Dirichlet process models

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

December 6, 2021

Outline

Last time:

- Bayesian neural networks

Today:

- Dirichlet processes

What's in a name?

Dirichlet processes – the name gives us a clue:

- Process: we saw this before in Gaussian processes
  - Extend a finite-dimensional distribution to infinite dimensions
  - Prior for a random function with certain probabilistic properties
- Dirichlet: a family of distributions useful for modeling a discrete probability distribution
  - Generalization of the Beta distribution

What's in a histogram?

When we're exploring the properties of a new variable y, with an unknown distribution

$$y_i \sim f(y_i)$$

often the first thing we do is make a histogram:

and think of this as en estimate of the underlying density.

What's in a histogram?

We don't usually think about:

- A histogram is a random variable
- Take a different sample, get a different histogram
- For fixed bins, this is just a vector of probabilities (or counts)

What do we do with random variables?

- We set priors and estimate them using Bayes

The Bayesian histogram

- fix bin edges $\xi = (\xi_0, \xi_1, \ldots, \xi_k)$
- assign a probability π_i to each bin $[\xi_{i-1}, \xi_i]$

Then our histogram is analagous to a piecewise constant density estimate

$$\tilde{f}(y) = \sum_{i=1}^{k} \mathbf{1}_{[\xi_{i-1}, \xi_i]} \frac{\pi_i}{(\xi_i - \xi_{i-1})}$$

Alternatively,

$$\tilde{f}(y) = \begin{cases} \frac{\pi_1}{(\xi_1 - \xi_0)} & \xi_0 < y < \xi_1 \\ \ldots & \\ \frac{\pi_k}{(\xi_k - \xi_{k-1})} & \xi_{k-1} < y < \xi_k \end{cases}$$

Prior for the Bayesian histogram

All we need is a prior for the vector $\pi = (\pi_1, \ldots, \pi_k)$.

- $$\pi \sim \text{Dirichlet}(a_1, \ldots, a_k)$$
- This distribution is:

$$p(\pi|a_k) \propto \pi_1^{a_1} \pi_2^{a_2} \cdots \pi_k^{a_k}$$

- Dirichlet is to Multinomial as Beta is to Binomial; useful as a prior for a discrete/categorical probability distribution
- Posterior is

$$\pi|y \sim \text{Dirichlet}(a_1 + n_1, \ldots, a_k + n_k)$$

From Bayesian histograms to Dirichlet processes

The major weakness is the same as in intro stats:

- Highly sensitive to the bin edges ξ
  - Too few bin edges and too many are both problems
  - A small change in position of bin edges can cause big changes in the estimate, especially if the observed data is discrete (including rounded)
- Less obvious: only allows a piecewise constant density estimate

The goal of the Dirichlet process is to generalize the histogram and free us from these problems

Defining the Dirichlet process

So, what can we preserve without specifying ξ or the form of the density estimate?

- Start with an arbitrary base distribution P_0 and a precision parameter α
- A Dirichlet process $\mathcal{DP}(\alpha, P_0)$ is a distribution on the space of probability distributions with the property:
  - for *any* partition $B_1, B_2, \ldots, B_k$ of the sample space,

$$P(B_1), \ldots, P(B_k) \sim \text{Dirichlet}(\alpha P_0(B_1), \ldots, \alpha P_0(B_k))$$

- What do draws from this distribution actually look like?

The stick-breaking process

Realizations of a DP

1. Begin with a "stick" of length 1, representing the total probability to be assigned
2. Sample an atom θ_1 from P_0.
3. Sample a value V_1 from $\mathrm{Beta}(1, \alpha)$.
4. Assign $\pi_1 = V_1$.
    - Idea: we broke off a piece of the probability "stick" and assigned it to θ_1.
5. Start again, with the "stick" consisting of the remaining $1 - V_1$ probability; break off a proportion of that equal to $V_2 \sim \mathrm{Beta}(1, \alpha)$

Realizations of a DP

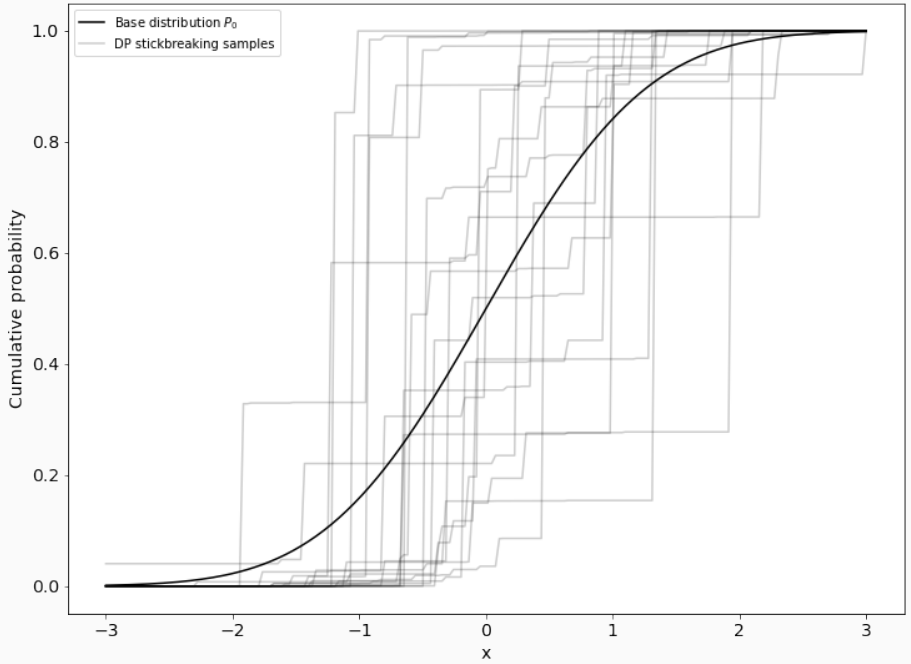
The resulting probability distribution:

$$
\begin{aligned}
p(y) &= \sum_{i=1}^{\infty} \pi_i \theta_i \\
\pi_i &= V_i \prod_{j=1}^{i-1} (1 - V_j) \\
V_i &\sim \mathrm{Beta}(1, \alpha) \\
\theta_i &\sim P_0
\end{aligned}
$$

is a draw from the $\mathcal{DP}(\alpha, P_0)$ prior.

