Generative Models and Simulation

Recipes for Simulating Synthetic Data from Assumptions

Generative Models and Simulation

What is a Generative Model?

> **Definition: Generative DAG**
>
> A generative DAG is a mathematical object (a directed acyclic graph) where nodes are random variables (or computed from random variables) and arrows show which random variables need to be computed before others. It serves as a step-by-step recipe for simulating synthetic data from assumptions.

Node Types (Only Two)

Stochastic Nodes (Random Draws)

Instruction: "Draw from a distribution."

Notation: $\sim$ (e.g. $x \sim \text{Bernoulli}(\theta)$)

Recipe analogy: "Take a random scoop from this bowl."

Deterministic Nodes (Functions)

Instruction: "Compute from inputs."

Notation: $=$ (e.g. R: `ifelse(X==1, 100, -100)` \ Python: `100 if X==1 else -100`)

Recipe analogy: "Mix these ingredients using this rule."

Example: $100 Bet on a Coin Flip

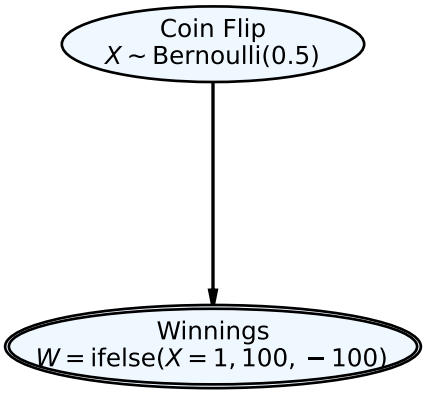

Figure 1: DAG for a $100 bet on a coin flip

Given the definition above,

Appendix: DAG Drawing Notes

Note to ourselves: We use a custom `DAG` class (inherits from `daft.PGM`) with three specialized node methods:

- `latentNode()` - aliceblue background, aspect=4
- `observedNode()` - purple background, aspect=4

- `deterministicNode()` - aliceblue background, aspect=5.4, alternate=True

The class includes a `truth` attribute and uses `partialmethod` to create these node methods with custom defaults.

Installation: `pip install 'daft-pgm'`

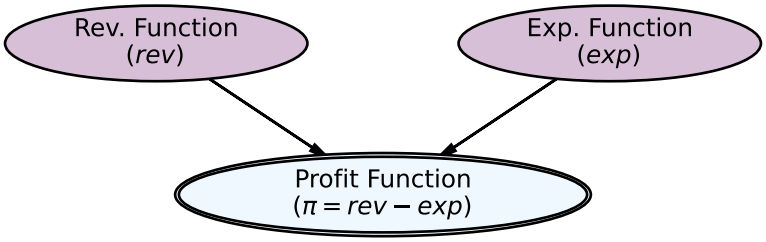

Figure 2: DAG constructor example