The IPython Notebook: open, reproducible scientific computing

Matthias Bussonnier[1], Jonathan Frederic[2], Bradley M. Froehle[3], Brian E. Granger[2], Paul Ivanov[3], Thomas Kluyver[3], Fernando Perez[3], Benjamin Ragan-Kelley[3] and Zachary Sailer[2]

[1]Affiliation of Matthias
[2]Cal Poly State University
[3]University of CA, Berkeley

June 30, 2013

Abstract

While computing has become a foundation of all it is challenging for researchers .

1 Introduction

Scientific research has become pervasively computational. In addition to experiment and theory, the notions of simulation and data-intensive discovery have emerged as third and fourth pillars of science [?]. Today, even theory and experiment are computational, as virtually all experimental work requires computing (whether in data collection, pre-processing or analysis) and most theoretical work requires symbolic and numerical support to develop and refine models. Scanning the pages of any major scientific journal, one is hard-pressed to find a publication in any discipline that doesn't depend on computing for its findings.

And yet, for all its importance, computing is often treated as an afterthought both in the training of our scientists and in the conduct of everyday research. Most working scientists have witnessed how computing is seen as a task of secondary importance that students and postdocs learn "on the go" with little training to ensure that results are trustworthy, comprehensible and ultimately a solid foundation for reproducible outcomes. Software and data are stored with poor organization, documentation and tests. A patchwork of software tools is used with limited attention paid to capturing the complex workflows that emerge, and the evolution of code is often not tracked over time, making it difficult to understand how a result was obtained. Finally, many of the software packages used by scientists in research are proprietary and closed-source, preventing the community from having a complete understanding of the final scientific results. The consequences of this cavalier approach are serious. Consider, just to name two widely publicized cases, the loss of public confidence in the "Climategate" fiasco [?] or the Duke cancer trials scandal, where sloppy computational practices likely led to severe health consequences for several patients [?].

This is a large and complex problem that requires changing the educational process for new scientists, the incentive models for promotions and rewards, the publication system, and more. We do not aim to tackle all of these issues here, but our belief is that a central element of this problem is the nature and quality of the software tools available for computational work in science.

2 Computing and the lifecycle of research

Based on our experience over the last decade as practicing researchers, educators and software developers, we propose an integrated approach to computing where the entire life-cycle of scientific research is considered, from the initial exploration of ideas and data to the presentation of final results. Briefly, this life-cycle can be broken down into the following phases:

- **Individual exploration:** a single investigator tests an idea, algorithm or question, likely with a small-scale test data set or simulation.
- **Collaboration:** if the initial exploration appears promising, more often than not some kind of collaborative effort ensues.
- **Production-scale execution:** large data sets and complex simulations often require the use of clusters, supercomputers or cloud resources in parallel.
- **Publication:** whether as a paper or an internal report for discussion with colleagues, results need to be presented to others in a coherent form.
- **Education:** ultimately, research results become part of the corpus of a discipline that is shared with students and colleagues, thus seeding the next cycle of research.

In this project, we tackle the following problem. **There are no software tools capable of spanning the entire lifecycle of computational research.** The result is that researchers are forced to use a large number of disjoint software tools in each of these phases in an awkward workflow that hinders collaboration and reduces efficiency, quality, robustness and reproducibility.

These can be illustrated with an example: a researcher might use Matlab for prototyping, develop high-performance code in C, run post-processing by twiddling controls in a Graphical User Interface (GUI), import data back into Matlab for generating plots, polish the resulting plots by hand in Adobe Illustrator, and finally paste the plots into a publication manuscript or PowerPoint presentation. But what if months later the researcher realizes there is a problem with the results? What are the chances they will be able to know what buttons they clicked, to reproduce the workflow that can generate the updated plots, manuscript and presentation? What are the chances that other researchers or students could reproduce these steps to learn the new method or understand how the result was obtained? How can reviewers validate that the programs and overall workflow are free of errors? Even if the researcher successfully documents each program and the entire workflow, they have to carry an immense cognitive burden just to keep track of everything.

3 The IPython Notebook

We propose that the open source IPython project [?] offers a solution to these problems; a single software tool capable of spanning the entire life-cycle of computational research. Amongst high-level open source programming languages, Python is today the leading tool for general-purpose source scientific computing (along with R for statistics), finding wide adoption across research disciplines, education and industry and being a core infrastructure tool at institutions such as CERN and the Hubble Space Telescope Science Institute [?, ?, ?]. The PIs created IPython as a system for interactive and parallel computing that is the *de facto* environment for scientific Python. In the last year we have developed the IPython Notebook, a web-based *interactive computational notebook* that combines code, text, mathematics, plots and rich media into a single document format (see Fig. ??). The IPython Notebook was designed to enable researchers to move fluidly between all the phases of the research life-cycle and has gained rapid adoption. It provides an integrated environment for all computation, without locking scientists into a specific tool or format: Notebooks can always be exported into regular scripts and IPython supports the execution of code in other languages such as R, Octave, bash, etc. In this project we will expand its capabilities and relevance in the following phases of the research cycle: interactive exploration, collaboration, publication and education.

3.1 Web application

3.2 Notebook document format

3.3 Display architecture

3.4 Installation

4 Collaboration

5 Broader ecosystem

6 Future directions

Dexy Snippets

This notebook uses format `3.0`.

Markdown Content

Here is an example of a markdown cell:

3What is the mean of λ_1 **given** we know τ is less than 45. That is, suppose we have new information as we know for certain that the change in behaviour occurred before day 45. What is the expected value of λ_1 now? (You do not need to redo the PyMC part, just consider all instances where ‘tau$_samples < 45$‘.)

Here is the same cell in a Verbatim block:

```
3\. What is the mean of $\lambda_1$ **given** we know $\tau$ is less than 45.  That is,
```

Here is the same cell presented as highlighted markup:

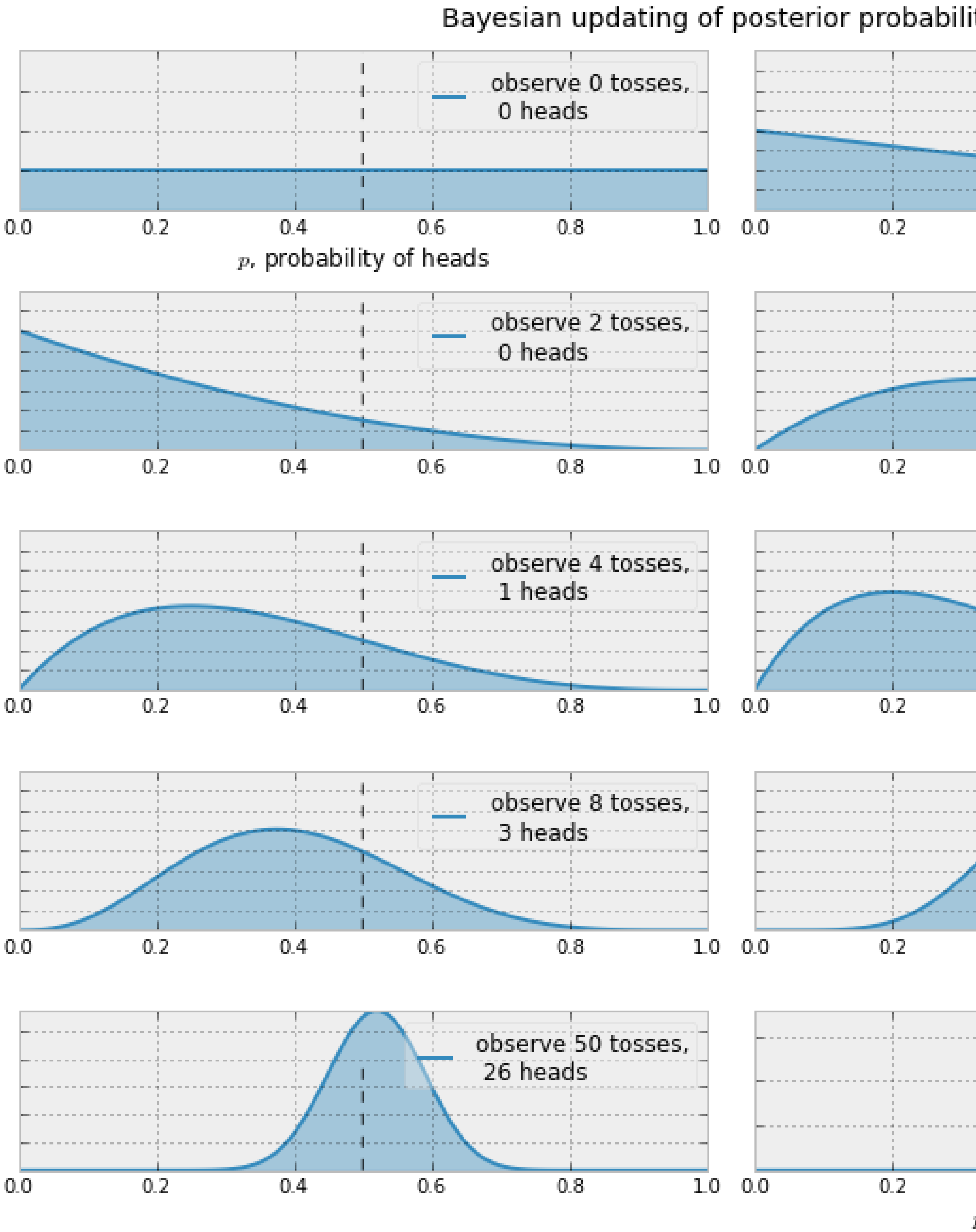
```
3\. What is the mean of $\lambda_1$ **given** we know $\tau$ is less than 45.  That is,
```

