Tutorial 5: Drift-diffusion models

BAMB! Summer School
Day 6

When to use a DDM

- You want to incorporate both choice and response time
  - If you don't care about RT, there are simpler models
- You have two alternatives
  - If you have only one alternative, you can fit a Wald distribution instead
  - If you have more than two alternatives, you (probably) need to use a race model

Tutorial overview

- Hour 1: Simulating the DDM by hand
  - Construct a DDM from first principles
- Hour 2: Simulating the DDM using PyDDM
  - Use efficient and higher-accuracy methods to perform simulations
- Hour 3: Fitting the DDM to data
  - Use PyDDM to fit the DDM to monkey random dot motion data
- Hour 4: Generalized drift diffusion models (GDDMs)
  - Create variants of the DDM which are specialized to specific tasks or encapsulate distinct strategies

Models from this course related to DDM

Bayesian (hierarchical) DDM
Parameter estimates with limited data, pooling across subjects, distributions of parameter estimates

DDM-HMM
When the emission is a response time and a choice

DDM

RL-DDM
Incorporating response time information into RL

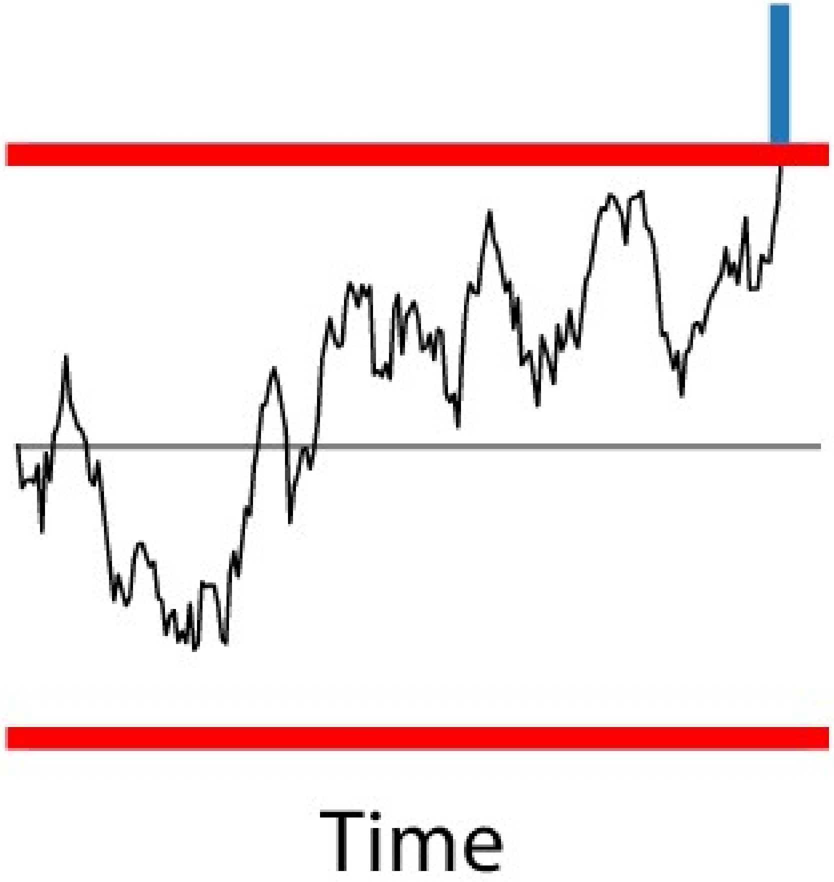
Likelihood approximation network
Neural network for fitting complex DDM-like models that cannot be easily simulated

Hour 1: Simulating the DDM by hand

- Basic algorithm
  - 1. Set x to starting point
  - 2. Set:

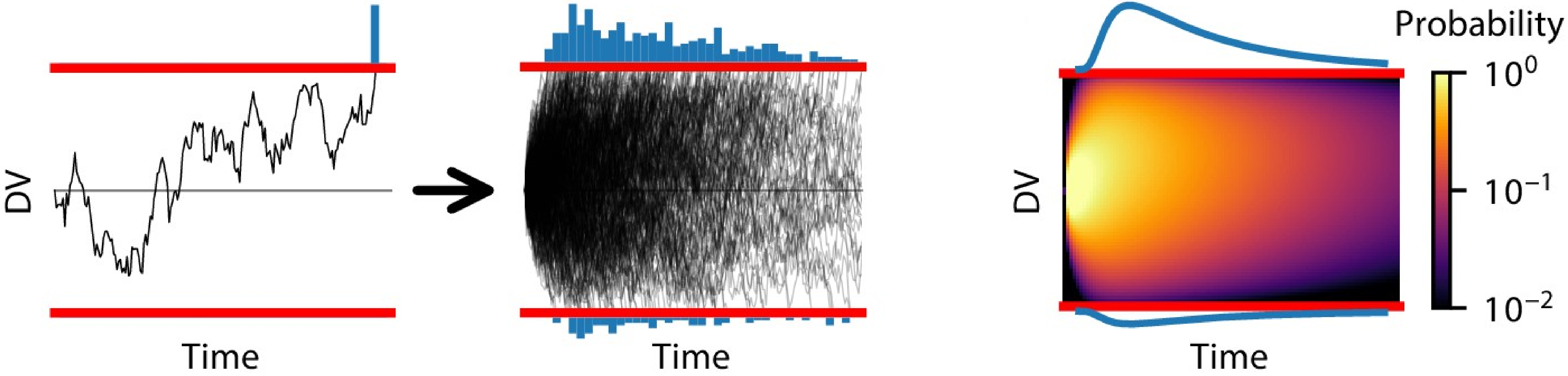
$$x_{t+1} = x_t + [\text{drift}]\Delta t + [\text{noise}] z_t \sqrt{\Delta t}$$
$$z_i \sim N(0,1)$$

  - 3. Check if x crosses a boundary. If so, you are done
  - 4. Otherwise, go to (2)



Hour 2: Simulating the DDM using PyDDM

- Use more efficient methods to simulate the probability distribution of a trajectory's position instead of one trial at a time



DDM libraries

	PyDDM	HDDM	EZ-Diffusion	CHaRTr	DMAT	fast-dm
Language	Python3	Python2/3	Matlab, R, Javascript, or Excel	Requires both R and C	Matlab	Command line
Solver	Fokker-Planck, analytical	Analytical numerical hybrid	None	None (Monte Carlo)	Analytical numerical hybrid	Fokker-Planck
Task parameters						
Time dependence of drift/noise	Any function	Constant	Constant	Any function	Constant	Constant
Position dependence of drift/noise	Any function	Constant	Constant	Any function	Constant	Constant
Bounds	Any function	Constant	Constant	Any function	Constant	Constant
Parameter dependence on task conditions	Any relationship for any parameter	Regression model	Categorical	Categorical	Linear	Categorical
Across-trial variability						
Across-trial drift variability	Slow discretization (via extension)	Normal distribution	None	Any distribution	Normal distribution	Normal distribution
Across-trial starting point variability	Any distribution	Uniform distribution	None	Any distribution	Uniform distribution	Uniform distribution
Across-trial non-decision variability	Any distribution	Uniform distribution	None	Any distribution	Uniform distribution	Uniform distribution
Model simulation and fitting						
Hierarchical fitting	No	Yes	No	No	No	No
Fitting methods	Any numerical (default: differential evolution)	MCMC	Analytical	Any numerical	Nelder-Mead	Nelder-Mead
Objective function	Any function (default: likelihood)	Likelihood	Mean/stdev RT and P(correct)	Any sampled (e.g. quantile maximum likelihood)	Quantile maximum likelihood or chi-squared	Likelihood, chi-squared, Kolmogorov-Smirnov
Mixture model	Any distribution(s)	Uniform	None (extendable)	None	Uniform and undecided guesses	Uniform

How PyDDM works:

- Construct a Model from its components
- Model components:
  - Drift rate
  - Noise
  - Bound
  - Initial Condition
  - Non-decision time
  - Mixture model

Many model components are built-in:

- Each component can be:
  - A constant value (e.g., `3`)
  - A fittable parameter, given by the the name (e.g., `param1`)
  - A function which depends on:
    - Parameters
    - Task conditions
    - Magic arguments

