```
begin
    import Pkg
    # activate the shared project environment
    Pkg.activate(Base.current_project())
    using Omega, Distributions, UnicodePlots, OmegaExamples
end
```

```
  Activating project at `~/Documents/GitHub/Omega.jl/OmegaExamples`
```

Inference by conditioning a generative model is a basic building block of Bayesian statistics. In cognitive science this tool can be used in two ways. If the generative model is a hypothesis about a person's model of the world, then we have a Bayesian *cognitive model*. If the generative model is instead the scientist's model of how the data are generated, then we have *Bayesian data analysis*. Bayesian data analysis can be an extremely useful tool to us as scientists, when we are trying to understand what our data mean about psychological hypotheses. This can become confusing: a particular modeling assumption can be something we hypothesize that people assume about the world, or can be something that we as scientists want to assume (but don't assume that people assume). A pithy way of saying this is that we can make assumptions about "Bayes in the head" (Bayesian cognitive models) or about "Bayes in the notebook" (Bayesian data analysis).

Bayesian data analysis (BDA) is a general-purpose approach to making sense of data. A BDA model is an explicit hypotheses about the generative process behind the experimental data – where did the observed data come from? For instance, the hypothesis that data from two experimental conditions came from two *different* distributions. After making explicit hypotheses, Bayesian inference can be used to invert the model: go from experimental data to updated beliefs about the hypotheses.

Parameters and predictives

In a BDA model the random variables are usually called parameters. Parameters can be of theoretical interest, or not (the latter are called nuisance parameters). Parameters are in general unobservable (or, "latent"), so we must infer them from observed data. We can also go from updated beliefs about parameters to expectations about future data, so call *posterior predictives*.

For a given Bayesian model (together with data), there are four conceptually distinct distributions we often want to examine. For parameters, we have priors and posteriors:

- The *prior distribution* over parameters captures our initial state of knowledge (or beliefs) about the values that the latent parameters could have, before seeing the data.
- The *posterior distribution* over parameters captures what we know about the latent parameters having updated our beliefs with the evidence provided by data.

From either the prior or the posterior over parameters we can then run the model forward, to get predictions about data sets:

- The prior predictive distribution tells us what data to expect, given our model and our initial beliefs about the parameters. The prior predictive is a distribution over data, and gives the relative probability of different observable outcomes before we have seen any data.
- The posterior predictive distribution tells us what data to expect, given the same model we started with, but with beliefs that have been updated by the observed data. The posterior predictive is a distribution over data, and gives the relative probability of different observable outcomes, after some data has been seen.

Loosely speaking, *predictive* distributions are in "data space" and *parameter* distributions are in "latent parameter space".

Example: Election surveys

Imagine you want to find out how likely Candidate A is to win an election. To do this, you will try to estimate the proportion of eligible voters in the United States who will vote for Candidate A in the election. Trying to determine directly how many (voting age, likely to vote) people prefer Candidate A vs. Candidate B would require asking over 100 million people (it's estimated that about 130 million people voted in the US Presidential Elections in 2008 and 2012). It's impractical to measure the whole distribution. Instead, pollsters measure a sample (maybe ask 1000 people), and use that to draw conclusions about the "true population proportion" (an unobservable parameter).

Here, we explore the result of an experiment with 20 trials and binary outcomes ("will you vote for Candidate A or Candidate B?").

20

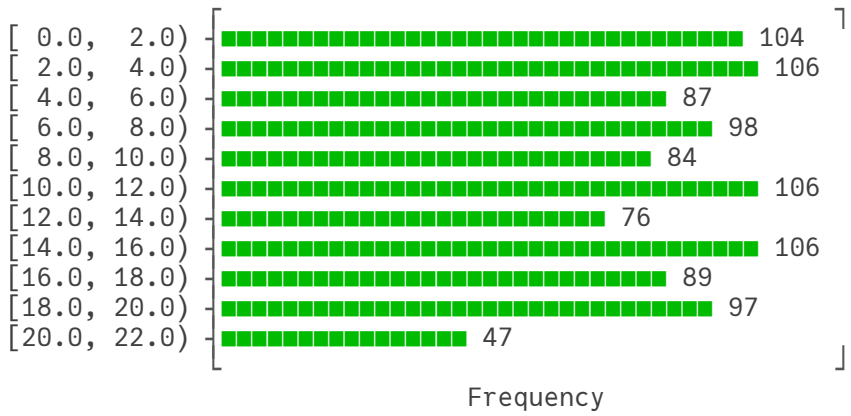
```
begin
    ## observed data
    k = 1  # number of people who support candidate A
    n = 20 # number of people asked
end
```

```
p = 840479291959638606 8@Distributions.Uniform{Float64}(a=0.0, b=1.0)
```

```
p = @~ Uniform(0, 1) # true population proportion who support candidate A
```

```
prior_predictive = 15305184762131525 71@#1
```

```
# recreate model structure, without conditioning
prior_predictive = @~ Binomial(n, p)
```



