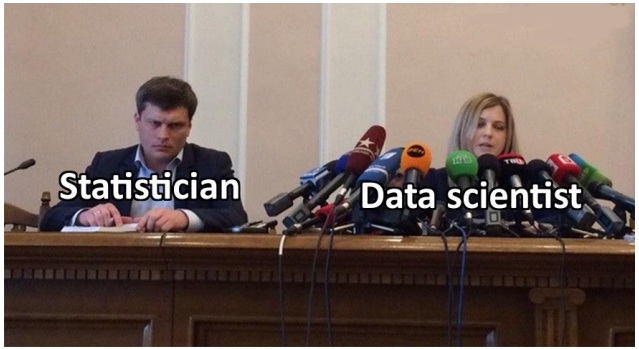
Bayesian modeling with spatial data using PyMC3

Shreya Khurana

GeoPython 2019
Muttenz Switzerland
Basel
June 24-26

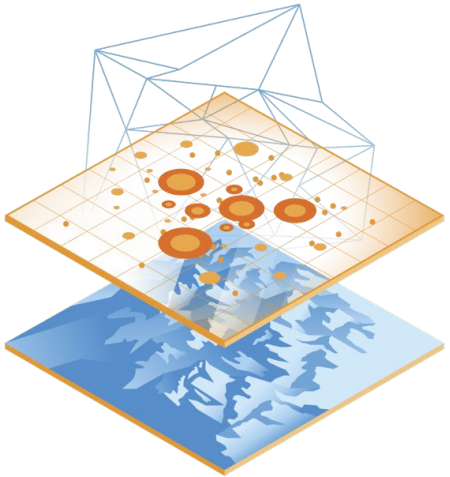
whoami

- Recently graduated with Masters in Statistics from University of Illinois, USA

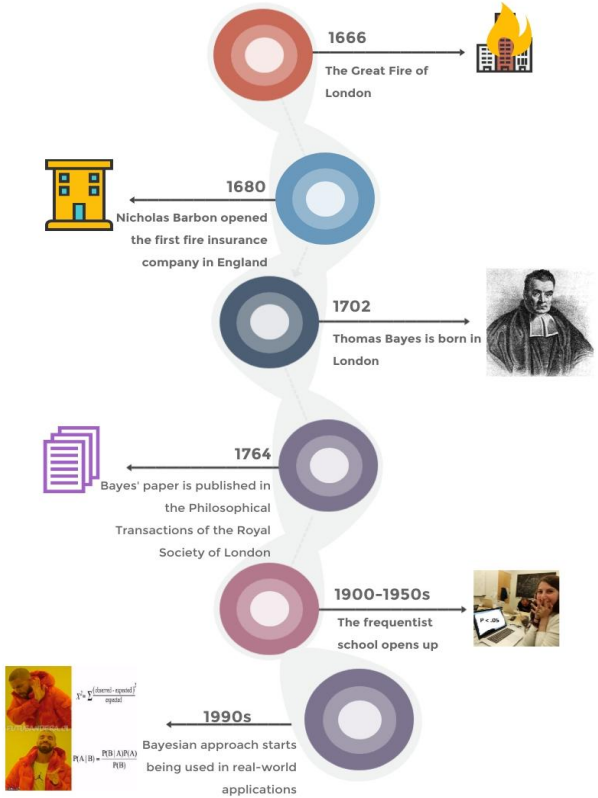


- Mostly work with:
    - Machine learning and deep learning models for language generation and health data applications
    - Bayesian models for disease mapping, population density mapping

What am I doing here?