Python Content

```
"""
The book uses a custom matplotlibrc file, which provides the unique styles for matplotli
If executing this book, and you wish to use the book’s styling, provided are two options
    1. Overwrite your own matplotlibrc file with the rc-file provided in the book’s styl
       See http://matplotlib.org/users/customizing.html
    2. Also in the styles is  bmh_matplotlibrc.json file. This can be used to update the
       in only this notebook. Try running the following code:

        import json
        s = json.load( open("../styles/bmh_matplotlibrc.json") )
        matplotlib.rcParams.update(s)

"""

#the code below can be passed over, as it is currently not important.
%pylab inline
figsize( 11, 9)

import scipy.stats as stats

dist = stats.beta
n_trials = [0,1,2,3,4,5,8,15, 50, 500]
data = stats.bernoulli.rvs(0.5, size = n_trials[-1] )
x = np.linspace(0,1,100)

for k, N in enumerate(n_trials):
    sx = subplot( len(n_trials)/2, 2, k+1)
    plt.xlabel("$p$, probability of heads") if k in [0,len(n_trials)-1] else None
    plt.setp(sx.get_yticklabels(), visible=False)
    heads = data[:N].sum()
    y = dist.pdf(x, 1 + heads, 1 + N - heads )
```

```
    plt.plot( x, y, label= "observe %d tosses,\n %d heads"%(N,heads) )
    plt.fill_between( x, 0, y, color="#348ABD", alpha = 0.4 )
    plt.vlines( 0.5, 0, 4, color = "k", linestyles = "--", lw=1 )

    leg = plt.legend()
    leg.get_frame().set_alpha(0.4)
    plt.autoscale(tight = True)


plt.suptitle( "Bayesian updating of posterior probabilities",
              y = 1.02,
              fontsize = 14);

plt.tight_layout()
```

```
Welcome to pylab, a matplotlib-based Python environment [backend: module://IPython.zmq.p
For more information, type 'help(pylab)'.
```


Bayesian updating of posterior probabilit
observe 0 tosses,
0 heads
p, probability of heads
observe 2 tosses,
0 heads
observe 4 tosses,
1 heads
observe 8 tosses,
3 heads
observe 50 tosses,
26 heads
0.0
0.2
0.4
0.6
0.8
1.0

Iterating over Cells and Documents

This notebook has 47 cells and 59 documents based on these cells.

Cells

Here is a list of cells:

- [u'markdown', {u'source': u'Probabilistic Programming\n=====\nand Bayesian Methods

- [u'markdown', {u'source': u'Chapter 1\n======\n***', u'cell_type': u'markdown', u'm

- [u'markdown', {u'source': u'The Philosophy of Bayesian Inference\n------\n \n> You

- [u'markdown', {u'source': u'\n###The Bayesian state of mind\n\n\nBayesian inference
  ), *interpretedastheprobabilityof* A*giventheevidence*X.*Wecalltheupdatedbeliefthe∗posteriorprobabil*
  soastocontrastitwiththepriorprobability.*Forexample*, *considertheposteriorprobabilities*(*read* :
  posteriorbeliefs)*oftheaboveexamples*, *afterobservingsomeevidence*X. : 1
  .P(A):
  ;
  ;*thecoinhasa*50*percentchanceofbeingheads*.P(A — X):
  ;
  ;*Youlookatthecoin*, *observeaheadshaslanded*, *denotethisinformation*X, *andtriviallyassignprobabili*
  .P(A):
  ;
  ;*Thisbig*, *complexcodelikelyhasabuginit*.P(A — X):
  ;
  ;*Thecodepassedall*X*tests*; *therestillmightbeabug*, *butitspresenceislesslikelynow*.3
  .P(A):
  ;
  ;*Thepatientcouldhaveanynumberofdiseases*.P(A — X):
  ;
  ;*Performingabloodtestgeneratedevidence*X, *rulingoutsomeofthepossiblediseasesfromconsideratio*
  re−*weightedtheprior*∗*toincorporatethenewevidence*(*i*.*e*.*weputmoreweight*, *orconfidence*, *onsomebe*
  lesswrong∗.*Thisisthealternativesideofthepredictioncoin*, *wheretypicallywetrytobe*∗
  moreright∗.′, *u*′*cell*$_t$*ype*′ : *u*′*markdown*′, *u*′*metadata*′ :]|[u'markdown', {u'source': u'\n###Baye

- [u'markdown', {u'source': u"### Our Bayesian framework\n\nWe are interested in beli
) =
  frac P(X — A) P(A) P(X)

  propto P(X — A) P(A)
  ;
  ; (

propto
textis proportional to)
endalignabove formula is not unique to Bayesian inference: it is a mathematical fact with uses outside Bayesian inference. Bayesian inference merely uses it to connect prior probabilities $P(A)$ with an updated posterior probabilities $P(A|X)$.", u'cell$_t$ype' : u'markdown', u'metadata' :]|[u'markdown', {u'source': u"##### Example: Mandatory coin-f

- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'text': u"
- [u'markdown', {u'source': u'The posterior probabilities are represented by the curv
- [u'markdown', {u'source': u'#####Example: Bug, or just sweet, unintended feature?\n
— A), i.e., theprobabilitythatthecodepassesXtests*given*therearenobugs?Well, itisequalto1, foraco
TheeventXcanbedividedintotwopossibilities, eventXoccurringeventhoughourcode*indeedhas*
bugs(denoted
sim A
;, spoken * notA*), oreventXwithoutbugs(A).P(X)canberepresentedas :', u'cell$_t$ype' :
u'markdown', u'metadata' :]|[u'markdown', {u'source': u'\\begin{align}\nP(X) & = P(X \
- [u'markdown', {u'source': u"We have already computed $P(XA)above.Ontheotherhand,P(X
—
sim A)issubjective : ourcodecanpasstestsbutstillhaveabuginit, thoughtheprobabilitythereisabugpres
sim A) = 0.5.Then
beginalign(A|X) =
frac1cdotp1cdotp + 0.5(1 − p)

=
frac2p1 + p
endalignistheposteriorprobability.WhatdoesitlooklikeasafunctionofourpriorP,p
in [0,1]?", u'cell$_t$ype' : u'markdown', u'metadata' :]|[u'code', {u'cell_type': u'code', u'langu
- [u'markdown', {u'source': u"We can see the biggest gains if we observe the X test
givenwesawalltestspass*, hence1-P(A—X)istheprobabilitythereisabug*givenalltestspassed*
.Whatdoesourposteriorprobabilitylooklike?Belowisagraphofboththepriorandtheposteriorprobabilit
u'markdown', u'metadata' :]|[u'code', {u'cell_type': u'code', u'language': u'python', u
- [u'markdown', {u'source': u'Notice that after we observed X occur, the probability
- [u'markdown', {u'source': u"_______\n\n##Probability Distributions\n\n\n**Let's qui
;
lambda
;
large] =
lambda

willusethispropertyoften, soit'ssomethingusefultoremember.Belowweplottheprobabilitymassdistr

lambda$values.The first thing to notice is that by increasing$
lambda$we add more probability to larger values occurring. Secondly, notice that although the graph ends a$
$negative integer.", u'cell_type': u'markdown', u'metadata':]|$[u'code', {u'cell_type': u'code',

- [u'markdown', {u'source': u'###Continuous Case\nInstead of a probability mass funct
lambda) =
lambda e$^{-lambdaz}$,
;
; z
$ge0 the Poisson random variable, an exponential random variable can only take on non-negative values.$
$any*non-negative values, like 4.25 or 5.612401. This makes it a poor choice for count data, which must bei$
$and positive * variable. Below are two probability density functions with different$
lambda$value. a random variable Z has an exponential distribution with parameter$
lambda, $we say *Z is exponential* and write Z$
sim
$textExp($
$lambda) a specific$
lambda, $the expected value of an exponential random variable is equal to the inverse of$
lambda, $that is: E[$
; Z
; |
;
$lambda$
;] =
$frac1lambda', u'cell_type': u'markdown', u'metadata':]|$[u'code', {u'cell_type': u'code', u'l
- [u'markdown', {u'source': u'\n###But what is $\\lambda \\;$?\n\n\n**This question i
- [u'markdown', {u'source': u"\n##### Example: Inferring behaviour from text-message
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'png': u'i
- [u'markdown', {u'source': u"Before we begin, with respect to the plot above, would
;
alpha] =
frac1
alpha

$$, and something I encourage the reader to try, is to have two priors: one for each$$

lambda$_i$; creating two exponential distributions with different
alpha values reflects a prior belief that the rate changed after some period.about
tau? Well, due to the randomness, it is too difficult to pick out when
tau might have occurred. Instead, we can assign an *uniform prior belief* to every possible day. This is equivalent to saying

beginalign
tau
sim
textDiscreteUniform(1,70)

