1. Introduction to Bayesian reasoning, computation and BUGS

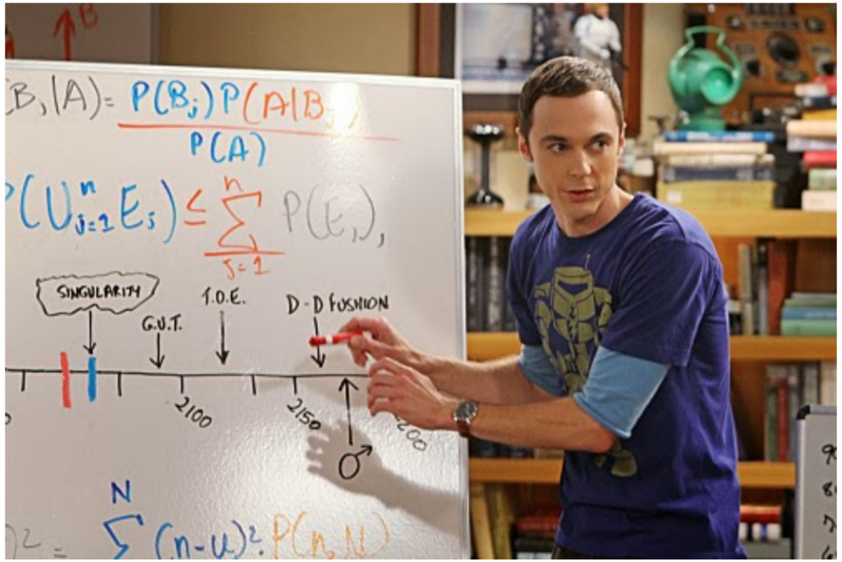
Gianluca Baio

Department of Statistical Science | University College London

g.baio@ucl.ac.uk
https://gianluca.statistica.it/
https://egon.stats.ucl.ac.uk/research/statistics-health-economics/
https://github.com/giabaio
https://github.com/StatisticsHealthEconomics
@gianlubaio

Bayesian Methods in Health Economics, Lausanne

Objective of this course



- Introduction to Bayesian analysis
  - MCMC methods
  - Using R and BUGS
- Apply Bayesian analysis to health economic evaluations
  - Cost-effectiveness analysis
  - Probabilistic sensitiity analysis
  - Advanced modelling

- Emphasis on **practical examples**
  - BUGS analysis
  - R/BUGS and BCEA
  - Problem-specific vs standardised analysis

Relevant resources

The course website contains all the relevant information

- Reading list
- Course description & assessment
- Full timetable
- Full syllabus
- Useful tips of the computer specification (for the practicals)

All the lecture slides are also available from the main page (see top menu under "Slides")

The material for the computer practicals is also available from the main page (see top menu under "Practicals")

> The relevant slides and practical material will be made available **before** the scheduled lecture.
> Annotated solutions to the practicals will also be made available **after** the sessions.

Relevant resources

The course website contains all the relevant information

- Reading list
- Course description & assessment
- Full timetable
- Full syllabus
- Useful tips of the computer specification (for the practicals)

All the lecture slides are also available from the main page (see top menu under "Slides")

The material for the computer practicals is also available from the main page (see top menu under "Practicals")

> The relevant slides and practical material will be made available **before** the scheduled lecture.
> Annotated solutions to the practicals will also be made available **after** the sessions.

- Some R resources:
  - This is a very comprehensive introduction.
  - This is also a very good introduction, particularly around many of the more modern features of R (e.g. the `tidyverse` package/approach).
- NICE Decision Support Unit website: http://nicedsu.org.uk/
- Moodle page (UCL-registered): https://moodle.ucl.ac.uk/course/view.php?id=8596

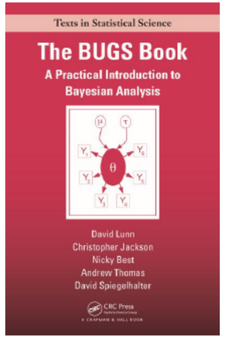
The BUGS Book

A Practical Introduction to Bayesian Analysis

Authors/Affiliations

David Lunn, MRC Biostatistics Unit, Cambridge, UK
Chris Jackson, MRC Biostatistics Unit, Cambridge, UK, MRC Biostatistics Unit, Cambridge, UK
Nicky Best, Imperial College, London, UK, Imperial College London, London, UK
Andrew Thomas, MRC Biostatistics Unit, Cambridge, UK, MRC Biostatistics Unit, Cambridge, UK
David Spiegelhalter, University of Cambridge, UK, University of Cambridge, UK

In recent years, Bayesian methods have become the most widely used statistical methods for data analysis and modelling. The BUGS software has become the most popular software for Bayesian analysis worldwide. Authored by the team that originally developed this software, **The BUGS Book** provides a practical introduction to this program and its use. The text presents complete coverage of all the functionalities of BUGS, including prediction, missing data, model criticism, and prior sensitivity. It also features a large number of worked examples, covering a wide range of applications from various disciplines, and exercises, solutions, code and data on a supplementary website.



Key Features

- Provides an accessible introduction to Bayesian analysis using the BUGS software
- Covers all the functionalities of BUGS, including prediction, missing data, model criticism, and prior sensitivity
- Features a large number of worked examples and applications from a wide range of disciplines
- Includes detailed exercises and solutions on the supporting website
- Authored by the team that developed the BUGS software.

Selected Contents

Introduction: Probability and Parameters. Monte Carlo Simulations using BUGS. Introduction to Bayesian Inference. Introduction to Markov Chain Monte Carlo Methods. Prior Distributions. Regression Models. Categorical Data. Model Checking and Comparison. Issues in Modeling. Hierarchical Models. Specialized Models. Different Implementations of BUGS. Appendices. Bibliography. Index.

Catalog no. C8490
October 2012, 399 pp.
ISBN: 978-1-58488-849-9
$52.95 / £25.99

Bayesian Methods in Health Economics

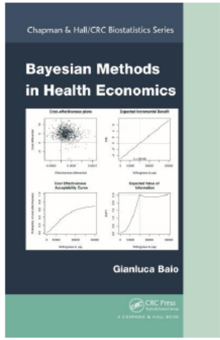
Author/Affiliation

Gianluca Baio, Department of Statistical Science, University College London, UK

Health economics is concerned with the study of the cost-effectiveness of health care interventions. This book provides an overview of Bayesian methods for the analysis of health economic data. After an introduction to the basic economic concepts and methods of evaluation, it presents Bayesian statistics using accessible mathematics. The next chapters describe the theory and practice of cost-effectiveness analysis from a statistical viewpoint, and Bayesian computation, notably MCMC. The final chapter presents three detailed case studies covering cost-effectiveness analyses using individual data from clinical trials, evidence synthesis and hierarchical models and Markov models. The text uses WinBUGS and JAGS with datasets and code available online.

Key Features

- Provides an overview of Bayesian methods for cost-effectiveness analysis, and includes all necessary background on economics and Bayesian statistics
- Presents three detailed case studies of the cost-effectiveness analysis of health care interventions
- Includes several worked examples to guide through the process of health economic evaluation
- Contains extensive coverage of the practice of making Bayesian analysis integrating software such as JAGS and R, specifically for the application of health economic analysis
- Systematically describes the methodological issues related to the application of Bayesian inference and decision process in health economics
- Designed as a reference for students and practitioners working in the field of health economic evaluations and medical statistics



The book is linked to code with which to replicate the examples, and an associated R package (BCEA) can be used in real applications to produce systematic health economic evaluations of Bayesian models.

Selected Contents

Introduction to Health Economic Evaluation. Introduction to Bayesian Inference. Statistical Cost-Effectiveness Analysis. Bayesian Analysis in Practice. Health Economic Evaluation in Practice.

Catalog no. K14236
November 2012, 243 pp.
ISBN: 978-1-4398-9555-9
$93.95 / £59.99

Disclaimer...

Manuela Joore
@ManuelaJoore

Best opening sentence #ISPOREurope from Gianluca Baio: “statisticians should rule the world and Bayesian statisticians should rule all statisticians”

> **Gianluca Baio** @gianlubaio
> Ready for our session on open siurce models & methods!

4:52 PM · Nov 4, 2019

16 Reply Copy link

Read 2 replies

...Just so you know what you're about to get into... 😉

Summary

- Deductive inference
  - "Standard" statistical methods
- Inductive inference
  - Bayesian reasoning
  - Basic ideas
- The Bayesian view making *probability statements about parameters*
  - *Quantities* that control reality (patient's risk of disease, treatment effects . . .)
  - Don't / can't know them for sure
  - Can express this *uncertainty* using tools of *probability*
- Examples of probability distributions
- Monte Carlo simulation for prediction under uncertainty
- Implementation in BUGS

References

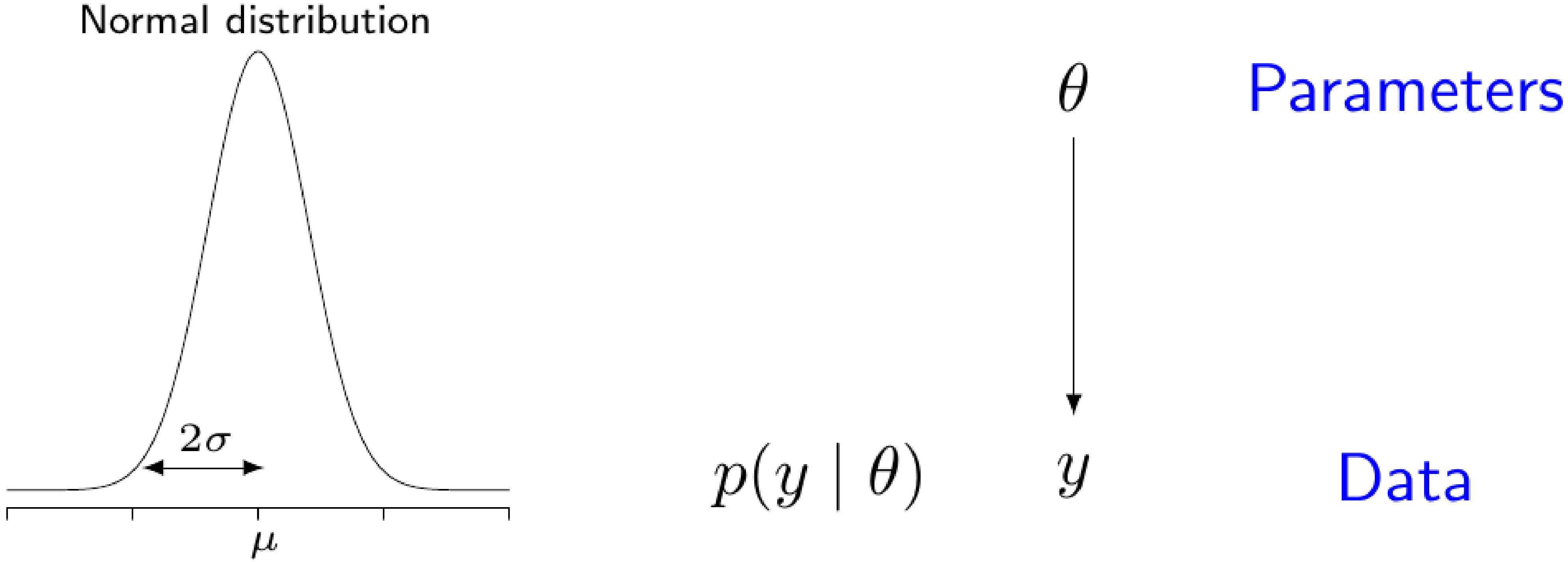
The BUGS Book, chapters 1, 2, 5 Book website

Bayesian Methods in Health Economics, chapters 2, 4 Book website (CRC) Book website Code

https://gianluca.statistica.it/teaching/intro-stats/

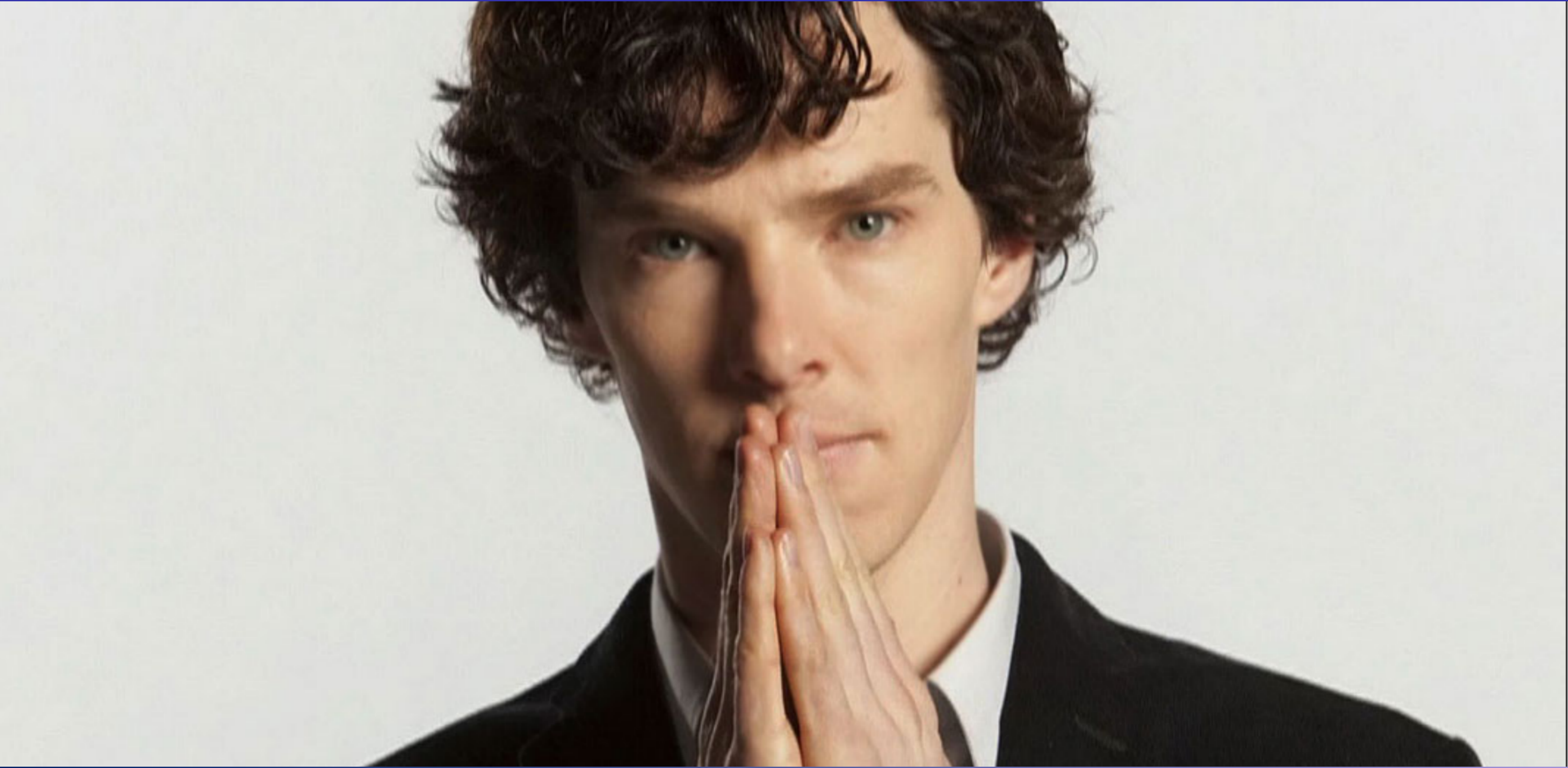
What is statistics all about?

- Typically, we observe some data and we want to use them to learn about some unobservable feature of the general population in which we are interested
- To do this, we use statistical models to describe the probabilistic mechanism by which (**we assume!**) that the data have arisen

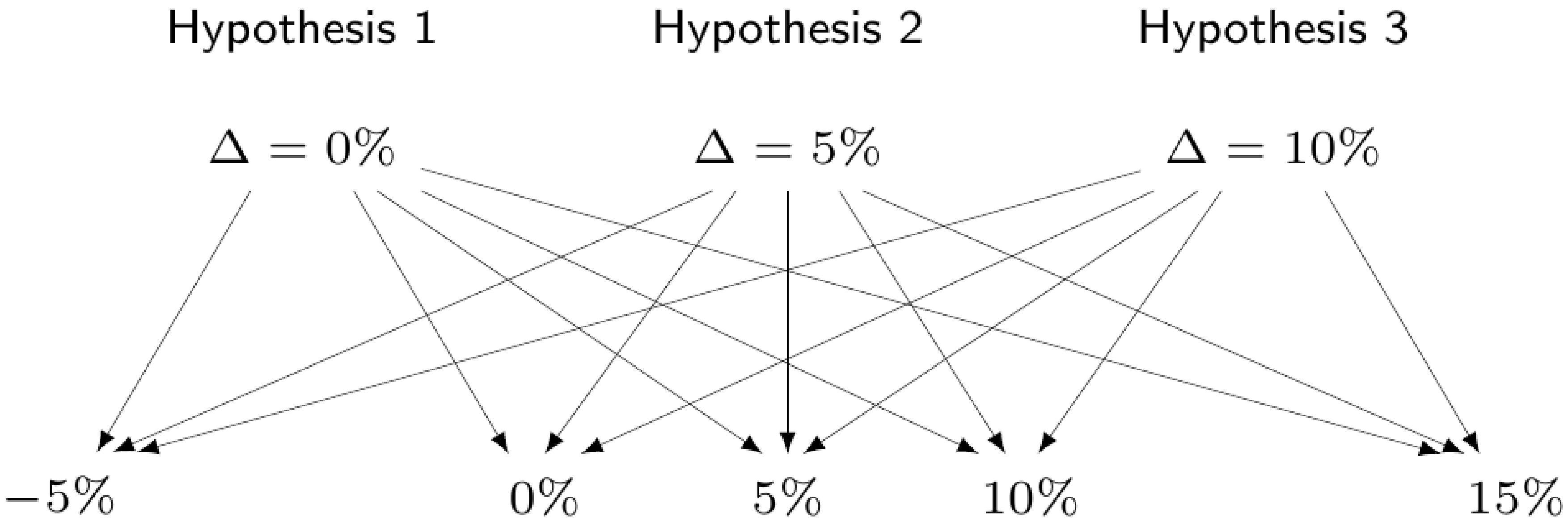


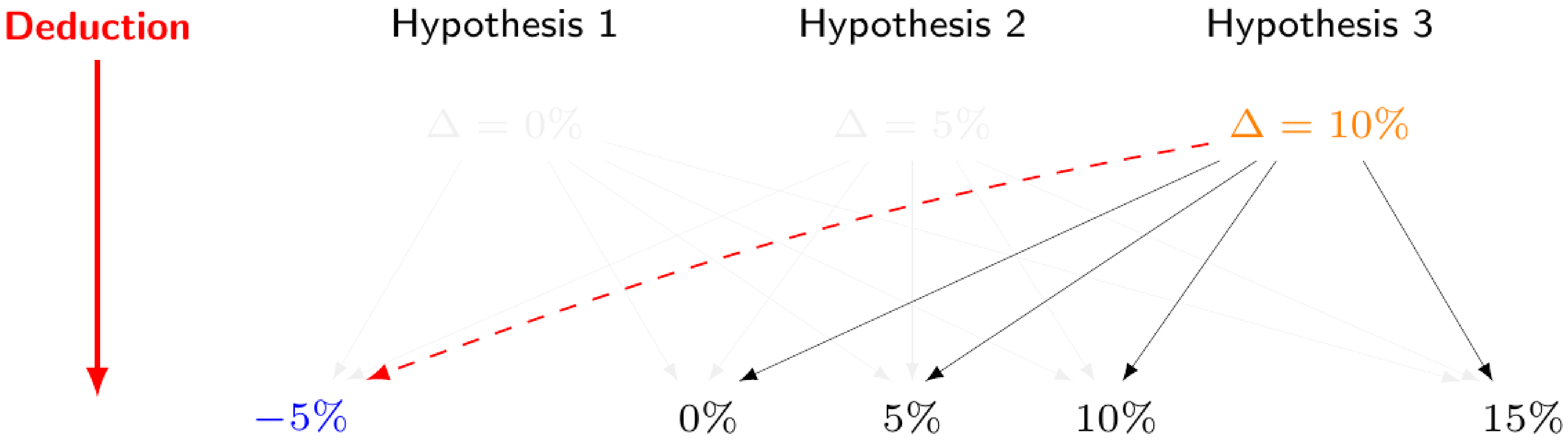
NB: Roman letters $(y$ or $x)$ typically indicate **observable data**, while Greek letters $(\theta, \mu, \sigma, \ldots)$ indicate **population parameters**

The Sherlock conundrum



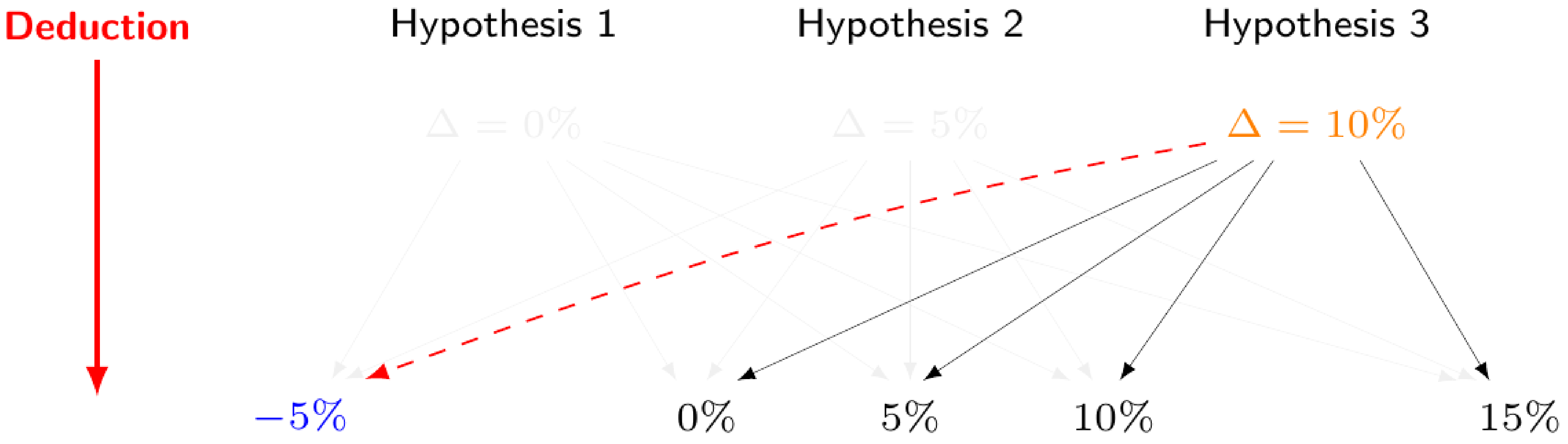
Deductive vs inductive inference





- Standard (frequentist) procedures fix the working hypotheses and, **by deduction**, make inference on the observed data:
  - If my hypothesis is true, what is the probability of randomly selecting the data that I actually observed? If small, then *deduce* weak support of the evidence to the hypothesis

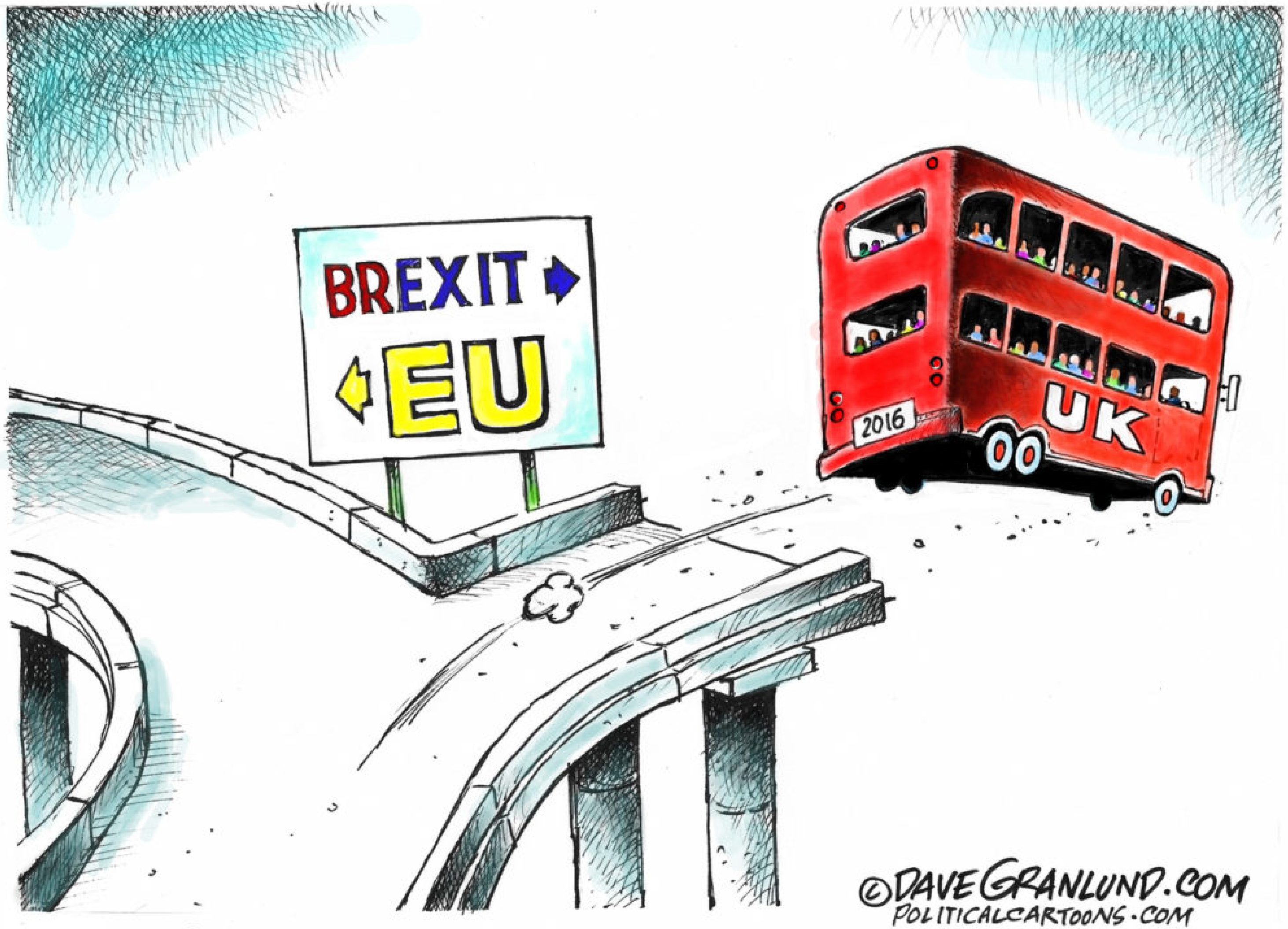
Deductive vs inductive inference



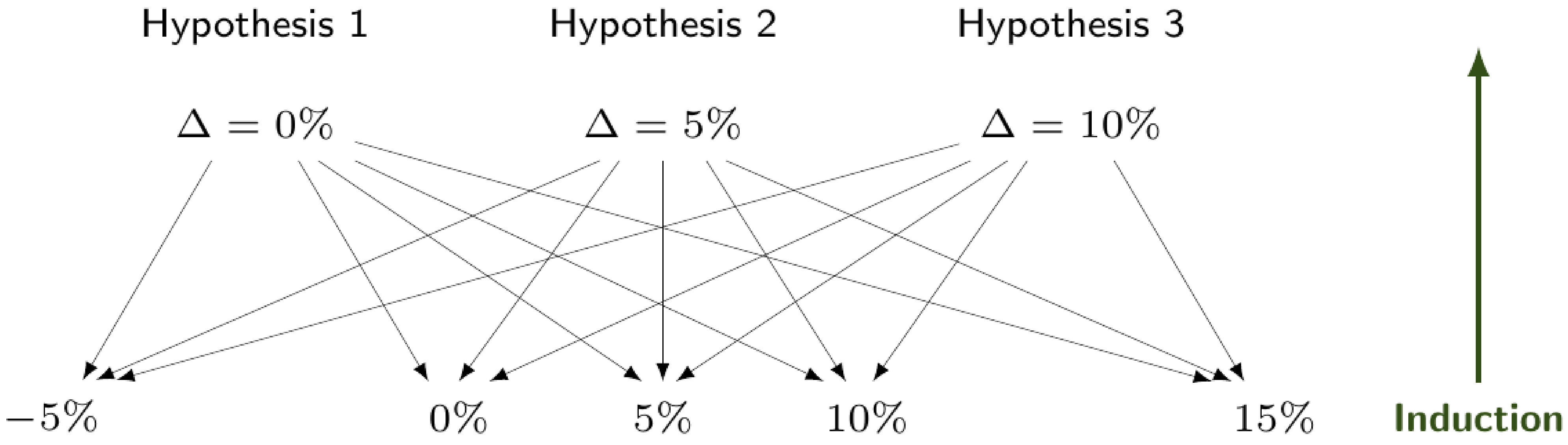
- Standard (frequentist) procedures fix the working hypotheses and, **by deduction**, make inference on the observed data:
  - If my hypothesis is true, what is the probability of randomly selecting the data that I actually observed? If small, then *deduce* weak support of the evidence to the hypothesis
  - Assess Pr(Observed data | Hypothesis)
  - Directly relevant for standard frequentist summaries, eg p-values, Confidence Intervals, etc
  - **NB**: Comparison with data that could have been observed, but haven't!

Adapted from Goodman (1999)

Is there another way?...

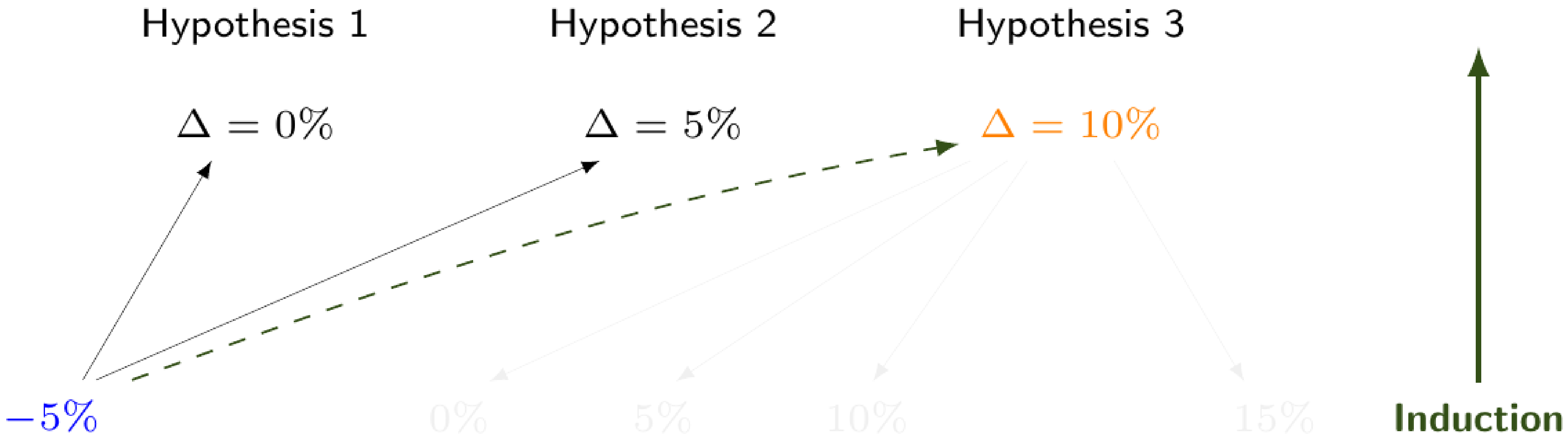


Deductive vs **inductive** inference



- The **Bayesian** philosophy proceeds fixing the value of the observed data and, **by induction**, makes inference on unobservable hypotheses
  - What is the probability of my hypothesis, given the data I observed? If less than the probability of other competing hypotheses, then weak support of the evidence to the hypothesis

Deductive vs **inductive** inference



- The **Bayesian** philosophy proceeds fixing the value of the observed data and, **by induction**, makes inference on unobservable hypotheses
  - What is the probability of my hypothesis, given the data I observed? If less than the probability of other competing hypotheses, then weak support of the evidence to the hypothesis
  - Assess Pr(Hypothesis | Observed data)
  - Can express in terms of an **interval** estimate: Pr$(a \leq$ parameter $\leq b \mid$ Data$)$
  - **NB**: Unobserved data have no role in the inference!

How did it all start?

In 1763, Reverend Thomas Bayes of Tunbridge Wells wrote

P R O B L E M.

Given the number of times in which an unknown event has happened and failed: *Required* the chance that the probability of its happening in a ſingle trial lies ſomewhere between any two degrees of probability that can be named.

In modern language, given $r \sim \text{Binomial}(\theta, n)$, what is $\Pr(\theta_1 < \theta < \theta_2 \mid r, n)$?

Some historical references

- http://www.bayesian.org/resources/bayes.html
- S. Bertsch McGrayne (2011). *The Theory That Would Not Die*
- S. Fienberg (2006). When did Bayesian inference become Bayesian?

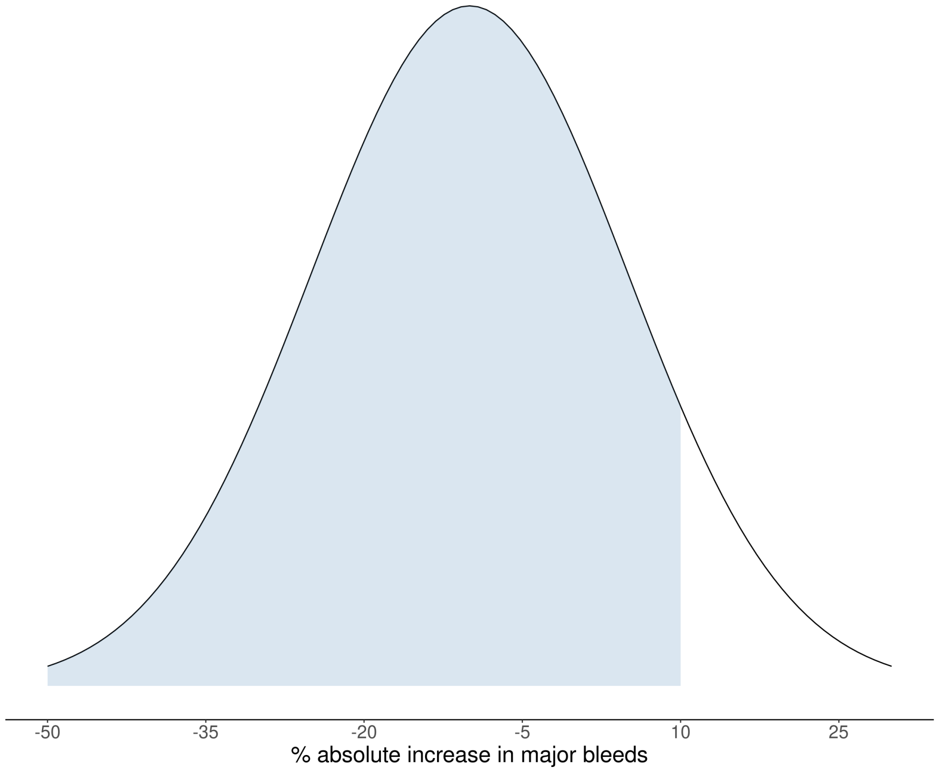
see Lecture

Basic ideas

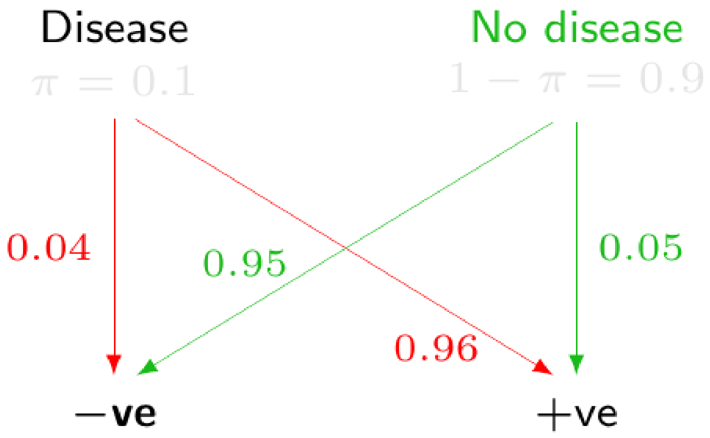
Direct expression of uncertainty about unknown parameters

"There is an 89% probability that the absolute increase in major bleeds is less than 10 percent with low-dose PLT transfusions"

(Tinmouth et al, *Transfusion*, 2004)

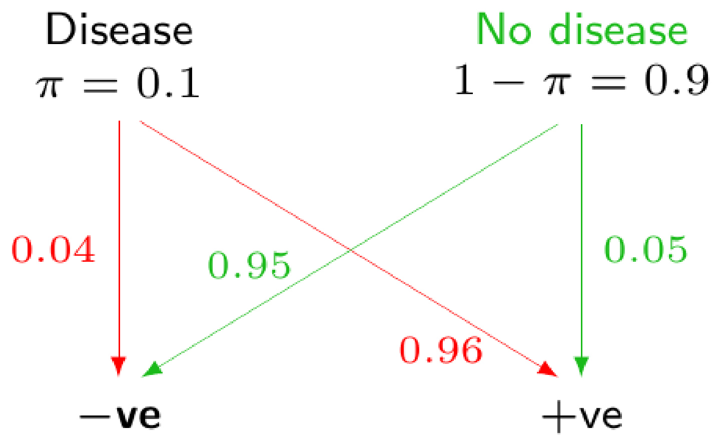


Basic ideas



- Suppose a patient is tested for HIV. The test comes up negative (–ve)
- Given the assumptions/model, this indicates **fairly strong** evidence against the hypothesis that the true status is "Disease", so basically $p = 0.04$

Basic ideas



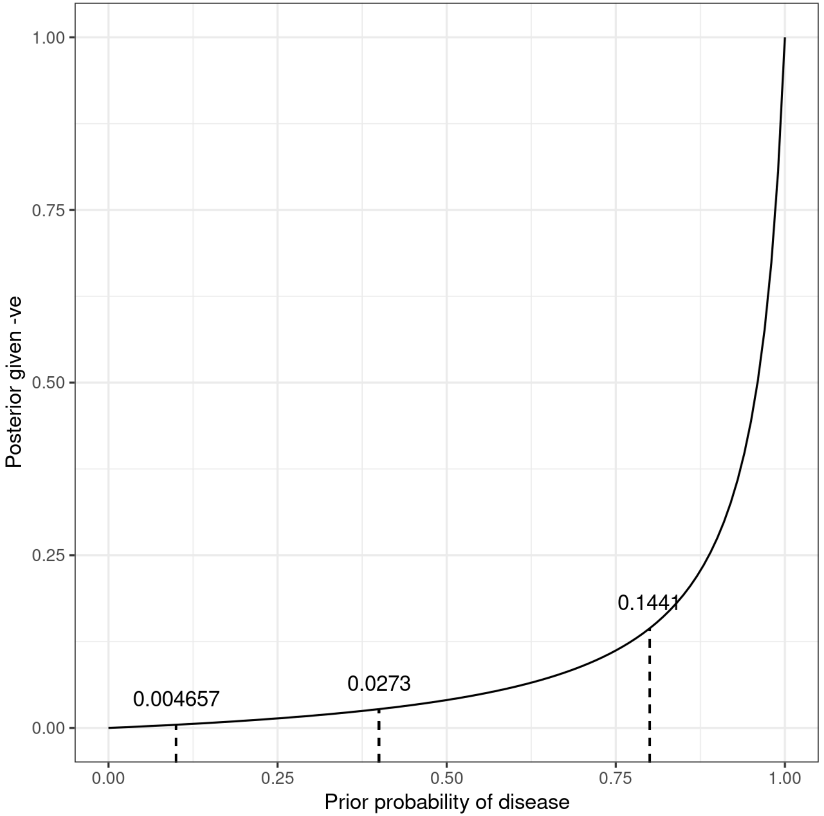
- Suppose a patient is tested for HIV. The test comes up negative (–ve)
- Given the assumptions/model, this indicates **fairly strong** evidence against the hypothesis that the true status is "Disease", so basically $p = 0.04$
- But: how **prevalent** is the disease in the population?
  - We can model our prior knowledge about this and combine this information with the evidence from the data (using **Bayes theorem**)

$$\Pr(\text{Disease} \mid \text{-ve}) = \frac{\Pr(\text{Disease})\Pr(\text{-ve} \mid \text{Disease})}{\Pr(\text{-ve})}$$

- Update uncertainty given the evidence provided by the data

Prior vs posterior

- The evidence **from the data alone** tells us that the observed result is extremely unlikely under the hypothesis of "Disease"
- This is strongly dependent on the **context**, as provided by the prior knowledge/epistemic uncertainty, though!



Basic ideas

- A Bayesian model specifies a **full probability distribution** to describe uncertainty
- This applies to
  - **Data**, which are subject to **sampling variability**
  - **Parameters** (or hypotheses), typically unobservable and thus subject to **epistemic uncertainty**
  - And even future, yet unobserved realisations of the observable variables (data)

Basic ideas

- A Bayesian model specifies a **full probability distribution** to describe uncertainty
- This applies to
  - **Data**, which are subject to **sampling variability**
  - **Parameters** (or hypotheses), typically unobservable and thus subject to **epistemic uncertainty**
  - And even future, yet unobserved realisations of the observable variables (data)
- Probability is the only languange in the Bayesian framework to assess any form of imperfect information or knowledge
  - No need to distinguish between probability and confidence
  - Before even seeing the data, we need to identify a suitable probability distribution to describe the overall uncertainty about the data $\boldsymbol{y}$ and the parameters $\boldsymbol{\theta}$

Basic ideas

- A Bayesian model specifies a **full probability distribution** to describe uncertainty
- This applies to
  - **Data**, which are subject to **sampling variability**
  - **Parameters** (or hypotheses), typically unobservable and thus subject to **epistemic uncertainty**
  - And even future, yet unobserved realisations of the observable variables (data)
- Probability is the only languange in the Bayesian framework to assess any form of imperfect information or knowledge
  - No need to distinguish between probability and confidence
  - Before even seeing the data, we need to identify a suitable probability distribution to describe the overall uncertainty about the data $\boldsymbol{y}$ and the parameters $\boldsymbol{\theta}$

$$p(\boldsymbol{y},\boldsymbol{\theta}) = p(\boldsymbol{\theta})p(\boldsymbol{y}\mid\boldsymbol{\theta}) = p(\boldsymbol{y})p(\boldsymbol{\theta}\mid\boldsymbol{y})$$

(see also Lecture 4) from which we derive Bayes Theorem

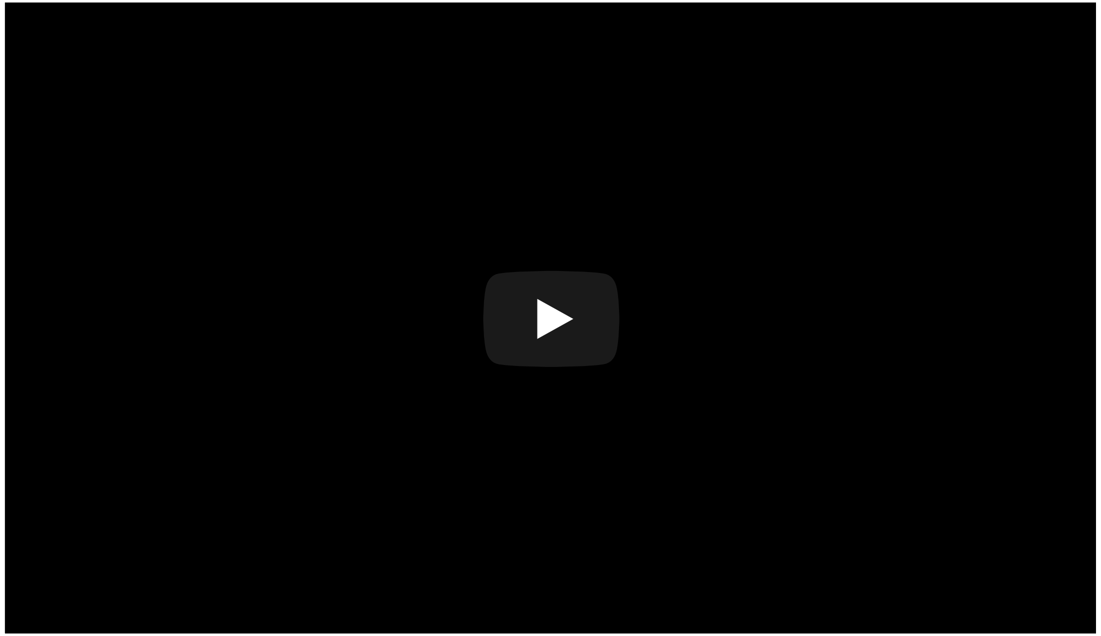
$$p(\boldsymbol{\theta}\mid\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})p(\boldsymbol{y}\mid\boldsymbol{\theta})}{p(\boldsymbol{y})}$$

- **Express beliefs in form of a probability distribution**

Bayesian computation

In artificially simplified modelling structures, Bayesian computations are just as easy as "standard" statistical models

- Thomas Bayes (1763)
  - Set up (what we now call) a Binomial model for number of "successes" out of a set number of "trials"
  - Applied to billiard balls:
- Pierre-Simon Laplace (1786)
  - Analysed data on christening in Paris from 1745 to 1770 using (what we now consider) a Bayesian model
  - Concludes that he was "morally certain" that Pr$($new born is boy $|$ data$) \geq 0.5$ (divine providence to account for the fact that males died at higher rates...)



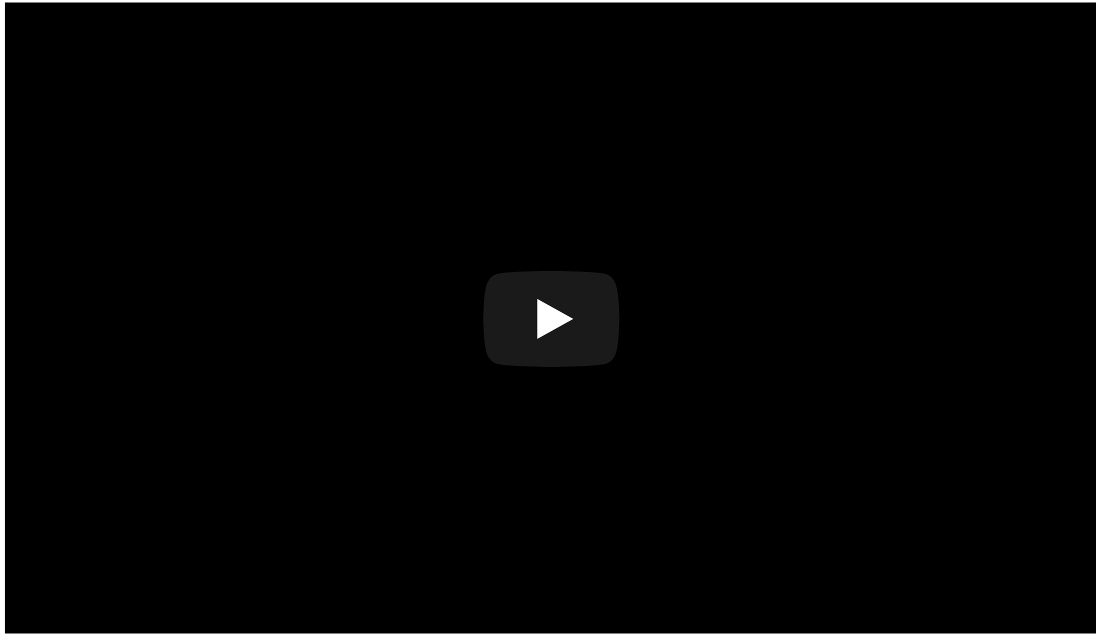
Bayesian computation

But they can become very complicated in realistic models

- Alan Turing (1940s): breaking the Enigma code 👉
  - Using "prior" information and guess a stretch of letters in an Enigma message, measure their belief in the validity of these guesses and more clues as they arrived

Since the 1990s, rely on computer simulations and a suite of algorithms called **Markov Chain Monte Carlo** (MCMC)

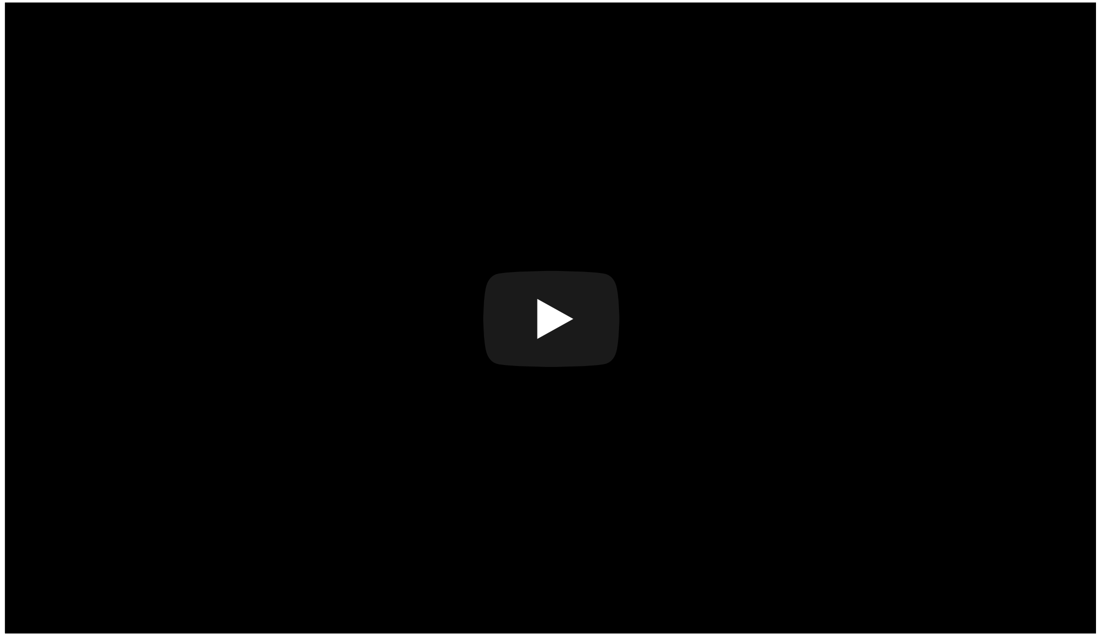
- Highly generalisable, can throw at it virtually any complexity
- Can still be computationally intensive, but variants of "vanilla" implementations can be made **very** efficient



Bayesian computation

But they can become very complicated in realistic models

- Alan Turing (1940s): breaking the Enigma code 👉
  – Using "prior" information and guess a stretch of letters in an Enigma message, measure their belief in the validity of these guesses and more clues as they arrived



Since the 1990s, rely on computer simulations and a suite of algorithms called **Markov Chain Monte Carlo** (MCMC)

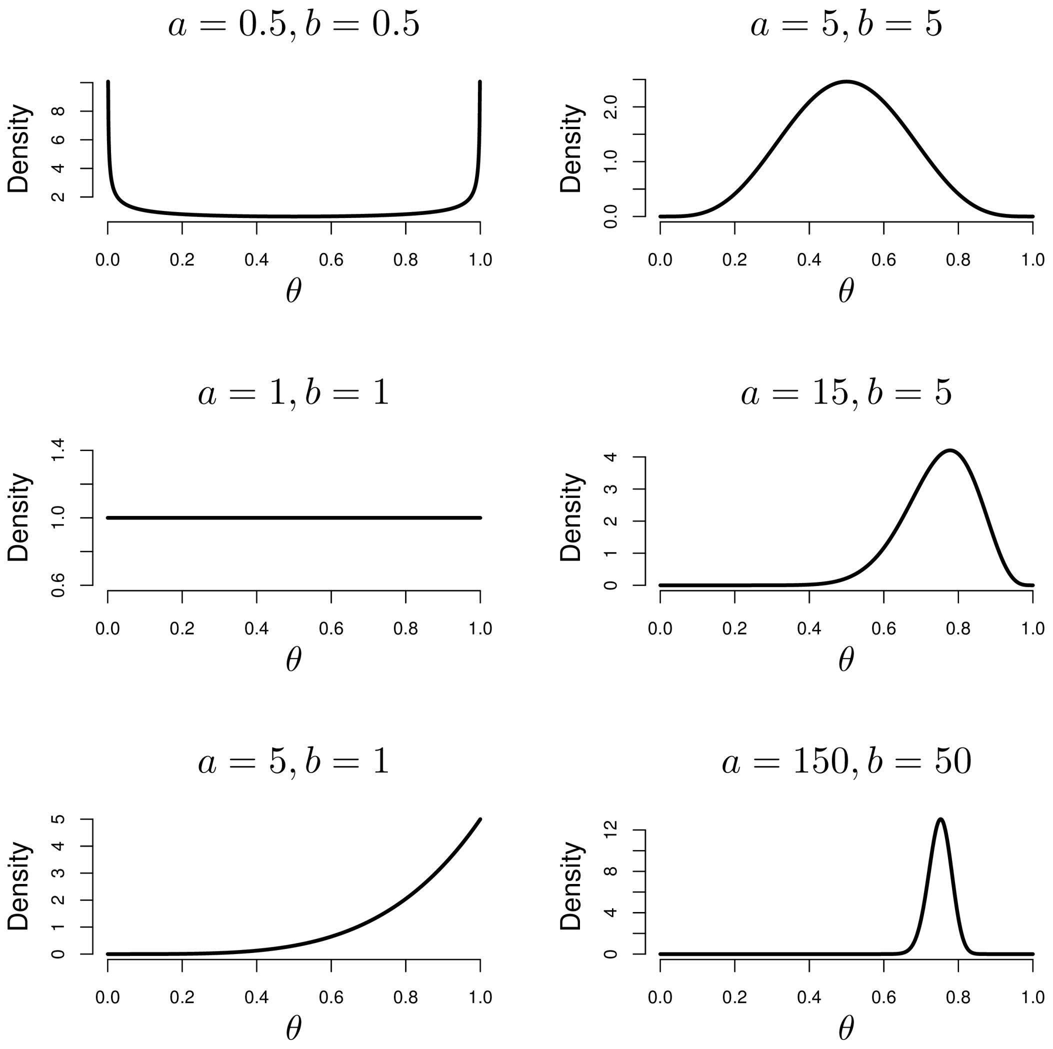
- Highly generalisable, can throw at it virtually any complexity
- Can still be computationally intensive, but variants of "vanilla" implementations can be made **very** efficient

Need to manipulate probability distributions to express both **sampling variability** and **epistemic uncertainty**!

- Choose most appropriate distribution for the quantity of interest

Beta distribution

General distribution representing uncertainty about a *proportion* or *probability* - values between *0 and 1*



$\theta \sim \text{Beta}(a, b)$ has density

$$p(\theta \mid a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\ \theta^{a-1}(1-\theta)^{b-1}$$

Mean and variance:

$$\mu = \frac{a}{a+b}$$

$$\sigma^2 = \frac{ab}{(a+b)^2(a+b+1)}$$

Expressing beliefs as a Beta distribution

1. Define mean and SD
   - Beta(a, b) has mean $\mu = \dfrac{a}{a+b}$, variance $\sigma^2 = \dfrac{ab}{(a+b)^2(a+b+1)}$
   - Solving gives a, b in terms of assumed mean and SD:

$$a = \mu\left(\frac{(1-\mu)\mu}{\sigma^2} - 1\right)$$

$$b = (1-\mu)\left(\frac{(1-\mu)\mu}{\sigma^2} - 1\right)$$

eg mean 0.4, sd 0.1 gives Beta(9.2, 13.8)

Expressing beliefs as a Beta distribution

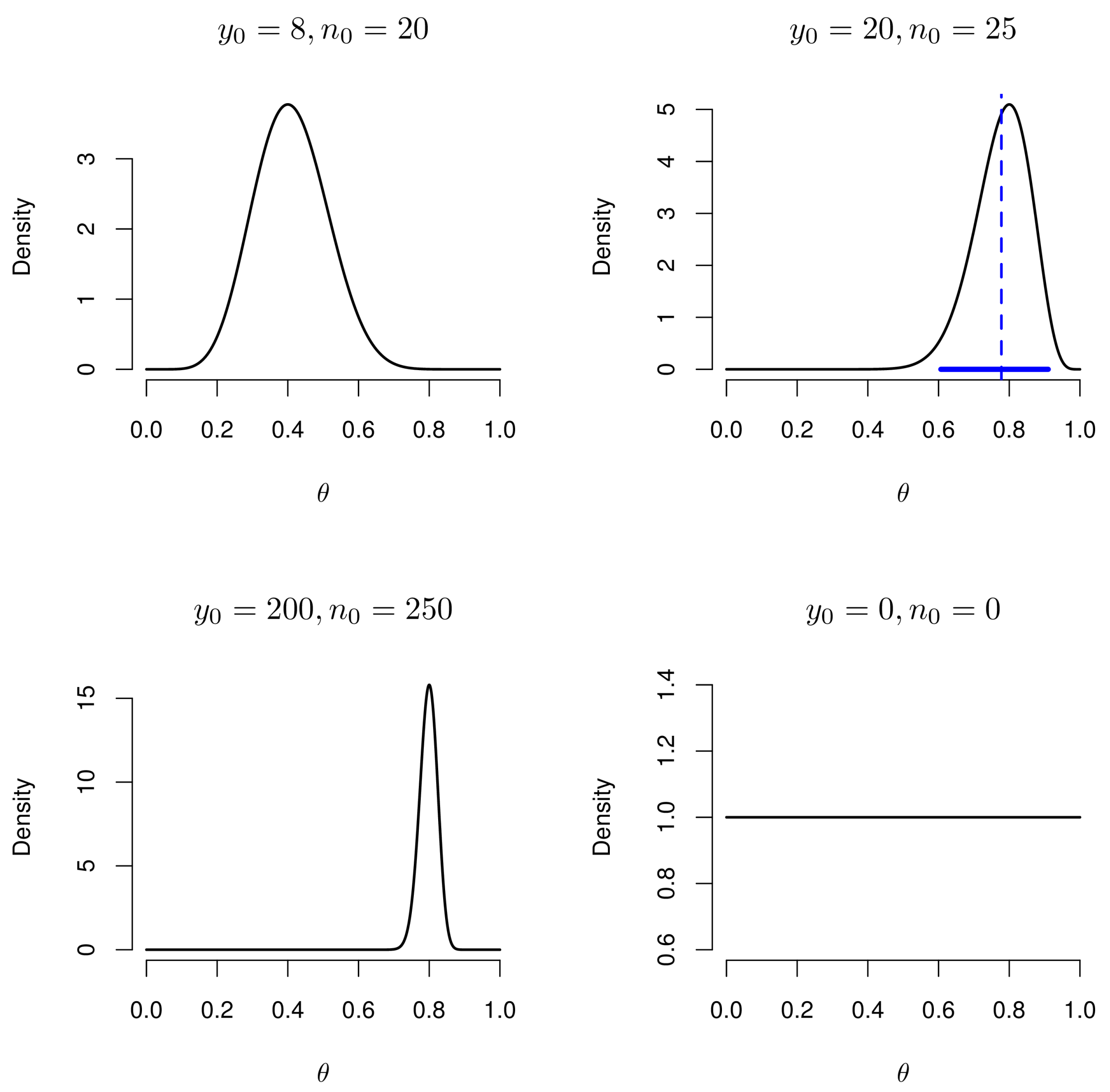
1. Define mean and SD
   - Beta(a, b) has mean $\mu = \dfrac{a}{a+b}$, variance $\sigma^2 = \dfrac{ab}{(a+b)^2(a+b+1)}$
   - Solving gives a, b in terms of assumed mean and SD:

$$a = \mu\left(\frac{(1-\mu)\mu}{\sigma^2} - 1\right)$$

$$b = (1-\mu)\left(\frac{(1-\mu)\mu}{\sigma^2} - 1\right)$$

   eg mean 0.4, sd 0.1 gives Beta(9.2, 13.8)

2. Use an *implicit* dataset
   - Imagine your beliefs about the success rate p are equivalent to observing, e.g. $y_0 = 8$ successes out of $n_0 = 20$ trials
   - This gives about Beta$(y_0 + 1, n_0 - y_0 + 1)$ – see Lecture 2 for theory!
   - Mean $(y_0+1)/(n_0+2) \approx 0.4$ and SD ≈ 0.1 here




```
> # Sets prior "successes" and "trials"
> y=20; n=25
>
> # Computes the mean of the distribution
> mean(rbeta(1000000,y+1,n-y+1))
```

```
[1] 0.7778048
```

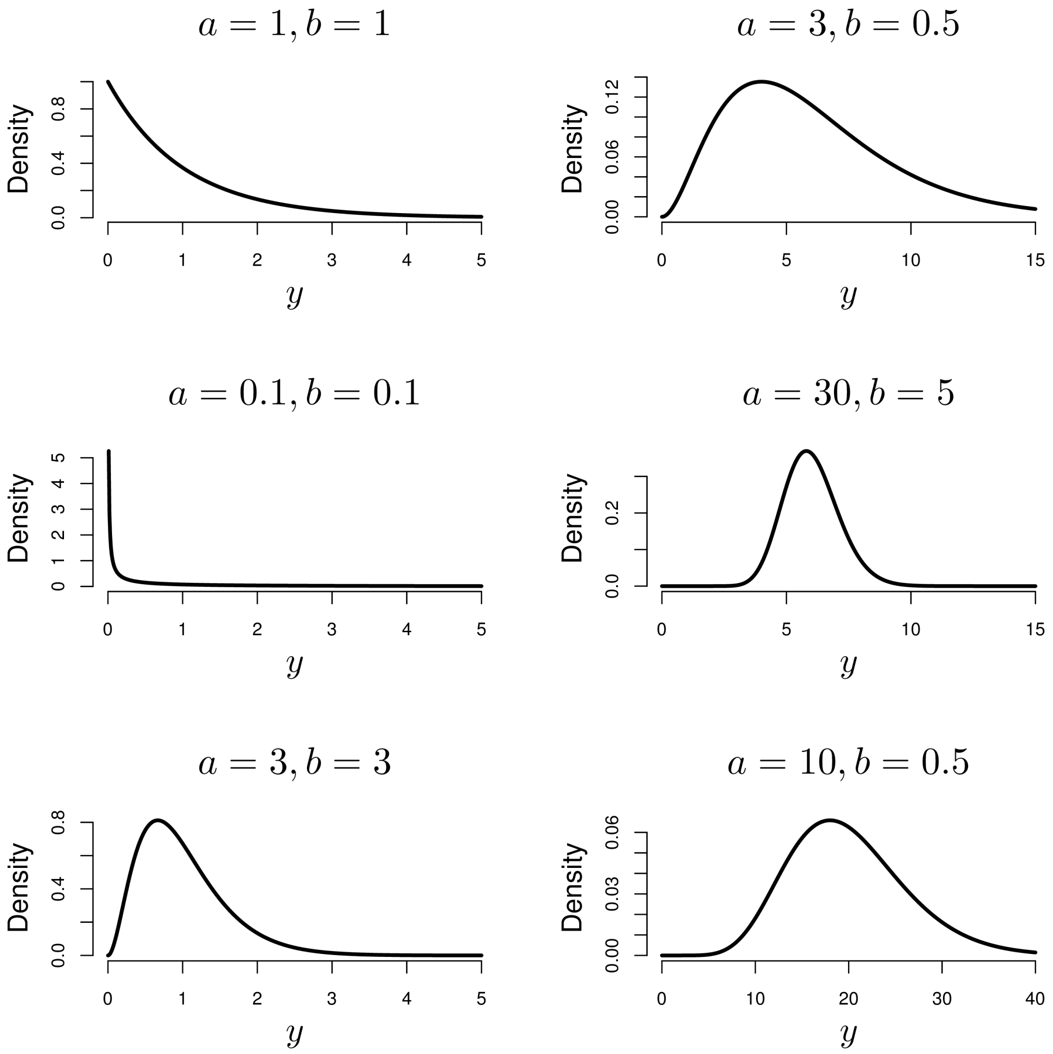
```
> # Computes summary statistics
> cbind("2.5%"=qbeta(.025,y+1,n-y+1),
+       "median"=qbeta(.5,y+1,n-y+1),
+       "97.5%"=qbeta(.975,y+1,n-y+1))
```

```
          2.5%    median     97.5%
[1,] 0.6064945 0.7847057 0.9102599
```

NB: $y_0 = n_0 = 0 \Rightarrow$ "complete ignorance" *non-informative* prior – Beta(1,1) = Uniform(0,1)

Gamma distribution

Positive, skewed quantities (eg costs, variance parameters)



If $Y \sim$ Gamma(a, b)

$$p(y \mid a, b) = \frac{b^a}{\Gamma(a)} y^{a-1} \, e^{-by}$$

$$\mathrm{E}(Y \mid a, b) = \frac{a}{b}$$

$$\mathrm{V}(Y \mid a, b) = \frac{a}{b^2}$$

log-Normal distribution is similar (but heavier tails than Gamma – see also Lecture 4!)

Broadly speaking, there are two types of Bayesian analysis:

Forward sampling (Monte Carlo): this lecture

- Express current knowledge as parameters with distributions
- Simulate parameters, make predictions from models based on the parameters
- Like a spreadsheet with randomness on cells. Familiar in health economics as "probabilistic sensitivity analysis" (see Lecture 3)
- Doesn't really need specialised software

Broadly speaking, there are two types of Bayesian analysis:

Forward sampling (Monte Carlo): this lecture

- Express current knowledge as parameters with distributions
- Simulate parameters, make predictions from models based on the parameters
- Like a spreadsheet with randomness on cells. Familiar in health economics as "probabilistic sensitivity analysis" (**see** Lecture 3)
- Doesn't really need specialised software

Model fitting using Bayes Theorem (Markov Chain Monte Carlo): **see** Lecture 2

- Combine prior knowledge with *learning from data*
- Searches for the unknown *posterior* distribution based on prior and data
- Needs to use/programme specific software (eg BUGS)

The BUGS language

Bayesian analysis **U**sing **G**ibbs **S**ampling

- Language for specifying Bayesian models as a *network* of *known and unknown* quantities