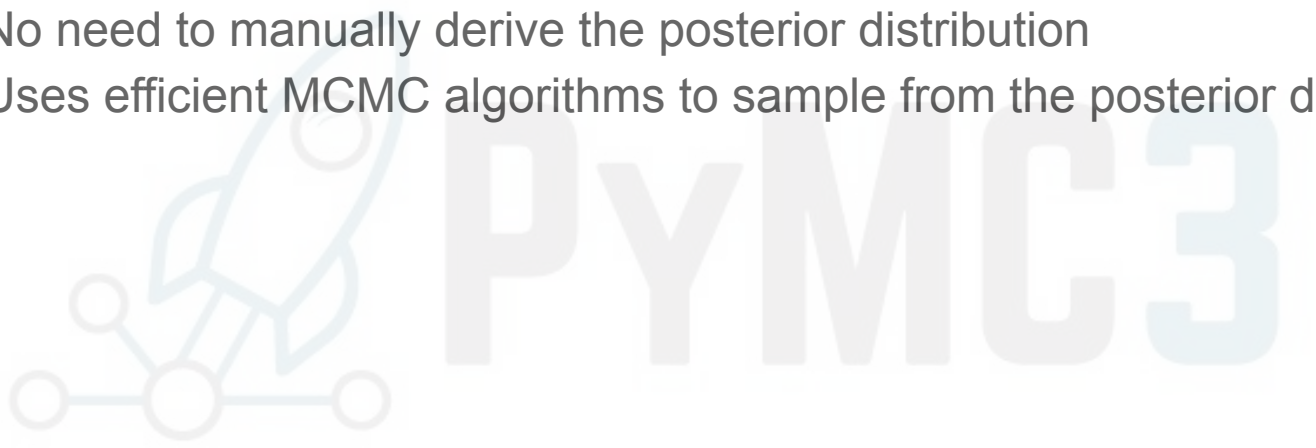


Bayesian School of Statistics



PyMC3

- Probabilistic programming framework
- Has various in-built distributions
- No need to manually derive the posterior distribution
- Uses efficient MCMC algorithms to sample from the posterior distribution



The Golden Rule

$$P(model \mid data) \propto P(model) \times P(data \mid model)$$

But why does Bayesian work for spatial data?

It's all about the definition!

Frequentist

Based on the assumption of repeatable experiments

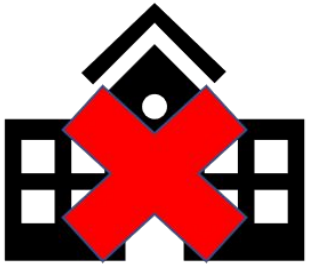
Bayesian

Subjective interpretation: how likely that event is to occur

How often can we get repeatable conditions?

Winter of 2007-08, USA

Mortgages



Businesses

Oil

Priors and Posteriors

- Priors = probabilities assessed before new data are gathered
    - Conjugate: beta prior and binomial likelihood

$$p(\pi|y) \propto \pi^{\alpha-1}(1-\pi)^{\beta-1}\pi^{y}(1-\pi)^{n-y}$$
$$= \pi^{\alpha+y-1}\ (1-\pi)^{\beta+n-y-1}$$

    - Informative, uninformative
- Posteriors = probabilities updated after new data are assessed

Priors and Posteriors

Sampling and estimating parameters

Solving Analytically:

$$\sigma_m^2|e.e. \sim IG\left(\frac{n_m}{2}+a, b+\frac{1}{2}\sum_{i=1}^{n_m}(x_{m,i}-\beta_0-\beta_1 x_i^{(m)})^2\right)$$

$$\sigma_g^2|e.e. \sim IG\left(\frac{n_g}{2}+a, b+\frac{1}{2}\sum_{i=1}^{n_g}(x_{g,i}-\alpha_0-\alpha_1 x_i^{(g)})^2\right)$$

$$\sigma_p^2|e.e. \sim IG\left(\frac{n_p}{2}+a, b+\frac{1}{2}\sum_{i=1}^{n_p}(x_{p,i}-\gamma_0-\gamma_1 x_i^{(p)})^2\right)$$

Finally, for the latent process:

$$\sigma^2|e.e. \sim IG\left(\frac{n}{2}+a, \frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}^{-1}\boldsymbol{x}+b\right)$$

$$\boldsymbol{x}|e.e. \sim N_n\left(\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}\right)$$

where,

$$\tilde{\boldsymbol{\mu}} = \frac{\beta_1}{\sigma_m^2}\boldsymbol{M}^T(\boldsymbol{x}_m-\beta_0\mathbf{1}_{n_m})+\frac{\alpha_1}{\sigma_g^2}\boldsymbol{G}^T(\boldsymbol{x}_g-\alpha_0\mathbf{1}_{n_g})+\frac{\gamma_1}{\sigma_p^2}\boldsymbol{P}^T(\boldsymbol{x}_p-\gamma_0\mathbf{1}_{n_p})$$

$$\tilde{\boldsymbol{\Sigma}} = \left[\boldsymbol{\Sigma}^{-1}+\frac{\beta_1^2}{\sigma_m^2}\boldsymbol{M}^T\boldsymbol{M}+\frac{\alpha_1^2}{\sigma_g^2}\boldsymbol{G}^T\boldsymbol{G}+\frac{\gamma_1^2}{\sigma_p^2}\boldsymbol{P}^T\boldsymbol{P}\right]^{-1}$$

$$\beta_0|e.e. \sim N\left(\frac{n_m}{\sigma_m^2\rho_m}(\bar{x}_m-\beta_1\bar{x}^{(m)}), \frac{1}{\rho_m}\right)$$

$$\alpha_0|e.e. \sim N\left(\frac{n_g}{\sigma_g^2\rho_g}(\bar{x}_g-\alpha_1\bar{x}^{(g)}), \frac{1}{\rho_g}\right)$$

$$\gamma_0|e.e. \sim N\left(\frac{n_p}{\sigma_p^2\rho_p}(\bar{x}_p-\gamma_1\bar{x}^{(p)}), \frac{1}{\rho_p}\right)$$

where $\rho_k = \frac{n_k}{\sigma_k^2}+\frac{1}{\tau}, k \in \{m,g,p\}$.

$$\beta_1|e.e. \sim N\left(\frac{\frac{n_m}{\sigma_m^2}(c_m-\beta_0\bar{x}^{(m)})+\frac{1}{\tau}}{\eta_m}, \frac{1}{\eta_m}\right)$$

$$\alpha_1|e.e. \sim N\left(\frac{\frac{n_g}{\sigma_g^2}(c_g-\alpha_0\bar{x}^{(g)})+\frac{1}{\tau}}{\eta_g}, \frac{1}{\eta_g}\right)$$

$$\gamma_1|e.e. \sim N\left(\frac{\frac{n_p}{\sigma_p^2}(c_p-\gamma_0\bar{x}^{(p)})+\frac{1}{\tau}}{\eta_p}, \frac{1}{\eta_p}\right)$$

```r
makeSymm <- function(m) {
  m[upper.tri(m)] <- t(m)[upper.tri(m)]
  return(m)
}
# Metropolis hastings algorithm
# MH <- function(old, sigma2, x1, mu0, sd) {
#   proposed <-  old + rnorm(1, mean=0, sd=sd)
#   ratio <- min(1, phi_target(proposed, sigma2, x1, mu0)/phi_target(old,
#   # print(ratio)
#   if (runif(1) <= ratio) { return(proposed) }
#   return(old)
# }

# Random-walk metropolis hastings
MH <- function(old, sigma2, x1, mu0) {
  proposed <-  old + rnorm(1, mean=0, sd=phi_sd)
  if ((proposed < A) | (proposed > B)){
    return(old)
  }
  # print(proposed)
  a = -1/(2*sigma2) * (t(x1 - mu0) %*% solve(V(proposed), (x1 - mu0)))
  b = -1/(2*sigma2) * (t(x1 - mu0) %*% solve(V(old), x1 - mu0))
  ratio <- min(1, sqrt(det(V(old)) / det(V(proposed))) * exp(a-b))

  if((runif(1) <= ratio)) { return(proposed) }
  return(old)
}
ptm <- proc.time()
for (i in 2:1000) {
  # Try for one timestep
```

```r
mu[i,] <- rnorm(1, (rep(1,n) %*% sinv %*% x[,i])/k, sqrt(1/k))
# print(mu[i])
# Sample sigma.2
bb <- 0.5*t(x[,i] - mu[i,]) %*% solve(V(phi[i-1]), (x[,i] - mu[i,])) + b
sigma.2[i,] <- 1/rgamma(1, shape = n/2 + a, scale = bb)

# Sample sigma2.m, sigma2.g, sigma2.p
aa <- n.m/2 + a
bb <- b + 0.5*sum((x.m - beta0[i-1] - beta1[i-1]*x[1:n.m,i])^2)
sigma2.m[i] <- 1/rgamma(1,shape=aa,scale=bb)

aa <- n.g/2 + a
bb <- b + 0.5*sum((x.g - alpha0[i-1] - alpha1[i-1]*x[(n.m+1):(n.m+n.g),i])^2)
sigma2.g[i] <- 1/rgamma(1,shape=aa,scale=bb)

aa <- n.p/2 + a
bb <- b + 0.5*sum((x.p - gamma0[i-1] - gamma1[i-1]*x[(n.m+n.g+1):n,i])^2)
sigma2.p[i] <- 1/rgamma(1,shape=aa,scale=bb)

# Sample beta1
c.k <- 1/n.m * (t(x.m) %*% x[1:n.m,i])
s.k <- 1/n.m * (t(x[,i]) %*% t(M) %*% M %*% x[,i])
eta.k <- n.m * s.k / sigma2.m[i] + 1/tau

mn <- n.m/sigma2.m[i] * (c.k - beta0[i-1]*mean(x[1:n.m,i])) + 1/tau

beta1[i] <- rnorm(1, mean = mn / eta.k, sd = sqrt(1/eta.k))

# Sample alpha1

c.k <- 1/n.g * (t(x.g) %*% x[(n.m+1):(n.m + n.g),i])
s.k <- 1/n.g * (t(x[,i]) %*% t(G) %*% G %*% x[,i])
eta.k <- n.g * s.k / sigma2.g[i] + 1/tau
```

Monte Carlo Markov Chain

A way to draw samples from very high-dimensional joint posterior densities

Metropolis Hastings

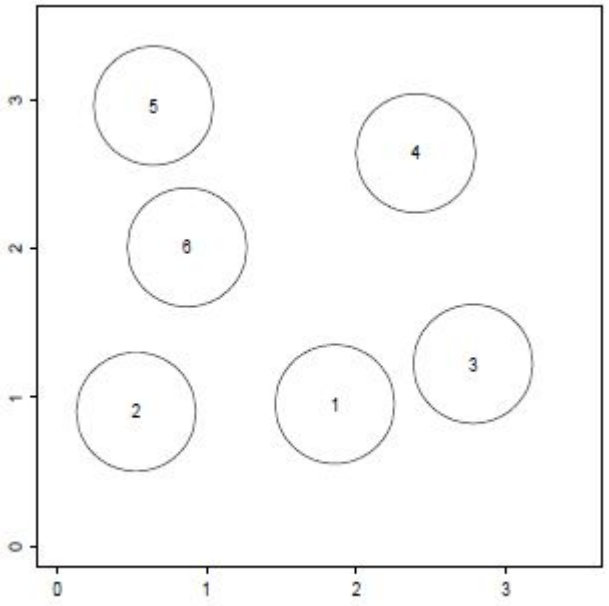
K non-overlapping equal diameter hard-shell balls are uniformly distributed in the box $[0,A]\times[0,B]$

1. pick a ball at random, say, the ball at position (x_i, y_i) ;
2. propose to move this ball to a new position :

   $(x'_i,y'_i) = (x_i+\delta_1,y_i+\delta_2)$

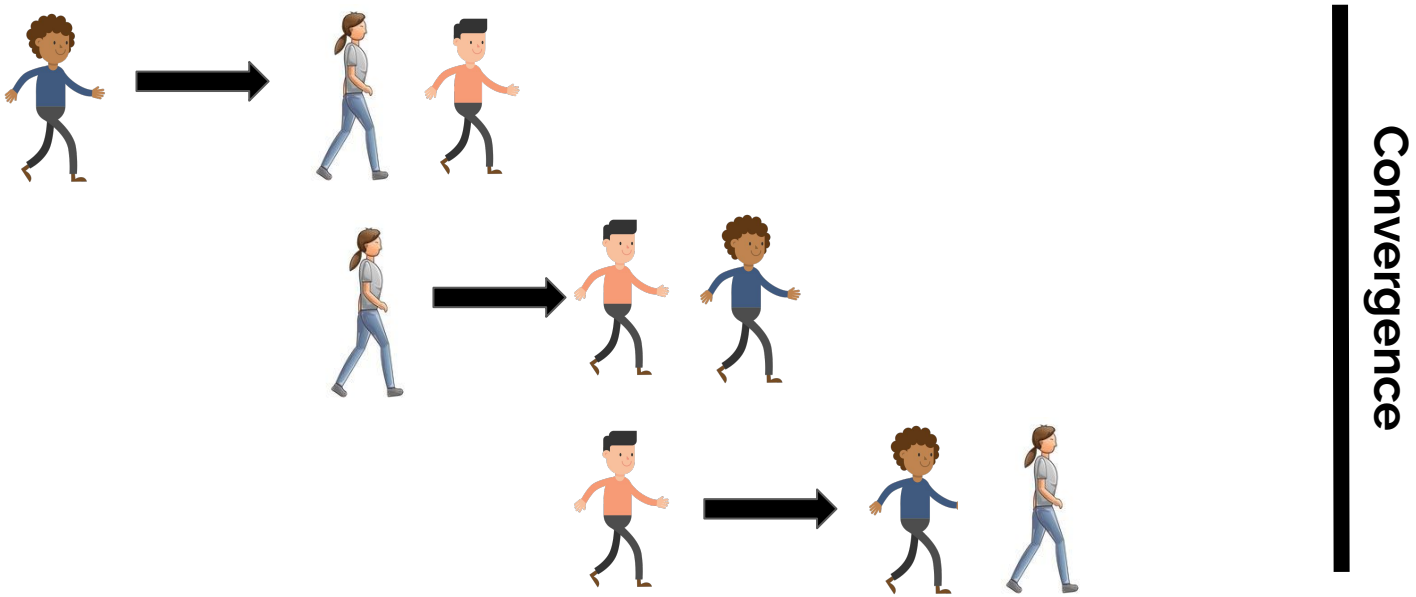
   where $\delta_j \sim N(0,\sigma_0^2)$;

3. Accept the proposed position (x'_i,y'_i) if it does not violate the constraints; otherwise stay put.



Gibbs Sampling

- Iteratively using conditional distributions, to construct Markov moves
- Underlying Markov chain is constructed by using a sequence



Let’s start modeling!

Bayesian Model

Level 1: Actual counts $y_i \sim Poisson(ƛ)$

Level 2: Parameter prior $ƛ \sim Unif(0, 20)$

Sampling and estimating parameters

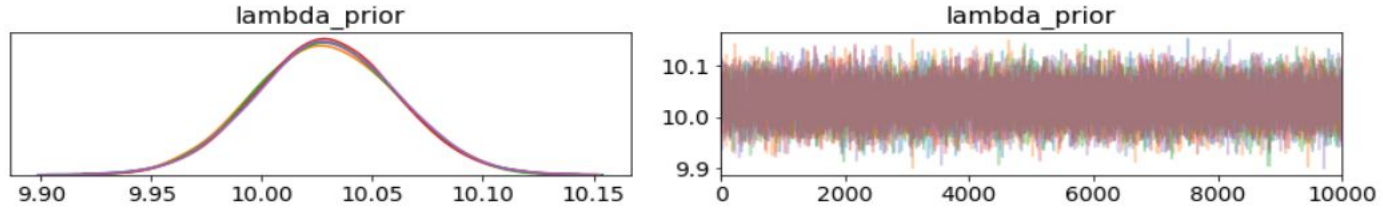
In [11]:

```
lambda_true = 10
n = 10000
y = np.random.poisson(lambda_true, n)

with pm.Model() as model:
    # Define prior
    lambda_prior = pm.Uniform('lambda_prior', 0, 20)

    # Define the observed level
    Y_obs = pm.Poisson('Y_obs', mu=lambda_prior, observed=y)

    # Sample
    trace = pm.sample(10000, chains = 5, tune = 2000)
```

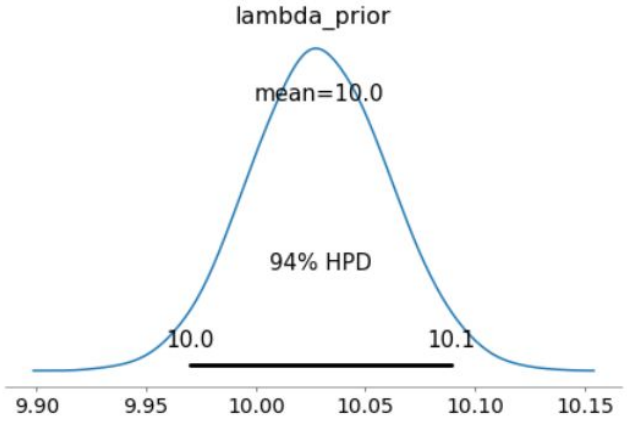
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (5 chains in 4 jobs)
NUTS: [lambda_prior]
Sampling 5 chains: 100%|██████████████████████████████████████████████| 60000/60000 [00:39<00:00, 1528.51draws/s]
```

Diagnostic Plots: Traceplot

```
pm.traceplot(trace);
```




```
pm.plot_posterior(trace);
```



Diagnostic Plots: Autocorrelation plot

In [17]:
```
burn_in = 5000
n_thin = 2
```

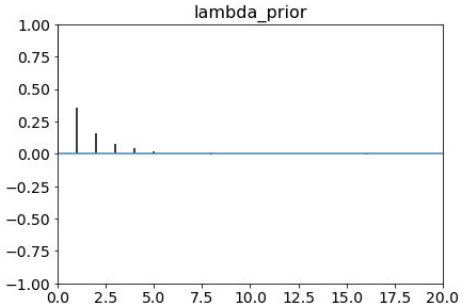
In [18]:
```
trace_thinned = trace[burn_in::n_thin]
```

In [19]:
```
pm.stats.summary(trace_thinned)
```

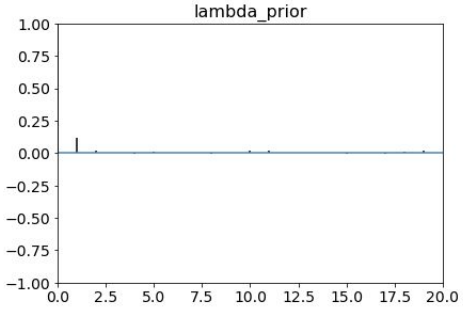
Out[19]:

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
lambda_prior	10.029377	0.031356	0.00036	9.966794	10.089653	9549.08365	1.000297

In [20]:
```
pm.autocorrplot(trace, max_lag=20, combined=True);
```



In [21]:
```
pm.autocorrplot(trace_thinned, max_lag=20, combined=True);
```



Bayesian Hierarchical Model: Disease Rate Mapping

Level 1: Actual counts

Level 2: Underlying spatial process

Level 3: Priors

$$y_i \sim Poisson(e^{Si}E_i)$$

$$S_i \sim p(.|\theta)$$

$$\theta \sim \pi()$$

i = spatial regions

S_i = Spatial process

E_i = expected counts of such populations in that region

$$y_i \sim Poisson(e^{Si}E_i)$$

$$S_i \sim MVN(0,\Sigma)$$

$$\Sigma_{ij} = \sigma^2\Omega_{ij}$$

$$\Omega_{ij} = \exp(-\phi d_{ij})$$

Priors:

$$\sigma^2 \sim IG(\alpha,\beta)$$

$$\phi \sim Unif(A,B)$$

Data

In [4]:

```
df.head()
```

Out[4]:

	state	actual	expected
9	Georgia	125	126.4
18	Maine	126	124.5
19	Maryland	132	131.4
30	New York	131	134.0
41	Texas	112	112.2

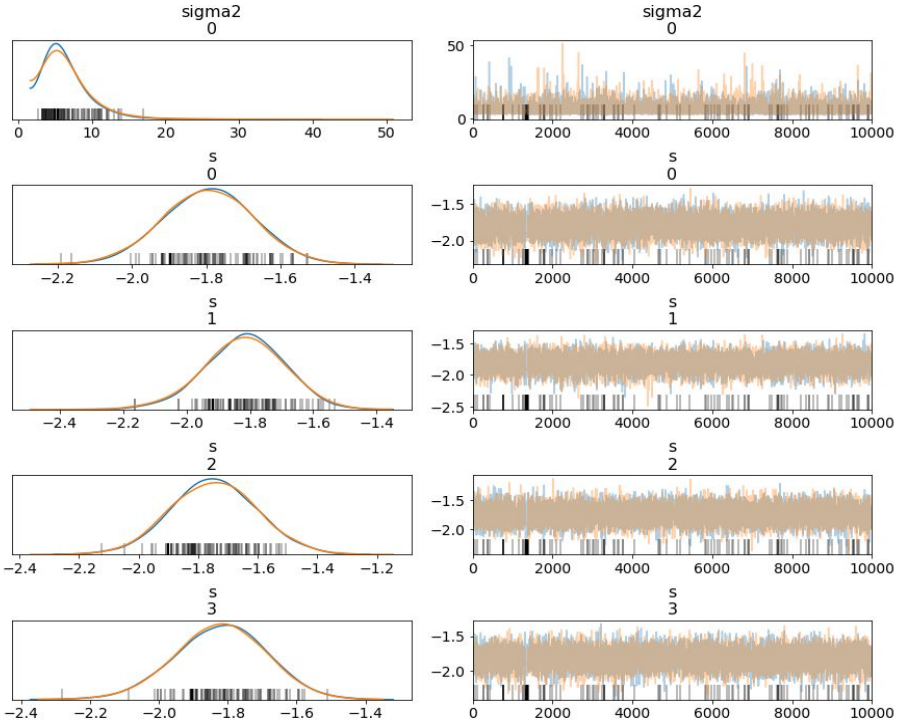
Model & Diagnostics

```
with pm.Model() as model:

    mu = np.zeros(N)

    sigma2 = pm.InverseGamma('sigma2', alpha=0.001, beta=0.001, shape=1)
    phi = pm.Uniform('phi', 0.0001, 100, shape=1)

    omega = T.exp(-phi * dist_mat)

    s = pm.MvNormal('s', mu=mu, cov=sigma2*omega, shape=5)

    p = pm.Poisson('y', mu=T.exp(s) * expected, observed=y)
    trace = pm.sample(10000, tune=5000, target_accept=0.95)
```

```
In [11]: pm.stats.summary(trace)
```

Out[11]:

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
s__0	-1.793581	0.122519	0.001450	-2.031428	-1.551101	8258.853827	1.000201
s__1	-1.816514	0.125030	0.001439	-2.068967	-1.575121	8108.114030	1.000150
s__2	-1.746722	0.143448	0.001451	-2.019468	-1.459145	9771.371065	0.999955
s__3	-1.820518	0.132165	0.001441	-2.079812	-1.562752	9481.177394	1.000047
sigma2__0	6.661579	3.339982	0.034284	2.240287	12.996738	9061.116703	1.000609
phi__0	0.023226	0.014208	0.000183	0.000593	0.046893	6970.850694	0.999971



Monitoring the model using diagnostic plots

- Trace plots not changing much
- Autocorrelation plots - check the need to thin
- Use burn-in period
- Gelman-Rubin statistic, MC error

Things to look out for

- Over-parametrized model - run simulations
- Sensitivity analysis with priors - robustness
- Extreme chain starting points

Key takeaways

1. Bayesian models are intuitive and go with the intuitive definition of probability.

2. Modeling spatial data in a Bayesian way gives you flexibility to incorporate prior information.

3. PyMC3 is a great framework to code models in and sample from the posteriors

Key takeaways

References

1. PyMC3 documentation: https://docs.pymc.io/
2. Thomas Wiecki's (the guy who wrote the PyMC3 library) blog: https://twiecki.io/
3. Good Intro blog: https://juanitorduz.github.io/intro_pymc3/
4. More about distributions: https://mlwhiz.com/blog/2017/09/14/discrete_distributions/

   Books:

5. Applied Bayesian statistics: with R and OpenBUGS examples Cowles, Mary Kathryn
6. Bayesian Analysis with Python, Osvaldo Martin

Thank you!

Slides and examples: http://bit.ly/2X5M0el

Stay in touch:

www.linkedin.com/in/shreya-khurana

https://shreyakhurana.github.io/