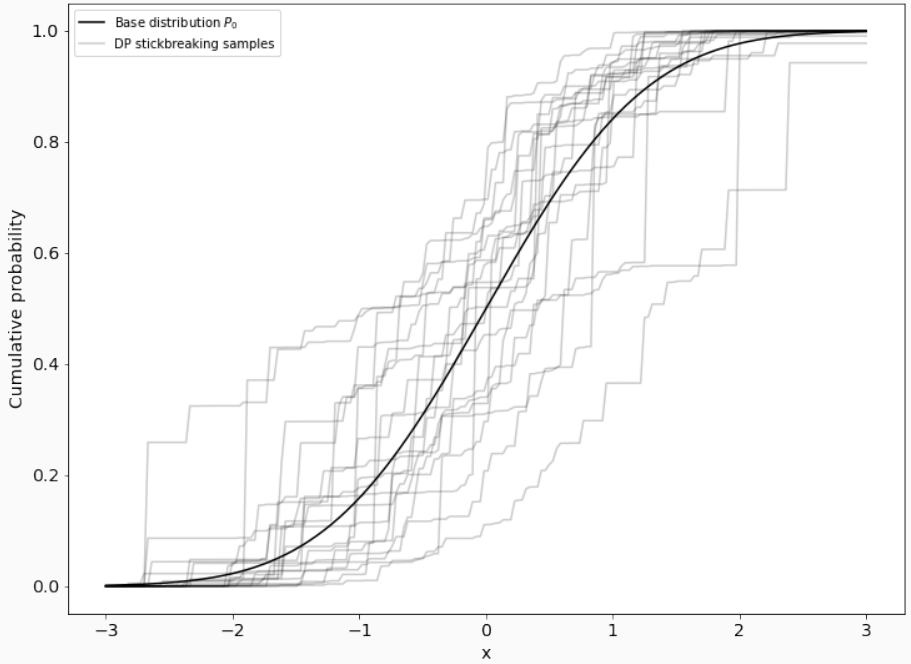
Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 1$



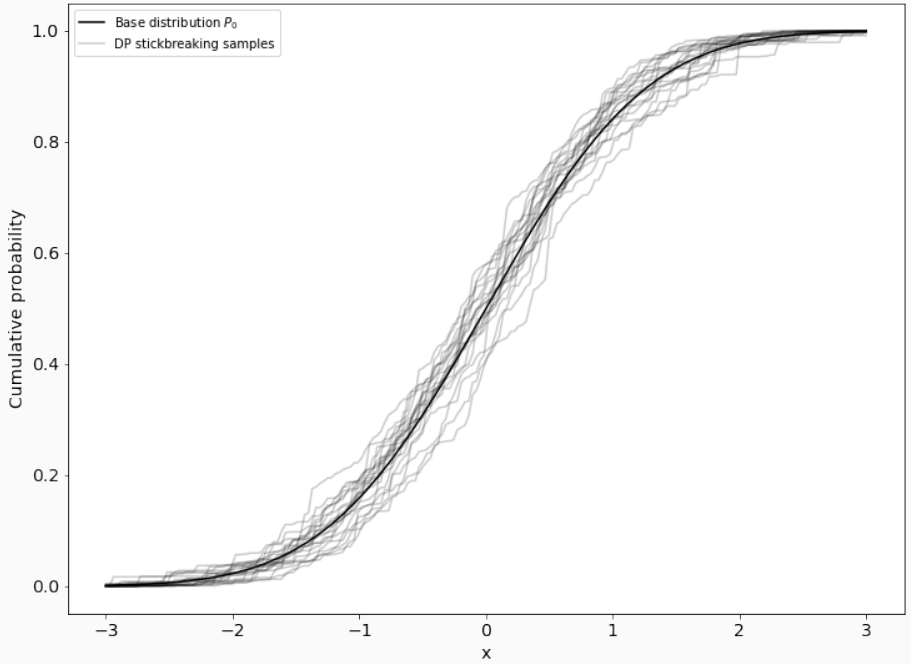
Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 10$



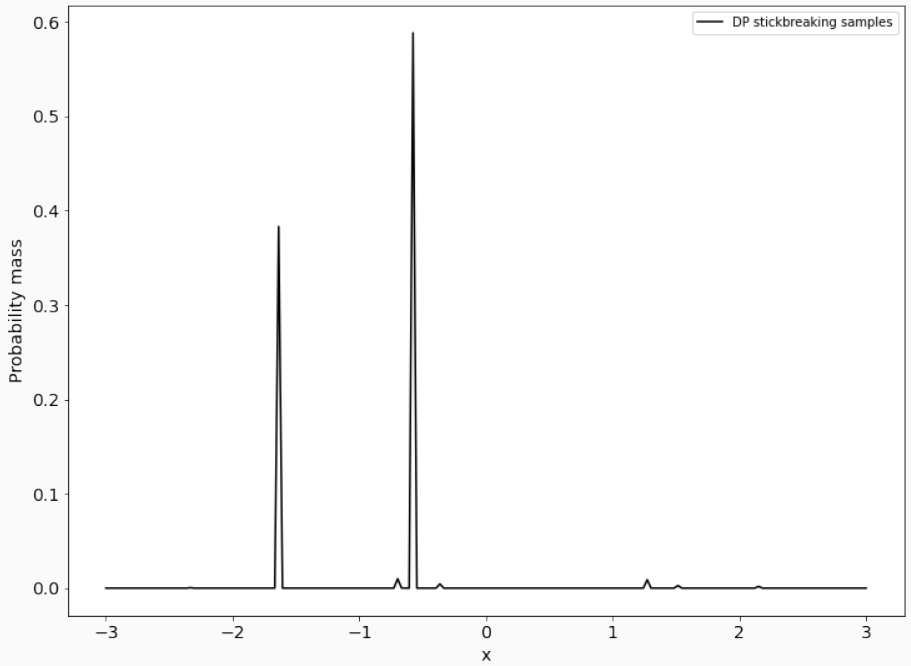
Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 100$



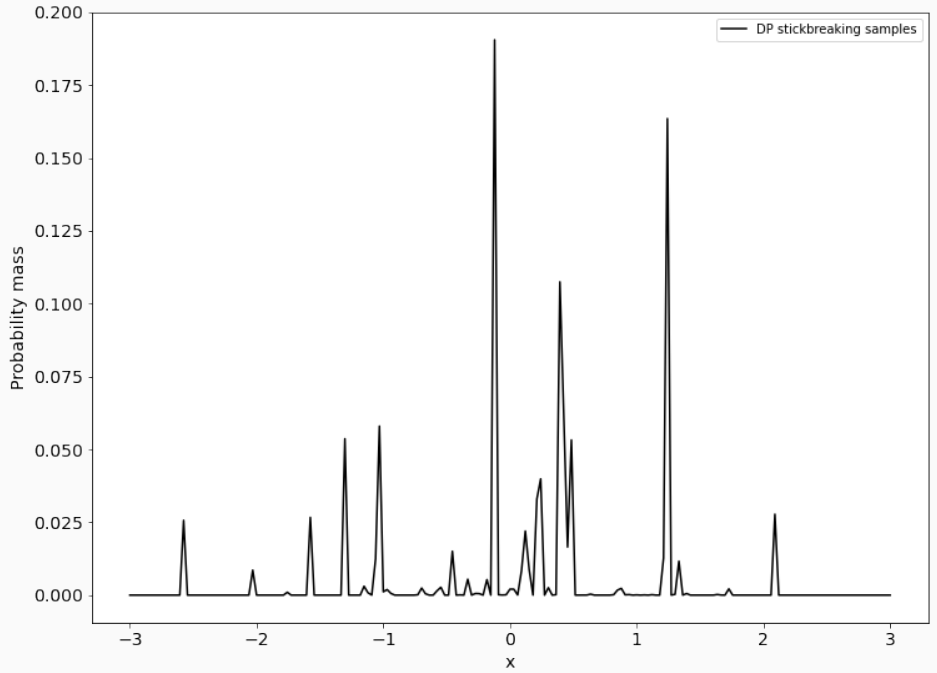
Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 1$



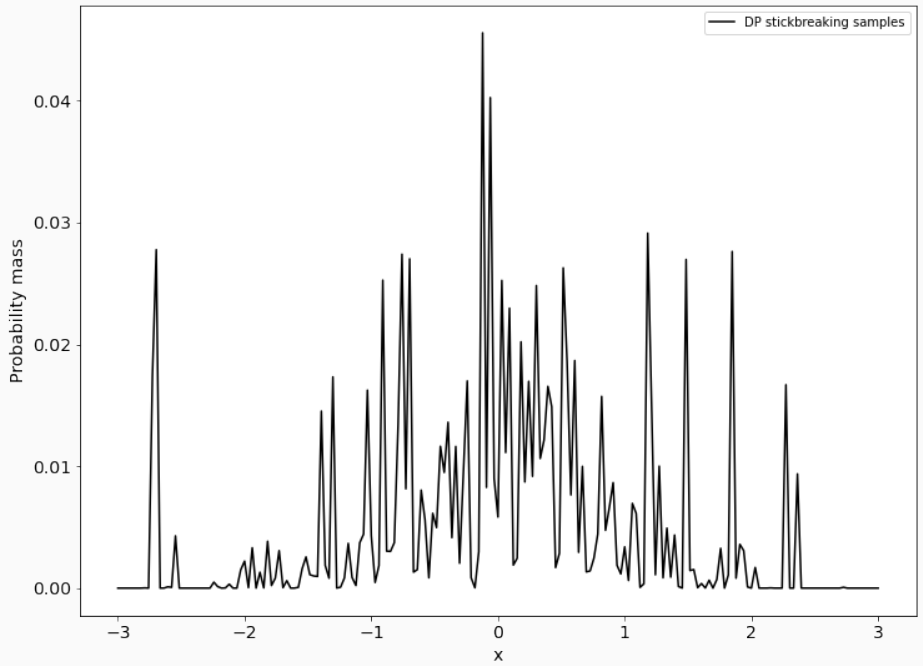
Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 10$



Trying out the stick-breaking process

- $P_0 = \text{Normal}(0, 1)$
- $\alpha = 100$



Dirichlet process mixtures

Making a mixture model

- Realizations from the stick-breaking process don't look like densities
- Even if $n \to \infty$, the resulting distribution does not have a continuous density function
- Alternative: build a mixture model. Estimated density is a weighted sum of (e.g.) normal densities, with the weights drawn from a DP
- A DP is like a histogram made Bayesian; a DP mixture is like a kernel density estimate made Bayesian

Making a mixture model

Broadly, a DP mixture is a hierarchical model, in which we mix components from a parametric family $\{f_\theta\}$:

$$
\begin{aligned}
y_i|\theta_i &\sim f_{\theta_i} \\
\theta_1, \ldots, \theta_n &\sim P \\
P &\sim \mathcal{DP}(\alpha, P_0)
\end{aligned}
$$

For example, a location mixture model of normals would have $f_\theta = \mathrm{Normal}(\theta, \sigma)$ for some fixed σ.

Fitting a DP mixture

In a sense, a DP mixture is an infinite mixture model:

- Allows for fitting a mixture model – such as a clustering model – without prespecifying the number of mixture components
- Problem: we can't actually store infinitely many mixture components
- There are clever Gibbs sampling tricks that allow adaptive modification of the number of components
- Simpler alternative: truncate the DP sum at some fixed K components, with K chosen large enough
    - Check K large enough by inspecting the posterior weights

Example

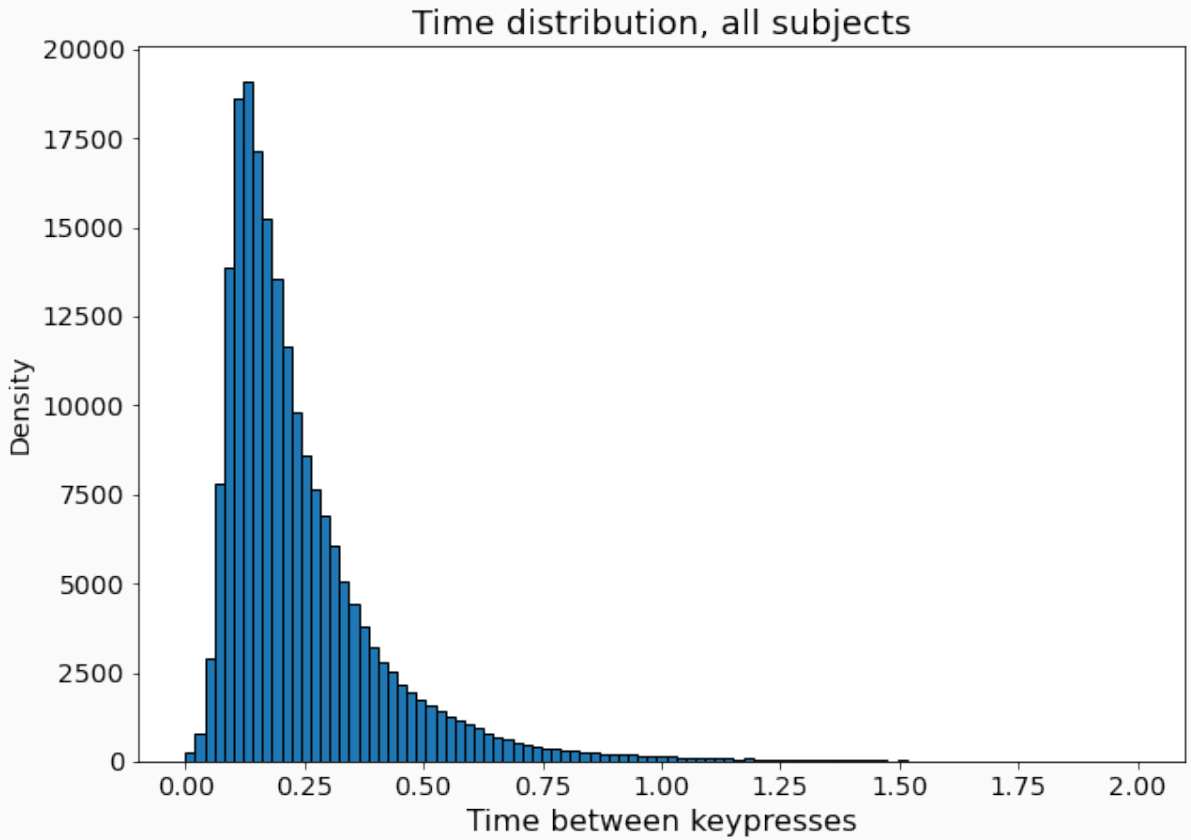
Example: analyzing keystroke rhythms for humans typing passwords

