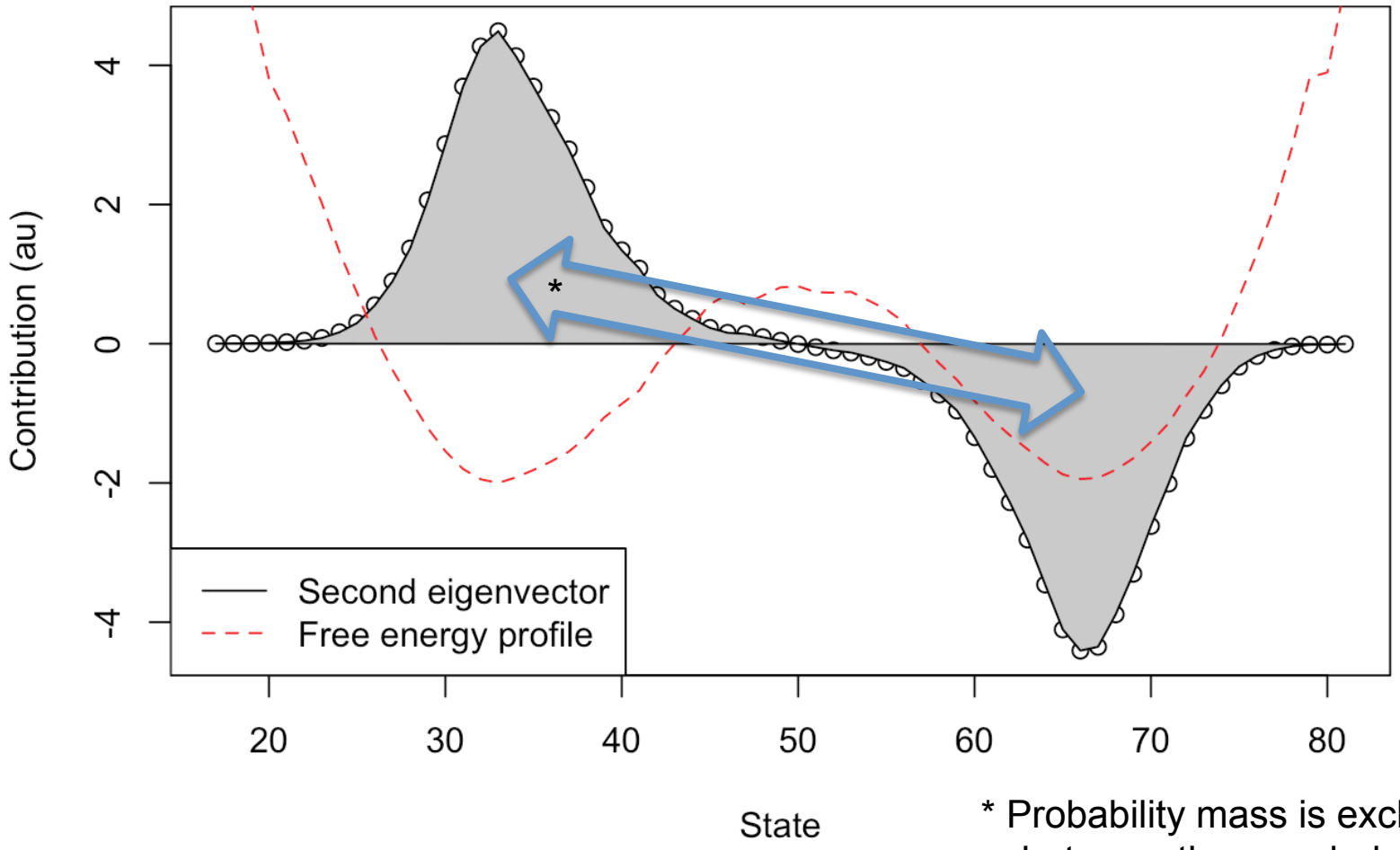
First eigenvector ($\mu_1 = 1$)

This is the stationary state (normalize so it sums to 1)



Second eigenvector (μ_2=0.997)