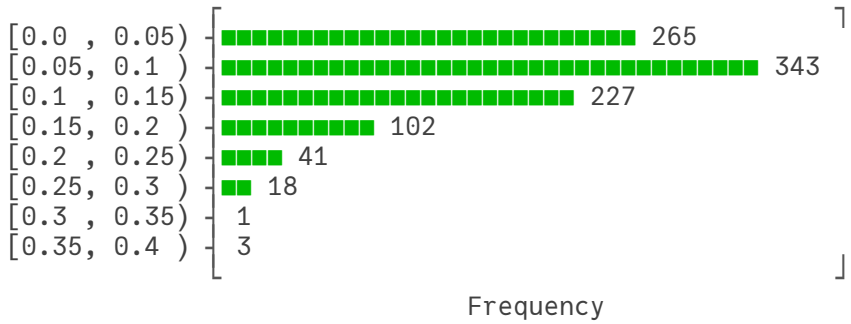

```
viz(randsample(prior_predictive, 1000))
```

```
posterior_predictive =
  Conditional(840479291959638606 8@Distributions.Uniform{Float64}(a=0.0, b=1.0), ==ₚ(ᵛ#7, 1))
```

```
# Observed k people support "A"
# Assuming each person's response is independent of each other
posterior_predictive = p |ᶜ (Variable(ω -> (@~ Binomial(n, p(ω)))(ω)) .== k)
```

```
posterior_predictive_samples =
  [0.0205908, 0.109047, 0.0758339, 0.107493, 0.0864416, 0.00591885, 0.058801, 0.115155, 0.012
```

```
posterior_predictive_samples = randsample(posterior_predictive, 1000)
```




```
viz(posterior_predictive_samples)
```

What can we conclude intuitively from examining these plots? First, because prior differs from posterior, the evidence has changed our beliefs. Second, the posterior predictive assigns quite high probability to the true data, suggesting that the model considers the data “reasonable”. Finally, after

observing the data, we conclude the true proportion of people supporting Candidate A is quite low – around 0.09, or anyhow somewhere between 0.0 and 0.15. Check your understanding by trying other data sets, varying both `k` and `n`.

Quantifying claims about parameters

How can we quantify a claim like "the true parameter is low"? One possibility is to compute the mean or expected value of the parameter, which is mathematically given by $\int x \cdot p(x) dx$ for a posterior distribution $p(x)$. Thus in the above election example we could:

```
0.09302382479376455
```

```
1 mean(posterior_predictive_samples)
```

This tells us that the mean is about 0.09. This can be a very useful way to summarize a posterior, but it eliminates crucial information about how *confident* we are in this mean. A coherent way to summarize our confidence is by exploring the probability that the parameter lies within a given interval. Conversely, an interval that the parameter lies in with high probability (say 90%) is called a *credible interval* (CI). Let's explore credible intervals for the parameter in the above model:

```
0.886
```

```
1 mean(0.01 .< posterior_predictive_samples .< 0.18)
```

Here we see that $[0.01, 0.18]$ is an (approximately) 90% credible interval – we can be about 90% sure that the true parameter lies within this interval. Notice that the 90% CI is not unique. There are different ways to choose a particular CI. One particularly common, and useful, one is the Highest Density Interval (HDI), which is the smallest interval achieving a given confidence. (For unimodal distributions the HDI is unique and includes the mean.)