Rightarrow P(
tau = k) =
frac170
endalignafter all this, what does our overall prior for the unknown variables look like? Frankly, *it doesn't matter*. What we should understand is that it would be an ugly, complicated, mess involving symbols only a mathematician would love. And things would only get uglier the more complicated our models become. Regardless, all we really care about is the posterior distribution. We next turn to PyMC, a Python library for performing Bayesian analysis, that is agnostic to the mathematical monster we have created. our first hammer: PyMC—–is a Python library for programming Bayesian analysis [3]. It is a fast, well-maintained library. The only unfortunate part is that documentation can be lacking in areas, especially the bridge between beginner to hacker. One of this book's main goals is to solve that problem, and also to demonstrate why PyMC is so cool.will model the above problem using the PyMC library. This type of programming is called *probabilistic programming*, an unfortunate misnomer that invokes ideas of randomly-generated code and has likely confused and frightened users away from this field. The code is not random. The title is given because we create probability models using programming variables as the model's components, that is, model components are first-class primitives in this framework. . Cronin [5] has a very motivating description of probabilistic programming:¿ Another way of thinking about this: unlike a traditional program, which only runs in the forward directions, a probabilistic program is run in both the forward and backward direction. It runs forward to compute the consequences of the assumptions it contains about the world (i.e., the model space it represents), but it also runs backward from the data to constrain the possible explanations. In practice, many probabilistic programming systems will cleverly interleave these forward and backward operations to efficiently home in on the best explanations.to its poorly understood title, I'll refrain from using the name *probabilistic programming*. Instead, I'll simply use *programming*, as that is what it really is. PyMC code is easy to follow along: the only novel thing should be the syntax, and I will interrupt the code to explain sections. Simply remember we are representing the model's components (
tau,
$lambda_1$,
$lambda_2$) as variables:", u'cell$_type' : u'markdown', u'metadata' :$]| `[u'code', {u'cell_type': u'co`

- `[u'markdown', {u'source': u"In the above code, we create the PyMC variables corresp`

- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'text': u'
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'markdown', {u'source': u'This code is creating a new function ‘lambda_‘, but rea
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'markdown', {u'source': u'The variable ‘observation‘ combines our data, ‘count_da
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'text': u'
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'png': u'i
- [u'markdown', {u'source': u"### Interpretation\n\nRecall that the Bayesian methodol
- [u'markdown', {u'source': u"###Why would I want samples from the posterior, anyways
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'png': u'i
- [u'markdown', {u'source': u"Our analysis shows strong support for believing the use
- [u'markdown', {u'source': u'##### Exercises\n\n1\\. Using ‘lambda_1_samples‘ and ‘
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'markdown', {u'source': u'2\\. What is the expected percentage increase in text-
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'markdown', {u'source': u'3\\. What is the mean of $\\lambda_1$ **given** we know
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps
- [u'markdown', {u'source': u'### References\n\n\n- [1] Gelman, Andrew. N.p.. Web. 2
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [{u'text': u'
- [u'code', {u'cell_type': u'code', u'language': u'python', u'outputs': [], u'collaps

Documents

Here is a list of documents:

- notebooks/Chapter1_Introduction--0.md
- notebooks/Chapter1_Introduction--1.md
- notebooks/Chapter1_Introduction--2.md
- notebooks/Chapter1_Introduction--3.md

- notebooks/Chapter1_Introduction--4.md
- notebooks/Chapter1_Introduction--5.md
- notebooks/Chapter1_Introduction--6.md
- notebooks/Chapter1_Introduction--7-input.py
- notebooks/Chapter1_Introduction--7-output-0.txt
- notebooks/Chapter1_Introduction--7-output-1.png
- notebooks/Chapter1_Introduction--8.md
- notebooks/Chapter1_Introduction--9.md
- notebooks/Chapter1_Introduction--10.md
- notebooks/Chapter1_Introduction--11.md
- notebooks/Chapter1_Introduction--12-input.py
- notebooks/Chapter1_Introduction--12-output-0.png
- notebooks/Chapter1_Introduction--13.md
- notebooks/Chapter1_Introduction--14-input.py
- notebooks/Chapter1_Introduction--14-output-0.png
- notebooks/Chapter1_Introduction--15.md
- notebooks/Chapter1_Introduction--16.md
- notebooks/Chapter1_Introduction--17-input.py
- notebooks/Chapter1_Introduction--17-output-0.png
- notebooks/Chapter1_Introduction--18.md
- notebooks/Chapter1_Introduction--19-input.py
- notebooks/Chapter1_Introduction--19-output-0.png
- notebooks/Chapter1_Introduction--20.md
- notebooks/Chapter1_Introduction--21.md
- notebooks/Chapter1_Introduction--22-input.py
- notebooks/Chapter1_Introduction--22-output-0.png
- notebooks/Chapter1_Introduction--23.md

- notebooks/Chapter1_Introduction--24-input.py
- notebooks/Chapter1_Introduction--25.md
- notebooks/Chapter1_Introduction--26-input.py
- notebooks/Chapter1_Introduction--26-output-0.txt
- notebooks/Chapter1_Introduction--27-input.py
- notebooks/Chapter1_Introduction--28.md
- notebooks/Chapter1_Introduction--29-input.py
- notebooks/Chapter1_Introduction--30.md
- notebooks/Chapter1_Introduction--31-input.py
- notebooks/Chapter1_Introduction--31-output-0.txt
- notebooks/Chapter1_Introduction--31-output-1.txt
- notebooks/Chapter1_Introduction--32-input.py
- notebooks/Chapter1_Introduction--33-input.py
- notebooks/Chapter1_Introduction--33-output-0.png
- notebooks/Chapter1_Introduction--34.md
- notebooks/Chapter1_Introduction--35.md
- notebooks/Chapter1_Introduction--36-input.py
- notebooks/Chapter1_Introduction--36-output-0.png
- notebooks/Chapter1_Introduction--37.md
- notebooks/Chapter1_Introduction--38.md
- notebooks/Chapter1_Introduction--39-input.py
- notebooks/Chapter1_Introduction--40.md
- notebooks/Chapter1_Introduction--41-input.py
- notebooks/Chapter1_Introduction--42.md
- notebooks/Chapter1_Introduction--43-input.py
- notebooks/Chapter1_Introduction--44.md
- notebooks/Chapter1_Introduction--45-input.py

- notebooks/Chapter1_Introduction--46-input.py

Here are the contents of documents:

Here are the contents of notebooks/Chapter1_Introduction--0.md:

```
Probabilistic Programming
=====
and Bayesian Methods for Hackers
========

#####Version 0.1
Welcome to *Bayesian Methods for Hackers*. The full Github repository, and additional ch
```

Here are the contents of notebooks/Chapter1_Introduction--1.md:

```
Chapter 1
======
***
```

Here are the contents of notebooks/Chapter1_Introduction--2.md:

```
The Philosophy of Bayesian Inference
------

> You are a skilled programmer, but bugs still slip into your code. After a particularly

If you think this way, then congratulations, you already are a Bayesian practitioner! Ba
```

Here are the contents of notebooks/Chapter1_Introduction--3.md:

```
###The Bayesian state of mind


Bayesian inference differs from more traditional statistical inference by preserving *un

The Bayesian world-view interprets probability as measure of *believability in an event*

For this to be clearer, we consider an alternative interpretation of probability: *Frequ

Bayesians, on the other hand, have a more intuitive approach. Bayesians interpret a prob

Notice in the paragraph above, I assigned the belief (probability) measure to an *indivi

- I flip a coin, and we both guess the result. We would both agree, assuming the coin is

-  Your code either has a bug in it or not, but we do not know for certain which is true
```

- A medical patient is exhibiting symptoms x, y and z. There are a number of dise

This philosophy of treating beliefs as probability is natural to humans. We employ it co

To align ourselves with traditional probability notation, we denote our belief about eve

John Maynard Keynes, a great economist and thinker, said "When the facts change, I chang

1\. $P(A): \;\;$ the coin has a 50 percent chance of being heads. $P(A | X):\;\;$ You lo

2\. $P(A): \;\;$ This big, complex code likely has a bug in it. $P(A | X): \;\;$ The

3\. $P(A):\;\;$ The patient could have any number of diseases. $P(A | X):\;\;$ Performi

It's clear that in each example we did not completely discard the prior belief after see

By introducing prior uncertainty about events, we are already admitting that any guess w

Here are the contents of `notebooks/Chapter1_Introduction--4.md`:

###Bayesian Inference in Practice

 If frequentist and Bayesian inference were programming functions, with inputs being sta

For example, in our debugging problem above, calling the frequentist function with the a

>   *YES*, with probability 0.8; *NO*, with probability 0.2

This is very different from the answer the frequentist function returned. Notice that th

####Incorporating evidence

As we acquire more and more instances of evidence, our prior belief is *washed out* by t

Denote N as the number of instances of evidence we possess. As we gather an *infinite*

One may think that for large N, one can be indifferent between the two techniques sinc

```
> Sample sizes are never large. If $N$ is too small to get a sufficiently-precise estima
```

```
### Are frequentist methods incorrect then?

**No.**

Frequentist methods are still useful or state-of-the-art in many areas. Tools like Least


#### A note on *Big Data*
Paradoxically, big data's predictive analytic problems are actually solved by relatively

The much more difficult analytic problems involve *medium data* and, especially troubles
```

Here are the contents of `notebooks/Chapter1_Introduction--5.md`:

```
### Our Bayesian framework

We are interested in beliefs, which can be interpreted as probabilities by thinking Baye

Secondly, we observe our evidence. To continue our buggy-code example: if our code passe

\begin{align}
 P( A | X ) = & \frac{ P(X | A) P(A) } {P(X) } \\\\[5pt]
& \propto P(X | A) P(A)\;\; (\propto \text{is proportional to } )
\end{align}

The above formula is not unique to Bayesian inference: it is a mathematical fact with us
```

Here are the contents of `notebooks/Chapter1_Introduction--6.md`:

```
##### Example: Mandatory coin-flip example

Every statistics text must contain a coin-flipping example, I'll use it here to get it o

We begin to flip a coin, and record the observations: either $H$ or $T$. This is our obs

Below we plot a sequence of updating posterior probabilities as we observe increasing am
```

Here are the contents of `notebooks/Chapter1_Introduction--7-input.py`:

```
"""
The book uses a custom matplotlibrc file, which provides the unique styles for matplotli
If executing this book, and you wish to use the book's styling, provided are two options
    1. Overwrite your own matplotlibrc file with the rc-file provided in the book's styl
```