This is the slowest relaxation mode: ITS τ_2 = 3610 time units



* Probability mass is exchanged between the + and - basins

Take-home message

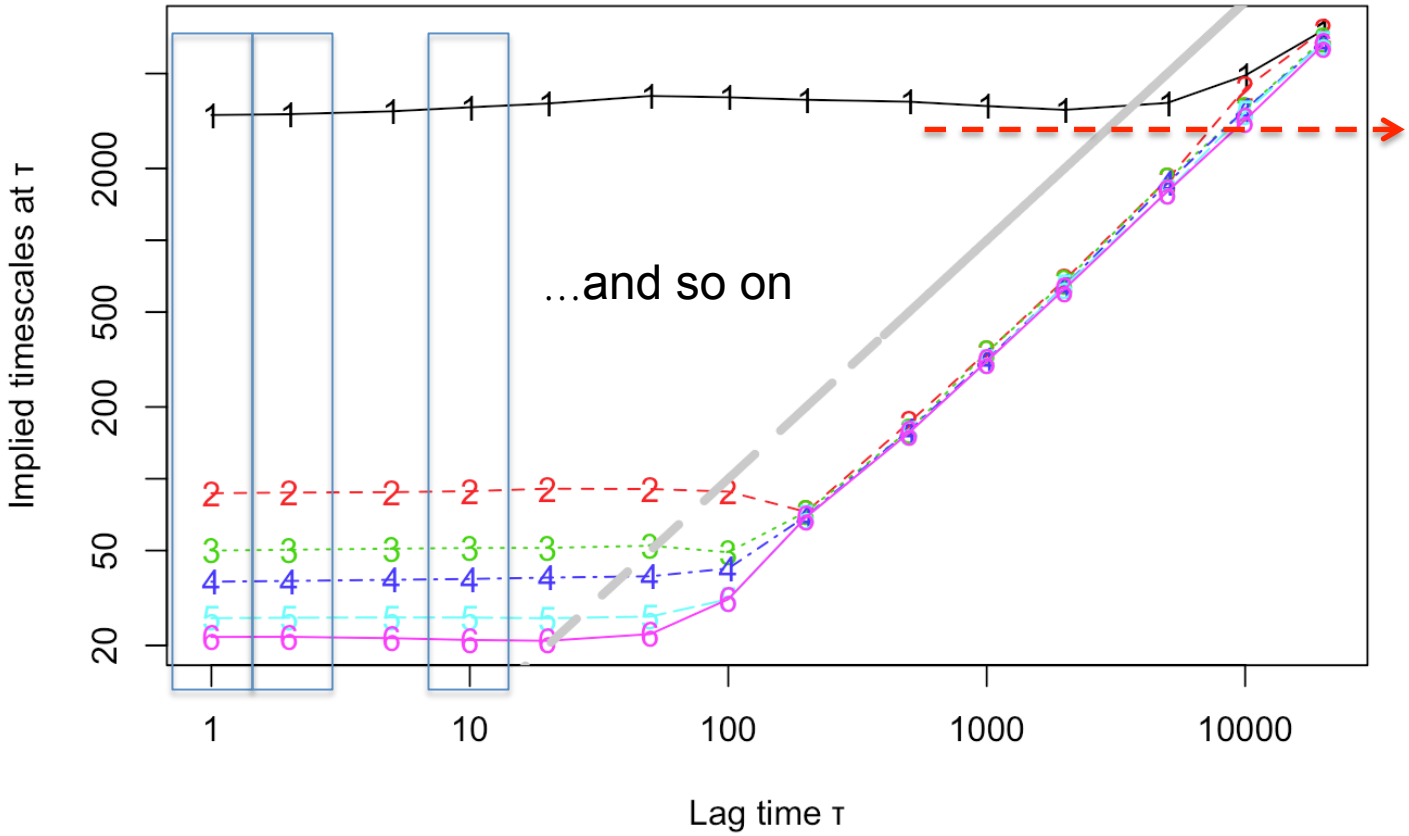
- Define states.
- Use trajectories to count transitions ➔ P_{ij}
- Eigenvalues: 1.0, μ_1, μ_2, …
  - These are the time-scales (after –log)
- Eigenvectors: s_∞, s_1, s_2, …
  - These are the equilibrium configuration, i.e. ΔG; and faster "oscillations" (kinetics)

Markovianity

- The state transition probabilities only depend on the current state.
- Examples
  - Today's weather, not yesterday's
  - Where the ligand is, not how did it got there
- The property may be false at short timescales but true at longer ones (system's memory)
- It does depend on the chosen states

Implied timescales plot

Repeat the eigenvalues determination for several lags. Check convergence.