Here is a quick way to approximate the HDI of a distribution:

```
cred (generic function with 1 method)
```

```
1 cred(d, low, up) = mean(low .< d .< up)
```

```
find_HDI (generic function with 1 method)
```

```
1 function find_HDI(targetp, d, low, up, eps)
2     if cred(d, low, up) < targetp
3         return [low, up]
4     end
5     y = cred(d, low + eps, up)
6     z = cred(d, low, up - eps)
7     return (y > z) ?
8     find_HDI(targetp,d,low+eps,up,eps) : find_HDI(targetp,d,low,up-eps,eps)
9 end
```

To test the function -

```
x = -5088485090087577783@Distributions.Normal{Float64}(μ=0.0, σ=1.0)
```

```
1 x = @~ Normal(0, 1)
```

```
samples =
 [1.55628, 2.42732, 0.549894, 1.13216, 0.655445, 0.0146558, 0.678079, 0.263732, 1.54772, 0.69
```

```
samples = randsample(x |ᶜ (x .> 0), 1000)
```

```
[-1.87905e-14, 1.8]
```

```
find_HDI(0.95, samples, -10, 10, 0.1)
```

Credible intervals are related to, but shouldn't be mistaken for the confidence intervals that are often used in frequentist statistics. (And these confidence intervals themselves should definitely not be confused with p-values....)

Example: logistic regression

Now imagine you are a pollster who is interested in the effect of age on voting tendency. You run the same poll, but you are careful to recruit people in their 20's, 30's, etc. We can analyze this data by assuming there is an underlying linear relationship between age and voting tendency. However since our data are Boolean (or if we aggregate, the count of people in each age group who will vote for candidate A), we must use a function to link from real numbers to the range $[0, 1]$. The logistic function is a common way to do so.

```
data =
 [(age = 20, n = 20, k = 1), (age = 30, n = 20, k = 5), (age = 40, n = 20, k = 17), (age = 50, n
```

```
data = [(age = 20, n = 20, k = 1),
    (age = 30, n = 20, k = 5),
    (age = 40, n = 20, k = 17),
    (age = 50, n = 20, k = 18)
]
```

```
ages =  [20, 30, 40, 50]
```

```
ages = map(d -> d.age, data)
```

```
logistic (generic function with 1 method)
```

```
logistic(x) = 1 / (1 + exp(-x))
```

```
a = 5579628292196151612@StdNormal{Float64}()
```

```
a = @~ StdNormal{Float64}() # true effect of age
```

```
b = -3237263054598830 39@StdNormal{Float64}()
```

```
b = @~ StdNormal{Float64}() # true intercept
```

```
bin (generic function with 1 method)
```

```
bin(ω, n, age, a, b, i) = (i ~ Binomial(n, logistic(a(ω) * age + b(ω))))(ω)
```

```
evidence = &ₚ(==ₚ(ᵛ#11, 1), ==ₚ(ᵛ#12, 5), ==ₚ(ᵛ#13, 17), ==ₚ(ᵛ#14, 18))
```

```
evidence = pw(&,
    (Variable(ω -> bin(ω, data[1].n, data[1].age, a, b, (@uid, 1))) .== data[1].k),
    (Variable(ω -> bin(ω, data[2].n, data[2].age, a, b, (@uid, 2))) .== data[2].k),
    (Variable(ω -> bin(ω, data[3].n, data[3].age, a, b, (@uid, 3))) .== data[3].k),
    (Variable(ω -> bin(ω, data[4].n, data[4].age, a, b, (@uid, 4))) .== data[4].k)
) # map gives the same `Variable`s for `bin`
```

```
false ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 1 000
```

```
viz(randsample(evidence, 1000))
```

```
a_posterior =
  Conditional(5579628292196151612@StdNormal{Float64}(), &ₚ(==ₚ(ᵛ#11, 1), ==ₚ(ᵛ#12, 5), ==ₚ(ᵛ
```

```
a_posterior = a |ᶜ evidence
```

```
b_posterior =
  Conditional(-3237263054598830390@StdNormal{Float64}(), &ₚ(==ₚ(ᵛ#11, 1), ==ₚ(ᵛ#12, 5), ==ₚ(ᵛ
```

```
b_posterior = b |ᶜ evidence
```

```
posterior = #33 (generic function with 1 method)
```

```
posterior = @joint a_posterior b_posterior
```

```
viz_marginals(randsample(posterior, 1000, alg = MH))
```

```
:a_posterior

   [-0.35, -0.3 ) ┤■ 8
   [-0.3 , -0.25) ┤■■■ 15
   [-0.25, -0.2 ) ┤■■■ 16
   [-0.2 , -0.15) ┤■■■■■■■■■■■■■■■■■ 94
   [-0.15, -0.1 ) ┤■■■■■■■■■■■ 63
   [-0.1 , -0.05) ┤■■■■■■■■■■■■■■■■■ 91
   [-0.05, -0.0 ) ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 193
   [ 0.0 ,  0.05) ┤■■■■■■■■■■■■■■■■■■■■■ 116
   [ 0.05,  0.1 ) ┤■■■■■■■■■■■■■■■■■■ 100
   [ 0.1 ,  0.15) ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 193
   [ 0.15,  0.2 ) ┤■■■■■■■■■■■■■■■■■■■ 104
   [ 0.2 ,  0.25) ┤■ 7

                              Frequency
:b_posterior

   [-0.55, -0.5 ) ┤■■■■■■■■■■■■■■■ 83
   [-0.5 , -0.45) ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 170
   [-0.45, -0.4 ) ┤■■■■■■■■■■■■ 64
   [-0.4 , -0.35) ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 172
   [-0.35, -0.3 ) ┤■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ 192
   [-0.3 , -0.25) ┤■■■■■■■■■■■■■■■■■■■ 103
   [-0.25, -0.2 ) ┤■■■■■■■■■■■ 60
   [-0.2 , -0.15) ┤■■■■■■■■■■■■■■■■■■■■ 110
   [-0.15, -0.1 ) ┤■■■■■ 27
   [-0.1 , -0.05) ┤■■■ 19

                              Frequency
```