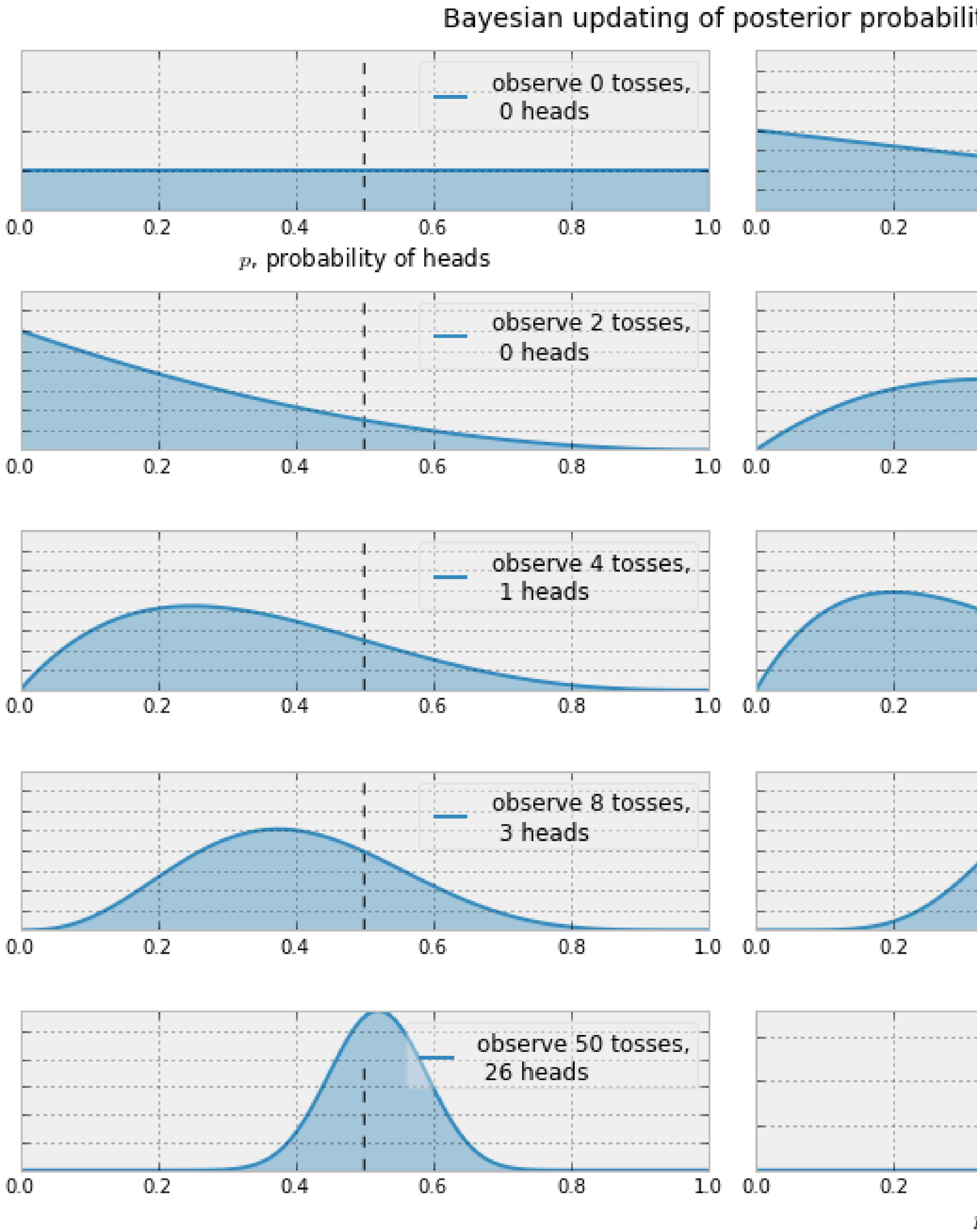
```
    See http://matplotlib.org/users/customizing.html
 2. Also in the styles is  bmh_matplotlibrc.json file. This can be used to update the
    in only this notebook. Try running the following code:

     import json
     s = json.load( open("../styles/bmh_matplotlibrc.json") )
     matplotlib.rcParams.update(s)

"""

#the code below can be passed over, as it is currently not important.
%pylab inline
figsize( 11, 9)

import scipy.stats as stats

dist = stats.beta
n_trials = [0,1,2,3,4,5,8,15, 50, 500]
data = stats.bernoulli.rvs(0.5, size = n_trials[-1] )
x = np.linspace(0,1,100)

for k, N in enumerate(n_trials):
    sx = subplot( len(n_trials)/2, 2, k+1)
    plt.xlabel("$p$, probability of heads") if k in [0,len(n_trials)-1] else None
    plt.setp(sx.get_yticklabels(), visible=False)
    heads = data[:N].sum()
    y = dist.pdf(x, 1 + heads, 1 + N - heads )
    plt.plot( x, y, label= "observe %d tosses,\n %d heads"%(N,heads) )
    plt.fill_between( x, 0, y, color="#348ABD", alpha = 0.4 )
    plt.vlines( 0.5, 0, 4, color = "k", linestyles = "--", lw=1 )

    leg = plt.legend()
    leg.get_frame().set_alpha(0.4)
    plt.autoscale(tight = True)


plt.suptitle( "Bayesian updating of posterior probabilities",
              y = 1.02,
              fontsize = 14);

plt.tight_layout()
```

Here are the contents of `notebooks/Chapter1_Introduction--7-output-0.txt`:

```
Welcome to pylab, a matplotlib-based Python environment [backend: module://IPython.zmq.p
```

```
For more information, type 'help(pylab)'.
```

Here are the contents of `notebooks/Chapter1_Introduction--7-output-1.png`:



Here are the contents of `notebooks/Chapter1_Introduction--8.md`:

```
The posterior probabilities are represented by the curves, and our confidence is proport

Notice that the plots are not always *peaked* at 0.5. There is no reason it should be: r

The next example is a simple demonstration of the mathematics of Bayesian inference.
```

Here are the contents of notebooks/Chapter1_Introduction--9.md:

```
#####Example: Bug, or just sweet, unintended feature?


Let $A$ denote the event that our code has **no bugs** in it. Let $X$ denote the event t

We are interested in $P(A|X)$, i.e. the probability of no bugs, given our debugging test

What is $P(X | A)$, i.e., the probability that the code passes $X$ tests *given* there a

$P(X)$ is a little bit trickier: The event $X$ can be divided into two possibilities, ev
```

Here are the contents of notebooks/Chapter1_Introduction--10.md:

```
\begin{align}
P(X ) & = P(X \text{ and } A) + P(X \text{ and } \sim A) \\\\[5pt]
 & = P(X|A)P(A) + P(X | \sim A)P(\sim A)\\\\[5pt]
& = P(X|A)p + P(X | \sim A)(1-p)
\end{align}
```

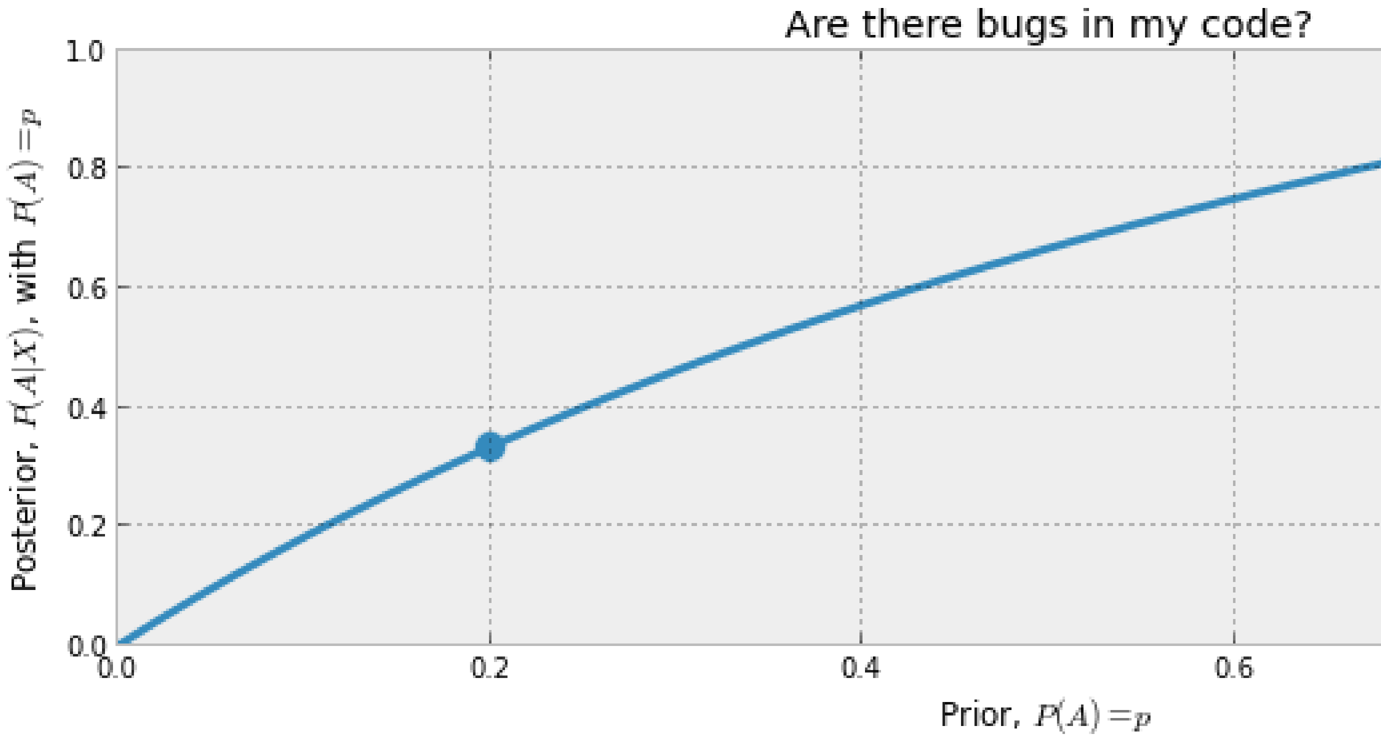
Here are the contents of notebooks/Chapter1_Introduction--11.md:

```
We have already computed $P(X|A)$ above. On the other hand, $P(X | \sim A)$ is subjectiv

\begin{align}
P(A | X) & = \frac{1\cdot p}{ 1\cdot p +0.5 (1-p) } \\\\
& = \frac{ 2 p}{1+p}
\end{align}
This is the posterior probability. What does it look like as a function of our prior, $p
```

