Bayesian Statistics with Python

An Example Method:
No Resampling Necessary

Charles Lindsey

Principal Data Scientist

Revionics®
an aptos company

Bayesian Statistics with Python

bayes_mapvar

Charles Lindsey
Principal Data Scientist

Revionics®
an aptos company

Bayesian Statistics with Python

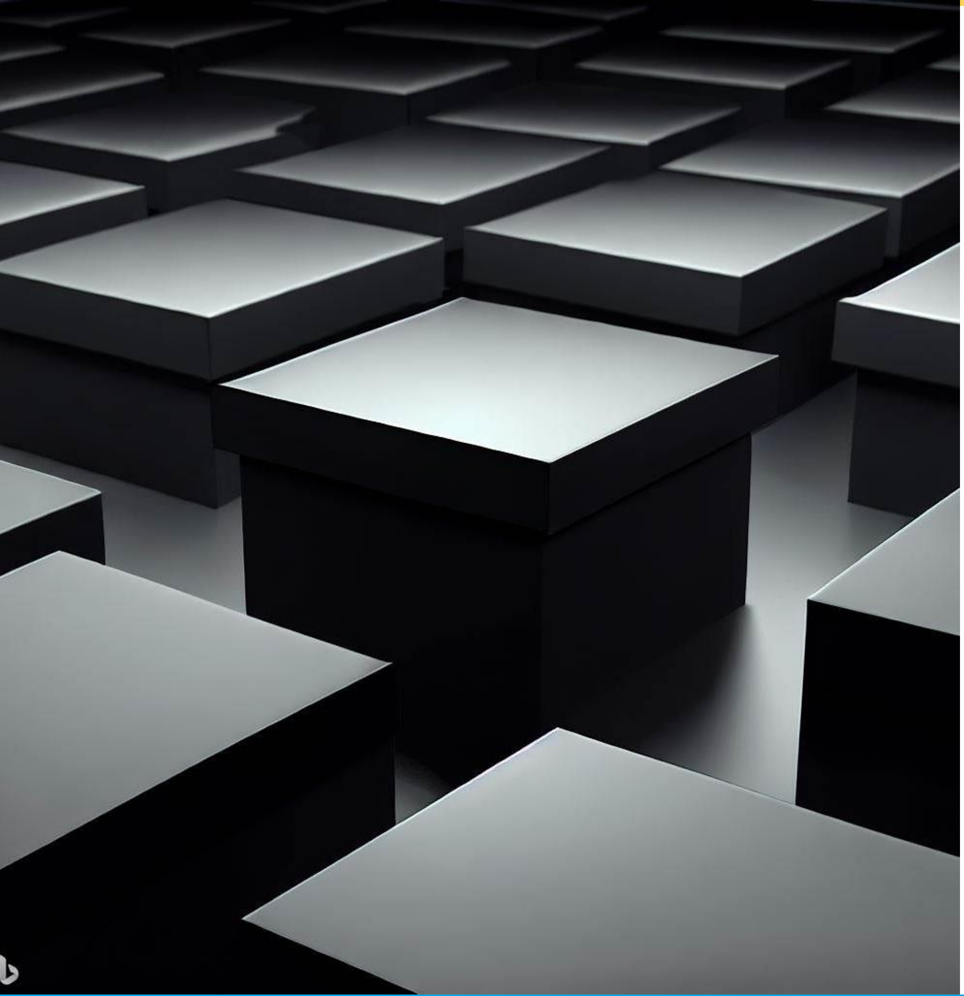
Data

- Everything begins with data.