Looking at the parameter posteriors we see that there seems to be an effect of age: the parameter `a` is greater than zero, so older people are more likely to vote for candidate A. But how well does the model really explain the data?

Posterior prediction and model checking

The posterior predictive distribution describes what data you should expect to see, given the model you've assumed and the data you've collected so far. If the model is able to describe the data you've collected, then the model shouldn't be surprised if you got the same data by running the experiment

again. That is, the most likely data for your model after observing your data should be the data you observed. It is natural then to use the posterior predictive distribution to examine the descriptive adequacy of a model. If these predictions do not match the data already seen (i.e., the data used to arrive at the posterior distribution over parameters), the model is descriptively inadequate.

A common way to check whether the posterior predictive matches the data, imaginatively called a posterior predictive check, is to plot some statistics of your data vs the expectation of these statistics according to the predictive. Let's do this for the number of votes in each age group:

```
predictive =  [#21, #22, #23, #24]
```

```
predictive = [
    ω -> (bin(ω, 20, 20, a_posterior, b_posterior, (@uid, @uid, 1)))
    ω -> (bin(ω, 20, 30, a_posterior, b_posterior, (@uid, @uid, 2)))
    ω -> (bin(ω, 20, 40, a_posterior, b_posterior, (@uid, @uid, 3)))
    ω -> (bin(ω, 20, 50, a_posterior, b_posterior, (@uid, @uid, 4)))
]
```

```
posterior_samples =
 [[0, 0, 0, 0, 0, 0, 0, 0, 0,    more ,0], [0, 0, 0, 0, 0, 0, 0, 0, 0,    more ,0], [0, 0, 0, 0, 0,
```

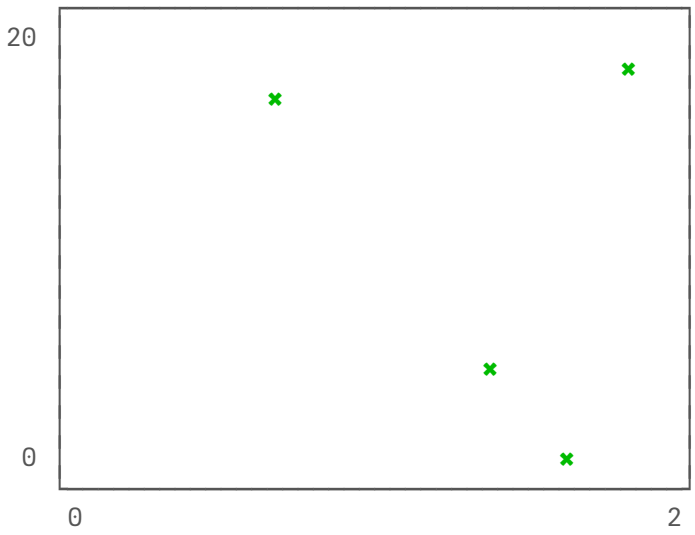
```
posterior_samples = randsample.(predictive, 1000, alg = MH)
```

```
ppstats =  [1.6, 1.38, 0.68, 1.84]
```

```
ppstats = mean.(posterior_samples)
```

```
datastats =  [1, 5, 17, 18]
```

```
datastats = map(d -> d.k, data)
```




```
scatterplot(ppstats, datastats, marker = :xcross)
```

This scatter plot will lie on the $x = y$ line if the model completely accomodates the data. Looking closely at this plot, we see that there are a few differences between the data and the predictives. First, the predictions are compressed compared to the data. This is likely because the prior encourages moderate probabilities. Second, we see hints of non-linearity in the data that the model is not able to account for. This model is pretty good, but certainly not perfect, for explaining this data.

One final note: There is an important, and subtle, difference between a model's ability to *accomodate* some data and that model's ability to *predict* the data. This is roughly the difference between a posterior predictive check and a *prior* predictive check.

Model selection

In the above examples, we've had a single data-analysis model and used the experimental data to learn about the parameters of the models and the descriptive adequacy of the models. Often as scientists, we are in fortunate position of having multiple, distinct models in hand, and want to decide if one or another is a better description of the data. The problem of *model selection* given data is one of the most important and difficult tasks of BDA.

Returning to the simple election polling example above, imagine we begin with a (rather unhelpful) data analysis model that assumes each candidate is equally likely to win, that is `p=0.5`. We quickly notice, by looking at the posterior predictives, that this model doesn't accommodate the data well at all. We thus introduce the above model where `p = @~ Uniform(0,1)`. How can we quantitatively decide which model is better? One approach is to combine the models into an uber model that decides which approach to take:

20

```
begin
    # observed data
    k_ = 5 # number of people who support candidate A
    n_ = 20  # number of people asked
end
```

x_ (generic function with 1 method)

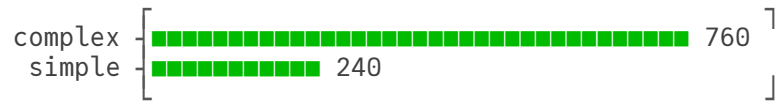
```
# binary decision variable for which hypothesis is better
x_(ω) = (@~ Bernoulli())(ω) ? "simple" : "complex"
```

p_ (generic function with 1 method)

```
p_(ω) = (x_(ω) == "simple") ? 0.5 : (@~ StdUniform{Float64}())(ω)
```

posterior_ = Conditional(x_ (generic function with 1 method), ==ₚ(ˣ#31, 5))

```
posterior_ = x_ |ᶜ (Variable(ω ->  (@~ Binomial(n_, p_(ω)))(ω)) .== k_)
```




```
viz(randsample(posterior_, 1000))
```

We see that, as expected, the more complex model is preferred: we can confidently say that given the data the more complex model is the one we should believe. Further we can quantify this via the posterior probability of the complex model.

This model is an example from the classical hypothesis testing framework. We consider a model that fixes one of its parameters to a pre-specified value of interest (here $H_0 : p = 0.5$). This is sometimes referred to as a *null hypothesis*. The other model says that the parameter is free to vary. In the classical

hypothesis testing framework, we would write: $H_1 : p \neq 0.5$. With Bayesian hypothesis testing, we must be explicit about what p is (not just what p is not), so we write $H_1 : p \sim Uniform(0, 1)$.