Here are the contents of notebooks/Chapter1_Introduction--12-input.py:

```
figsize(12.5,4)
p = np.linspace( 0,1, 50)
plt.plot( p, 2*p/(1+p), color = "#348ABD", lw = 3 )
#plt.fill_between( p, 2*p/(1+p), alpha = .5, facecolor = ["#A60628"])
plt.scatter( 0.2, 2*(0.2)/1.2, s = 140, c ="#348ABD"   )
plt.xlim( 0, 1)
plt.ylim( 0, 1)
plt.xlabel( "Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title( "Are there bugs in my code?");
```

Here are the contents of notebooks/Chapter1_Introduction--12-output-0.png:



Here are the contents of notebooks/Chapter1_Introduction--13.md:

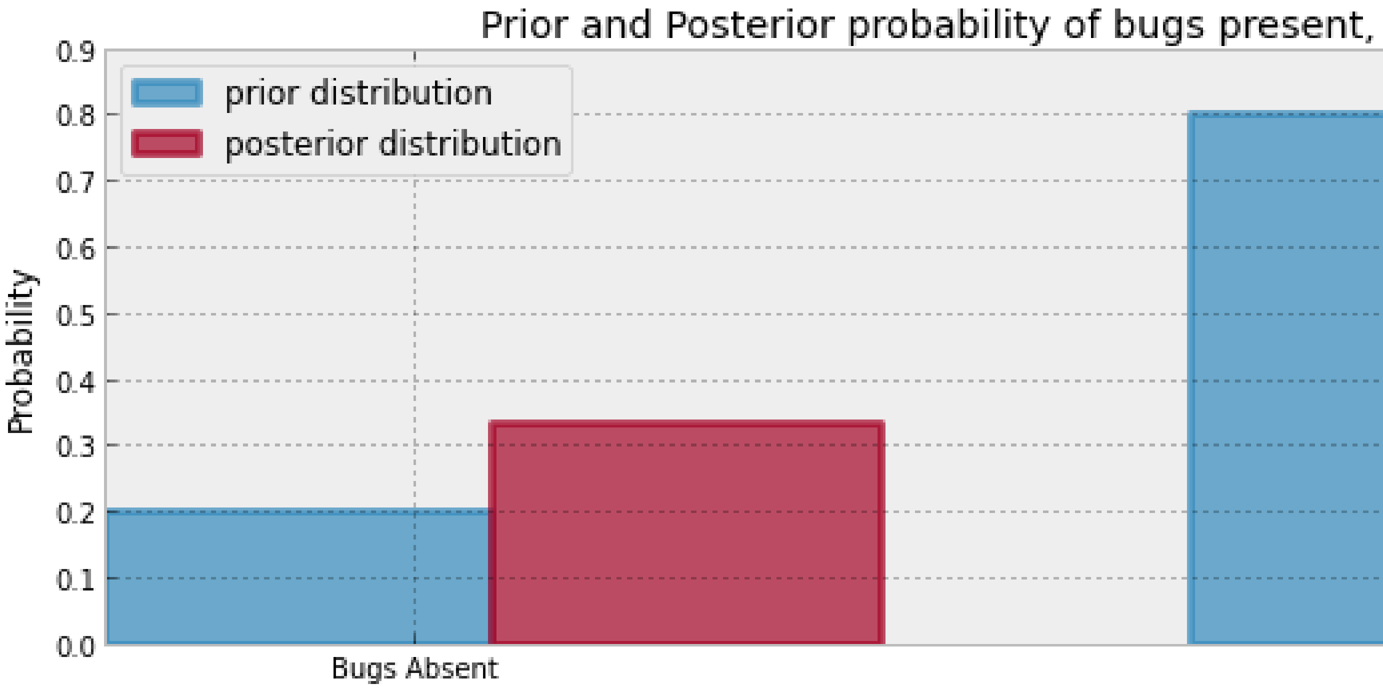
```
We can see the biggest gains if we observe the $X$ tests passed when the prior probabili

Recall that the prior is a probability: $p$ is the prior probability that there *are no

Similarly, our posterior is also a probability, with $P(A | X)$ the probability there is
```

Here are the contents of notebooks/Chapter1_Introduction--14-input.py:

```
figsize( 12.5, 4 )
colours = ["#348ABD", "#A60628"]

prior = [0.20, 0.80]
posterior =  [1./3, 2./3]
plt.bar( [0,.7], prior ,alpha = 0.70, width = 0.25, \
            color = colours[0], label = "prior distribution",
             lw = "3", edgecolor = colours[0])


plt.bar( [0+0.25,.7+0.25], posterior ,alpha = 0.7, \
          width = 0.25, color =  colours[1],
          label = "posterior distribution",
          lw = "3", edgecolor = colours[1])
```

```
plt.xticks( [0.20,.95], ["Bugs Absent", "Bugs Present"] )
plt.title("Prior and Posterior probability of bugs present, prior = 0.2")
plt.ylabel("Probability")
plt.legend(loc="upper left");
```

Here are the contents of `notebooks/Chapter1_Introduction--14-output-0.png`:



Here are the contents of `notebooks/Chapter1_Introduction--15.md`:

```
Notice that after we observed $X$ occur, the probability of bugs being absent increased.

This was a very simple example of Bayesian inference and Bayes rule. Unfortunately, the
```

Here are the contents of `notebooks/Chapter1_Introduction--16.md`:

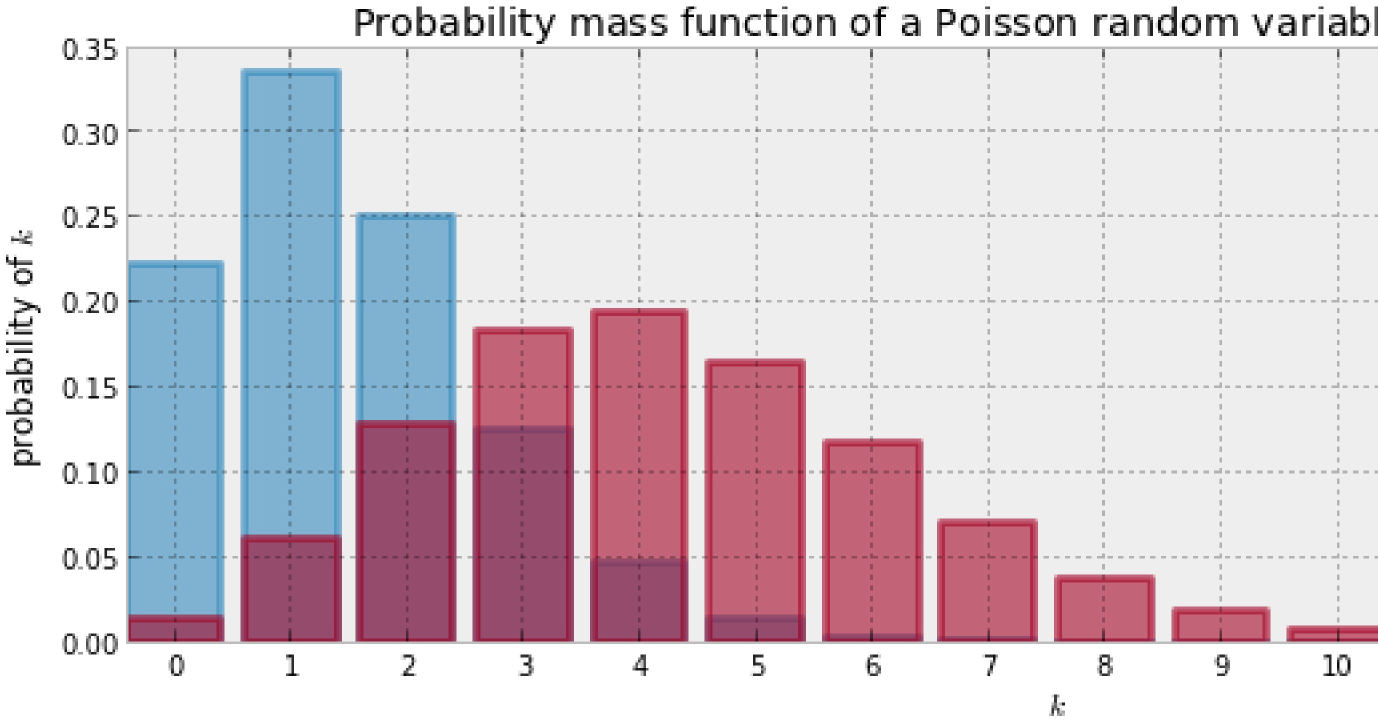
```
-------

##Probability Distributions


**Let's quickly recall what a probability distribution is:** Let $Z$ be some random vari

We can divide random variables into three classifications:

-   **$Z$ is discrete**: Discrete random variables may only assume values on a specified

-   **$Z$ is continuous**: Continuous random variable can take on arbitrarily exact valu
```

```
- **$Z$ is mixed**: Mixed random variables assign probabilities to both discrete and con

###Discrete Case
If $Z$ is discrete, then its distribution is called a *probability mass function*, which

$$P(Z = k) =\frac{ \lambda^k e^{-\lambda} }{k!}, \; \; k=0,1,2, \dots $$

What is $\lambda$? It is called the parameter, and it describes the shape of the distrib

Unlike $\lambda$, which can be any positive number, the value $k$ in the above formula m

If a random variable $Z$ has a Poisson mass distribution, we denote this by writing

$$Z \sim \text{Poi}(\lambda) $$

One very useful property of the Poisson random variable, given we know $\lambda$, is tha

$$E\large[ \;Z\; | \; \lambda \;\large] = \lambda $$

We will use this property often, so it's something useful to remember. Below we plot the
```