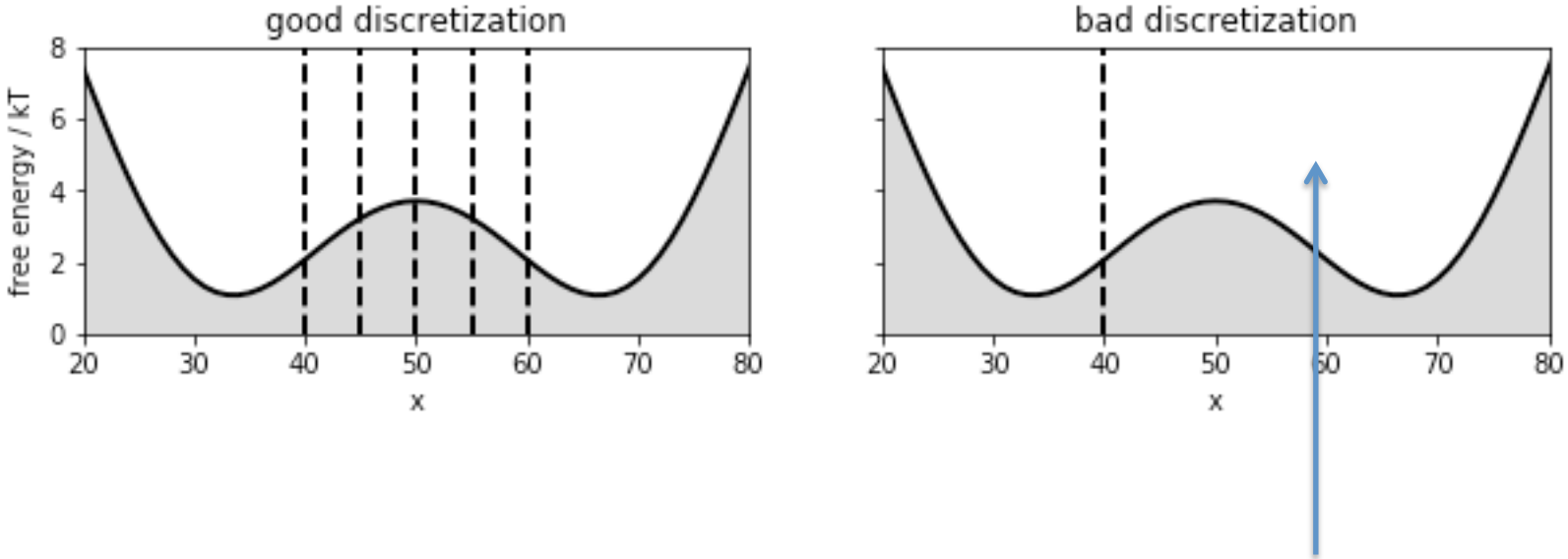


Here, convergence is achieved very early.

Reasons:
(a) true two-state dynamics;
(b) absence of orthogonal degrees of freedom;
(c) fine space discretization