Bayesian Statistics with Python

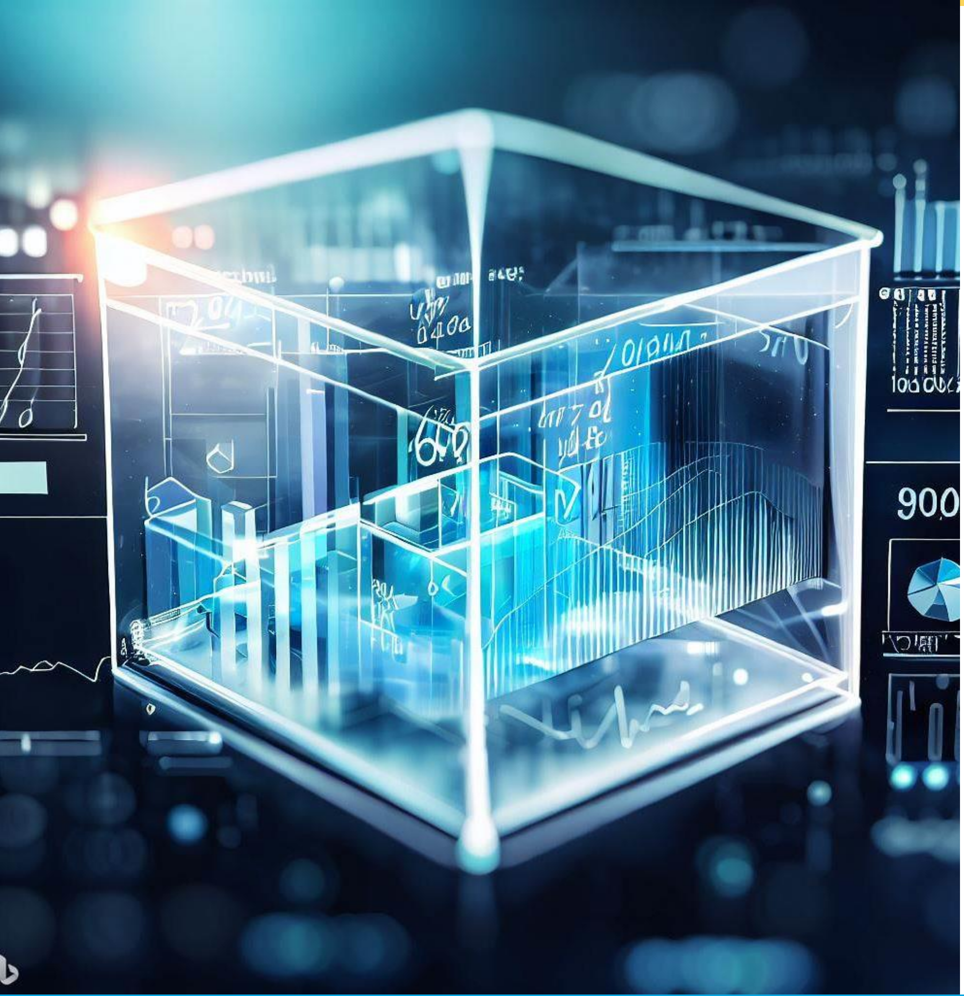
Data

- Everything begins with data.
- Once collected and organized, we want to learn from it.
- What methods should we use?



Black box methods

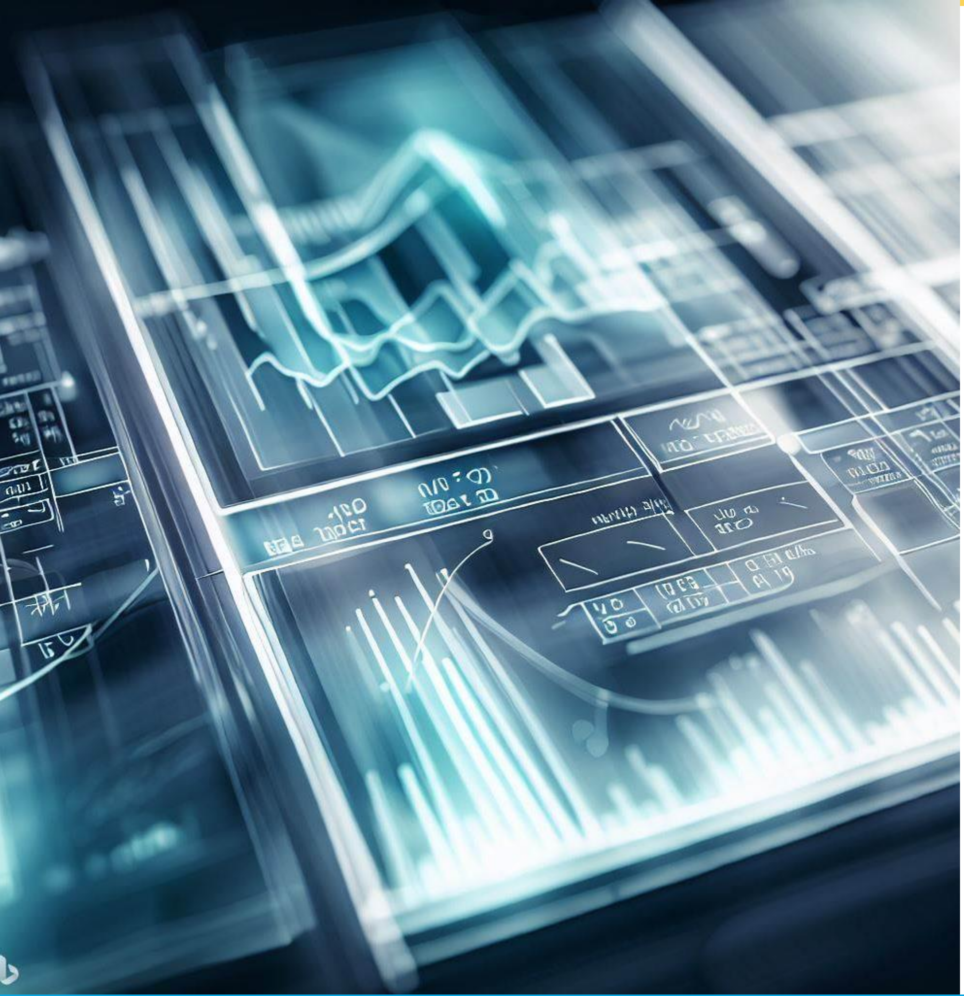
- Some techniques are very powerful at making accurate predictions.
- But the model they are providing predictions from is not easily interpretable.
- This can lead to misapplication of the methods as well, as analysts don't understand.
- Which can lead to bad prediction outside of training data.



Bayesian Statistics with Python

Bayesian Parametric

- Parametric models have parameters and an easily interpreted structure.
- They are based on statistical principles that let us interpret predictions and parameter estimates in terms of probabilities.
- Some black box methods only provide point estimates. No probabilities or confidence intervals. Not even a standard error to estimate how precise your answer is.
- And Bayesian?



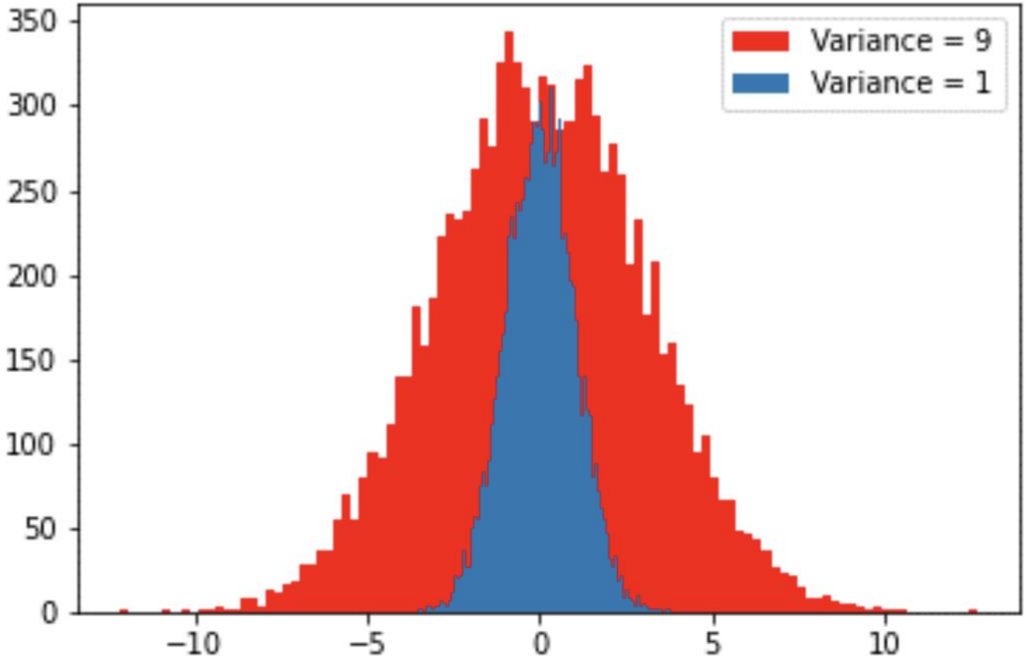
Bayesian Statistics with Python

Bayesian Prior

- Frequentist methods see parameters as fixed.
- You get new data and you re-estimate the parameters.
- You ignore the past.
- You ignore domain expertise (beyond setting up the observed data model).
- Bayesian analysis incorporates the past and domain expertise into a **prior**.
- The prior density is the model for the belief in the parameters before any data is observed.

Bayesian Statistics with Python

Prior Distributions



- We might believe that elasticity should be centered below 1 for products with static demand in the past.
- The variation of average temperature may be greater in the desert than an on the coast.
- Some parameters are constrained to be positive, while others could range over reals.
- We can have more confidence in some parameters than others being closer to their center.