Here are the contents of `notebooks/Chapter1_Introduction--17-input.py`:

```python
figsize( 12.5, 4)

import scipy.stats as stats
a = np.arange( 16 )
poi = stats.poisson
lambda_ = [1.5, 4.25 ]

plt.bar( a, poi.pmf( a, lambda_[0]), color=colours[0],
        label = "$\lambda = %.1f$"%lambda_[0], alpha = 0.60,
        edgecolor = colours[0], lw = "3")

plt.bar( a, poi.pmf( a, lambda_[1]), color=colours[1],
         label = "$\lambda = %.1f$"%lambda_[1], alpha = 0.60,
          edgecolor = colours[1], lw = "3")

plt.xticks( a + 0.4, a )
plt.legend()
plt.ylabel("probability of $k$")
plt.xlabel("$k$")
plt.title("Probability mass function of a Poisson random variable; differing \
$\lambda$ values");
```

Here are the contents of notebooks/Chapter1_Introduction--17-output-0.png:



Here are the contents of notebooks/Chapter1_Introduction--18.md:

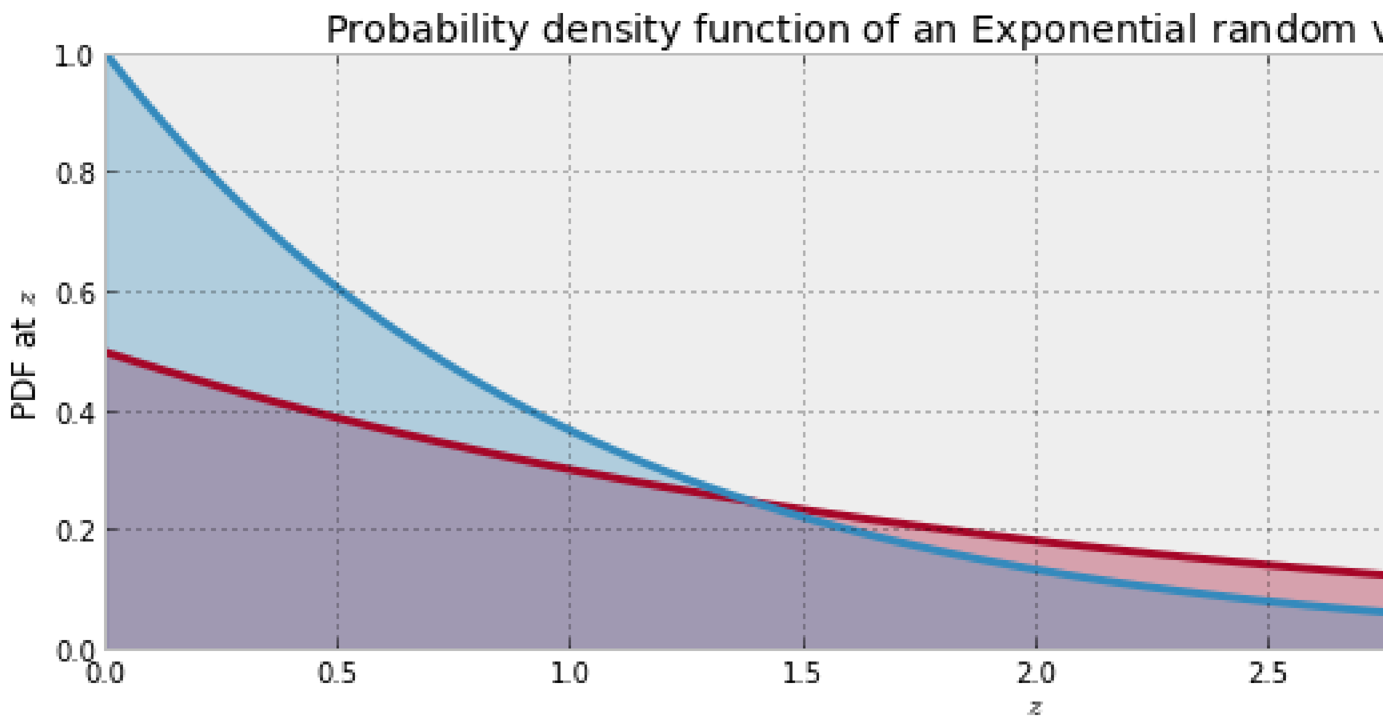
```
###Continuous Case
Instead of a probability mass function, a continuous random variable has a *probability

$$f_Z(z | \lambda) = \lambda e^{-\lambda z }, \;\; z\ge 0$$

Like the Poisson random variable, an exponential random variable can only take on non-ne

When a random variable $Z$ has an exponential distribution with parameter $\lambda$, we

$$Z \sim \text{Exp}(\lambda)$$

Given a specific $\lambda$, the expected value of an exponential random variable is equa

$$E[\; Z \;|\; \lambda \;] = \frac{1}{\lambda}$$
```

Here are the contents of notebooks/Chapter1_Introduction--19-input.py:

```
a = np.linspace(0,4, 100)
expo = stats.expon
lambda_ = [0.5, 1]

for l,c in zip(lambda_,colours):
```

```
    plt.plot( a, expo.pdf( a, scale=1./l), lw=3,
                color=c, label = "$\lambda = %.1f$"%l)
    plt.fill_between( a, expo.pdf( a, scale=1./l), color=c, alpha = .33)

plt.legend()
plt.ylabel("PDF at $z$")
plt.xlabel("$z$")
plt.title("Probability density function of an Exponential random variable;\
 differing $\lambda$");
```

Here are the contents of `notebooks/Chapter1_Introduction--19-output-0.png`:



Here are the contents of `notebooks/Chapter1_Introduction--20.md`:

```
###But what is $\lambda \;$?


**This question is what motivates statistics**. In the real world, $\lambda$ is hidden f

Bayesian inference is concerned with *beliefs* about what $\lambda$ is. Rather than try

This might seem odd at first: after all, $\lambda$ is fixed, it is not (necessarily) ran
```

Here are the contents of `notebooks/Chapter1_Introduction--21.md`:

```
##### Example: Inferring behaviour from text-message data
```

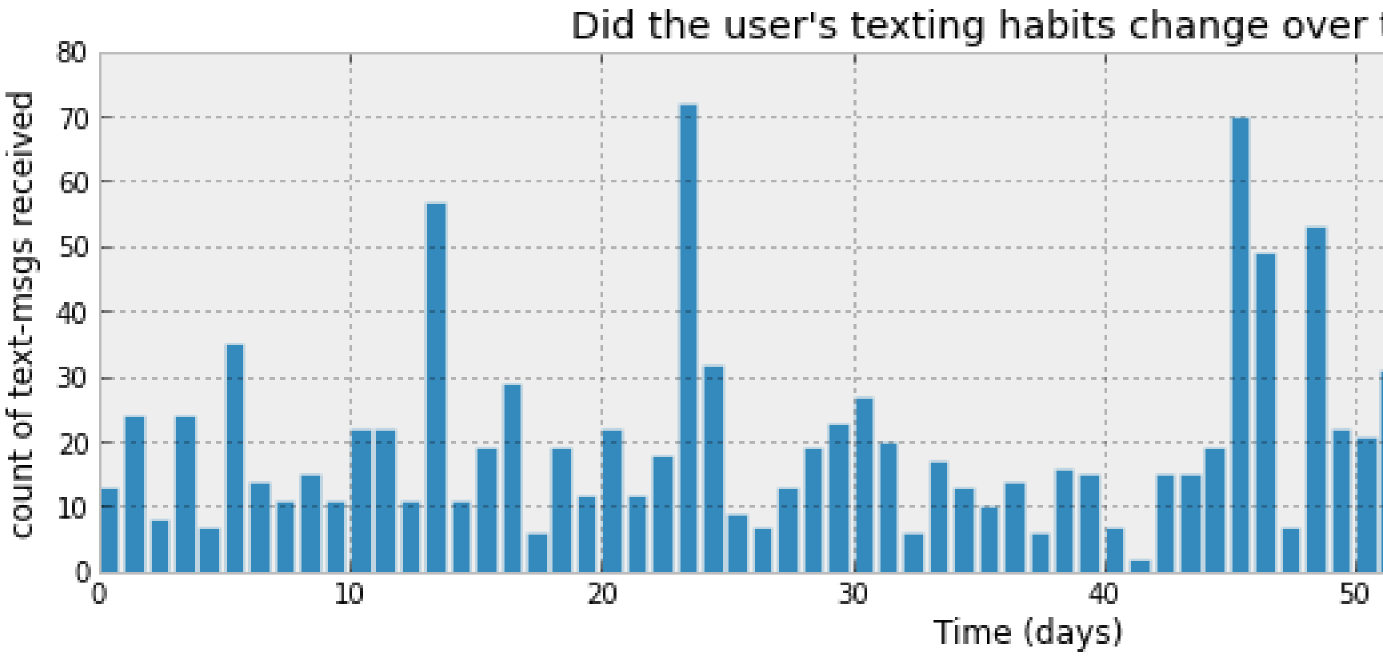
Let's try to model a more interesting example, concerning text-message rates:

> You are given a series of text-message counts from a user of your system. The data, p

Here are the contents of `notebooks/Chapter1_Introduction--22-input.py`:

```
figsize( 12.5, 3.5 )
count_data = np.loadtxt("data/txtdata.csv")
n_count_data = len(count_data)
plt.bar( np.arange( n_count_data ), count_data, color ="#348ABD"  )
plt.xlabel( "Time (days)")
plt.ylabel("count of text-msgs received")
plt.title("Did the user's texting habits change over time?")
plt.xlim( 0, n_count_data );
```

Here are the contents of `notebooks/Chapter1_Introduction--22-output-0.png`:



Here are the contents of `notebooks/Chapter1_Introduction--23.md`:

Before we begin, with respect to the plot above, would you say there was a change in beh
during the time period?