Macrostates

A bad choice of the discretization breaks the Markovianity assumption

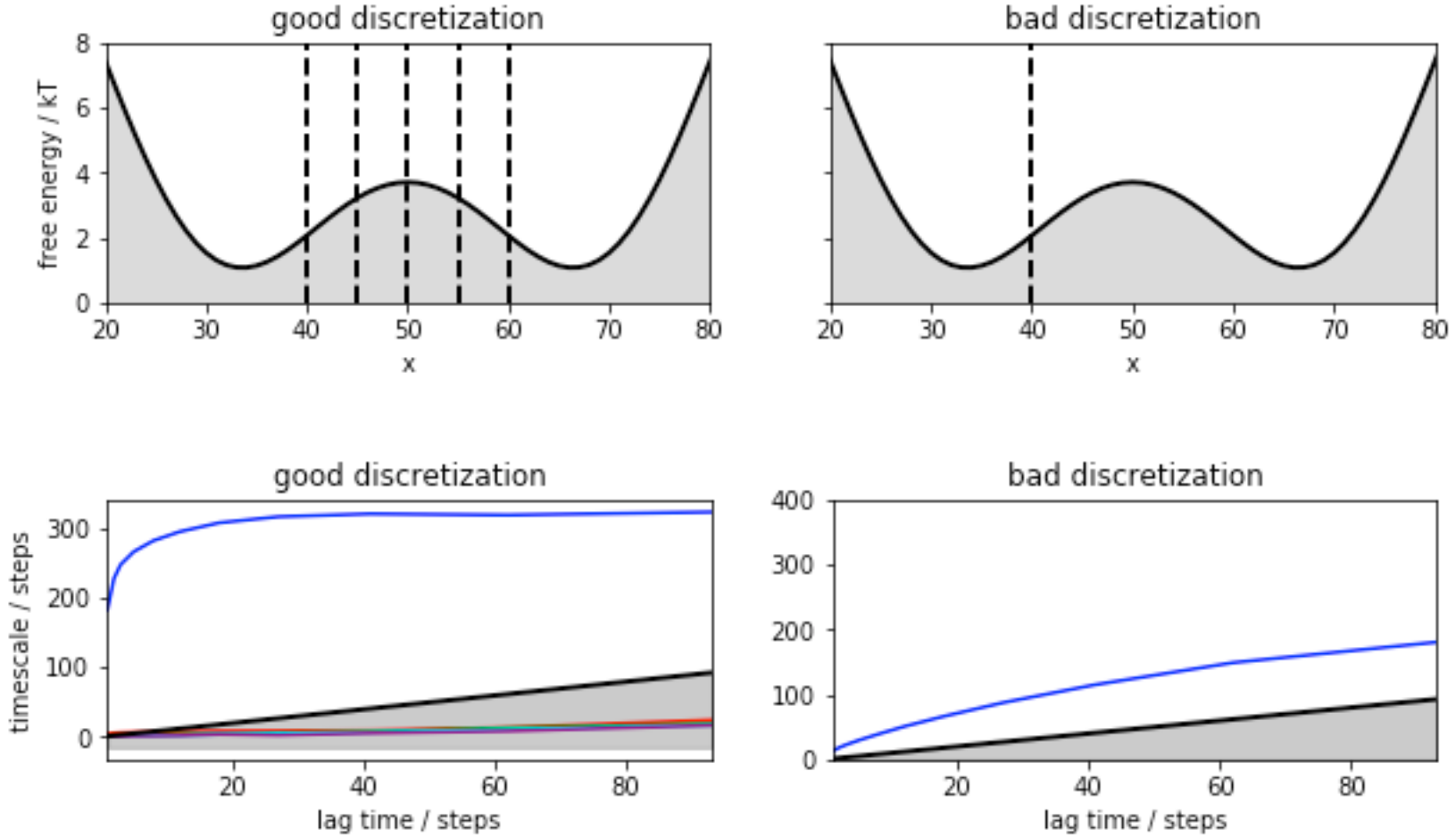


In the “bad discretization” case, the barrier is embedded in one of the states. This generates a “long term memory” effect: the rightmost state could actually be short-lived (if we are on the left of the barrier) or long-lived (if we are on its right). These two cases are convoluted into the same, so that the present state information itself is not sufficient to predict the “future” of the system any more.

Based on http://www.emma-project.org/v2.0.4/generated/model_selection_validation.html

