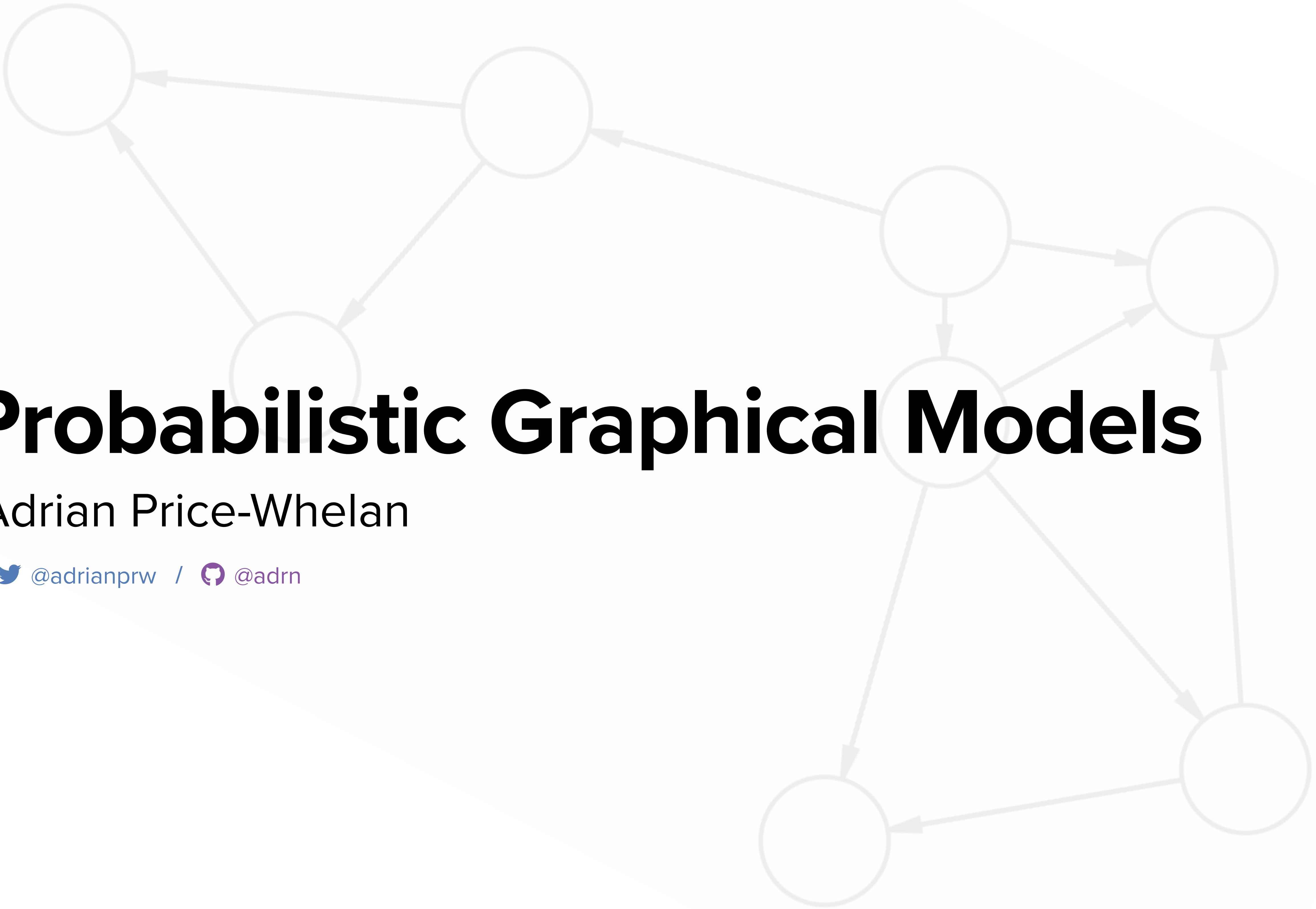


Probabilistic Graphical Models

Adrian Price-Whelan

@adrianprw / @adrn

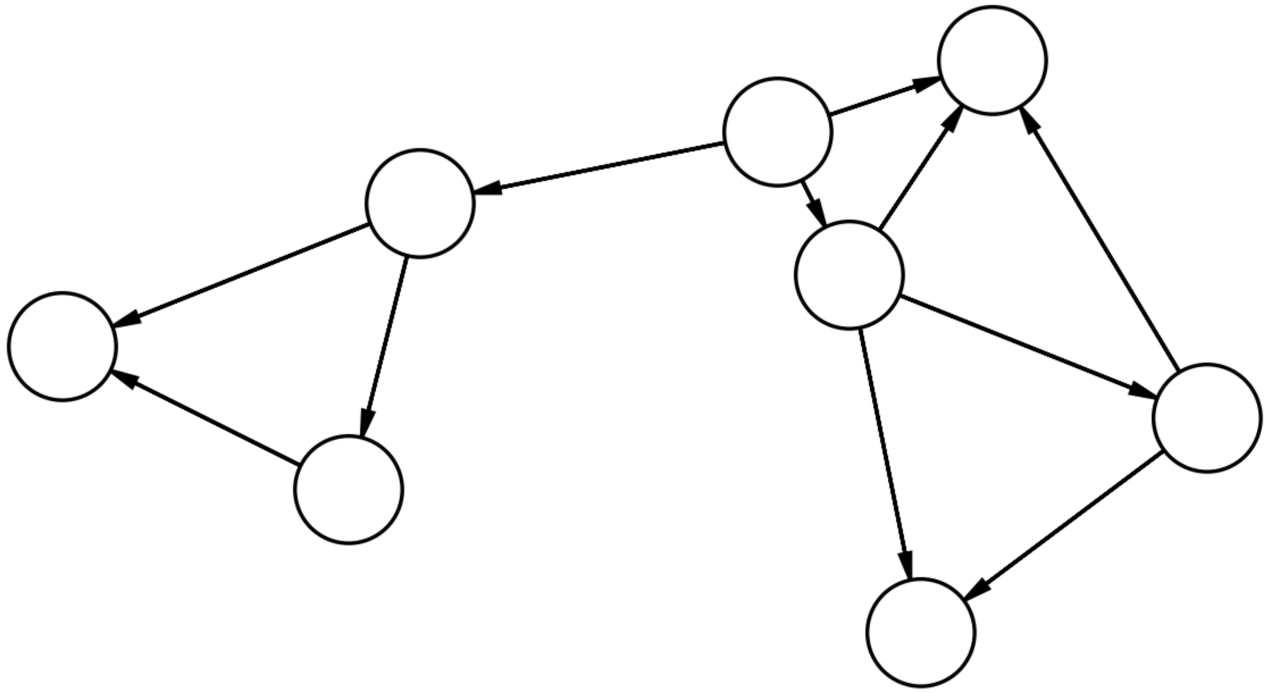
Probabilistic Graphical Models

or: Graphical Models, PGMs

Apply concepts from *graph theory* to help represent and conceptualize relationships between random variables

PGMs are useful for understanding models, for debugging, and for thinking about extensions or generalizations of models

I find them primarily useful for visualizing and mapping out relationships between parameters and data

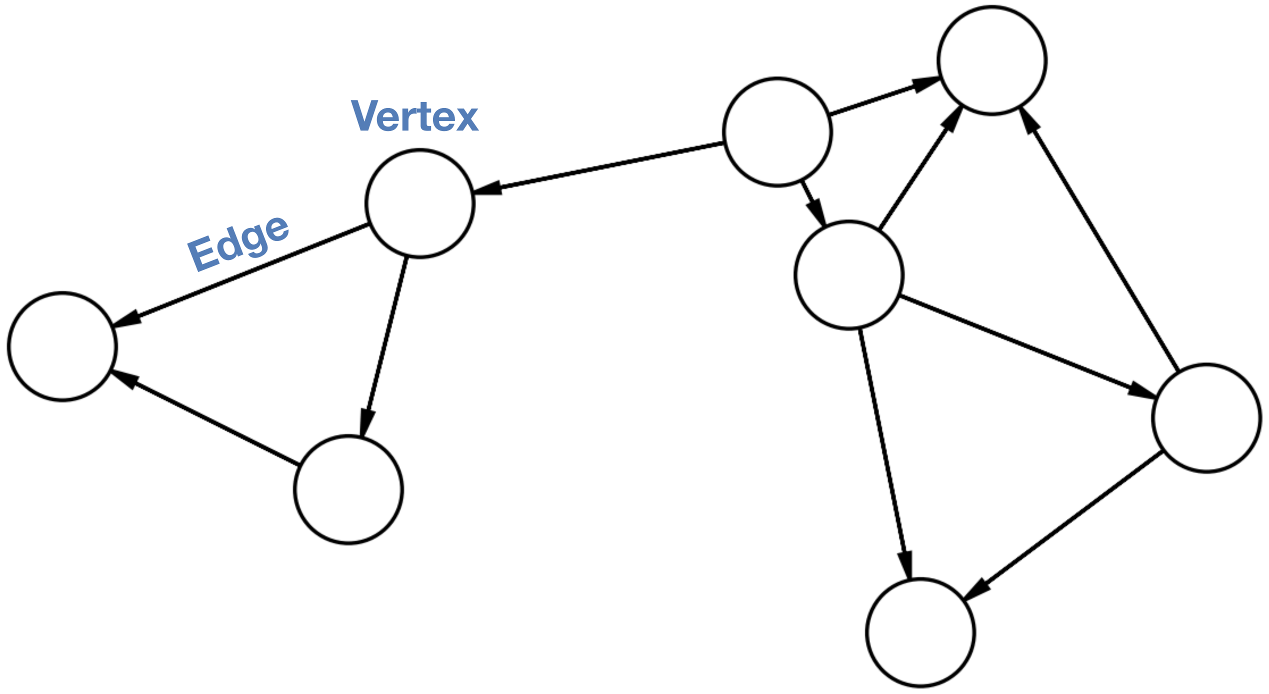


Graph Theory

A *graph* is a mathematical structure for expressing relationships between objects that together might belong to a larger network

A *graph* is made of two main types of objects:

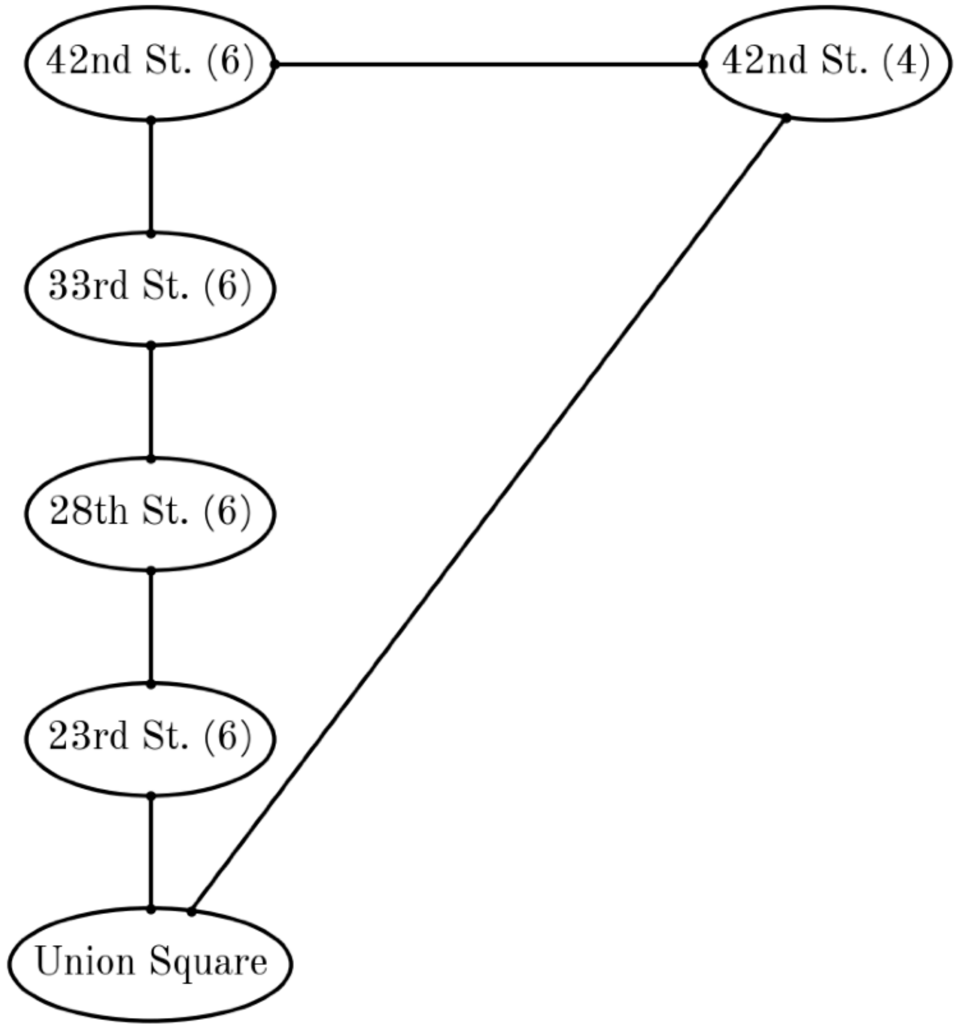
- **Vertex**: The vertices in a graph represent objects (sometimes also called *nodes*)
- **Edge**: The edges of a graph represent the relationships between objects. Edges can either be *directed* or *undirected*



Undirected Graphs

Used when there is no sense of direction or hierarchy between vertices (objects).

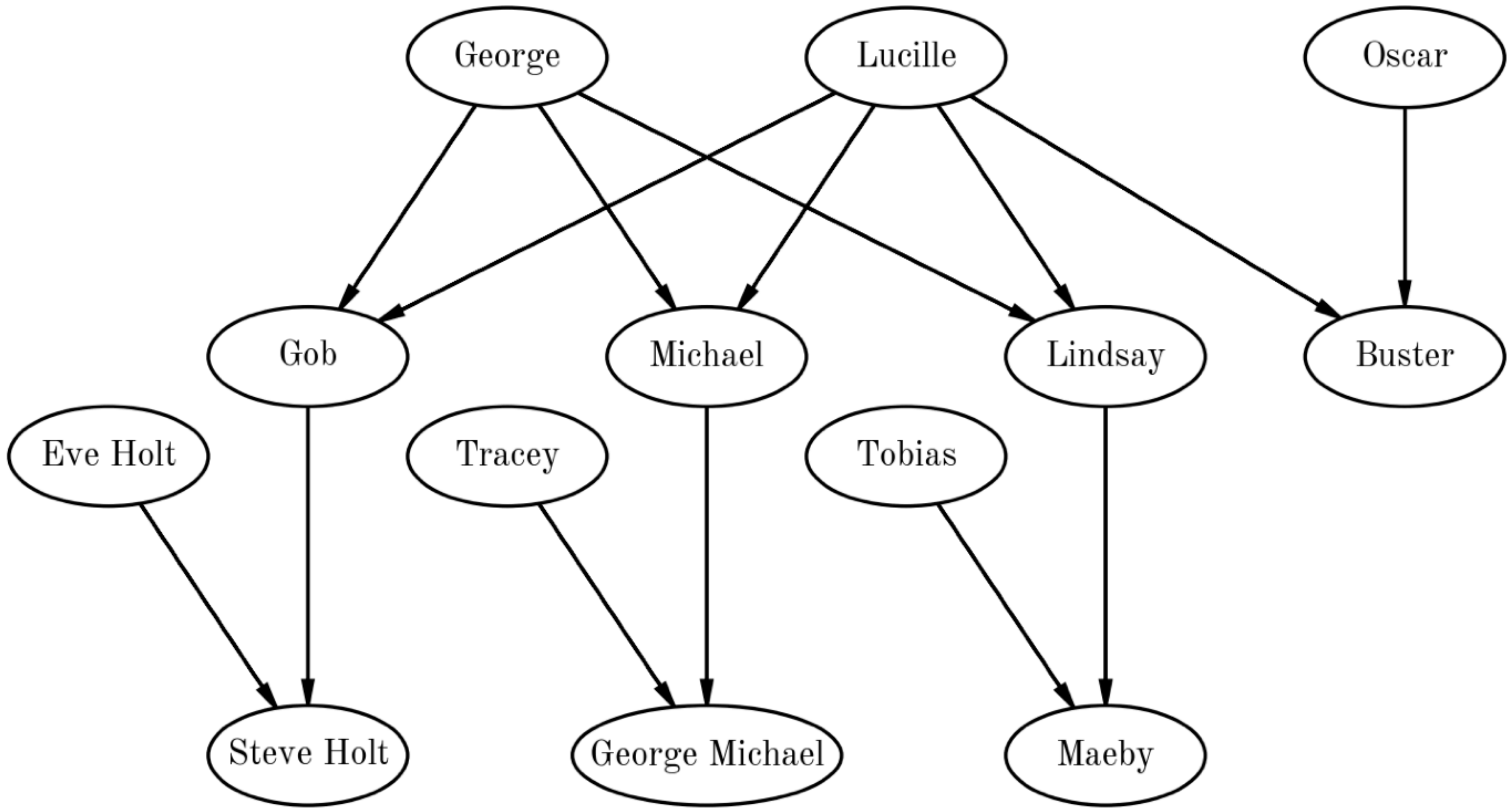
For example, a subway map: You can move either way between stations, and the most important thing to represent are the connections between stations or subway lines.



Directed Graphs

Used when there is inherent directionality or asymmetry in the relationships between vertices.

For example, a family tree: The people in a family are the vertices, and the edges represent generational relationships

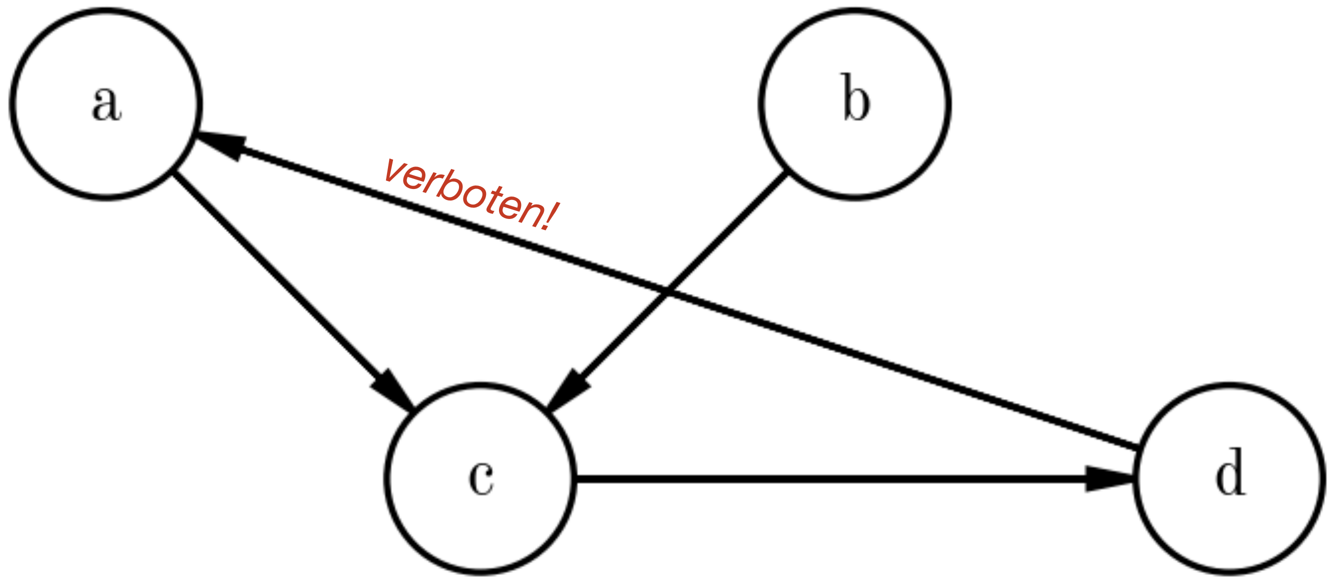


Directed Acyclic Graphs (DAGs)

In all model-building contexts I have come across, I have used directed *acyclic* graphs to represent models:

> **Acyclic** refers to the fact that there are no closed loops (i.e. cycles) in the graph.