How can we start to model this? Well, as I conveniently already introduced, a Poisson ra

$$ C_i \sim \text{Poisson}(\lambda) $$

We are not sure about what the λ parameter is though. Looking at the chart above

How can we mathematically represent this? We can think, that at some later date (call it

```
$$
\lambda =
\begin{cases}
\lambda_1  & \text{if } t \lt \tau \cr
\lambda_2 & \text{if } t \ge \tau
\end{cases}
$$


 If, in reality, no sudden change occurred and indeed $\lambda_1 = \lambda_2$, the $\lam

We are interested in inferring the unknown $\lambda$s. To use Bayesian inference, we nee

\begin{align}
&\lambda_1 \sim \text{Exp}( \alpha ) \\\
&\lambda_2 \sim \text{Exp}( \alpha )
\end{align}

$\alpha$ is called a *hyper-parameter*, or a *parent-variable*, literally a parameter th

$$\frac{1}{N}\sum_{i=0}^N \;C_i \approx E[\; \lambda \; |\; \alpha ] = \frac{1}{\alpha}$

Alternatively, and something I encourage the reader to try, is to have two priors: one f

What about $\tau$? Well, due to the randomness, it is too difficult to pick out when $\t

\begin{align}
& \tau \sim \text{DiscreteUniform(1,70) }\\\\
& \Rightarrow P( \tau = k ) = \frac{1}{70}
\end{align}

So after all this, what does our overall prior for the unknown variables look like? Fran


Introducing our first hammer: PyMC
-----

PyMC is a Python library for programming Bayesian analysis [3]. It is a fast, well-maint

We will model the above problem using the PyMC library. This type of programming is call

B. Cronin [5] has a very motivating description of probabilistic programming:

>   Another way of thinking about this: unlike a traditional program, which only runs in
```

```
Due to its poorly understood title, I'll refrain from using the name *probabilistic prog
```

```
The PyMC code is easy to follow along: the only novel thing should be the syntax, and I
```

Here are the contents of `notebooks/Chapter1_Introduction--24-input.py`:

```
import pymc as mc



alpha = 1.0/count_data.mean() #recall count_data is
                              #the variable that holds our txt counts

lambda_1 = mc.Exponential( "lambda_1",  alpha )
lambda_2 = mc.Exponential( "lambda_2", alpha )

tau = mc.DiscreteUniform( "tau", lower = 0, upper = n_count_data )
```

Here are the contents of `notebooks/Chapter1_Introduction--25.md`:

```
In the above code, we create the PyMC variables corresponding to $\lambda_1, \; \lambda_
```

Here are the contents of `notebooks/Chapter1_Introduction--26-input.py`:

```
print "Random output:", tau.random(),tau.random(), tau.random()
```

Here are the contents of `notebooks/Chapter1_Introduction--26-output-0.txt`:

```
Random output: 58 31 17
```

Here are the contents of `notebooks/Chapter1_Introduction--27-input.py`:

```
@mc.deterministic
def lambda_( tau = tau, lambda_1 = lambda_1, lambda_2 = lambda_2 ):
    out = np.zeros( n_count_data )
    out[:tau] = lambda_1 #lambda before tau is lambda1
    out[tau:] = lambda_2 #lambda after tau is lambda2
    return out
```

Here are the contents of `notebooks/Chapter1_Introduction--28.md`:

```
This code is creating a new function `lambda_`, but really we think of it as a random va
```

Here are the contents of `notebooks/Chapter1_Introduction--29-input.py`:

```
observation = mc.Poisson( "obs", lambda_, value = count_data, observed = True)

model = mc.Model( [observation, lambda_1, lambda_2, tau] )
```

Here are the contents of `notebooks/Chapter1_Introduction--30.md`:

```
The variable `observation` combines our data, `count_data`, with our proposed data-gener

The below code will be explained in the Chapter 3, but this is where our results come fr
```

Here are the contents of `notebooks/Chapter1_Introduction--31-input.py`:

```
### Mysterious code to be explained in Chapter 3.
mcmc = mc.MCMC(model)
mcmc.sample( 40000, 10000, 1 )
```

Here are the contents of `notebooks/Chapter1_Introduction--31-output-0.txt`:

```
[****************100%******************]  40000 of 40000 complete
```

Here are the contents of `notebooks/Chapter1_Introduction--31-output-1.txt`:

Here are the contents of `notebooks/Chapter1_Introduction--32-input.py`:

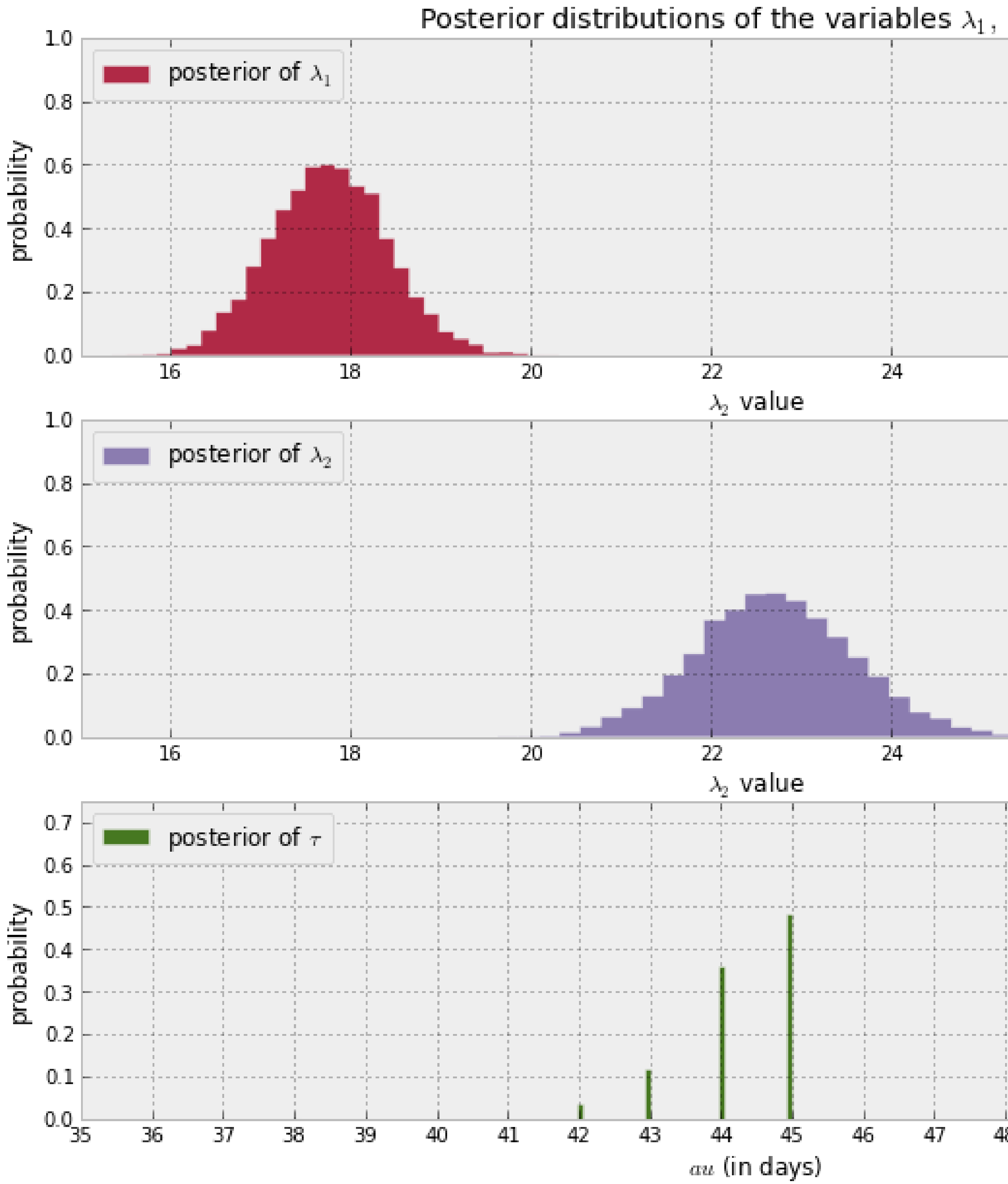
```
lambda_1_samples = mcmc.trace( 'lambda_1' )[:]
lambda_2_samples = mcmc.trace( 'lambda_2' )[:]
tau_samples = mcmc.trace( 'tau' )[:]
```