REVIONICS an aptos company

Bayesian Statistics with Python

Regularization

- The prior information can regularize the modeling, preventing overfitting on the training data.
- The regularization is easily interpretable
- It is based on the meaning of the parameters, not their sizes.

Prior, Likelihood, Posterior

- The parameters and their confidence are evaluated using the parameter's probability distributions, characterized by their density.
- **Prior** density $\pi(\boldsymbol{\theta})$ represents the prior belief in parameters $\boldsymbol{\theta}$.
- **Likelihood** of the observed data $\mathbf{x}$, $L(\boldsymbol{\theta}|\mathbf{x})$ is the probability density of the observed data conditional on the parameters. You could view this as $\pi(\mathbf{x}|\boldsymbol{\theta})$
- **Posterior** density $\pi(\boldsymbol{\theta}|\mathbf{x})$ represents the updated belief in the parameters based on the observed data.
- $\pi(\boldsymbol{\theta}|\mathbf{x}) \propto L(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})$

Posterior Distribution

- That proportionality does not give us a distribution.
- **Resampling** methods like Markov Chain Monte Carlo methods can be used to estimate the posterior distribution. These make potential draws from the distribution and accept or reject the draws based on some criteria.
- **Computationally** intensive is a good adjective for resampling.
- **Asymptotically,** the posterior density is thankfully normal.
- So as the observed data **x** grows large in sample size we can approximate the posterior density with a multivariate normal distribution.

Posterior Distribution Approximation

- Details are given in the proceedings paper and attached references.
- At the posterior mode estimate, we can approximate the posterior with

$$\log \pi(\theta|\mathbf{x}) \approx \log \pi(\hat{\theta}|\mathbf{x}) + (\theta - \hat{\theta})^T S(\theta)|_{\theta=\hat{\theta}} + \frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta})$$

- **Score** S(**θ**), is the derivative of the log posterior with respect to the parameters **θ**.
- **Hessian** H(**θ**), is the second derivative of the log posterior with respect **θ**.
- Our approximation gets more accurate as the amount of observed data increases and the mode estimate gets closer to **θ** (like sample mean getting closer to population mean).
- Using Calculus, this means S(**θ**) goes to zero at the posterior mode and so...

Posterior Distribution Approximation

$$\pi(\theta|\mathbf{x}) \approx \pi(\hat{\theta}|\mathbf{x}) \exp(\frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta}))$$

- This is proportional to a normal density.
- Optimizing parameters over the real line can be easier than optimizing parameters on constrained spaces. So, we can formulate our true parameter space as being **unconstrained**.
- **Transformed**, or **constrained** parameters, $\boldsymbol{\theta}_c$ = $\mathbf{g}(\boldsymbol{\theta})$ are the final parameters that we will interpret. Then using the **Delta** method, we get:

$$\theta_c|x = g(\theta)|x \approx_D N\left(g(\hat{\theta}), -\frac{\delta}{\delta\hat{\theta}} g(\hat{\theta})^T H(\hat{\theta})^{-1} \frac{\delta}{\delta\hat{\theta}} g(\hat{\theta}) \right)$$

bayes_mapvar

- Finally, time for some Python.
- The package **bayes_mapvar** takes a Bayesian model specification and observed data, then outputs:
- Posterior Mode estimate, called the Maximum A Priori (MAP)
  - This is provided for the **unconstrained** parameters and a transform of it for the **constrained**.
- Posterior Variance estimate
  - **Hessian** estimate
  - **Delta** matrix (derivative of g) estimate
  - Posterior Variance of **Constrained** parameters estimate

TensorFlow Probability

- Bayesian model is specified using a dictionary of **TensorFlow Probability (TFP)** Distributions.
- TFP is a **Probabilistic Programming Language**. This lets us build powerful models in a modular way and estimate them automatically.
- **TensorFlow**, a leading software library for machine learning and artificial intelligence is the engine used to implement TFP.
- The **Distribution** class is how TFP represents **a** probability distribution, the set of rules that govern the likelihood of how a random variable (vector, or even general tensor) takes its values.
- Together with callables that ingest and output distributions, you can form a complex Bayesian model using a single dictionary.

C we think there might be a logistic model for winning a best picture oscar based on the covariates of whether the director has won an oscar previously, and the number of movie stars in the project, give some sample data for the model

View other drafts

Sure, here is some sample data for the logistic model for winning a Best Picture Oscar based on the covariates of whether the director has won an Oscar previously, and the number of movie stars in the project:

Movie	Director Won Oscar Previously	Number of Movie Stars	Won Oscar
The Godfather	Yes	3	Yes
The Shawshank Redemption	No	2	No
Titanic	Yes	4	Yes
The Departed	Yes	3	Yes
Parasite	No	1	Yes

Enter a prompt here

Bard may display inaccurate or offensive information that doesn't represent Google's views.