Macrostates

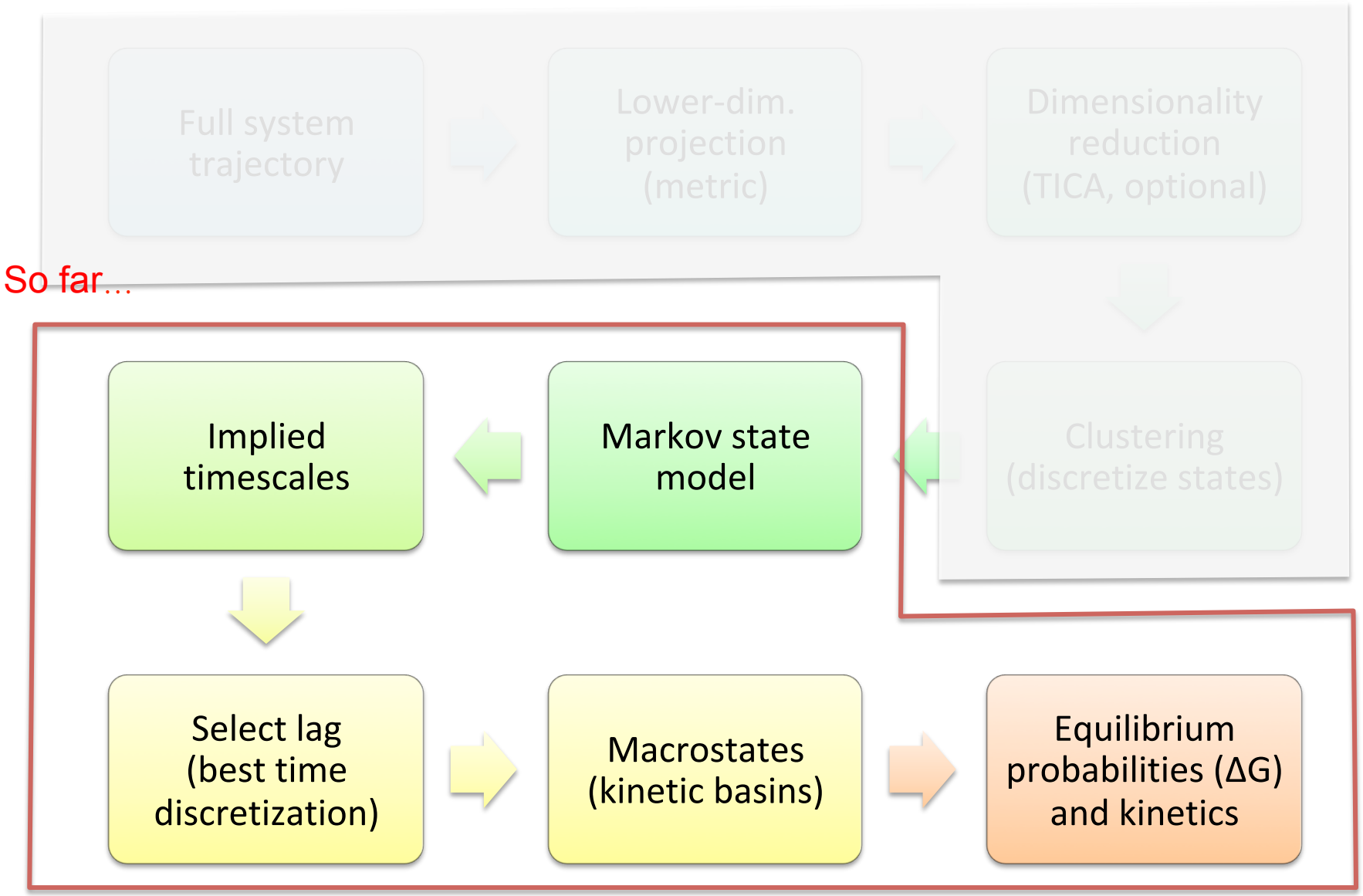
A bad choice of the discretization breaks the Markovianity assumption



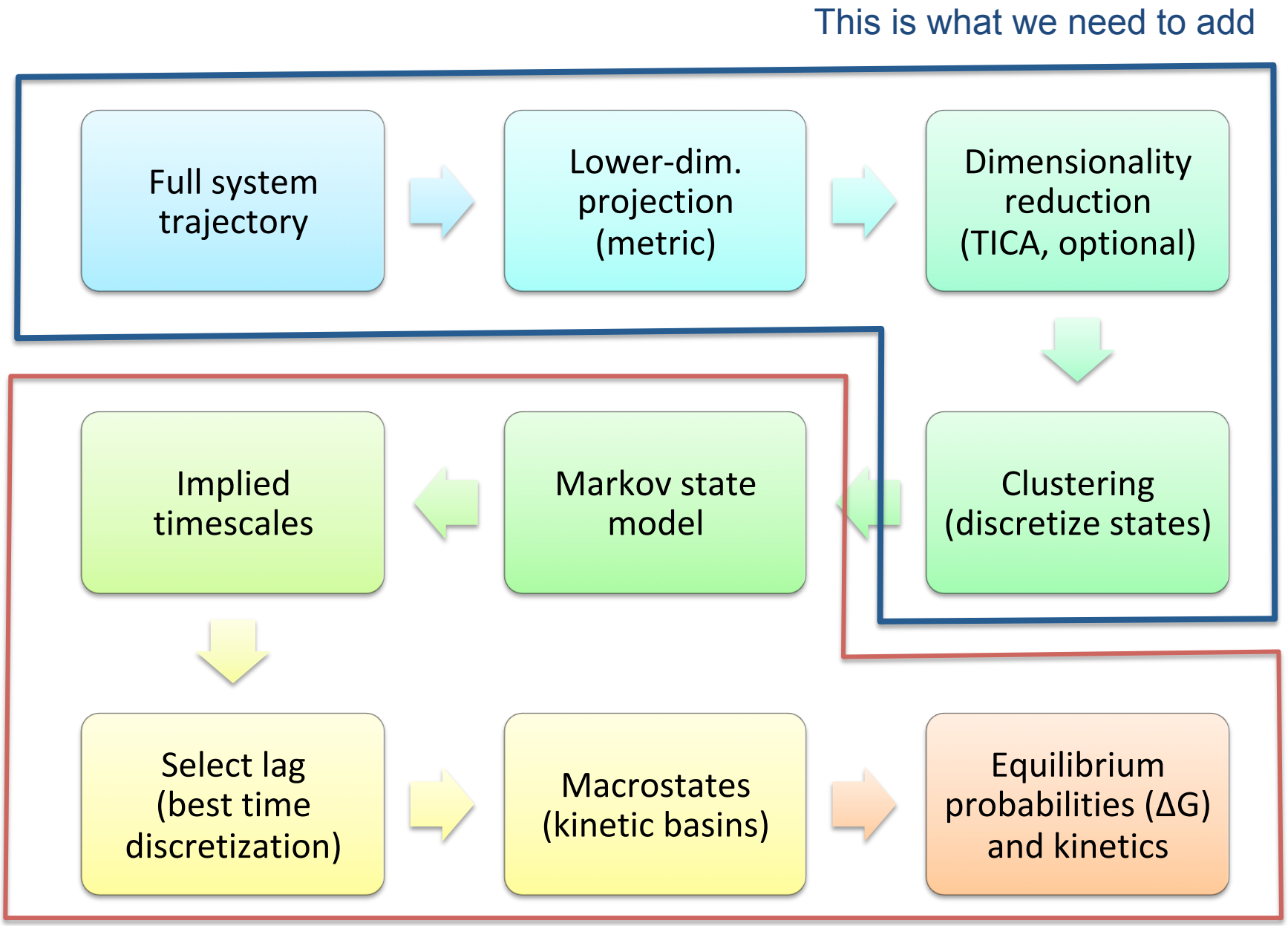
Based on http://www.emma-project.org/v2.0.4/generated/model_selection_validation.html

Back to molecules:
>>1 dimension, continuous:
clustering

Dealing with trajectories…

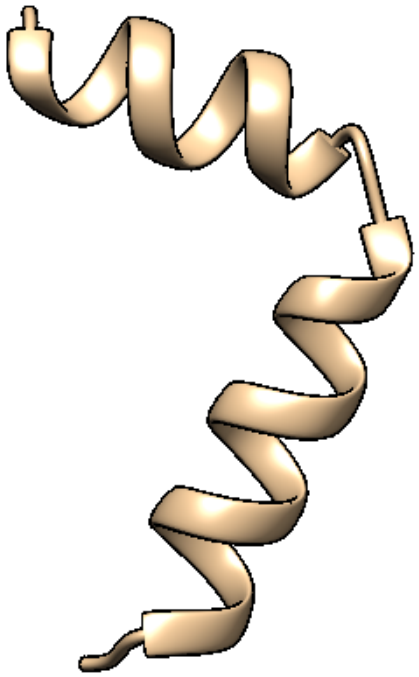


Dealing with trajectories…



Metric projection

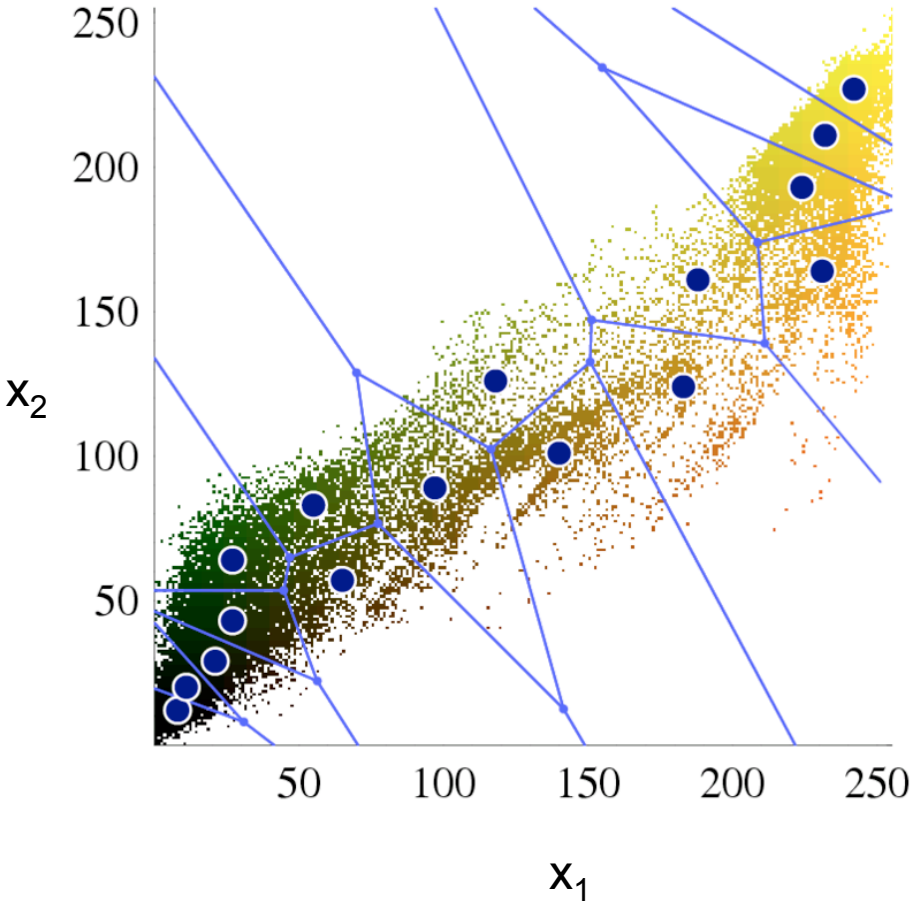
The first step is to project the system state in a lower-dimensional space ("metric"). Many choices are available, e.g.



- Manually chosen distances
- Atom coordinates
- *N* phi/psi Ramachandran angles
- Distance matrix
- Contact matrix
- …

Clustering (1st level)

- Now move to a discrete state space
- Reduction from the low-dimensional space is done by *clustering*
  - Usually called "microstates"
- Several algorithms are implemented in MSM packages (e.g. grid; *k*-means; etc. We won't discuss them)



https://en.wikipedia.org/wiki/K-means_clustering

Conclusions and Resources

Conclusions

- MSM methods make efficient use of *unbiased* sampling
- Cluster → P_{ij} → eigenvectors → free energies (K_D) and kinetics (k_{on}/k_{off})
- Still require *huge* (but not unreachable) amounts of sampling/simulation for biologically interesting systems
- Strong mathematical foundation, with good software available

Software + Tutorials

- All are Python-based
  - They include clear walkthroughs (highly recommended) with datasets
- **HTMD** – www.htmd.org
  - Also analysis + system build + adaptive …
  - Can aggregate large-scale datasets
- PyEMMA – www.emma-project.org
- MSMBuilder - msmbuilder.org

Related topics

If you liked this, you may also like…

- Estimation of reversible transition matrices
- Hidden Markov Models doi:10.1063/1.4828816
- TICA
- Conformational fluxes
- Adaptive sampling
- Role of drug residence time in drug discovery

Literature

- Buch et al., 10.1073/pnas.1103547108
  - Rigid ligand + protein association, simplest case
- Swinney, PMID:19152211
  - Importance of kinetics in drug design
- Chodera and Noe, 10.1016/j.sbi.2014.04.002
  - Excellent overview (1)
- Voelz et al., 10.1021/ja9090353
  - Full reconstruction of a millisecond folding
- Pande et al., 10.1016/j.ymeth.2010.06.002
  - Excellent overview (2)
- Paul et al., 10.1038/s41467-017-01163-6
  - Peptide-peptide association
- Doerr et al., 10.1021/ct400919u
  - Adaptive sampling
- Bryan et al., "The $25 B eigenvector"
  - The original PageRank algorithm; e.g. jdc.math.uwo.ca/M1600b/l/pagerank-1600.pdf

End