Parameters and conditions

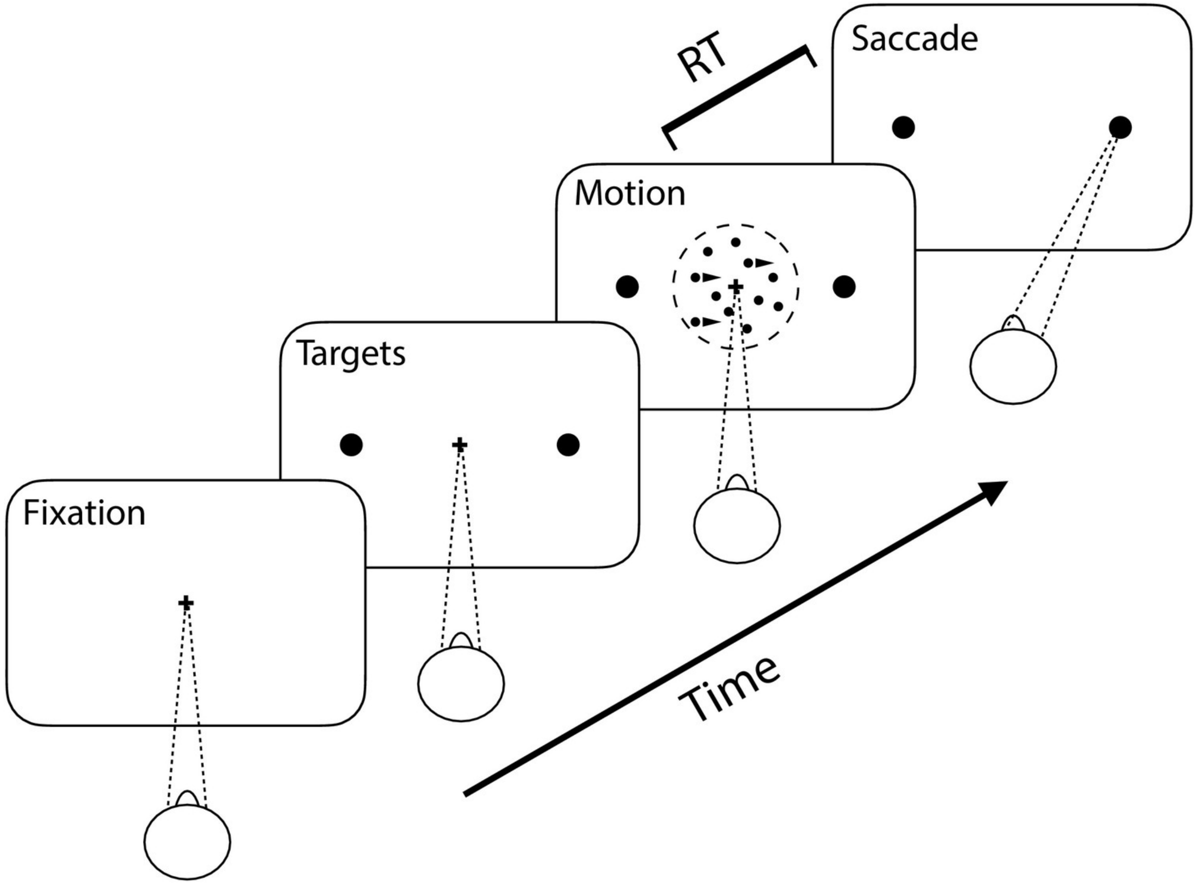
- Parameters: Have the same value for the entire dataset
  - E.g. bound height
- Conditions: May change from trial to trial
  - E.g. strength of motion coherence

Three objects to remember in PyDDM

- Model: created by the gddm() function
  - May need to call model.fit() before using if there are parameters
- Solution: Created using model.solve(conditions={...})
- Sample: RT and choice data, either experimental or simulated data

Hour 3: Fitting the DDM to data

- Dataset: Monkeys performing the random dot motion task (Roitman and Shadlen, 2002)
- Several levels of motion coherence