Bayesian Statistics

Simple Example

- We'll use Bard to get some data.
- Then we will setup a dictionary of distributions in TFP.
- We stick with a fun example with publicly available data.

Example

Setting up priors in a Bayesian analysis, such as in Bayesian logistic regression, requires careful consideration and understanding of your problem domain. Below are some general tips for setting priors for the coefficients in a logistic regression model in the context of predicting an Oscar win.

1. **Previous Oscar Win for the Director:** This could be a significant predictor, as one might expect a director who has previously won an Oscar to have a higher likelihood of winning again. However, it's important to note that many directors win only once, and some never win despite numerous nominations. A reasonable prior for this coefficient might be a normal distribution with mean 0 (i.e., no effect), but with a standard deviation that allows for a substantial potential effect, such as a standard deviation of 1 or 2.
2. **Number of Movie Stars in a Movie:** Defining "movie star" can be subjective, and there may not be a clear linear relationship between the number of movie stars in a movie and the probability of an Oscar win. Movies with a larger number of stars could have a higher chance of winning, but only up to a certain point. For this reason, a reasonable prior might again be a normal distribution centered around 0, but the standard deviation might be larger, indicating more uncertainty about the potential effect. If you have domain knowledge suggesting a maximum or minimum plausible effect size, you might consider a truncated normal prior, which is a normal distribution that's cut off at some boundary.

These priors are fairly uninformativ Regenerate response ongly commit to any particular

- We asked ChatGPT about priors for the coefficients.
- We **will** constrain the movie stars coefficient to be positive. Star power!
- See collab notebook at [link].

REVIONICS an aptos company

SciPy Optimization

- The dictionary of tensor parameter values are stacked together into a vector, and this is passed into **scipy.optimize** to maximize the posterior density.
- We use **method**= 'L-BFGS-B' to use the Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm in optimization.
- This is classified as a quasi-Newton method, being based on the Scores and an approximation of the Hessian matrix.
- TFP's automatic differentiation function **tfp.math.value_and_gradient** provides the gradient / score of the log posterior density to the loss function used by scipy.optimize

Variance Estimation

- Details on variance estimation are in the proceedings paper.
- We use numeric derivatives of the scores to estimate the Hessian of the unconstrained parameters.
- Numeric derivatives are also used for calculating the Delta matrix for the constrained parameters.
- The final variance matrix is computed as

```
np.matmul(
    np.matmul(dg,
              np.linalg.inv(hessian)),
    np.transpose(dg))
```

Simulation

- A simulation study was performed to corroborate that bayes_mapvar is estimating correctly. Details are in the proceedings paper, and this collab.
- The map estimates, variance estimates, and asymptotic normality of the posterior distribution was checked by comparing the mapvar results using MCMC methods implemented in TFP and the Anderson Darling tests from **scipy.stats**.
- We simulated draws from a linear regression model with an intercept that had a $\chi 2$ prior for the intercept and a normal prior for the slope.

Simulation

- Standardizing based on the estimated map and posterior variance from bayes_mapvar, we should find that the posterior density is approximately standard normal. The Anderson Darling test was used to assess this.
- We performed a successful hypothesis test of whether the rejection rate is .05 by checking whether .05 is in the confidence interval for the proportion. We used the **proportion_confint** function from **statsmodels.**

Statistics	Lower	Upper
α AD Reject	0.030	0.056
β A.D. Reject	0.033	0.060

TABLE 2
A.D. Confidence Intervals, $n_{pre} = 1000$, $n_{post} = 600$, $n_{MCMC} = 500$.

References

- Full references are in the proceedings paper.

- [1] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, Bayesian Data Analysis, Third Edition. Chapman & Hall/CRC Texts in Statistical Science, Taylor & Francis, 2013.
- [2] G. W. Oehlert, “A note on the delta method,” The American Statistician, vol. 46, no. 1, pp. 27–29, 1992.
- [3] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: a survey,” 2018.
- [4] W. Press and S. Teukolsky, Numerical Recipes 3rd Edition: The Art of Scientific Computing. Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, 2007.

Bayesian Statistics with Python

Thank you!

- Happy Modeling Everybody!
- Thanks for your time.
- GitHub: https://github.com/cdlindsey/bayes_mapvar
- Proceedings Paper (temporary): https://procbuild.scipy.org/download/cdlindsey-23_lindsey