Here are the contents of `notebooks/Chapter1_Introduction--33-input.py`:

```
figsize(12.5, 10)
#histogram of the samples:

ax = plt.subplot(311)
ax.set_autoscaley_on(False)

plt.hist( lambda_1_samples, histtype='stepfilled', bins = 30, alpha = 0.85,
        label = "posterior of $\lambda_1$", color = "#A60628",normed = True )
plt.legend(loc = "upper left")
plt.title(r"Posterior distributions of the variables $\lambda_1,\;\lambda_2,\;\tau$")
plt.xlim([15,30])
plt.xlabel("$\lambda_2$ value")
plt.ylabel("probability")
```

```
ax = plt.subplot(312)
ax.set_autoscaley_on(False)

plt.hist( lambda_2_samples,histtype='stepfilled', bins = 30, alpha = 0.85,
          label = "posterior of $\lambda_2$",color="#7A68A6", normed = True )
plt.legend(loc = "upper left")
plt.xlim([15,30])
plt.xlabel("$\lambda_2$ value")
plt.ylabel("probability")

plt.subplot(313)


w = 1.0/ tau_samples.shape[0] * np.ones_like( tau_samples )
plt.hist( tau_samples, bins = n_count_data, alpha = 1,
         label = r"posterior of $\tau$",
         color="#467821", weights=w, rwidth =2. )
plt.xticks( np.arange( n_count_data ) )

plt.legend(loc = "upper left");
plt.ylim([0,.75])
plt.xlim([35, len(count_data)-20])
plt.xlabel("$\tau$ (in days)")
plt.ylabel("probability");
```

Here are the contents of `notebooks/Chapter1_Introduction--33-output-0.png`:



Here are the contents of `notebooks/Chapter1_Introduction--34.md`:

```
### Interpretation

Recall that the Bayesian methodology returns a *distribution*, hence we now have distrib

Also notice that the posterior distributions for the $\lambda$'s do not look like any ex

Our analysis also returned a distribution for what $\tau$ might be. Its posterior distri
```

Here are the contents of `notebooks/Chapter1_Introduction--35.md`:

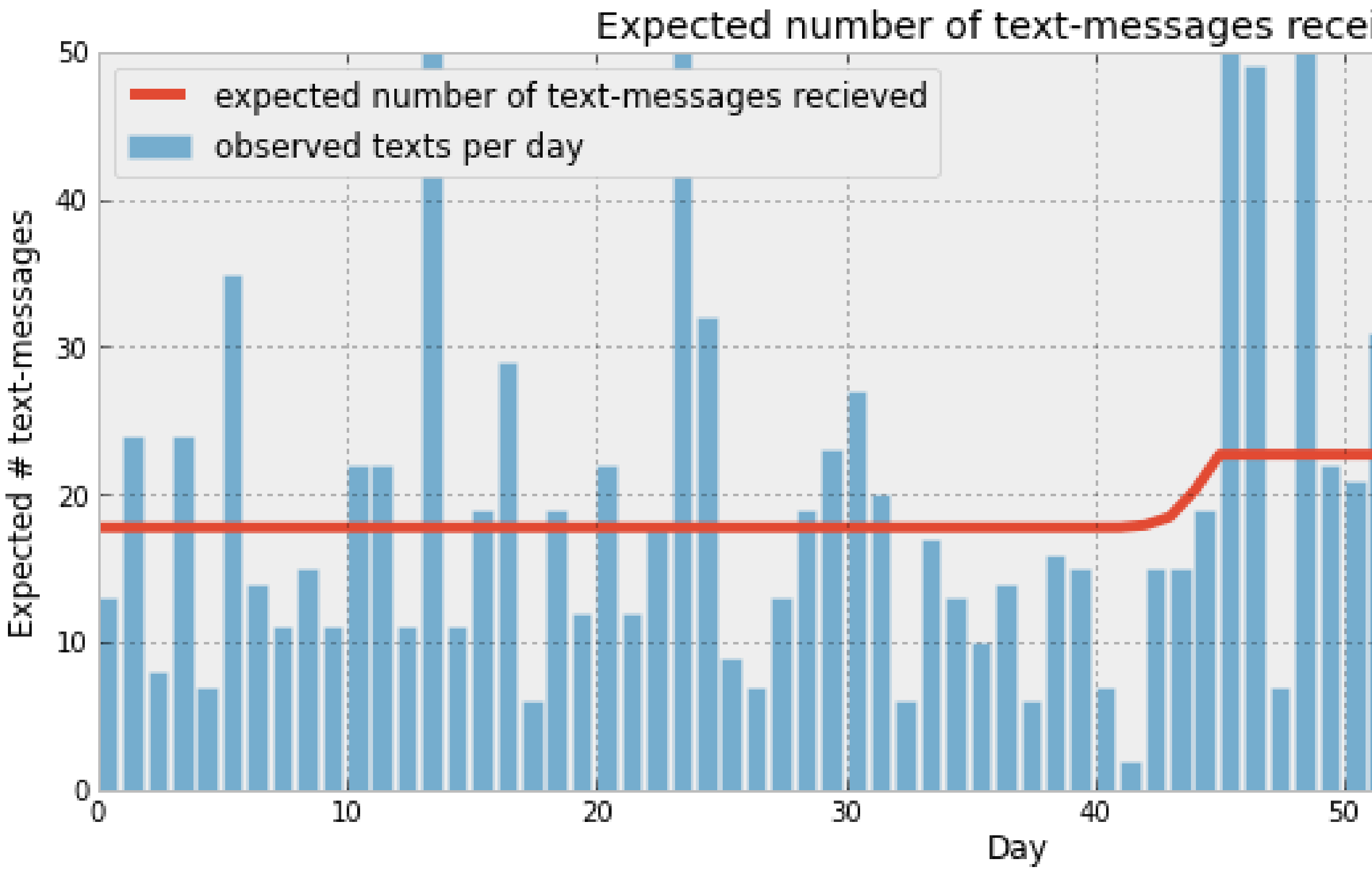
```
###Why would I want samples from the posterior, anyways?


We will deal with this question for the remainder of the book, and it is an understateme

In the code below, we are calculating the following: Let $i$ index samples from the post
```

Here are the contents of `notebooks/Chapter1_Introduction--36-input.py`:

```
figsize( 12.5, 5)
# tau_samples, lambda_1_samples, lambda_2_samples contain
# N samples from the corresponding posterior distribution
N = tau_samples.shape[0]
expected_texts_per_day = np.zeros(n_count_data)
for  day in range(0, n_count_data):
    # ix is a bool index of all tau samples corresponding to
    # the switchpoint occurring prior to value of 'day'
    ix = day < tau_samples
    # Each posterior sample corresponds to a value for tau.
    # for each day, that value of tau indicates whether we're "before"
    # (in the lambda1 "regime") or
    #  "after" (in the lambda2 "regime") the switchpoint.
    # by taking the posterior sample of lambda1/2 accordingly, we can average
    # over all samples to get an expected value for lambda on that day.
    # As explained, the "message count" random variable is Poisson distributed,
    # and therefore lambda (the poisson parameter) is the expected value of "message cou
    expected_texts_per_day[day] = (lambda_1_samples[ix].sum()
                                   + lambda_2_samples[~ix].sum() ) /N


plt.plot( range( n_count_data), expected_texts_per_day, lw =4, color = "#E24A33",
         label = "expected number of text-messages recieved")
plt.xlim( 0, n_count_data )
plt.xlabel( "Day" )
plt.ylabel( "Expected # text-messages" )
```

```
plt.title( "Expected number of text-messages received")
plt.ylim( 0, 50 )
plt.bar( np.arange( len(count_data) ), count_data, color ="#348ABD", alpha = 0.65,
            label="observed texts per day")

plt.legend(loc="upper left");
```

Here are the contents of `notebooks/Chapter1_Introduction--36-output-0.png`:



Here are the contents of `notebooks/Chapter1_Introduction--37.md`:

```
Our analysis shows strong support for believing the user's behavior did change ($\lambda
```

Here are the contents of `notebooks/Chapter1_Introduction--38.md`:

```
##### Exercises

1\.  Using `lambda_1_samples` and `lambda_2_samples`, what is the mean of the posterior
```

Here are the contents of `notebooks/Chapter1_Introduction--39-input.py`:

```
#type your code here.
```

Here are the contents of `notebooks/Chapter1_Introduction--40.md`:

```
2\.  What is the expected percentage increase in text-message rates? `hint:` compute the
```

Here are the contents of `notebooks/Chapter1_Introduction--41-input.py`:

```
#type your code here.
```

Here are the contents of `notebooks/Chapter1_Introduction--42.md`:

```
3\. What is the mean of $\lambda_1$ **given** we know $\tau$ is less than 45.  That is,
```

Here are the contents of `notebooks/Chapter1_Introduction--43-input.py`:

```
#type your code here.
```

Here are the contents of `notebooks/Chapter1_Introduction--44.md`:

```
### References


-  [1] Gelman, Andrew. N.p.. Web. 22 Jan 2013. <http://andrewgelman.com/2005/07/n_is_nev
-  [2] Norvig, Peter. 2009. [*The Unreasonable Effectiveness of Data*](http://www.csee.w
- [3] Patil, A., D. Huard and C.J. Fonnesbeck. 2010.
PyMC: Bayesian Stochastic Modelling in Python. Journal of Statistical
Software, 35(4), pp. 1-81.
- [4] Jimmy Lin and Alek Kolcz. Large-Scale Machine Learning at Twitter. Proceedings of
- [5] Cronin, Beau. "Why Probabilistic Programming Matters." 24 Mar 2013. Google, Online
```

Here are the contents of `notebooks/Chapter1_Introduction--45-input.py`:

```
from IPython.core.display import HTML
def css_styling():
    styles = open("../styles/custom.css", "r").read()
    return HTML(styles)
css_styling()
```

Here are the contents of `notebooks/Chapter1_Introduction--46-input.py`: