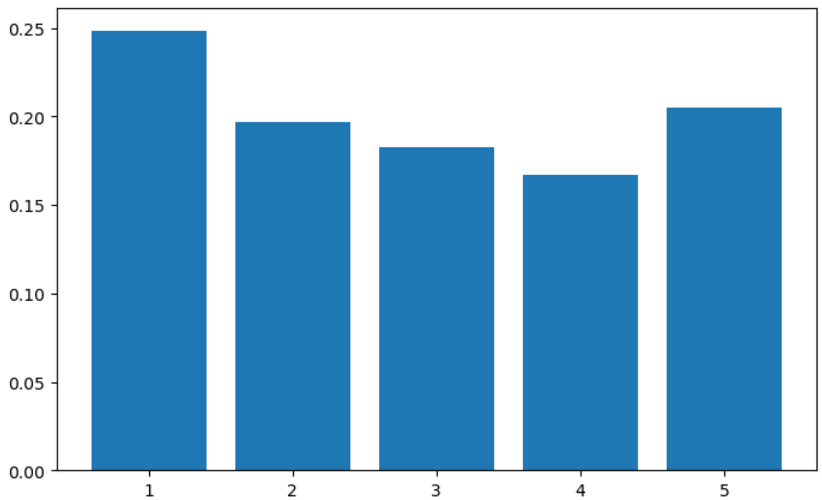
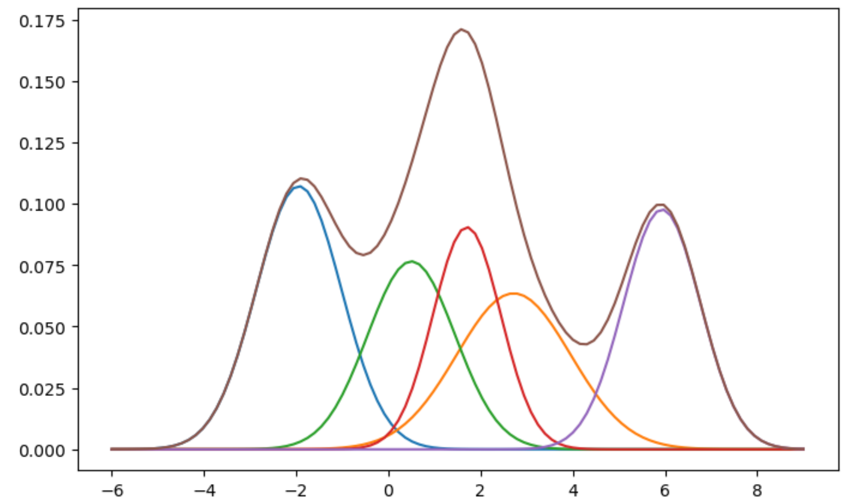
Class Activity 8: (Mixture Model) Preview [8 minutes]

In [1]:
```
import numpy as np; from scipy import stats; import matplotlib.pyplot as plt; np.random.seed(9)

k = 5
alpha = [2]*k
p_true = stats.dirichlet(alpha).rvs(1)[0] # p_true.sum() # 1

mu_k_true = stats.norm(0,3).rvs(k); support = np.linspace(-6,9,100); population_pdf = 0*support
sigma2_k_true = stats.halfnorm().rvs(k)

fig,ax = plt.subplots(1,2,figsize=(18,5)); ax[0].bar(x=np.linspace(1,5,5), height=p_true)
for j in range(k):
    subpopulation_pdf = p_true[j]*stats.norm(mu_k_true[j],sigma2_k_true[j]**0.5).pdf(support)
    ax[1].plot(support, subpopulation_pdf); population_pdf += subpopulation_pdf

ax[1].plot(support, population_pdf);
```





Class Activity 8: (Mixture Model) Preview [7 minutes]

In [2]:
```
n = 1000; v_true = stats.multinomial(n=1,p=p_true).rvs(n); v_true[:3,:]
```

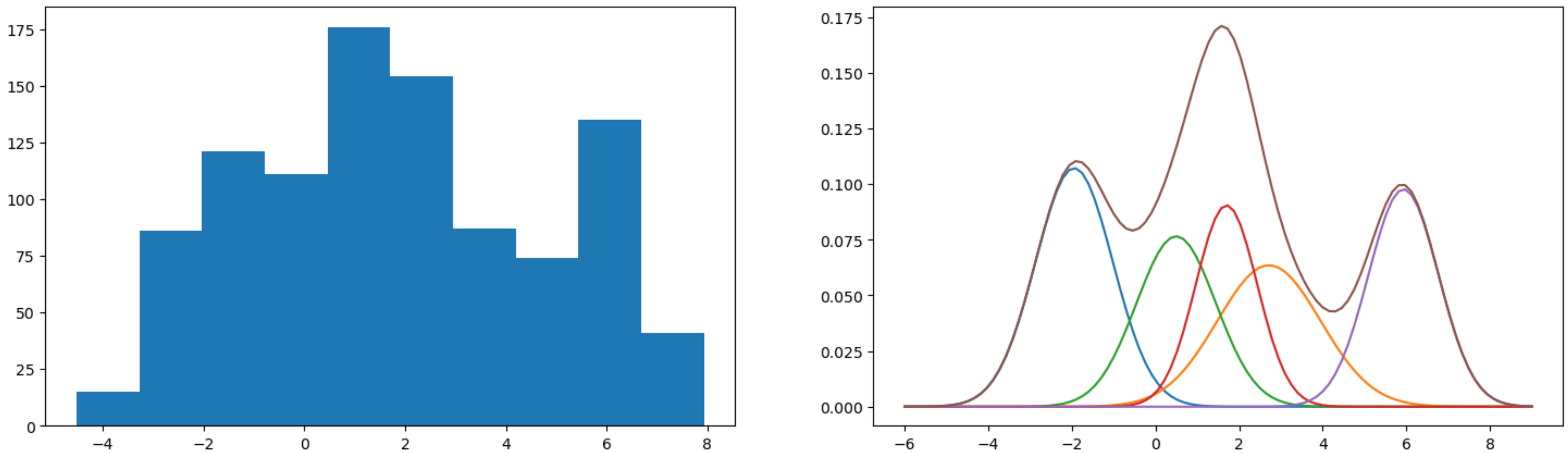
Out[2]:
```
array([[1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 0, 1, 0, 0]])
```

In [3]:
```
print(mu_k_true); print((v_true*mu_k_true)[:3,:]); print((v_true*mu_k_true).sum(axis=1)[:3])
```

```
[-1.94384238  2.715657    0.49645017  1.70025387  5.93193509]
[[-1.94384238  0.          0.          0.          0.        ]
 [-1.94384238  0.          0.          0.          0.        ]
 [-0.          0.          0.49645017  0.          0.        ]]
[-1.94384238 -1.94384238  0.49645017]
```

In [4]:
```
x = stats.norm((v_true*mu_k_true).sum(axis=1), (v_true*sigma2_k_true).sum(axis=1)**0.5).rvs(); ax[0].hist(x); fig
```

Out[4]:



Class Activity 8: (Mixture Model) Quiz [15 minutes]

$$x_i \sim \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(\mu_k, \sigma_k^2) \qquad \mu_k \sim \mathcal{N}(\mu_{k0}, \sigma_{k0}^2) \quad \sigma_k^2 \sim \text{Inverse-Gamma}(\alpha_{k0}, \beta_{k0})$$

$$\overset{\overset{\text{multinomial}}{\mathbf{v}_i \sim \text{MN}(\mathbf{p},\, n=1)}}{\Pr(\mathbf{v}_i | E[\mathbf{v}_i] = \mathbf{p}, n=1)} = \frac{n!}{v_1! \cdots v_K!} p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \qquad \sum_{j=1}^n \mathbf{v}_{ik} = 1 \quad \mathbf{v}_{ik} \in \{0,1\} \quad \text{latent (unknown) subpulation membership } \mathbf{v}$$

$$\underset{\text{Dirichlet}}{\overset{p \sim \text{Dir}(\boldsymbol{\alpha})}{p(\mathbf{p}|\boldsymbol{\alpha})}} = \frac{1}{\text{B}(\boldsymbol{\alpha})} \prod_{k=1}^K p_k^{\alpha_k - 1} \qquad \sum_{j=1}^n p_k = 1 \quad \text{B}(\boldsymbol{\alpha}) = \prod_{k=1}^K \Gamma(\alpha_k) \Big/ \Gamma\left(\sum_{k=1}^K \alpha_k\right) \quad E[p_k] = \alpha_k \Big/ \sum_{k=1}^K \alpha_k$$

$$p(\mu_k | -) \propto \mathcal{N}(\mu_k | \mu_{k0}, \sigma_{k0}^2) \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i | \mu_k, \sigma_k^2) \qquad p(\sigma_k^2 | -) \propto \underset{\text{Inverse-Gamma}}{\text{IG}(\sigma_k^2 | \alpha_{k0}, \beta_{k0})} \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i | \mu_k, \sigma_k^2)$$

$$\Pr(\mathbf{v}_{ik} = 1 | -) \propto p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i | \mu_k, \sigma_k^2) \qquad p(\mathbf{p} | -) \propto \prod_{k=1}^K p_k^{\alpha_k - 1} \prod_{i=1}^n p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}}$$

1. If $\mathbf{v}_i$ are known, what type of ***priors*** do μ_k and σ_k^2 have? Besides being normal and inverse gamma...
2. Again given $\mathbf{v}_i$, what is the distribution and ***hyperparameters*** of the ***full conditional distribution*** $\Pr(\mathbf{p}|-)$?
3. What is the actual probability $p_{\mathbf{v}_{ik}} = \Pr(\mathbf{v}_{ik} = 1|-)$? Hint: $\sum_{k=1}^n p_{\mathbf{v}_{ik}} = 1$
4. What type of MCMC sampler could be created out the ***full conditionals*** provided above?
5. What is $x_i \sim \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(\mu_k, \sigma_k^2)$ if $\mathbf{v}_i$ is integrated out of the expression before any data is observed?

Class Activity 8: (Mixture Model) Quiz [10 minutes]

$$x_i \sim \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(\mu_k, \sigma_k^2) \qquad \mu_k \sim \mathcal{N}(\mu_{k0}, \sigma_{k0}^2) \quad \sigma_k^2 \sim \text{Inverse-Gamma}(\alpha_{k0}, \beta_{k0})$$

$$\Pr(\overset{\overset{\text{multinomial}}{\mathbf{v}_i \sim \text{MN}(\mathbf{p},\, n=1)}}{\mathbf{v}_i|E[\mathbf{v}_i] = \mathbf{p}, n=1}) = \frac{n!}{v_1! \cdots v_K!} p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \qquad \sum_{j=1}^n \mathbf{v}_{ik} = 1 \quad \mathbf{v}_{ik} \in \{0,1\} \quad \text{latent (unknown) subpulation membership } \mathbf{v}$$

$$\underset{\text{Dirichlet}}{\overset{p\sim\text{Dir}(\boldsymbol{\alpha})}{p(\mathbf{p}|\boldsymbol{\alpha})}} = \frac{1}{\text{B}(\boldsymbol{\alpha})}\prod_{k=1}^K p_k^{\alpha_k - 1} \qquad \sum_{j=1}^n p_k = 1 \quad \text{B}(\boldsymbol{\alpha}) = \prod_{k=1}^K \Gamma(\alpha_k) \Big/ \Gamma\left(\sum_{k=1}^K \alpha_k\right) \quad E[p_k] = \alpha_k \Big/ \sum_{k=1}^K \alpha_k$$

$$p(\mu_k|-) \propto \mathcal{N}(\mu_k|\mu_{k0}, \sigma_{k0}^2) \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2) \qquad p(\sigma_k^2|-) \propto \underset{\text{Inverse-Gamma}}{\text{IG}(\sigma_k^2|\alpha_{k0}, \beta_{k0})} \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2)$$

$$\Pr(\mathbf{v}_{ik} = 1|-) \propto p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2) \qquad p(\mathbf{p}|-) \propto \prod_{k=1}^K p_k^{\alpha_k - 1} \prod_{i=1}^n p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}}$$

1. ***Conjugate priors*** for ***normal-normal*** and ***normal-IG*** models, e.g., $\mathcal{N}(\mu_k|\mu_{k0}, \sigma_{k0}^2) \prod_{i:\mathbf{v}_{ik}=1} \mathcal{N}(x_i|\mu_k, \sigma_k^2)$
2. ***Dirichlet*** $\propto \prod_{k=1}^K p_k^{\alpha_k - 1 + \sum_{i=1}^n \mathbf{v}_{ik}}$ with number of observations having $\mathbf{v}_{ik} = 1$ added to prior parameter
3. $\Pr(\mathbf{v}_{ik} = 1|-) = p_k\mathcal{N}(x_i|\mu_k, \sigma_k^2) / \sum_{j=1}^K \mathcal{N}(x_i|\mu_j, \sigma_j^2)$ is a ***discrete distribution*** so we ***self normalize*** it
4. A ***Gibbs sampler*** (all full conditionals analytically known and n ***latent multinomial*** $\mathbf{v}_i$ resampled at each cycle)
5. $x_i \sim \sum_{k=1}^K p_k\mathcal{N}(\mu_k, \sigma_k^2)$ if $\mathbf{v}_i$ with $x_i|\mathbf{v} \sim \sum_{k=1}^K((\alpha_k + \sum_{i=1}^n \mathbf{v}_{ik}) / \sum_{j=1}^K(\alpha_j + \sum_{i=1}^n \mathbf{v}_{ij}))\mathcal{N}(\mu_k, \sigma_k^2)$

Class Activity 8: (Mixture Model) Quiz [10 minutes]

$$x_i \sim \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(\mu_k, \sigma_k^2) \qquad \mu_k \sim \mathcal{N}(\mu_{k0}, \sigma_{k0}^2) \quad \sigma_k^2 \sim \text{Inverse-Gamma}(\alpha_{k0}, \beta_{k0})$$

$$\Pr(\overset{\overset{\text{multinomial}}{\mathbf{v}_i \sim \text{MN}(\mathbf{p},\, n=1)}}{\mathbf{v}_i|E[\mathbf{v}_i] = \mathbf{p}, n=1}) = \frac{n!}{v_1! \cdots v_K!} p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \qquad \sum_{j=1}^n \mathbf{v}_{ik} = 1 \quad \mathbf{v}_{ik} \in \{0,1\} \quad \text{latent (unknown) subpulation membership } \mathbf{v}$$

$$\underset{\text{Dirichlet}}{\overset{p\sim\text{Dir}(\boldsymbol{\alpha})}{p(\mathbf{p}|\boldsymbol{\alpha})}} = \frac{1}{\text{B}(\boldsymbol{\alpha})}\prod_{k=1}^K p_k^{\alpha_k - 1} \qquad \sum_{j=1}^n p_k = 1 \quad \text{B}(\boldsymbol{\alpha}) = \prod_{k=1}^K \Gamma(\alpha_k) \Big/ \Gamma\left(\sum_{k=1}^K \alpha_k\right) \quad E[p_k] = \alpha_k \Big/ \sum_{k=1}^K \alpha_k$$

$$p(\mu_k|-) \propto \mathcal{N}(\mu_k|\mu_{k0}, \sigma_{k0}^2) \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2) \qquad p(\sigma_k^2|-) \propto \underset{\text{Inverse-Gamma}}{\text{IG}(\sigma_k^2|\alpha_{k0}, \beta_{k0})} \prod_{i=1}^n \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2)$$

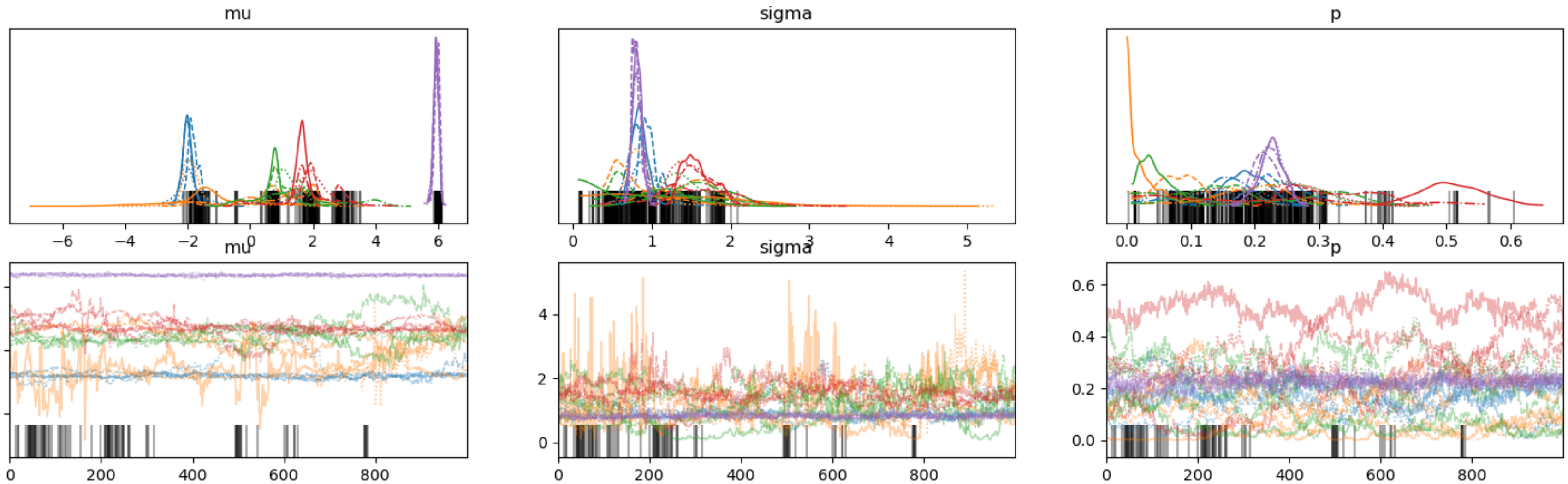
$$\Pr(\mathbf{v}_{ik} = 1|-) \propto p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}} \sum_{k=1}^K \mathbf{v}_{ik}\mathcal{N}(x_i|\mu_k, \sigma_k^2) \qquad p(\mathbf{p}|-) \propto \prod_{k=1}^K p_k^{\alpha_k - 1} \prod_{i=1}^n p_1^{\mathbf{v}_{i1}} \cdots p_K^{\mathbf{v}_{iK}}$$

6. Write `PyMC` code specifying this mixture model. Hint: define latent subpopulation membership with `v_cat = pm.Categorical('v', p=p, shape=n)` [`v_cat` an $n \times 1$ column] with `v[i]` $\in \{1, \cdots, k\}$] instead of `v_mn = pm.Multinomial('v', n=1, p=p, shape=n)` [`v_mn` an $n \times k$ matrix]; so, e.g., if `mu = pm.Normal('mu', size=k)` [`mu` the k subpopulation means] then `pm.Normal('data', mu=mu[v], sigma=sigma[v], observed=x)` specifies, e.g., the mean `x[i]` to be `mu[v[i]]`

Class Activity 8: (Mixture Model) Review [10 minutes]

In [368…

```
fig,ax = plt.subplots(2,3,figsize=(18,5)); import arviz as az; az.plot_trace(idata, var_names=['mu','sigma','p'], axes=ax.T);
```



In [367…

```
import pymc as pm
with pm.Model() as mixture_model:
    p = pm.Dirichlet('p', a=[1]*k); v = pm.Categorical('v', p=p, size=n) #v = pm.Multinomial('v', n=1, p=p, size=n)
    mu = pm.Normal('mu', mu=[-4,-2,0,2,4], sigma=2, size=k); sigma = pm.HalfNormal('sigma', sigma=2, size=k)
    pm.Normal('y', mu=mu[v], sigma=sigma[v], observed=x) #y = pm.Normal('y', mu=(mu*v).sum(axis=1), sigma=(sigma2**0.5*v).sum(axis=1), observed=x)
    idata = pm.sample()
```

```
Multiprocess sampling (4 chains in 4 jobs)
CompoundStep
>NUTS: [p, mu, sigma]
>CategoricalGibbsMetropolis: [v]
```

100.00% [8000/8000 01:55<00:00 Sampling 4 chains, 94 divergences]

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 116 seconds.
The rhat statistic is larger than 1.01 for some parameters. This indicates problems during sampling. See https://arxiv.org/abs/1903.08008 for details
The effective sample size per chain is smaller than 100 for some parameters.  A higher number is needed for reliable rhat and ess computation. See http
s://arxiv.org/abs/1903.08008 for details
There were 94 divergences after tuning. Increase `target_accept` or reparameterize.
```

(Mixture Model) Posterior Predictive Distributions [5 minutes]

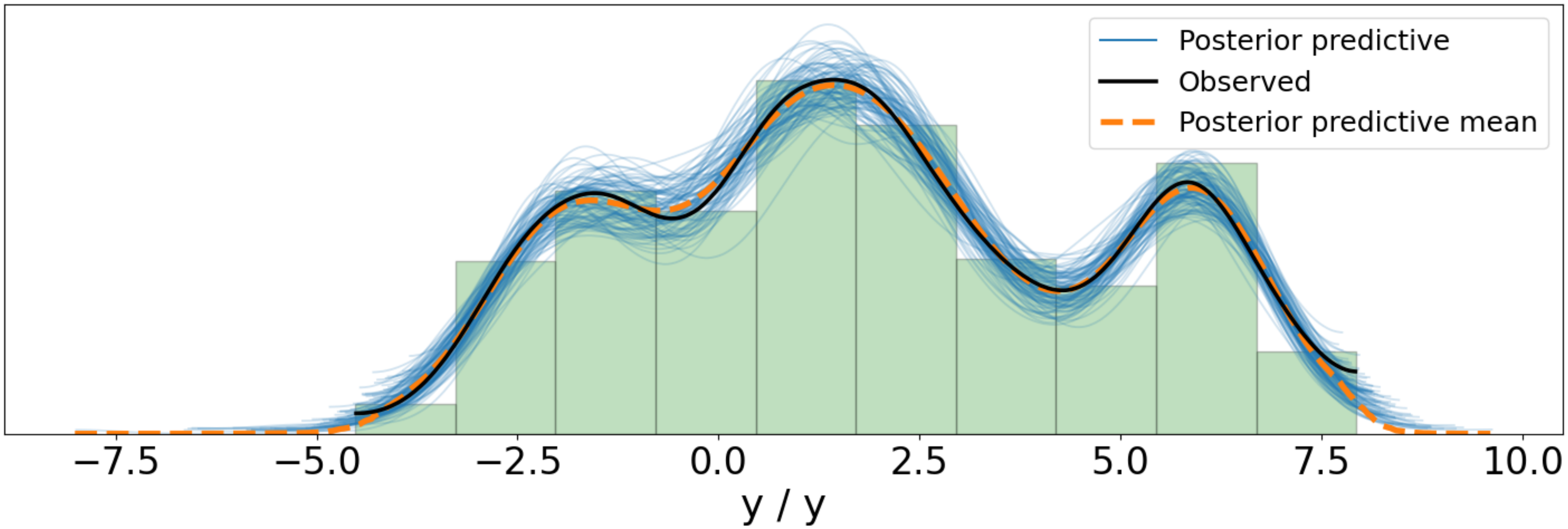
https://www.pymc.io/projects/docs/en/stable/learn/core_notebooks/posterior_predictive.html

In [369…
```
with mixture_model:
    pm.sample_posterior_predictive(idata, extend_inferencedata=True)
```

```
Sampling: [y]
```

100.00% [4000/4000 00:00<00:00]

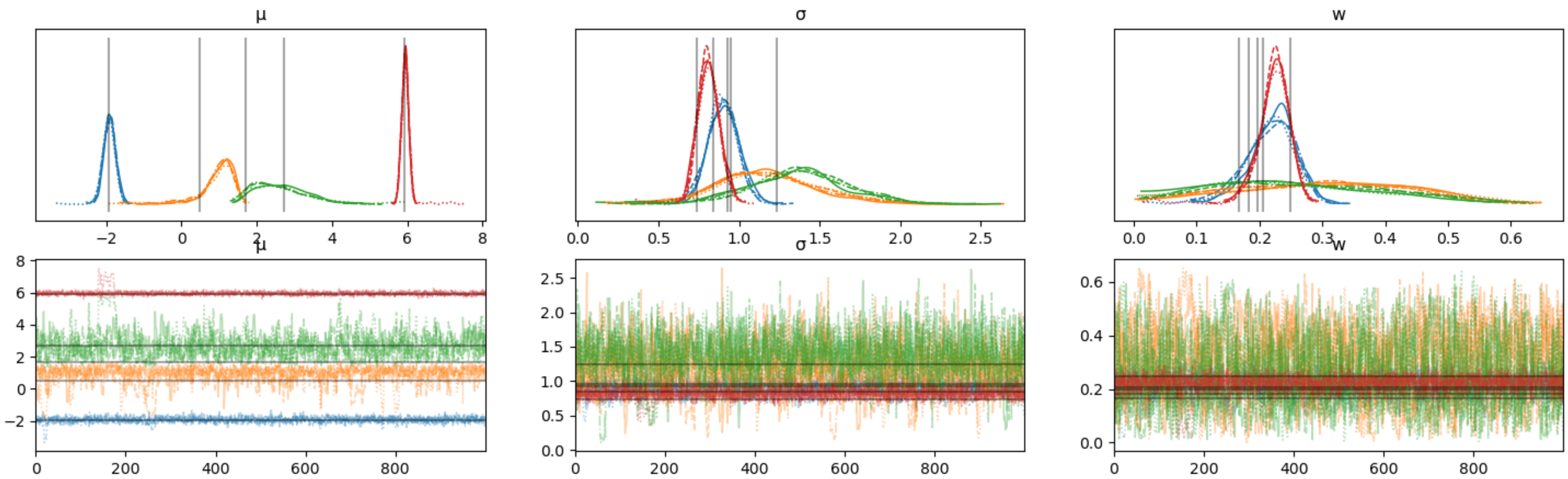
In [370…
```
fig = az.plot_ppc(idata, num_pp_samples=100, figsize=(18,5)); fig.hist(x, density=True, color='green', edgecolor='black', alpha=0.25);
```



`pm.NormalMixture`
label switching, and the number of components [10 minutes]

https://www.pymc.io/projects/examples/en/latest/mixture_models/gaussian_mixture_model.html

In [376…
```
fig,ax = plt.subplots(2,3,figsize=(18,5)); az.plot_trace(idata, var_names=["μ", "σ", "w"], lines=[("μ", {}, [mu_k_true]), ("σ", {}, [sigma2_k_true**0.
```



In [375…

```
with pm.Model(coords={"cluster": range(k-1)}) as model:
    μ = pm.Normal("μ", mu=[-2, 0, 2, 4], sigma=2,
                  transform=pm.distributions.transforms.univariate_ordered, initval=[-2, 0, 2, 4], dims="cluster")
    σ = pm.HalfNormal("σ", sigma=1, dims="cluster")
    weights = pm.Dirichlet("w", 2*np.ones(k-1), dims="cluster")
    pm.NormalMixture("y", w=weights, mu=μ, sigma=σ, observed=x); idata = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [μ, σ, w]
```

100.00% [8000/8000 00:22<00:00 Sampling 4 chains, 0 divergences]

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 23 seconds.
The rhat statistic is larger than 1.01 for some parameters. This indicates problems during sampling. See https://arxiv.org/abs/1903.08008 for details
```

(Mixture Model) Posterior Predictive Distributions [2 minutes]

https://www.pymc.io/projects/docs/en/stable/learn/core_notebooks/posterior_predictive.html

In [380…

```
with model:
    pm.sample_posterior_predictive(idata, extend_inferencedata=True)
```

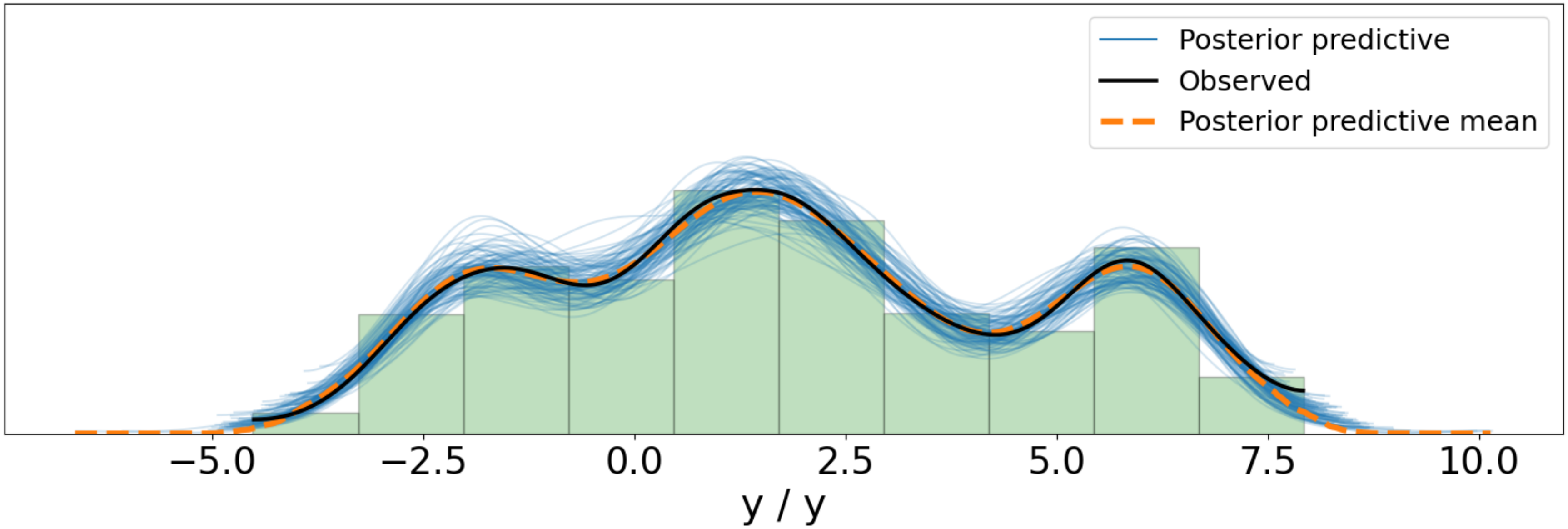
```
Sampling: [y]
```

100.00% [4000/4000 00:06<00:00]

In [381…

```
fig = az.plot_ppc(idata, num_pp_samples=100, figsize=(18,5)); fig.hist(x, density=True, color='green', edgecolor='black', alpha=0.25); fig.set_ylim([0
```

Out[381… (0.0, 0.25)

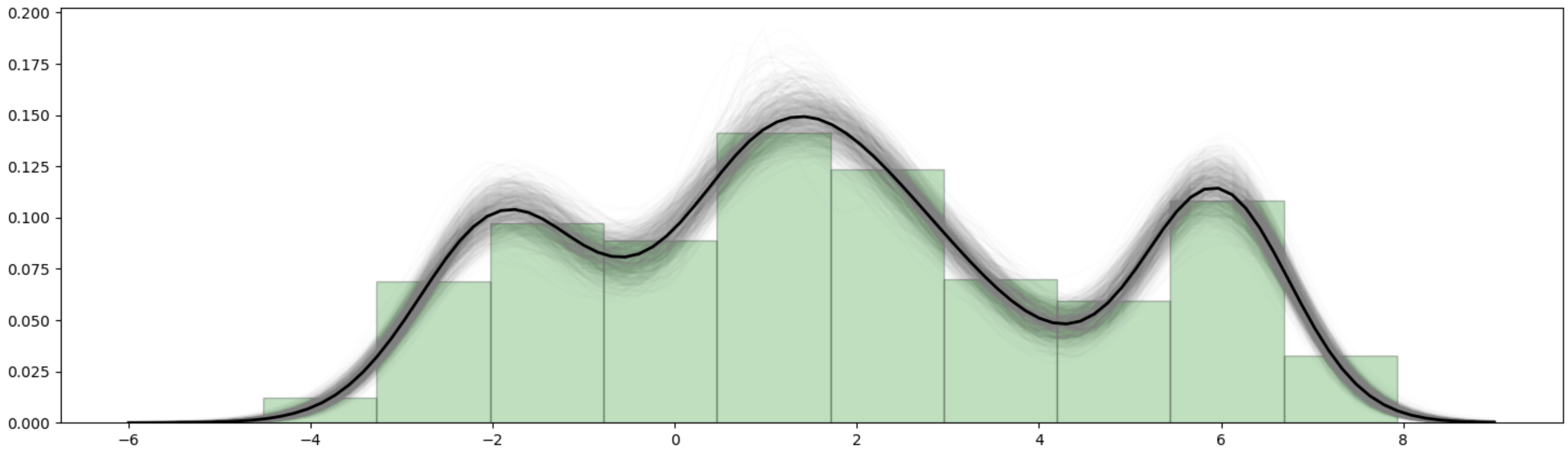


(Mixture Model) Posterior Predictive Distributions [8 minutes]

https://www.pymc.io/projects/docs/en/stable/learn/core_notebooks/posterior_predictive.html

In [167…

```
plt.figure(figsize=(18,5)); ave_mixture=0*support; reps=1000
for j in range(reps):

    chain,draw=np.random.choice([0,1,2,3]),np.random.choice(np.linspace(0,999,1000,dtype=int))

    mixture = stats.norm(idata.posterior['μ'][chain,draw],
                         idata.posterior['σ'][chain,draw]).pdf(np.array([support]*4).reshape(4,100).T)
    mixture = (mixture*idata.posterior['w'][chain,draw].values).sum(axis=1)

    plt.plot(support, mixture, c='gray', alpha=0.01); ave_mixture += mixture/reps
plt.plot(support, ave_mixture, 'k', lw=2)
plt.hist(x, density=True, color='green', edgecolor='black', alpha=0.25);
```



Homework 8: (a) Posterior Predictive Distributions and (b) Missing Data Imputation

1. Describe how the posterior predictive distribution is created for mixture models

2. Describe how the posterior predictive distribution is created in general

3. Have glance through this and then describe how, if you were doing a regression of y on X but X had some missing values, you could perform a Bayesian analysis without throwing away the rows with missing values in X

- **Hint: latent variables v indicating the subpopulation are competely missing values that we simply treat as paramters to be inferred though posterior analysis... the same sort of thing can be done with missing values in data that need to be imputed... we should just be careful about the MCAR assumption...**

4. Work on your course project

Dirichlet Processes: a distribution of distributions (20 minutes)

For any partition of support $S = \cup_{i=1}^k S_i$ the distribution p is distributed according to the ***Dirichlet process***

$$\overset{\text{Dirichlet Process}}{p \sim \text{DP}(\alpha, p_0)} \quad \text{if} \quad (p(S_1), \cdots, p(S_k)) \sim \text{Dir}\left(\alpha p_0(S_1), \cdots, \alpha p_0(S_k)\right)$$

- Distribution p sampled from a DP has probabilities over the partition S_i which are more similar to the ***base distribution*** p_0 the larger α is

For $x_i \sim p$ the ***posterior distribution*** $f(p|\mathbf{x})$ is

$$p|\mathbf{x} \sim \text{DP}\left(\alpha + n, \frac{\alpha}{\alpha+n}p_0 + \frac{\sum_{i=1}^n \delta_{x_i}}{\alpha+n}\right) \quad \text{with} \quad \underset{\text{function}}{\overset{\text{indicator}}{\delta_{x_i}(S_j)}} = \begin{cases} 1: & \text{if } x_i \in S_j \\ 0: & \text{otherwise} \end{cases}$$

Letting $p = \sum_{i=1}^{\infty} w_i \delta_{y_i}$ where $y_i \sim \tilde{p}_0$ and **weights** $\sum_{i=1}^{\infty} \underset{w_i>0}{w_i} = 1$ are the **stick-breaking process**

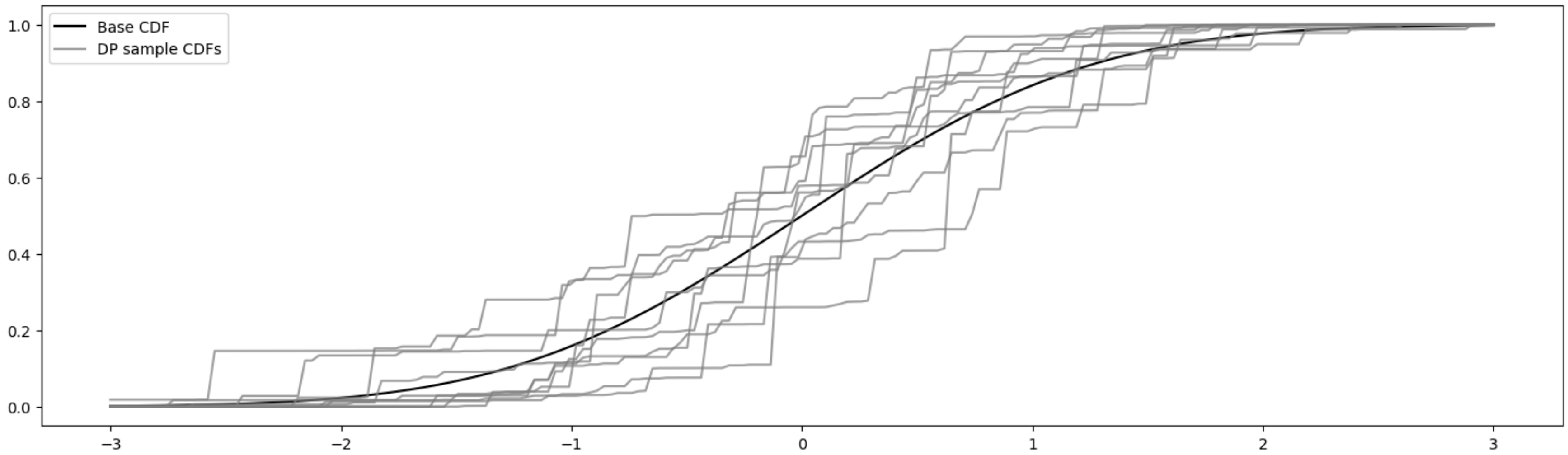
$$w_1 = \beta_1 \text{ and } \underset{\text{for } j>1}{w_j} = \beta_j \underbrace{\prod_{i=1}^{j-1}(1-\beta_i)}_{\text{stick length at } j-1} \text{ for } \underset{E[\beta_i]=1/(1+\alpha)=\alpha/(\alpha+\beta)}{\beta_i \sim \text{Beta}(\alpha \text{ [for Beta]} = 1, \beta = \alpha \text{ [from DP]})}$$

actualizes sample $p \sim \text{DP}(\alpha, \tilde{p}_0)$ so $\left(\sum_{i=1}^{\infty} w_i\delta_{y_i}(S_1), \cdots, \sum_{i=1}^{\infty} w_i\delta_{y_i}(S_k)\right) \sim \text{Dir}\left(\alpha\tilde{p}_0(S_1), \cdots, \alpha\tilde{p}_0(S_k)\right)$

Dirichlet Processes: a distribution of distributions (10 minutes)

https://www.pymc.io/projects/examples/en/latest/mixture_models/dp_mix.html

In [17]:
```
alpha=10
support=np.linspace(-3, 3, 200); p0=stats.norm # could be a posterior mixed p0 + discrete point mass posterior

n=10
k=alpha*100

beta = stats.beta.rvs(1, alpha, size=(n,k))
w = np.zeros(beta.shape); w[:, 0] = beta[:,0]; w[:,1:] = beta[:,1:] * (1-beta[:,:-1]).cumprod(axis=1)

y = p0.rvs(size=(n,k))

plt.figure(figsize=(18,5));plt.plot(support, p0.cdf(support), c="k", label="Base CDF")
sample_cdfs = (w[..., np.newaxis] * np.less.outer(y, support)).sum(axis=1)
plt.plot(support, sample_cdfs[0], c="gray", alpha=0.75, label="DP sample CDFs"); plt.plot(support, sample_cdfs[1:].T, c="gray", alpha=0.75); plt.legen
```



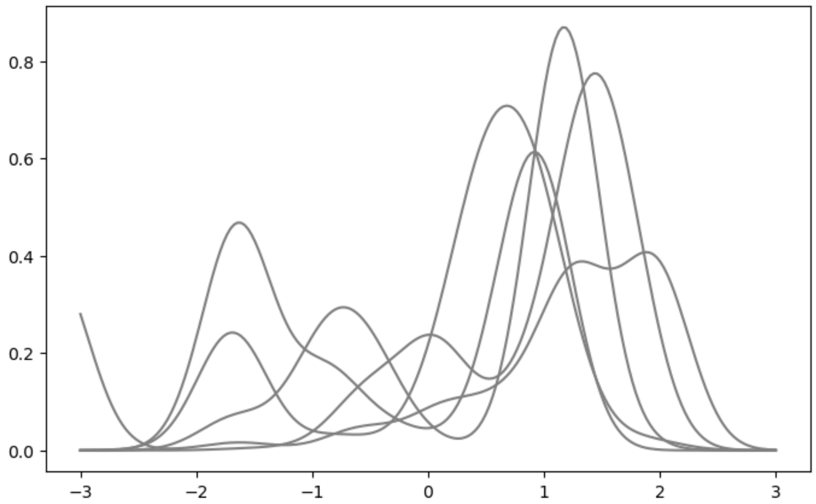
Dirichlet Process Mixtures: Nonparametric Density Estimation (15 minutes)

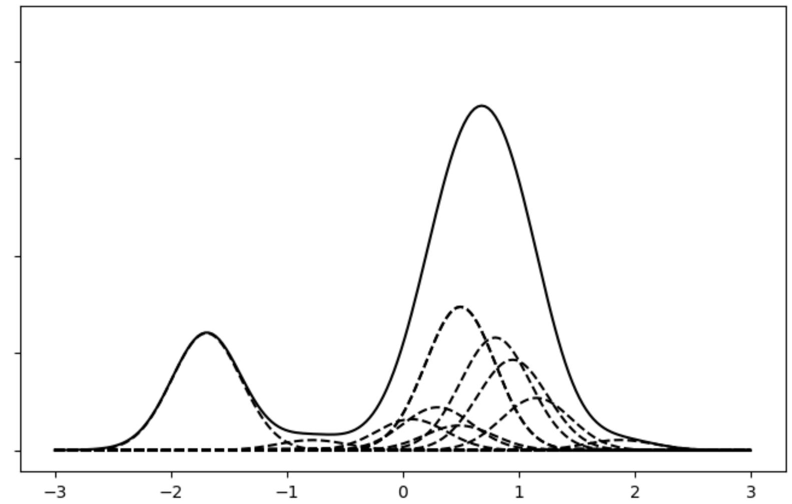
https://www.pymc.io/projects/examples/en/latest/mixture_models/dp_mix.html#dirichlet-process-mixtures

$$
\begin{array}{llll}
x_i \sim f_{\theta_i} & x_i \sim \mathcal{N}(\theta_i, \sigma = 0.3) & & \implies p(x^*) = \sum_{i=1}^{\infty} w_i \mathcal{N}(y_i, \sigma = 0.3) \\
\theta_i \sim p & \theta_i \sim p = \sum_{j=1}^{\infty} w_j \delta_{y_j} & y_j \sim \mathcal{N}(0,1) & \approx \sum_{i=1}^{n} w_i \mathcal{N}(y_i, \sigma = 0.3) \\
p \sim DP(\underset{2}{\alpha}, \underset{\mathcal{N}(0,1)}{p_0}) & w_j = \beta_j \prod_{k=1}^{j-1}(1-\beta_k) & \beta_j \sim \text{Beta}(\alpha \text{ [for Beta]} = 1, \beta = \alpha = 2 \text{ [from DP]}) &
\end{array}
$$

In [22]:

```
n,k=5,30; alpha,p0,f=2,stats.norm,lambda x,theta: stats.norm(theta, 0.3).pdf(x)
beta = stats.beta.rvs(1, alpha, size=(n,k)); w = np.zeros(beta.shape); w[:,0] = beta[:,0]; w[:,1:] = beta[:,1:]*(1-beta[:,:-1]).cumprod(axis=1)
theta = p0.rvs(size=(n,k)); dpm_pdf_components = f(support, theta[..., np.newaxis]); dpm_pdfs = (w[..., np.newaxis] * dpm_pdf_components).sum(axis=1);
```





(Truncated) Dirichlet Process Mixture Models: Nonparametric Density Estimation (10 minutes)

$$x_i \sim \sum_{j=1}^{k} w_j \mathcal{N}(\mu_j, \tau_j) \qquad \mu_j \sim \mathcal{N}(0, \lambda_j) \qquad \tau_j, \lambda_j \sim \mathrm{Gamma}(10, 1)$$

$$w_j = \beta_j \prod_{i=1}^{j-1}(1-\beta_i) \qquad \beta_j \sim \mathrm{Beta}(1, \alpha) \qquad \alpha \sim \mathrm{Gamma}(1, 1)$$

In [319…

```
import pytensor.tensor as pt
def stick_breaking(beta):
    portion_remaining = pt.concatenate([[1], pt.extra_ops.cumprod(1 - beta)[:-1]])
    return beta * portion_remaining
k = 30
with pm.Model(coords={"component": np.arange(k), "obs_id": np.arange(n)}) as model:
    alpha = pm.Gamma("alpha", 1.0, 1.0);
    beta = pm.Beta("beta", 1.0, alpha, dims="component")
    w = pm.Deterministic("w", stick_breaking(beta), dims="component")
    tau = pm.Gamma("tau", 1.0, 1.0, dims="component"); lambda_ = pm.Gamma("lambda_", 10.0, 1.0, dims="component")
    mu = pm.Normal("mu", 0, tau=lambda_ * tau, dims="component")
    obs = pm.NormalMixture("obs", w, mu, tau=lambda_*tau, observed=x, dims="obs_id")
    trace = pm.sample()
    # old_faithful_df.std_waiting.values
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, tau, lambda_, mu]
```

100.00% [8000/8000 01:19<00:00 Sampling 4 chains, 510 divergences]

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 79 seconds.
The rhat statistic is larger than 1.01 for some parameters. This indicates problems during sampling. See https://arxiv.org/abs/1903.08008 for details
The effective sample size per chain is smaller than 100 for some parameters.  A higher number is needed for reliable rhat and ess computation. See http
s://arxiv.org/abs/1903.08008 for details
There were 510 divergences after tuning. Increase `target_accept` or reparameterize.
```

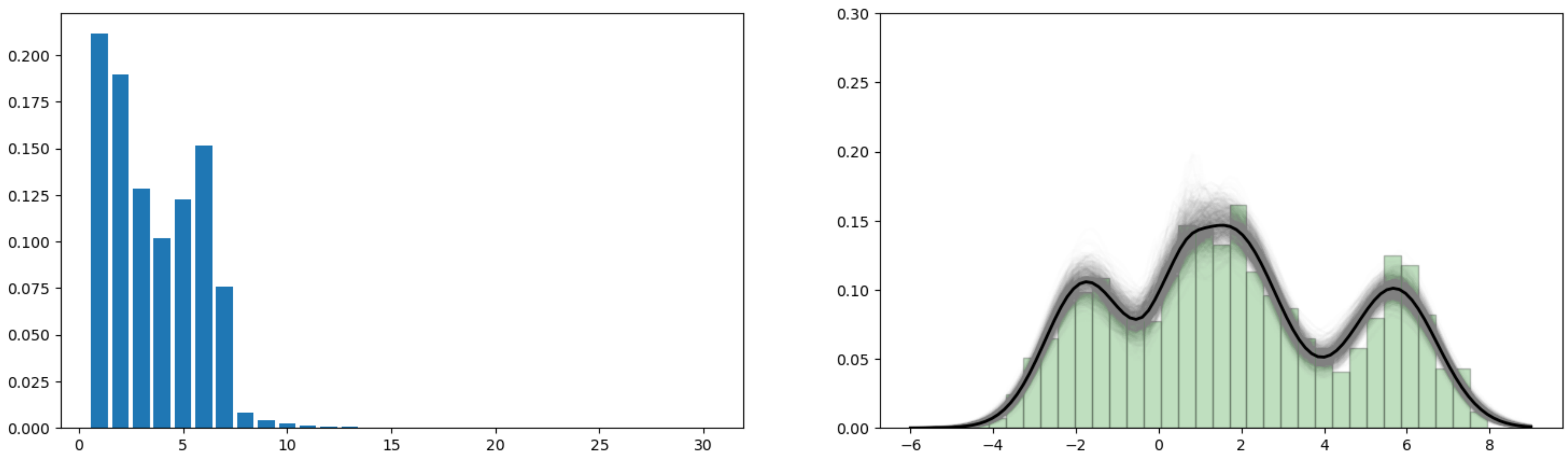
(Truncated) Dirichlet Process Mixture Models: Nonparametric Density Estimation (5 minutes)

In [320…

```
fig,ax=plt.subplots(1,2,figsize=(18,5)); ave_mixture=0*support; reps=1000
for j in range(reps):
    chain,draw=np.random.choice([0,1,2,3]),np.random.choice(np.linspace(0,999,1000,dtype=int))
    mixture = stats.norm(trace.posterior['mu'][chain,draw],(trace.posterior['tau'][chain,draw]*trace.posterior['lambda_'][chain,draw])**-0.5).pdf(np.a
    mixture = (mixture*trace.posterior['w'][chain,draw].values).sum(axis=1)
    plt.plot(support, mixture, c='gray', alpha=0.01); ave_mixture += mixture/reps
ax[0].bar(x=np.linspace(1,k,k), height=trace.posterior['w'].mean(("chain", "draw"))); ax[1].plot(support, ave_mixture, 'k', lw=2)
ax[1].hist(x, density=True, color='green', edgecolor='black', alpha=0.25, bins=30); ax[1].set_ylim([0,0.3])
```

Out[320…

```
(0.0, 0.3)
```



(Truncated) Dirichlet Process Mixture Models: Nonparametric Density Estimation (3 minutes)

$$x_i \sim \sum_{j=1}^{k} w_j \mathcal{N}(\mu_j, \tau_j) \qquad \mu_j \sim \mathcal{N}(0, \lambda_j) \qquad \tau_j, \lambda_j \sim \mathrm{Gamma}(10, 1)$$

$$w_j = \beta_j \prod_{i=1}^{j-1} (1 - \beta_i) \qquad \beta_j \sim \mathrm{Beta}(1, \alpha) \qquad \alpha \sim \mathrm{Gamma}(1, 1)$$

In [322…

```
with model:
    trace2 = pm.sample(tune=5000, init="advi", target_accept=0.95)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using advi...
```

16.98% [33963/200000 00:18<01:28 Average Loss = inf]

```
Convergence achieved at 34100
Interrupted at 34,099 [17%]: Average Loss = 3,473.9
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, tau, lambda_, mu]
```

100.00% [24000/24000 09:42<00:00 Sampling 4 chains, 77 divergences]

```
Sampling 4 chains for 5_000 tune and 1_000 draw iterations (20_000 + 4_000 draws total) took 583 seconds.
The rhat statistic is larger than 1.01 for some parameters. This indicates problems during sampling. See https://arxiv.org/abs/1903.08008 for details
The effective sample size per chain is smaller than 100 for some parameters.  A higher number is needed for reliable rhat and ess computation. See http
s://arxiv.org/abs/1903.08008 for details
There were 77 divergences after tuning. Increase `target_accept` or reparameterize.
```

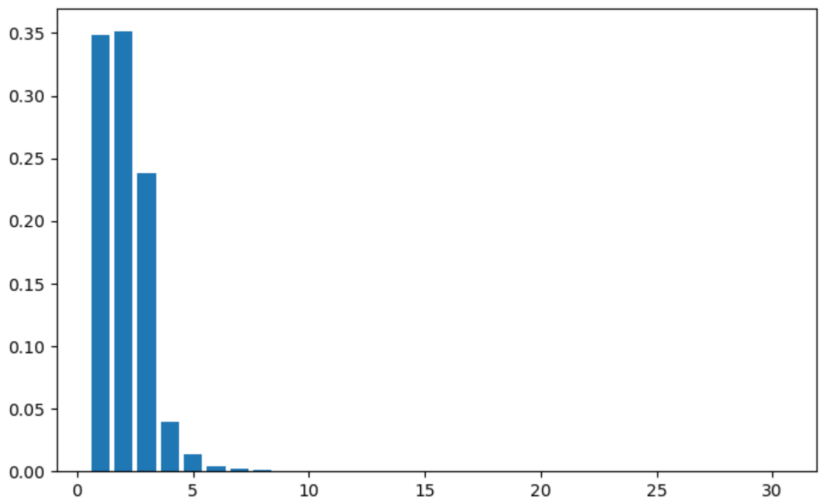
(Truncated) Dirichlet Process Mixture Models: Nonparametric Density Estimation (2 minutes)

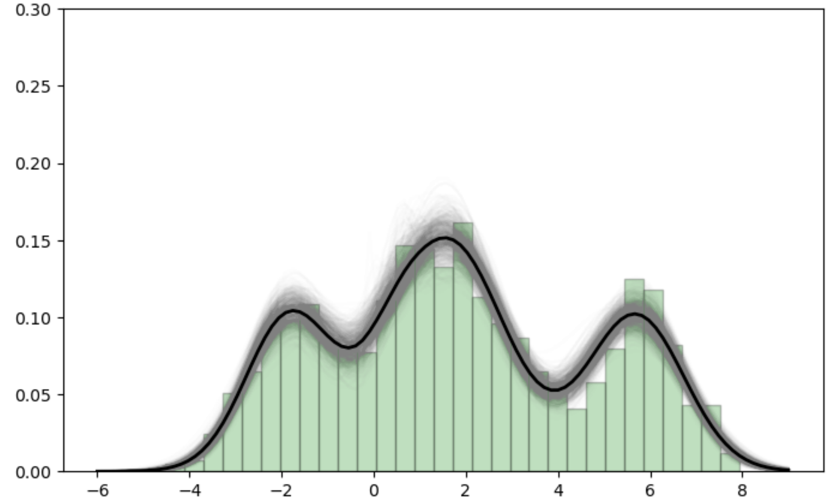
In [323…

```
fig,ax=plt.subplots(1,2,figsize=(18,5)); ave_mixture=0*support; reps=1000
for j in range(reps):
    chain,draw=np.random.choice([0,1,2,3]),np.random.choice(np.linspace(0,999,1000,dtype=int))
    mixture = stats.norm(trace2.posterior['mu'][chain,draw],(trace2.posterior['tau'][chain,draw]*trace2.posterior['lambda_'][chain,draw])**-0.5).pdf(n
    mixture = (mixture*trace2.posterior['w'][chain,draw].values).sum(axis=1)
    plt.plot(support, mixture, c='gray', alpha=0.01); ave_mixture += mixture/reps
ax[0].bar(x=np.linspace(1,k,k), height=trace2.posterior['w'].mean(("chain", "draw"))); ax[1].plot(support, ave_mixture, 'k', lw=2)
ax[1].hist(x, density=True, color='green', edgecolor='black', alpha=0.25, bins=30); ax[1].set_ylim([0,0.3])
```

Out[323…

```
(0.0, 0.3)
```


0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30


0.30
0.25
0.20
0.15
0.10
0.05
0.00
−6
−4
−2
0
2
4
6
8