- Different people have different typing patterns
- Time between keypresses may be characteristic of a particular person
- More broadly, time between keypresses may characterize human typing patterns
- Data: from a CMU study[1]
  - 51 subjects typed a fixed password `.tie5Roanl` 400 times
  - Time between each keypress recorded

[1] Kevin S. Killourhy and Roy A. Maxion. "Comparing Anomaly Detectors for Keystroke Dynamics," in Proceedings of the 39th Annual International Conference on Dependable Systems and Networks (DSN-2009), pages 125-134, Estoril, Lisbon, Portugal, June 29-July 2, 2009. IEEE Computer Society Press, Los Alamitos, California, 2009.

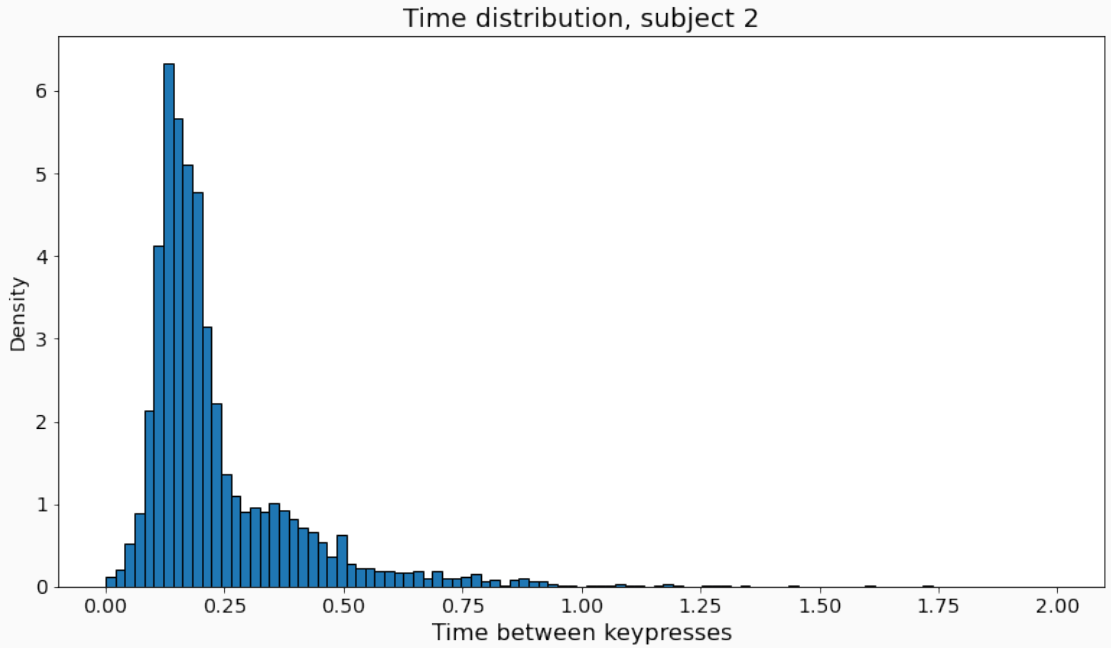
Exploring the data

Pooling together all time gaps and all subjects:



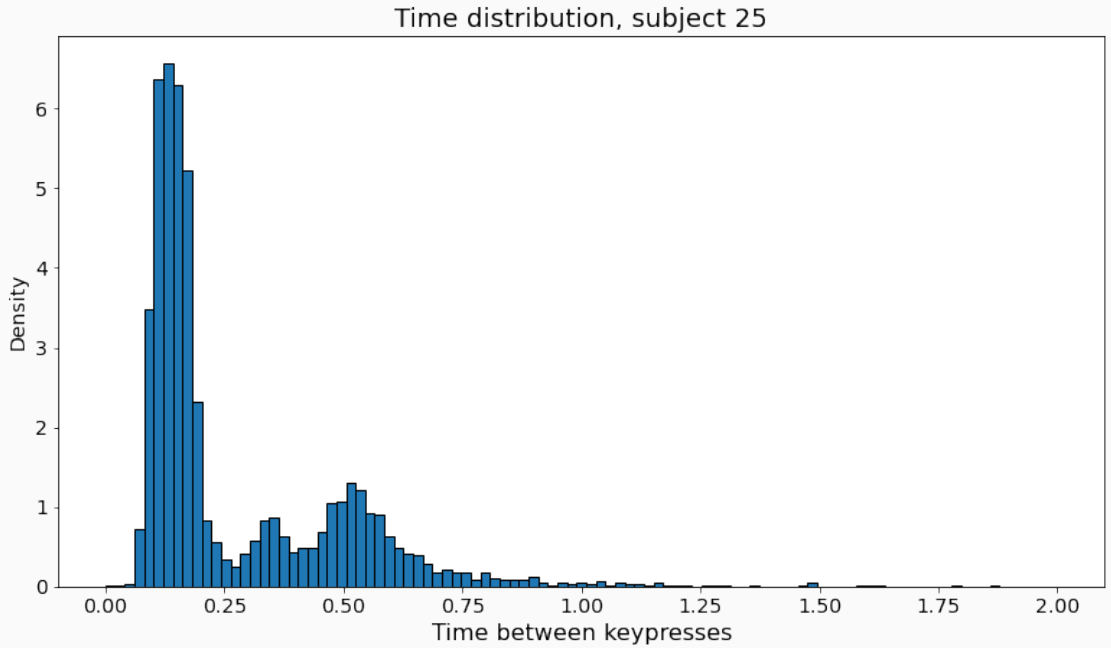
Exploring the data

However, histograms for time-between-keypress look pretty different for different subjects:



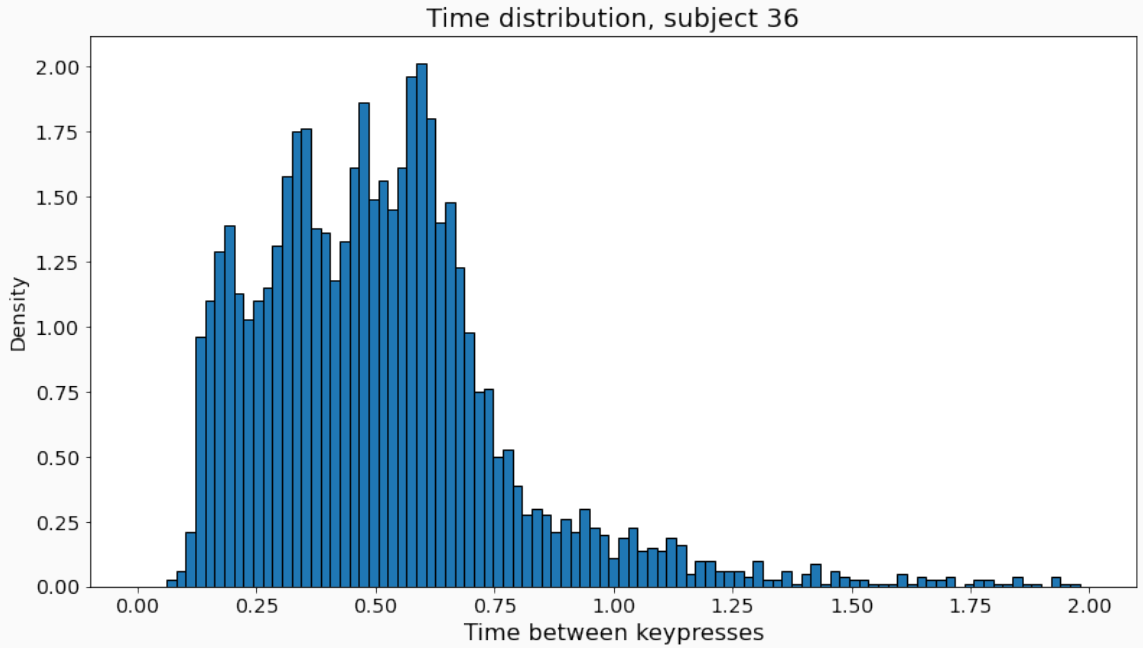
Exploring the data

However, histograms for time-between-keypress look pretty different for different subjects:



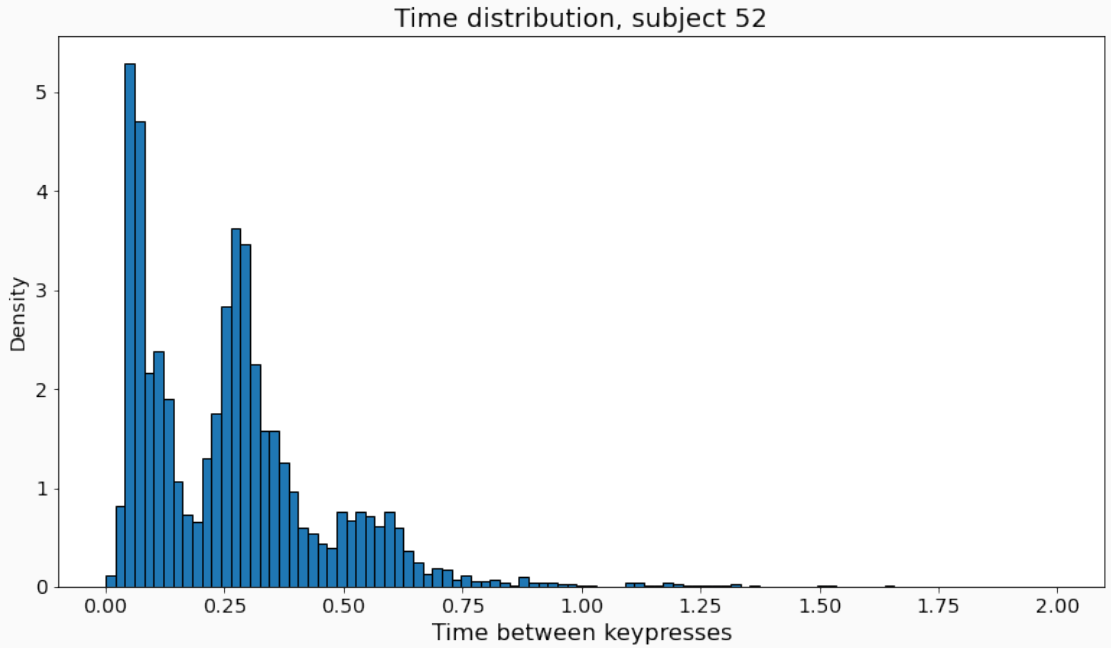
Exploring the data

However, histograms for time-between-keypress look pretty different for different subjects:



Exploring the data

However, histograms for time-between-keypress look pretty different for different subjects:



Building a model

We'll try to fit a mixture of normals:

$$
\begin{aligned}
y &\sim \sum_{i=1}^{K} w_i \text{Normal}(\mu_i, \sigma_i) \\
\mu_i &\sim \text{Normal}(0, \sigma_\mu) \\
\sigma_i &\sim \text{Exponential}(1) \\
\sigma_\mu &\sim \text{Exponential}(1) \\
w_i &\sim \text{stick}(i, \alpha) \\
\alpha &\sim \text{Gamma}(1, 1)
\end{aligned}
$$

with the second-to-last line meaning that the weights w_i come from the stick-breaking process with parameter α.