A closed cycle (not allowed in many PGMs) would look like:



PGMs as DAGs

In the context of probabilistic models, the **vertices** (nodes) in a graph represent *parameters* or *data*

The **edges** represent *relationships* between the parameters / data

Importantly: A graph representation of a probabilistic model gives a map of ways you can *factorize the joint probability* distribution over all of the parameters and data

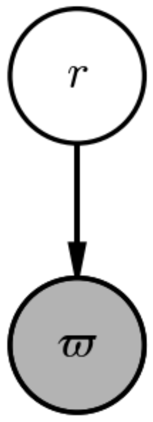
Example: Distance Given Parallax

Let's start with an example that is relevant in the *Gaia* era:

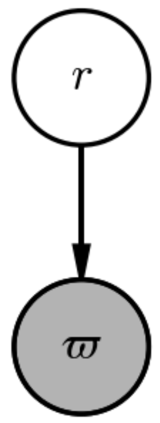
Infer the distance r to a star given its observed parallax ϖ, where

$$\varpi = 1/r$$

In this case, the only unobserved random variable (i.e. parameter) in our model is the distance, so a PGM for this model is simple:



Example: Distance Given Parallax

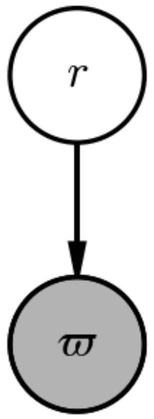


The gray vertex here uses a common convention to indicate *observed* quantities (i.e. ϖ is data, not a random variable)

Unobserved ("latent") random variables are unfilled circles

Earlier I stated "a *PGM tells us ways we can factorize the joint probability distribution*" — what does that mean for this example?

Example: Distance Given Parallax



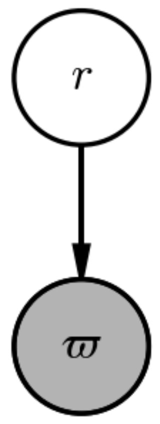
The joint pdf for this model is:

$$p(r, \varpi)$$

The PGM tells us that ϖ depends on the value of r, but r has no dependencies in this graph. This means:

$$p(r, \varpi) = p(\varpi \mid r)\, p(r)$$

Example: Distance Given Parallax



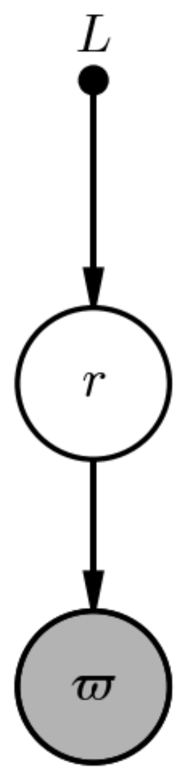
This is a valid PGM and we could stop here. But I want to introduce another convention you may see.

In the context of a probabilistic model, $p(r)$ is the prior pdf — this prior may have its own parameters that we fix to some values when running our inference method

For example, maybe $p(r) = \frac{1}{2\,L^3}\,\frac{r^2}{e^{r/L}}$ with $L = 1$

Example: Distance Given Parallax

You will sometimes see graphs with *fixed* nodes represented as closed, small markers like:



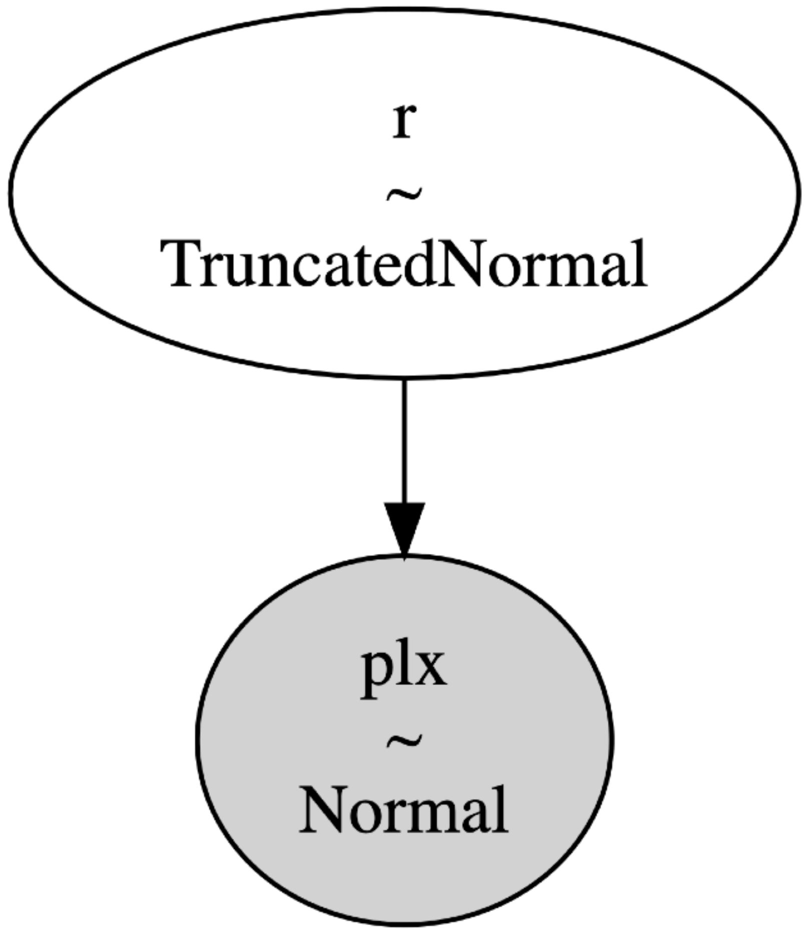
These fixed nodes are used for keeping track of parameters that are not random variables, and are not observed, but are fixed values that you may want to keep track of and visualize

Example: Distance Given Parallax

BTW: As you'll see in the accompanying Jupyter notebook, I drew these graphs manually using a Python package called daft

pymc will make PGM representations of your models automatically, they just aren't as visually nice, e.g.:

```
with model:
    pm.model_to_graphviz(model)
```



Problem 1: Inferring the tangential velocity and distance of a star

Scroll down in the accompanying notebook to Problem 1

Your task is to draw (with daft, pen and paper, or whiteboard) a PGM for a model to infer the tangential velocity and distance of a star given its observed parallax and total proper motion

Problem 2: The distance to a cepheid using a period–luminosity relation

Scroll down in the accompanying notebook to Problem 2

With daft, pen and paper, or whiteboard: draw a PGM for a model to infer the distance and luminosity of a Cepheid variable star given its observed mean (bolometric) flux and period

Note: the functional form doesn't really matter here, but the luminosity and flux are related via

$$\langle f \rangle = \frac{\mathcal{L}}{4\pi\, r^2}$$

Example: Fitting a sinusoid to data

We are given some time series data (e.g., fluxes in a light curve) that we think displays sinusoidal variability

We assume that the times are known perfectly (no uncertainty), but the fluxes have Gaussian uncertainties

Our dataset is then: times t_n, fluxes f_n, and flux uncertainties σ_n where the subscript indexes the separate observations in the time series.

Our goal is to infer period, amplitude, and mean value of the flux

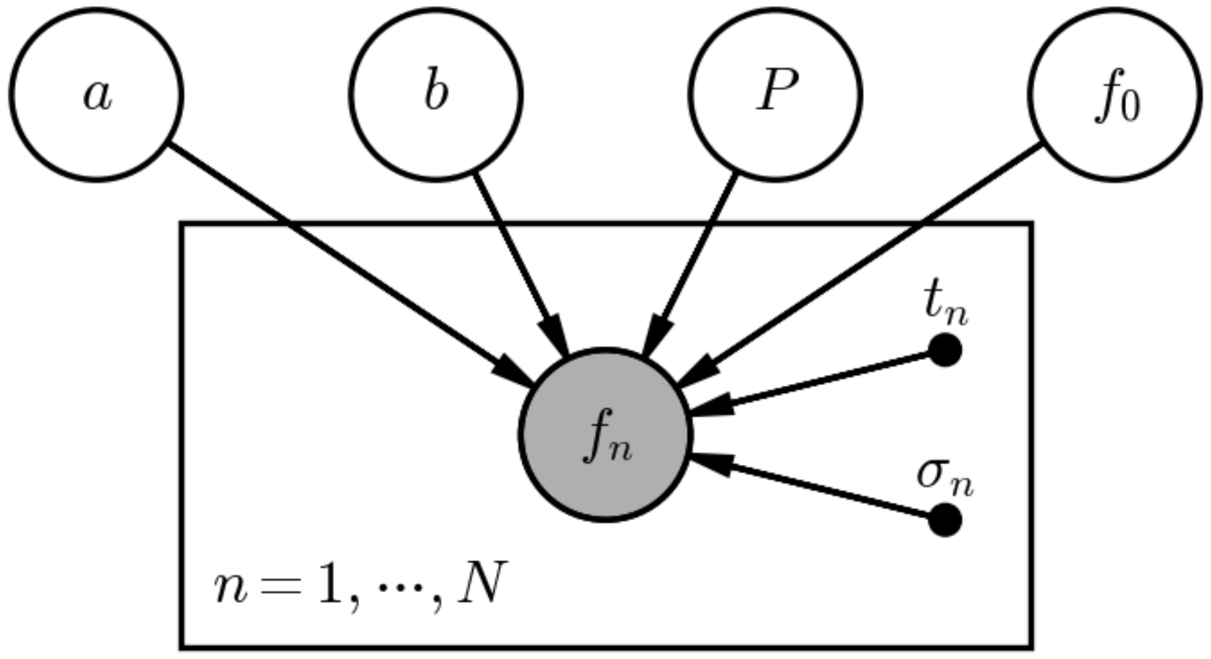
The parametric model we will use is:

$$f(t) = f_0 + a\,\cos\left(\frac{2\pi\,t}{P}\right) + b\,\sin\left(\frac{2\pi\,t}{P}\right)$$

Example: Fitting a sinusoid to data

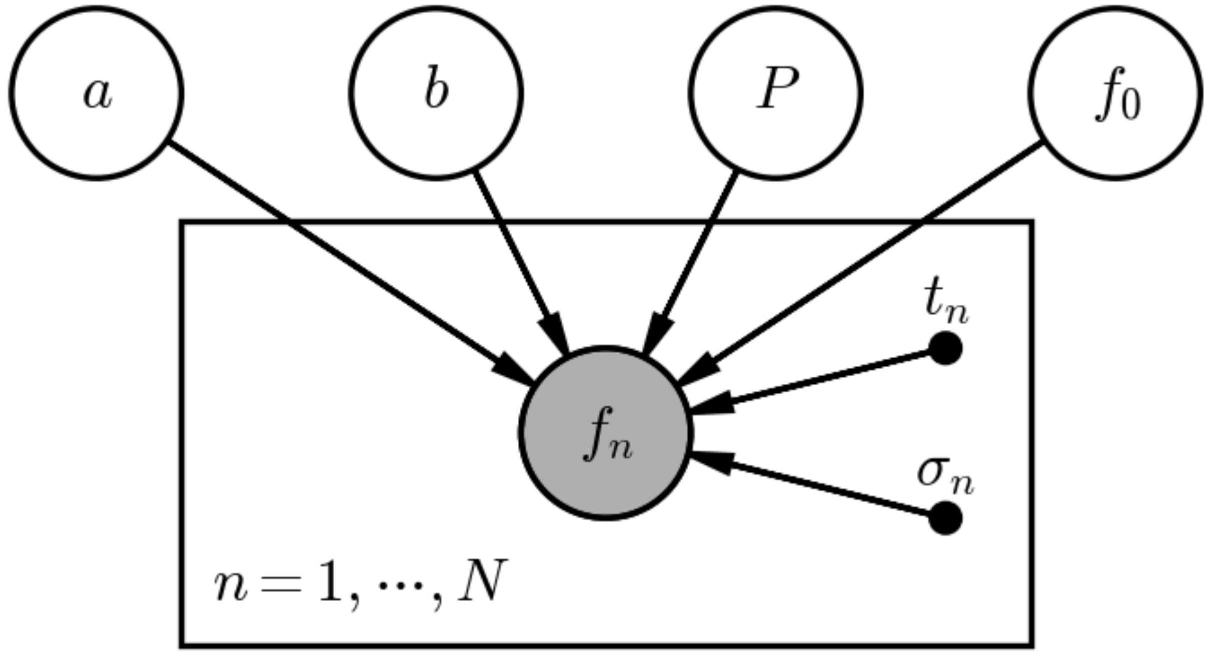
$$f(t) = f_0 + a \cos\left(\frac{2\pi t}{P}\right) + b \sin\left(\frac{2\pi t}{P}\right)$$

Note that this has 4 parameters: f_0, a, b, P



The box or *plate* denotes that the interior part of the model should be repeated some number of times (here, *N* times)

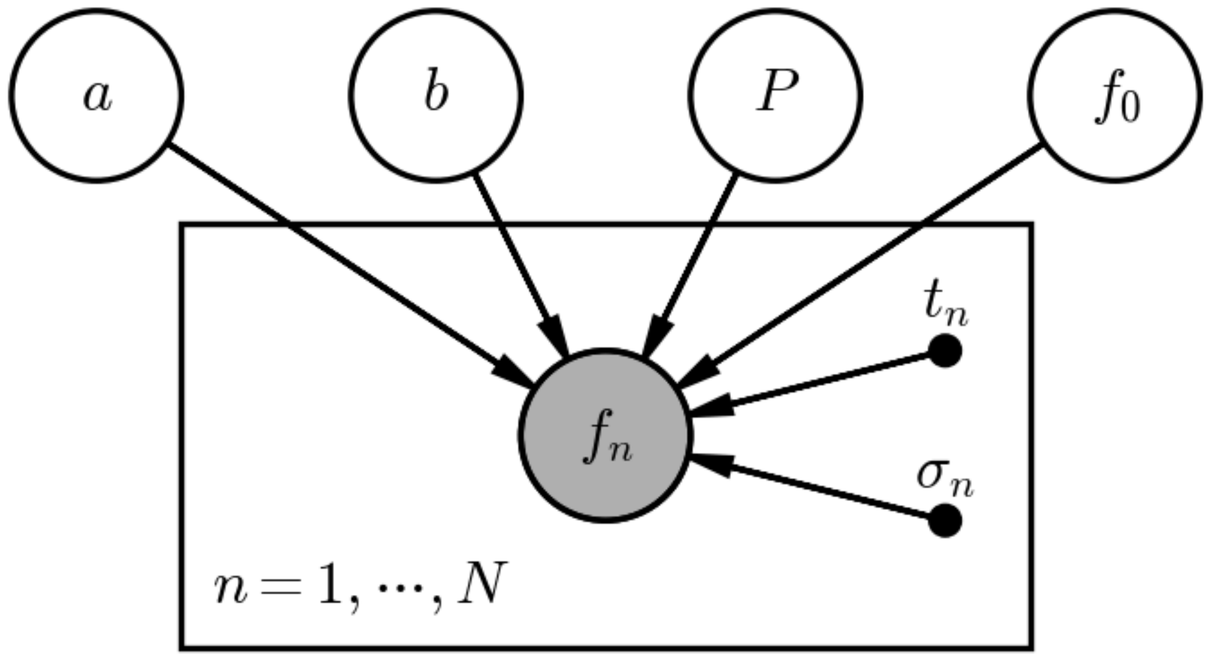
Example: Fitting a sinusoid to data



Question: What does this tell us about factorizing the joint pdf?

$$p(a, b, P, f_0, \{f_n\}_N)$$

Example: Fitting a sinusoid to data



Question: What does this tell us about factorizing the joint pdf?

$$p(a, b, P, f_0, \{f_n\}_N) = p(a)\, p(b)\, p(P)\, p(f_0) \prod_n^N p(f_n | a, b, P, f_0)$$

Problem 3: Fitting a straight line to data

Scroll down in the accompanying notebook to Problem 3

With daft, pen and paper, or whiteboard: draw a PGM for the straight line model you saw yesterday

(see the problem specification in the notebook for more prompt)

Problem 4: Inferring the radial velocity from an Echelle spectrum

This is an advanced problem! If you get stuck, that's totally fine

Scroll down in the accompanying notebook to Problem 4

(see the problem specification in the notebook for more prompt)

Preview of tomorrow: Going Hierarchical

We now have worked with all of the key components of building a Graphical Model: *random variable vertices*, *edges*, *plates*, *fixed vertices*, and *observed vertices*

Well done!

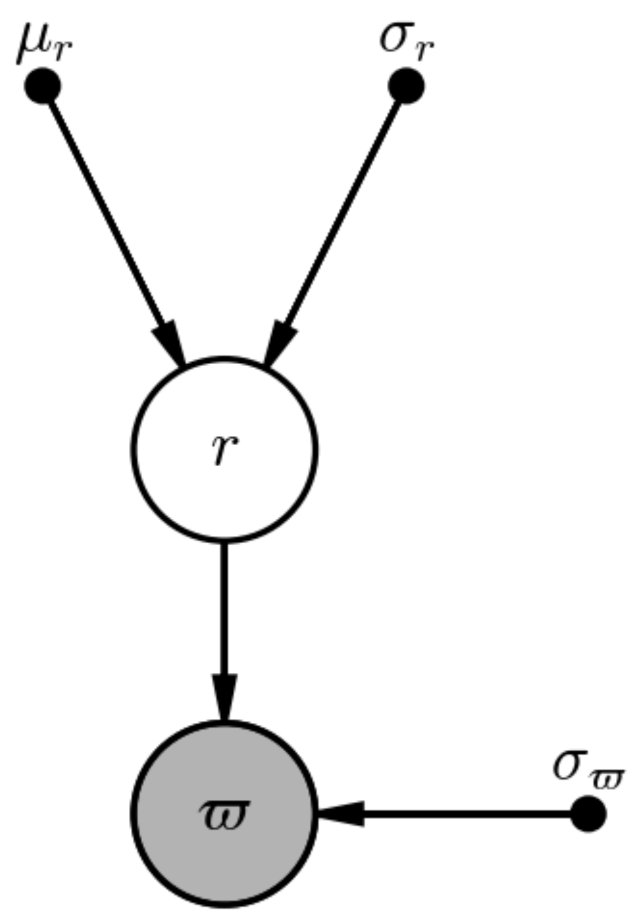
As a final example, let's look at how we might extend a one-level model to become a *hierarchical model*

Let's return to the example of inferring the distance to a star given its parallax

Example: The mean distance of a star cluster

Still thinking about a single star, imagine now that our prior on the distance is a Gaussian with mean μ_r and standard deviation σ_r, both fixed to the known distance and size of a star cluster

A PGM for this model might look like:

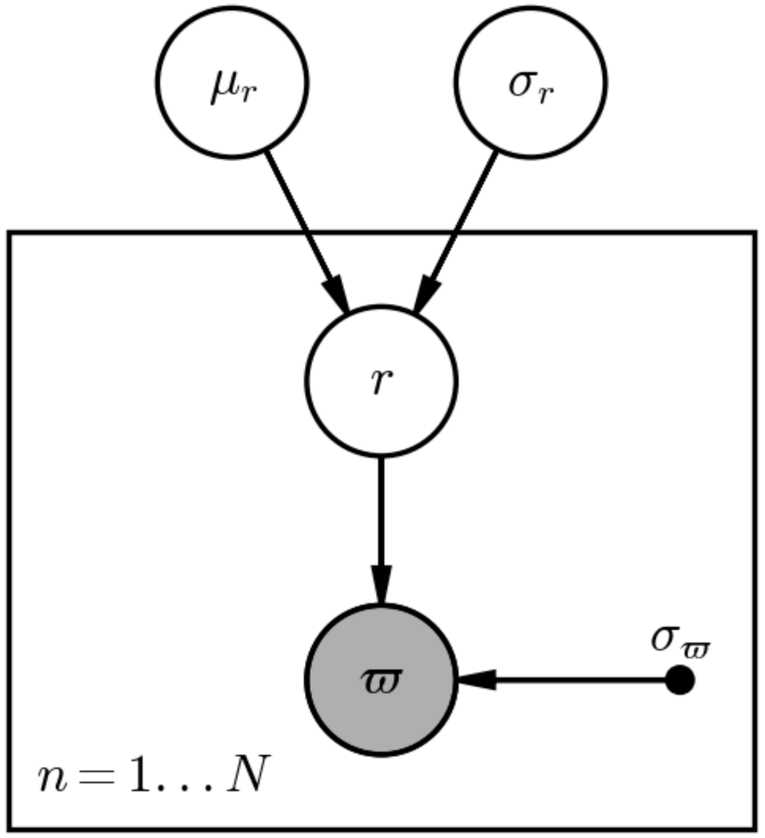


Example: The mean distance of a star cluster

Now assume we observe *N* stars in the cluster and we want to infer its distance and size from these stars

To represent this model, we need to:

1. add a plate around the distance and parallax vertices (we will have a distance and parallax for each star), and
2. make the cluster mean and standard deviations random variables instead of fixed nodes:


μ_r
σ_r
r
ϖ
σ_ϖ
$n = 1 \ldots N$