One might have a conceptual worry: Isn't the second model just a more general case of the first model? That is, if the second model has a uniform distribution over p, then $p : 0.5$ is included in the second model. Fortunately, the posterior on models automatically penalizes the more complex model when it's flexibility isn't needed. (To verify this, set `k=10` in the example.) This idea is called the principle of parsimony or *Occam's razor*, and will be discussed at length later. For now, it's sufficient to know that more complex models will be penalized for being more complex, intuitively because they will be diluting their predictions. At the same time, more complex models are more flexible and can capture a wider variety of data (they are able to bet on more horses, which increases the chance that they will win some money). Bayesian model comparison lets us weigh these costs and benefits.

Bayes' factor

What we are plotting above are *posterior model probabilities*. These are a function of the marginal likelihoods of the data under each hypothesis and the prior model probabilities (here, defined to be equal: `Bernoulli(0.5)`). Sometimes, scientists feel a bit strange about reporting values that are based on prior model probabilities (what if scientists have different priors as to the relative plausibility of the hypotheses?) and so often report the ratio of marginal likelihoods, a quantity known as a *Bayes Factor*.

Let's compute the Bayes' Factor, by computing the likelihood of the data under each hypothesis.

```
simple_model = -3098464983855197482@Distributions.Binomial{Float64}(n=20, p=0.5)
```

```
1 simple_model = @~ Binomial(n_, 0.5)
```

```
complex_model (generic function with 1 method)
```

```
1 complex_model(ω) = (@~ Binomial(n_, (@~ StdUniform{Float64}())(ω)))(ω)
```

```
1 # simple_likelihood = logpdf(simple_model, k_)
```

```
1 # complex_likelihood = logpdf(complex_model, k_)
```

```
1 # bayes_factor = simple_likelihood / complex_likelihood
```

Savage-Dickey method

Sometimes the Bayes factor can be obtained by computing marginal likelihoods directly. (As in the example) However, it is sometimes hard to get good estimates of the two marginal probabilities. In the case where one model is a special case of the other, called *nested model comparison*, there is another option. The Bayes factor can also be obtained by considering only the more complex hypothesis, by looking at the distribution over the parameter of interest (here, p) at the point of interest (here, $p = 0.5$). Dividing the probability density of the posterior by the density of the prior (of the parameter at the point of interest) gives you the Bayes Factor! This, perhaps surprising, result was described by Dickey and Lientz (1970), and they attribute it to Leonard "Jimmie" Savage. The method is

called the *Savage-Dickey density ratio* and is widely used in experimental science. We would use it like so:

```
complex_model_prior = -5462228074815098354@StdUniform{Float64}()
```

```
complex_model_prior = @~ StdUniform{Float64}()
```

```
complex_model_posterior (generic function with 1 method)
```

```
function complex_model_posterior(n, k)
    p = (@~ StdUniform{Float64}())
    p |ᶜ (Variable(ω -> (@~ Binomial(n, p(ω)))(ω)) .== k)
end
```

```
savage_dickey_denomenator = 0.109
```

```
savage_dickey_denomenator = mean(0.45 .< randsample(complex_model_prior, 1000) .< 0.55)
```

```
savage_dickey_numerator = 0.028
```

```
savage_dickey_numerator = mean(0.45 .< randsample(complex_model_posterior(n_, k_), 1000) .< 0.55)
```

```
savage_dickey_ratio = 0.25688073394495414
```

```
savage_dickey_ratio = savage_dickey_numerator/savage_dickey_denomenator
```

(Note that we have approximated the densities by looking at the expectation that p is within 0.05 of the target value $p = 0.5$.)

BDA of cognitive models

In this chapter we have described how we can use generative models of data to do data analysis. In the rest of this book we are largely interested in how we can build *cognitive models* by hypothesizing that people have generative models of the world that they use to reason and learn. That is, we view people as intuitive Bayesian statisticians, doing in their heads what scientists do in their notebooks.

Of course when we, as scientists, try to test our cognitive models of people, we can do so using BDA! This leads to more complex models in which we have an "outer" Bayesian data analysis model and an "inner" Bayesian cognitive model. What is in the inner (cognitive) model captures our scientific hypotheses about what people know and how they reason. What is in the outer (BDA) model represents aspects of our hypotheses about the data that we *are not* attributing to people: linking functions, unknown parameters of the cognitive model, and so on. This distinction can be subtle.