Model code

```
def stick_breaking(beta):
    portion_remaining = tt.concatenate([[1], tt.extra_ops.cumprod(1 - beta)[:-1]])

    return beta * portion_remaining

with pm.Model() as model:
    alpha = pm.Gamma("alpha", 1.0, 1.0)
    beta = pm.Beta("beta", 1.0, alpha, shape=n_components)
    w = pm.Deterministic("w", stick_breaking(beta))

    sigma_components = pm.Exponential("sigma_components", 1.0, shape=n_components)
    sigma_mu = pm.Exponential("sigma_mu", 1)
    mu = pm.Normal("mu", 0, sigma_mu, shape=n_components)
    obs = pm.NormalMixture(
        "obs",
        w,
        mu,
        sigma=sigma_components,
        observed=groups['s052'].drop('subject', axis = 1).values.reshape(-1, 1)
    )
```

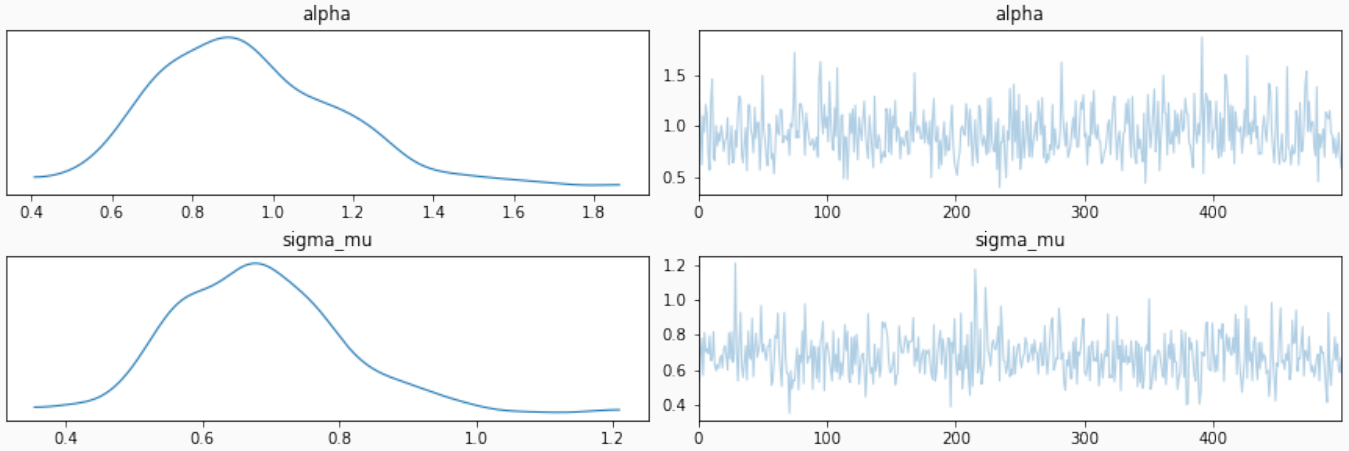
The mixture model

In summary:

- Mixture of Gaussians, each with independently drawn standard deviations σ_i
- Means of the Gaussians μ_i are drawn from a common $N(0, \sigma_\mu)$
- Weights drawn from a stick-breaking process with parameter α
- Underlying Dirichlet process has precision α and base distribution $P_0 = N(0, \sigma_\mu)$

