UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
Statistics as Pottery: Bayesian Data Analysis using Probabilistic Programs (Tutorial)

Permalink
https://escholarship.org/uc/item/8hn1k8qp

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 40(0)

Author
Tessler, Michael Henry

Publication Date
2018

eScholarship.org Powered by the California Digital Library
University of California

Statistics as Pottery: Bayesian Data Analysis using Probabilistic Programs (Tutorial)

Michael Henry Tessler (mhtessler@stanford.edu), **Noah D. Goodman (ngoodman@stanford.edu)**
Department of Psychology, Stanford University

Abstract

Probability theory is the "logic of science" (Jaynes, 2003) and Bayesian data analysis (BDA) is the glue that brings that logic to data. BDA is a general, flexible alternative to standard statistical approaches (e.g., NHST) that provides the scientist with clarity and ease to address their personal scientific questions. Doing BDA in a probabilistic programming language (PPL) affords several additional advantages: a compositional approach to writing models, separation of model specification from algorithmic implementation (a la `lm()` in R), and continuity from articulating data analytic models to Bayesian cognitive models. Furthermore, specifying one's model and data analysis in a PPL allows you to search for "optimal experiments" for free. This tutorial will walk the participant through the basics of BDA to state-of-the-art applications, using an interactive online web-book and tools for integrating BDA into their existing workflow.

Keywords: bayesian data analysis; bayesian cognitive modeling; probabilistic programming

Significance

Learning statistics is like learning pottery. With pottery, you can learn how to make different shapes (e.g., a bowl, a vase, a spoon) without understanding general principles. Another way is to learn the basic strokes of forming pottery (e.g., how to mold a curved surface, a flat surface, long pointy things). In this course, we are going to learn the basic strokes of statistics by building generative models of data. We'll compose these strokes to make shapes you've seen before (e.g., a linear model), some shapes you've probably never seen before (e.g., a single model of a two-alternative forced-choice task and a three-alternative task), and explore how you would make new shapes if you needed to. We won't learn what tests apply to what data types but instead foster the ability to reason through data analysis. We will do this through the lens of Bayesian statistics, though the basic ideas will aid your understanding of classical (frequentist) statistics as well.

Bayesian data analysis (BDA) is a general-purpose, flexible data analytic approach for drawing inferences about hypotheses from data. Recent years have elucidated the shortcomings of the "classical" statistical framework—Null Hypothesis Significance Testing (NHST). NHST is opaque and rife with assumptions the scientist is often forced to adopt, which is partially credited for the reproducibility crises in psychology and related fields. BDA provides a principled alternative to NHST: Intuitive conclusions are drawn from intuitive, explicit assumptions. Major journals have made public their openness to alternatives to NHST (e.g., Lindsay, 2015), and Bayesian techniques are precisely that alternative.

For many years, BDA was a specialist's methodology because it required advanced computational skills to implement and fit Bayesian models. No longer: *Probabilistic programming languages* (PPLs) abstract away nuisance details and allow one to build models from scratch in few lines of code; they further provide black-box inference algorithms that can be used with little expertise. PPLs are programming languages have special operators for representing probabilities and randomness. Crucially, they provide an abstraction barrier between *model specification* (*what* we wish to compute) and the algorithm for returning the result (*how* we try to compute it), analogous to how the function `lm()` (in the programming language R) runs a linear model: The user simply specifies the functional form of the regression model (e.g., additive, interactive,...) and, under the hood, computation is done to return the answer. The PPL approach enhances the transparency of a model, which in turn allows the scientist to reason through and revise the model more easily.

PPLs are more than just a single model, however, they are full-blown programming languages that allow you to build *any* model you might need. Strange data types or experiment designs that violate assumptions of classical tests are no problem; all you have to do is be explicit about how you believe the data were generated (e.g., in the simplest case, that two experimental conditions gave rise to potentially different distributions of responses). If your hypothesis can be made more precise, PPLs provide an avenue to formalize it in a cognitive model. Have a cognitive model in hand but can't think of a way to precisely integrate it with the raw behavioral data? You could build a simple regression model to take your predictors (predictions of the cognitive model) into the appropriate data space. All of this can be achieved without leaving the same language. This kind of open-ended model specification cannot be achieved using probabilistic libraries or restricted probabilistic languages (e.g, STAN, BUGS).

State-of-the-art data analytic technologies are to be found in PPLs. For example, with a formal model in hand, one can automatically search for *good experiments* to run (i.e., experiments that, regardless of their outcome, strongly update your beliefs about a scientific question) using "optimal experiment design" (OED). OED searches from a space of possible experiments for those that distinguish a scientist's alternatives models. OED has been implemented in the PPL that this course will use (Ouyang, Tessler, Ly, & Goodman, 2016), and we will introduce the system at the end of the course.

This tutorial will be of broad interest to the Cognitive Science community because it touches upon a variety of distinct but related topics in the empirical disciplines. Foremost, it is a tutorial in modern data analysis and modeling, assuming no background knowledge of either Bayesian statistics or

probabilistic programming. By learning how to create simple Bayesian models in a PPL, participants will more easily grasp the relationship between the *generative process* of the data and the inferences drawn from observed data. Second, we draw the connection between simple data-analytic models (e.g., regression models) and Bayesian cognitive models, fostering an integrative theoretical view of data analysis and modeling. Finally, we demonstrate the power of explicit data modeling by showing how Optimal Experiment Design comes for free once a Bayesian data analysis has been chosen. Throughout, we show participants how to incorporate this kind of modeling into their everyday scientific workflow. By the end of the tutorial, participants will be able to

1. **Build** Bayesian statistical models for simple and complex problems using a probabilistic programming language
2. **Interpret** the various components of such a model in terms of one's scientific hypotheses
3. **Relate** Bayesian models to more orthodox statistical models (e.g., a linear model)
4. **Defend** a particular model specification (priors, likelihood) in a way that other cognitive scientists will understand

Structure

This one-day hands-on tutorial will introduce participants to Bayesian Data Analysis, providing a set of tools and techniques that will allow researchers to conduct BDA on their own. We will use the probabilistic programming language WebPPL (Goodman & Stuhlmüller, 2015), which is freely available to run in the web browser or on the command line, as well as via the programming language R,[1] to examine data sets from behavioral experiments. The course will be taught from interactive course notes that will be posted on the web. A schematic website can be found at `https://mhtess.github.io/bdappl/`

- Introduction to probabilistic programming
  - Theory: Basics of probability, conditional probability
  - Introduction to WebPPL / RWebPPL,
  - Building generative models with programs
- Simple Bayesian data analysis
  - Theory: Bayes' rule
  - Bayesian inference in WebPPL
  - Highest posterior density intervals
  - Model comparison (e.g., Bayes Factors)
- Joint inference models
  - Modeling data contamination
  - Building custom regression models from scratch
  - Hierarchical modeling / participant-level parameters
- Analyzing cognitive models
  - Theory: Cognitive models as model of a hypothesis
  - Learning about the parameters of a cognitive model
- Optimal Experiment Design
  - Theory: Representing multiple hypotheses as models
  - Practical: Running `webppl-oed` on models

Each participant will need a laptop but no additional materials are required for this tutorial.

Organizer Credentials

Tessler and Goodman are experts in Bayesian cognitive modeling and use BDA extensively in their own research. Tessler is a 5th year PhD student (*expected graduation June 2018*) who has designed and taught a 10-week graduate-level course on Bayesian Data Analysis at Stanford,[2] 1- to 2-week courses on Bayesian Data Analysis and probabilistic programming at the European Summer School for Logic, Language, and Information (ESSLLI), the Fall School for Computational Linguistics (German Linguistics Society), the Probabilistic Programming and Machine Learning Summer School (PPAML), and has been invited to teach a one-week course on these topics at ESSLLI in 2018.[3] He has co-authored an interactive web-book for probabilistic modeling of language (Scontras & Tessler, 2017), a chapter on Bayesian Data Analysis,[4] and an R package for interfacing with WebPPL.

Goodman is an Associate Professor in the Departments of Psychology and Computer Science at Stanford. He has taught Bayesian modeling for roughly a decade and co-authored `probmods.org`, a highly accessible interactive web-book, now in its 2nd edition, that is used for teaching Bayesian modeling at universities and institutes around the world (Goodman & Tenenbaum, 2016). Goodman developed the probabilistic programming language WebPPL, as well as two other PPLs: Church (Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum, 2008) and Pyro.

References

Goodman, N. D., Mansinghka, V. K., Roy, D. M., Bonawitz, K., & Tenenbaum, J. B. (2008). Church: a language for generative models..

Goodman, N. D., & Stuhlmüller, A. (2015). *The design and implementation of probabilistic programming languages*.

Goodman, N. D., & Tenenbaum, J. B. (2016). *Probabilistic Models of Cognition* (Second ed.). `http://probmods.org/v2`. (Accessed: 2018-1-29)

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge University Press.

Lindsay, D. S. (2015). *Replication in psychological science.* SAGE Publications Sage CA: Los Angeles, CA.

Ouyang, L., Tessler, M. H., Ly, D., & Goodman, N. (2016). Practical optimal experiment design with probabilistic programs. Retrieved from `http://arxiv.org/abs/1608.05046`

Scontras, G., & Tessler, M. H. (2017). *Probabilistic language understanding: An introduction to the Rational Speech Act framework.* `https://gscontras.github.io/probLang/`. (Accessed: 2017-10-18)

[1] `https://github.com/mhtess/rwebppl`

[2] `https://web.stanford.edu/class/psych201s/`

[3] `http://stanford.edu/%7Emtessler/short-courses/2017-computational-pragmatics/`

[4] `https://probmods.org/chapters/14-bayesian-data-analysis.html`