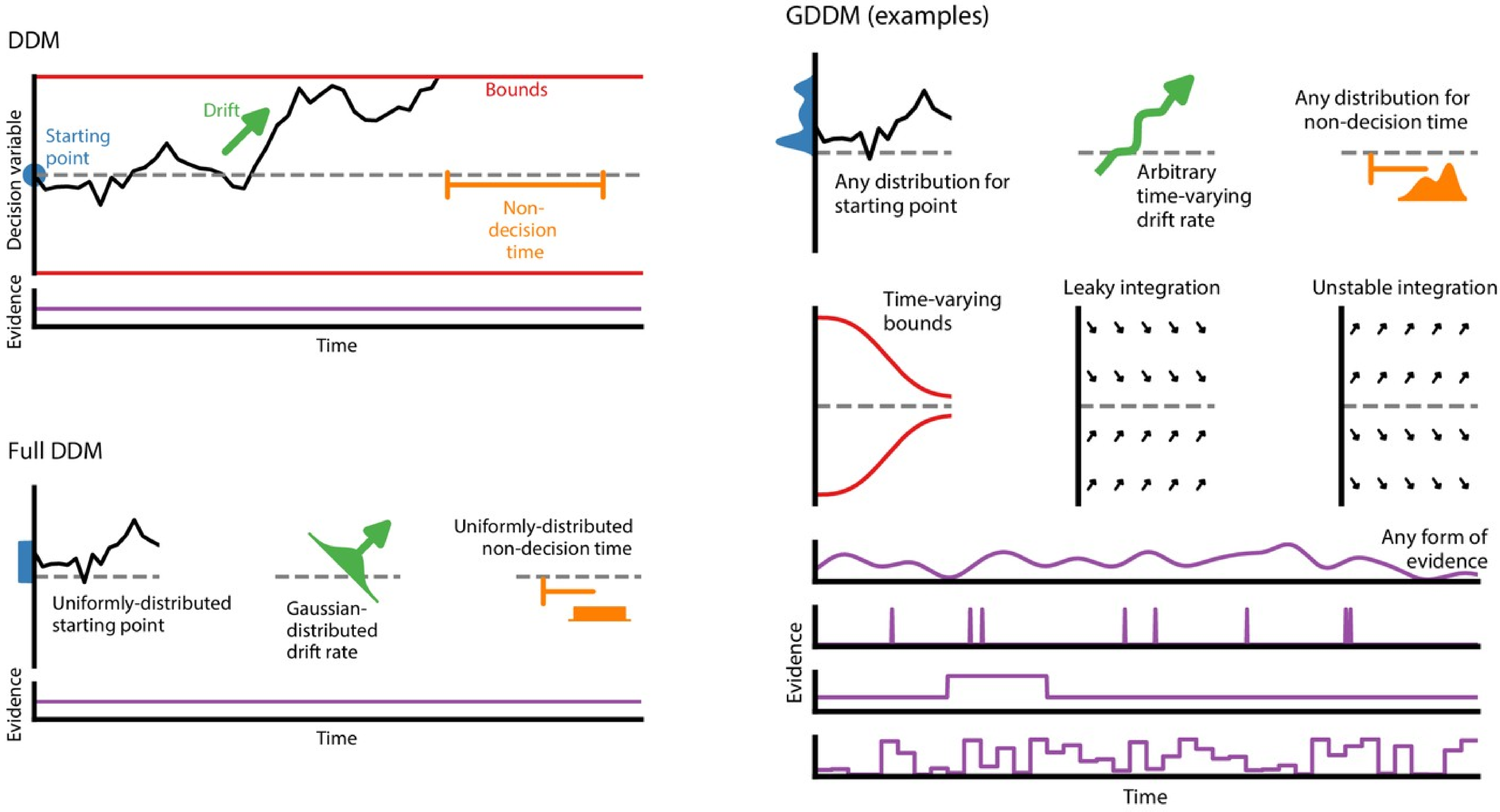


https://shadlenlab.columbia.edu/resources/RoitmanDataCode.html

Hour 4: Generalized DDMs (GDDMs)

- Construct a more complex model, or a model for a more complex task
- Magic arguments:
    - Time in the simulation `t`
    - Positions of the decision variable `x`
    - A vector of all simulation times `T`

Example GDDMs



When should you use these GDDMs?

- Response time distribution is not skewed
  - Consider leaky integration
- The speed-accuracy tradeoff may change across the trial
  - Consider collapsing bounds
- I think the agent may be more likely to choose one choice over another or have a prior
  - Consider a starting point or drift bias
- Evidence is not constant in my task or it requires multisensory integration
  - Consider a more complex drift rate function
- There is a large variability in motor actions
  - Consider non-decision time variability *(but be careful! This can make the model non-recoverable)*