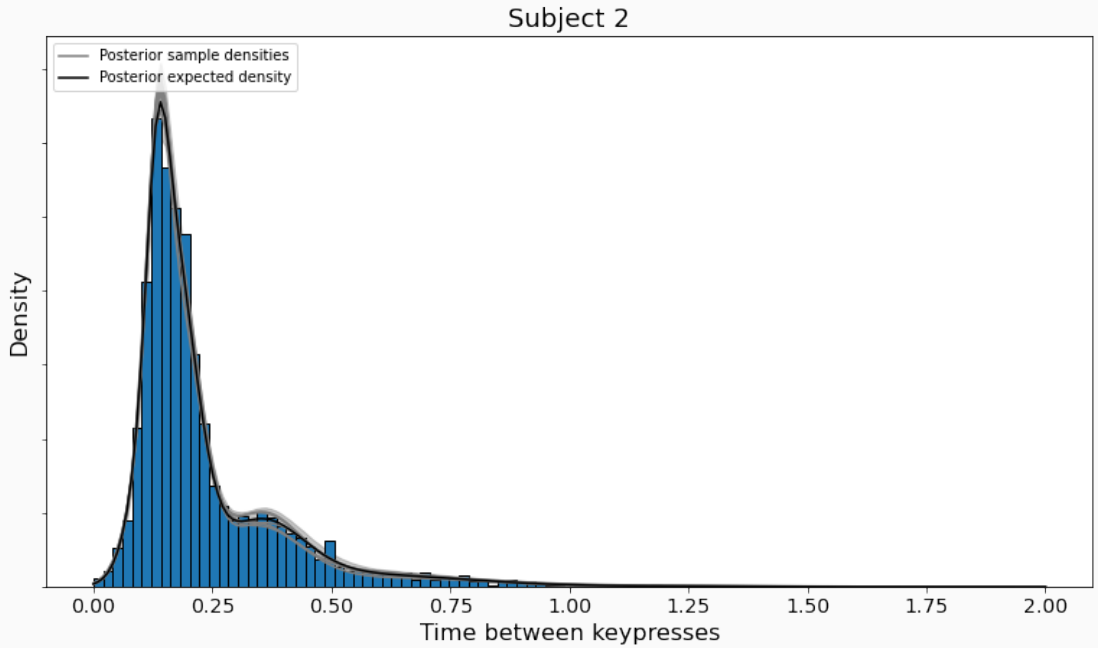
Fitting the model

- Fit the model using ADVI (MCMC is pretty slow here)
- Check out a traceplot



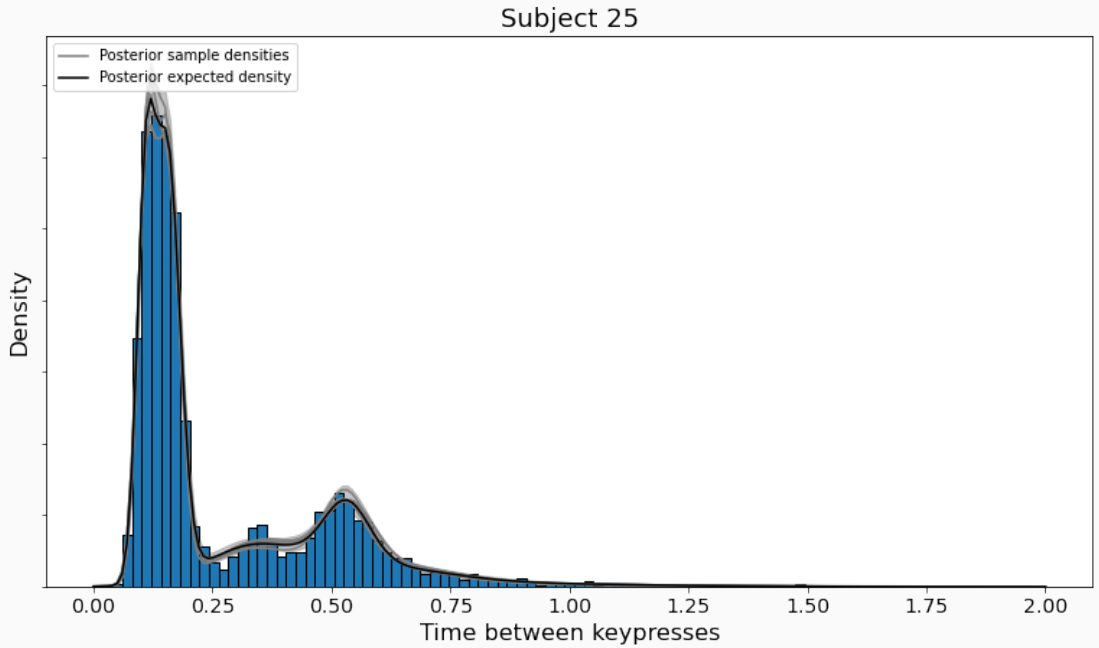
Exploring the data

DP mixture fit for keypress density:



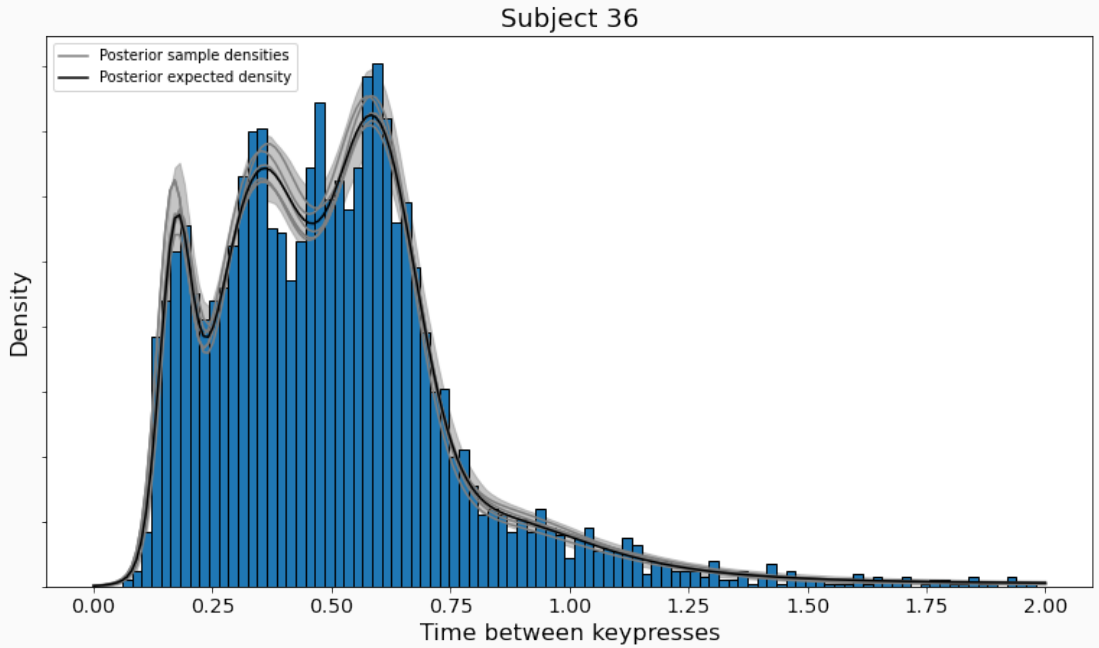
Exploring the data

DP mixture fit for keypress density:



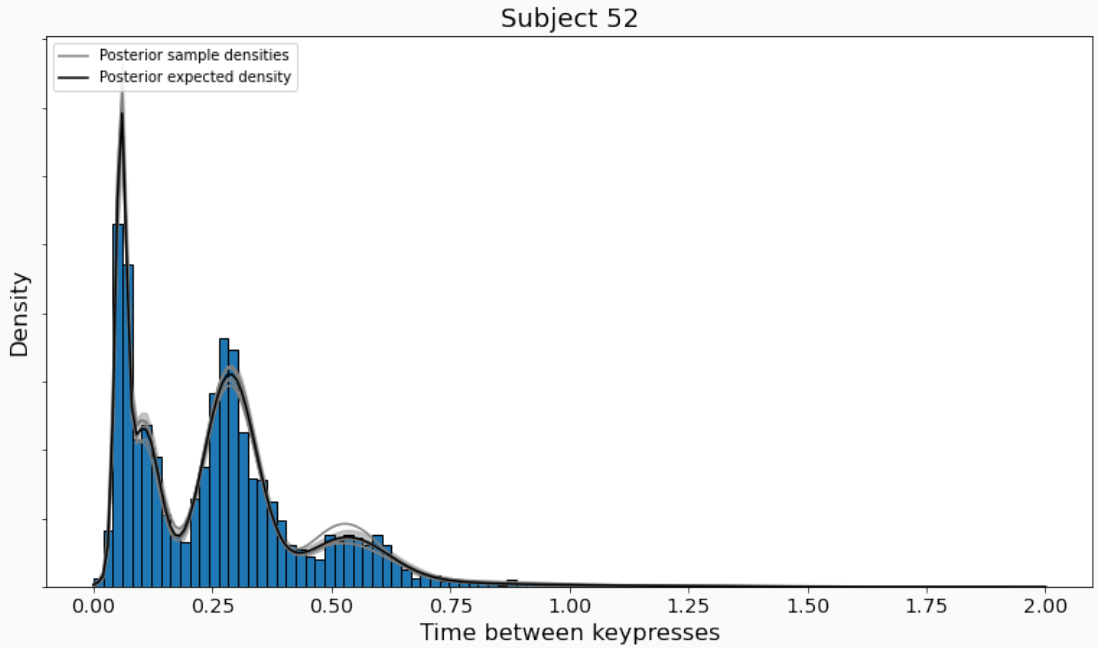
Exploring the data

DP mixture fit for keypress density:



Exploring the data

DP mixture fit for keypress density:



Checking the weights

- Instead of any clever adaptive tricks, we chose to simply truncate the Dirichlet process, limiting the number of components
- This is equivalent to replacing

$$p(y) = \sum_{i=1}^{\infty} \pi_i \theta_i$$

with

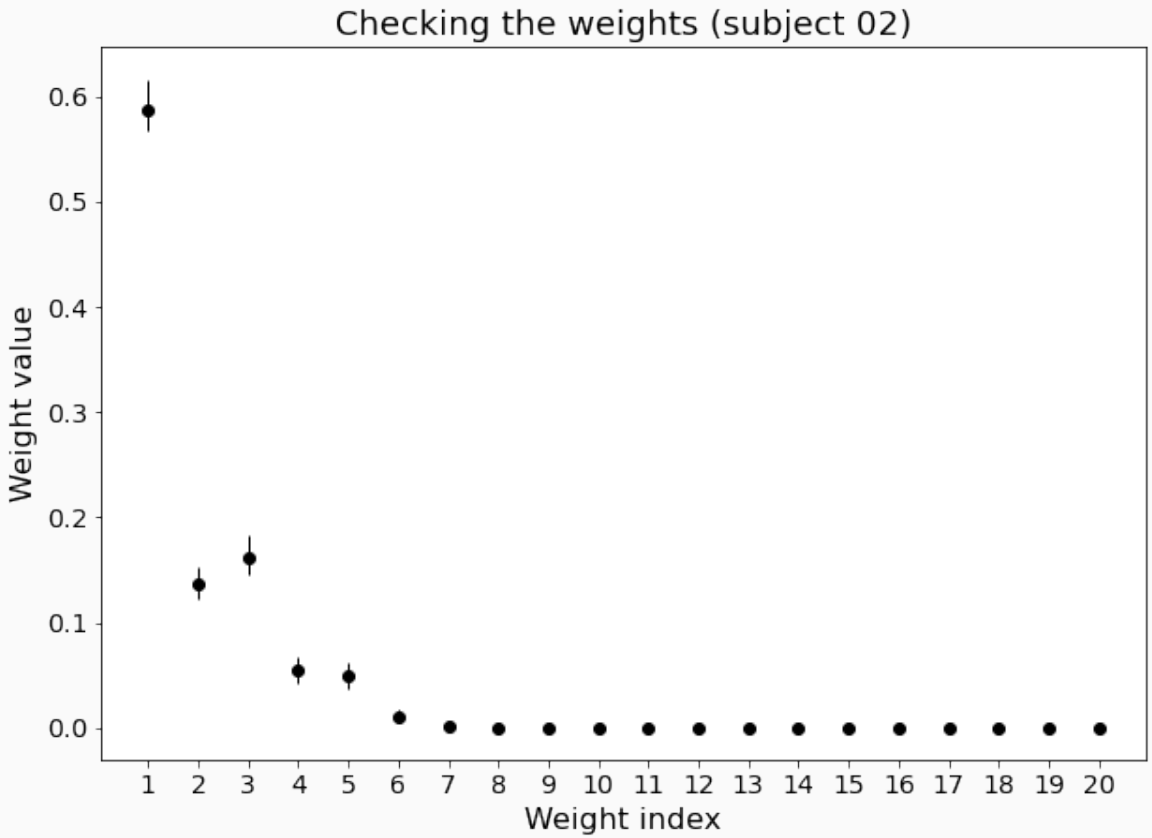
$$p(y) = \sum_{i=1}^{K} \pi_i \theta_i$$

in the stick-breaking process

- Did we get enough components?

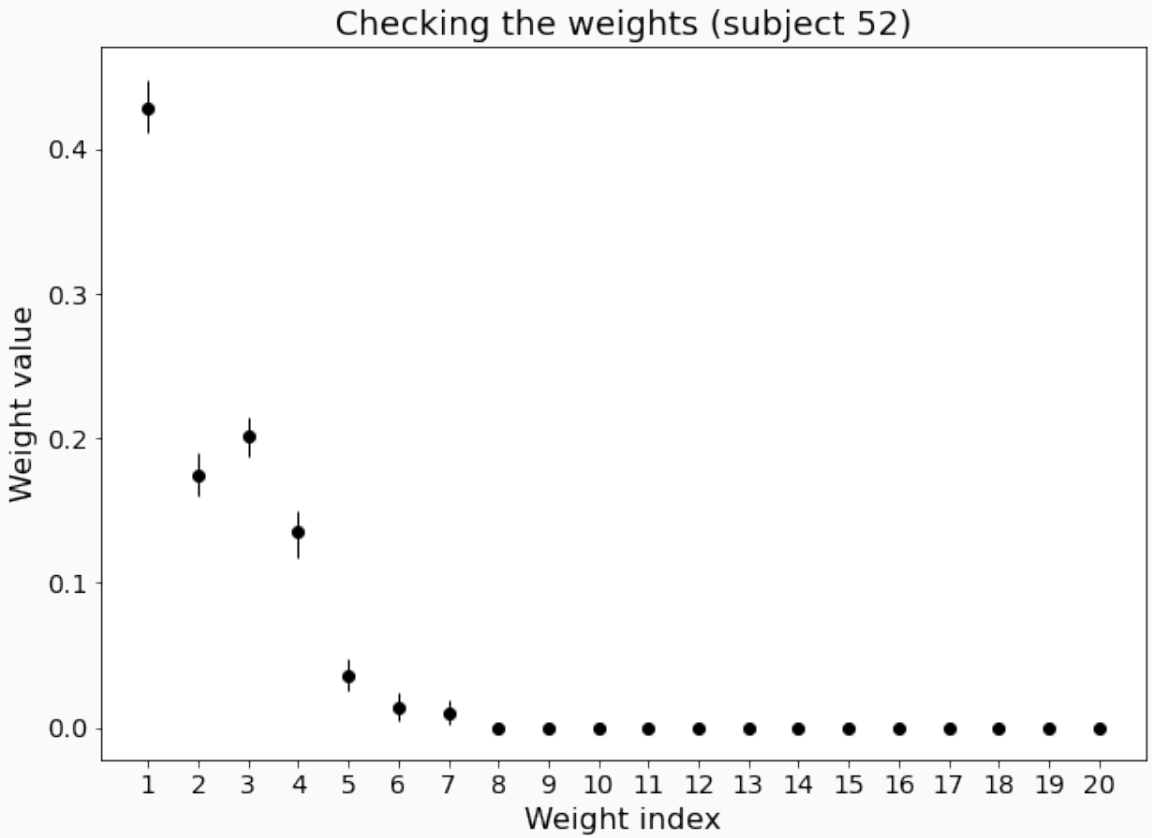
Checking the weights

Check that the higher-indexed components have negligible weight:



Checking the weights

Check that the higher-indexed components have negligible weight:



Summary

Today:

- Dirichlet process models
  - Flexible models for density estimation
  - Other applications in relaxing assumptions in hierarchical models
  - See BDA3 ch. 23 for examples

Next time:

- particle filter algorithms (aka sequential importance sampling)
- end of the semester!