SDSS Lecture Solutions

Bayes Intro

Placenta previa example

```
from scipy.stats import beta, binom

x = np.linspace(0, 1, 1000)
y_prior = beta.pdf(x, 1, 1)
y_post = beta.pdf(x, 438, 544)
like = binom.pmf(k=437, n=980, p=x)
like *= y_post.max() / like.max()



plt.figure(figsize=(10, 5))
plt.plot(x, y_prior, label='Prior', linewidth=5)
plt.plot(x, like, label='Likelihood', linewidth=5)
plt.plot(x, y_post, label='Posterior', linewidth=5, alpha=0.4,
    linestyle='dashed')
plt.legend()

x = np.linspace(0, 1, 1000)
y_prior = beta.pdf(x, 48.5, 51.5)
y_post = beta.pdf(x, 437+48.5, 543+51.5)
like = binom.pmf(k=437, n=980, p=x)
like *= y_post.max() / like.max()



plt.figure(figsize=(10, 5))
plt.plot(x, y_prior, label='Prior', linewidth=5)
plt.plot(x, like, label='Likelihood', linewidth=5)
plt.plot(x, y_post, label='Posterior', linewidth=5, alpha=0.4,
    linestyle='dashed')
plt.legend();
```

PyMC Intro

LD50 Model

```
invlogit = pm.math.invlogit

with pm.Model() as bioassay:

    a = pm.Normal('a', 0, 5)
    b = pm.Normal('b', 0, 5)

    p = invlogit(a + b*np.array(log_dose))

    y = pm.Binomial('y', n=n, p=p, observed=np.array(deaths))

    LD50 = pm.Deterministic('LD50', -a/b)
```

MCMC

Random walk MCMC

1. The Metropolis-Hastings acceptance criteria simplifies quite a bit - work out what $A(x_{t + 1} | x_t)$ is now.
2. scipy.stats is doing a lot of work: `st.norm().rvs()` is ~1000x slower than `np.random.randn()`. Rewrite `metropolis_hastings` with the acceptance criteria, and without using `st.norm().rvs()` to provide proposals.

```
def rw_metropolis_hastings(pdf, proposal_sd, init=0):
    current = init
    while True:
        prop = current + np.random.randn() * proposal_sd
        # prop = st.norm(current, proposal_sd).rvs()
        p_accept = min(1, pdf(prop) / pdf(current))
        accept = np.random.rand() < p_accept
        if accept:
            current = prop

        yield current, accept
```

```
pdf = mixture_of_gaussians()

samples, accepts = gen_samples(40_000, rw_metropolis_hastings(pd

fig, ax = plt.subplots(figsize=(10, 7))

t = np.linspace(samples.min(), samples.max(), 500)
ax.hist(samples, bins='auto', density=True)
ax.plot(t, pdf(t))

ax.set_title(f'{samples.size:,d} draws from the pdf with {100 *
```

Experimenting with step methods

```
with model:
    trace0 = pm.sample(draws=1000, nuts_sampler="numpyro")
az.plot_trace(trace0, var_names=['early_lambda'])
```

```
with model:
    trace1 = pm.sample(draws=1000, step=pm.Slice())
az.plot_trace(trace1, var_names=['early_lambda'])
```

```
with model:
    trace2 = pm.sample(draws=1000, step=pm.Metropolis())
az.plot_trace(trace2, var_names=['early_lambda'])
```

Speed test

```
with model:
    trace_numpyro = pm.sample(draws=500, tune=2000, target_accep
az.plot_trace(trace_numpyro, var_names='early_lambda')
```

Hierarchical Models

Varying slope model

```
with pm.Model(coords=coords) as varying_slope:

    # Priors
    mu_b = pm.Normal('mu_b', mu=0., sigma=1e5)
    sigma_b = pm.HalfCauchy('sigma_b', 5)

    # Common intercepts
    mu = pm.Normal('mu', mu=0., sigma=1e5)
    # Random slopes
    beta = pm.Normal('beta', mu=mu_b, sigma=sigma_b, dims='count

    # Model error
    sigma_y = pm.HalfCauchy('sigma_y',5)

    # Expected value
    y_hat = mu + beta[county] * floor_measure
    # Data likelihood
    y_like = pm.Normal('y_like', mu=y_hat, sigma=sigma_y, obser
```

Non-centered varying intercept and slope model

```
with pm.Model(coords=coords) as varying_intercept_slope_noncente

    mu_a = pm.Normal('mu_a', mu=0., sigma=1e5)
    sigma_a = pm.HalfCauchy('sigma_a', 5)

    mu_b = pm.Normal('mu_b', mu=0., sigma=1e5)
    sigma_b = pm.HalfCauchy('sigma_b', 5)

    # Random intercepts
    z_a = pm.Normal('z_a', mu=0, sigma=1, dims='county')
    mu = pm.Deterministic('mu', mu_a + z_a * sigma_a, dims='cour
    # Random slopes
    z_b = pm.Normal('z_b', mu=0, sigma=1, dims='county')
    beta = pm.Deterministic('beta', mu_b + z_b * sigma_b, dims=

    # Model error
    sigma_y = pm.Uniform('sigma_y', lower=0, upper=100)

    # Expected value
    y_hat = mu[county] + beta[county] * floor_measure

    # Data likelihood
    y_like = pm.Normal('y_like', mu=y_hat, sigma=sigma_y, obser

    pm.model_to_graphviz(varying_intercept_slope_noncentered)
```