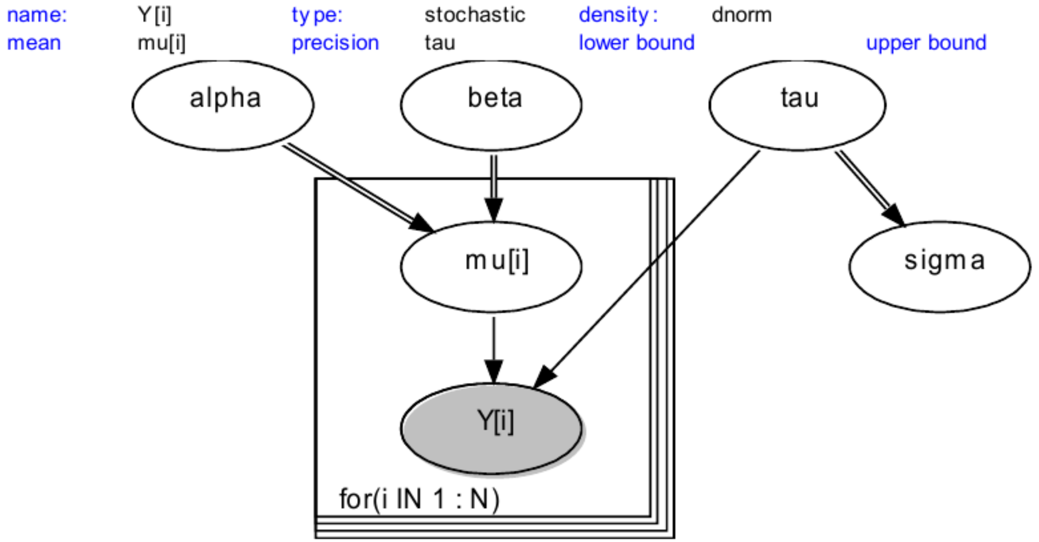
eg linear regression

$$
\begin{aligned}
Y_i &\sim \text{Normal}(\mu_i, \tau) \\
\mu_i &= \alpha + \beta x_i \\
\tau &= 1/\sigma^2
\end{aligned}
$$

```
for (i in 1:N) {
  Y[i] ~ dnorm(mu[i], tau)
  mu[i] <- alpha + beta*x[i]
}
tau <- 1/(sigma*sigma)

# Prior knowledge
alpha ...
beta ...
sigma ...
)
```

Works by internally constructing *directed acyclic graph* - model code *equivalent* to a graph



Simulates distributions of unknowns conditional on prior distributions and data

Different versions of BUGS

- `WinBUGS` 1.4.3
  - Original release 1997 – runs only on `Windows`
  - Stable but no longer developed (latest release: August 2007)
  - Freely available from http://www.mrc-bsu.cam.ac.uk/bugs
- `OpenBUGS` http://www.openbugs.net
  - Open-source offshoot, also runs on `Linux` and `MacOS`
  - Works just as well, stable

Different versions of BUGS

- WinBUGS 1.4.3
  - Original release 1997 – runs only on Windows
  - Stable but no longer developed (latest release: August 2007)
  - Freely available from http://www.mrc-bsu.cam.ac.uk/bugs
- OpenBUGS http://www.openbugs.net
  - Open-source offshoot, also runs on Linux and MacOS
  - Works just as well, stable

"Rivals"/alternatives

- JAGS http://mcmc-jags.sourceforge.net
  - Language essentially identical, Work just as well, stable
  - Runs natively on Mac/Unix/Windows
- Stan http://mc-stan.org/
  - *Probabilistic* language – slightly different than BUGS/JAGS
  - Based on different algorithm – can be more efficient in some cases

Interfaces exist to run these from other software, eg R (R2OpenBUGS, R2jags, rstan) Excel, S-Plus, SAS, Matlab, Stata, ...

A Bayesian workflow...

1. Pre-process data
   - Import dataset from spreadsheet
   - Create new variables
   - Subset data
   - ...

A Bayesian workflow...

1. Pre-process data
   - Import dataset from spreadsheet
   - Create new variables
   - Subset data
   - ...
2. Sets up model
   - Code up distributional assumptions
   - Define initial values (**more on this in Lecture 2!**)

A Bayesian workflow...

1. Pre-process data
   - Import dataset from spreadsheet
   - Create new variables
   - Subset data
   - ...
2. Sets up model
   - Code up distributional assumptions
   - Define initial values (**more on this in Lecture 2!**)
3. "Run" the model and obtain simulations from the target (posterior) distributions
   - Typically done using "Gibbs Sampling" (eg BUGS)
   - But other engines/MCMC algorithms/approximation methods available

A Bayesian workflow...

1. Pre-process data
   - Import dataset from spreadsheet
   - Create new variables
   - Subset data
   - ...
2. Sets up model
   - Code up distributional assumptions
   - Define initial values (**more on this in Lecture 2!**)
3. "Run" the model and obtain simulations from the target (posterior) distributions
   - Typically done using "Gibbs Sampling" (eg BUGS)
   - But other engines/MCMC algorithms/approximation methods available
4. Use simulations to summarise results
   - Make histograms of target (posterior) distributions
   - Compute means, sd, medians, quantiles of target (posterior) distributions
   - Derive distributions for functions of original parameters $g(\theta)$ – **more on this in Lecture 5!**

Some aspects of the BUGS language

Not a conventional programming language

- BUGS is for *describing a model*
  - ***not*** for performing a sequence of tasks in order
- Every line defines a model quantity in relation to others

> **Random (stochastic) dependence**
>
> e.g.: `r ~ dunif(a,b)`

- Simulate data r from model, or fit model to observed data r

> **Fixed (logical) dependence**
>
> e.g.: `m <- a + b*x`

- Define any quantity as deterministic function of another (as in a spreadsheet)

Functions in the BUGS language

Use in definitions of logical quantities, eg mathematical functions

- `tau <- 1 / pow(s,2)` sets $\tau = 1/s^2$
- `s <- 1 / sqrt(tau)` sets $s = 1/\sqrt{\tau}$

Useful data processing tricks, eg

- `p <- step(x - 0.7)` = 1 if x $\geq$ 0.7, 0 otherwise. Hence monitoring p and recording its mean will give the probability that x $\geq$ 0.7
- `p <- equals(x, 0.7)` = 1 if x $=$ 0.7, 0 otherwise

See "Model Specification/Logical nodes" in manual for full syntax

Some common distributions

Expression	**Distribution**	**Usage**
`dbin`	Binomial	`r ~ dbin(p, n)`
`dnorm`	Normal	`x ~ dnorm(mu, tau)`
`dpois`	Poisson	`r ~ dpois(lambda)`
`dunif`	Uniform	`x ~ dunif(a, b)`
`dgamma`	Gamma	`x ~ dgamma(a, b)`

- **NB**: The normal is parameterised in terms of its mean and ***precision*** = 1/ variance = $1/\mathrm{sd}^2$
- Functions cannot be used as arguments in distributions (you need to create new nodes)

See "Model Specification/The BUGS language: stochastic nodes/Distributions" in manual for full syntax

Use arrays and loops for sets of related quantities

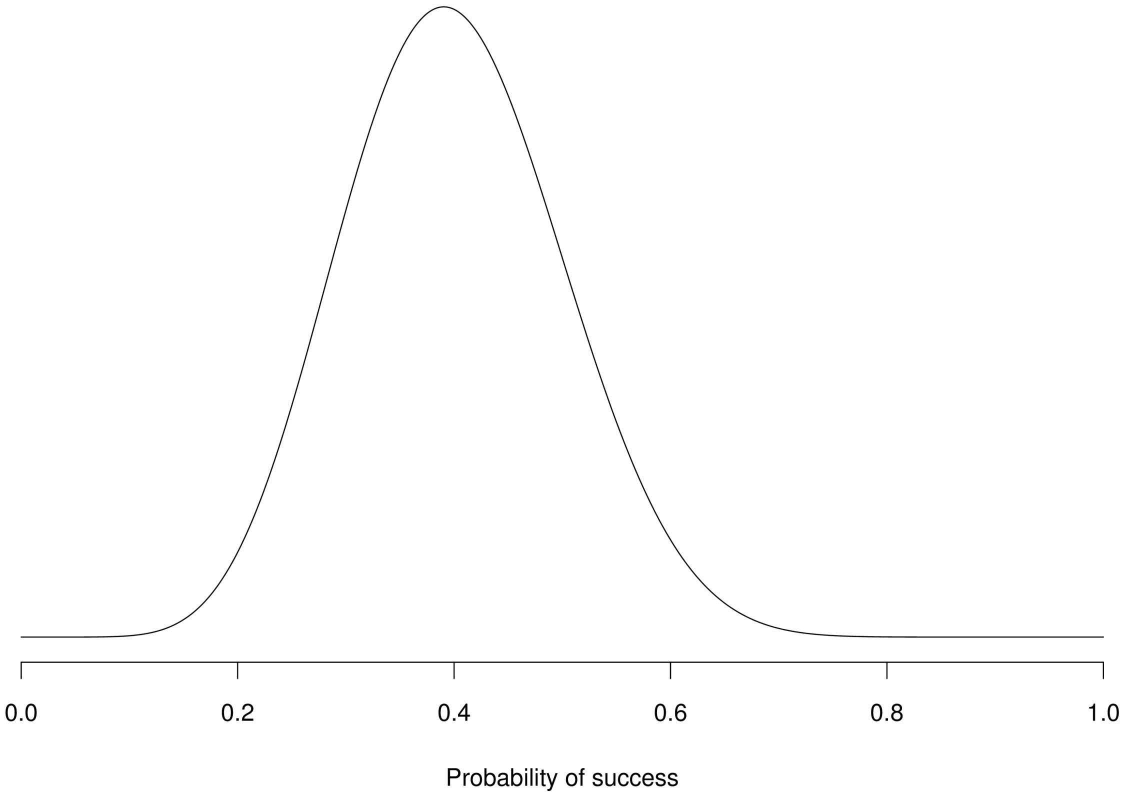
```
for (i in 1:n) {
  r[i] ~ dbin(p[i],n[i])
  p[i] ~ dunif(0,1)
}
```

Array functions: eg `mean(p[])` to take mean of whole array, `mean(p[m:n])` to take mean of elements `m` to `n`. Similarly, `sum(p[])`

See "Hints on using `OpenBUGS`" handouts, or the `OpenBUGS` manual for full information or `BUGS` syntax

Forward sampling

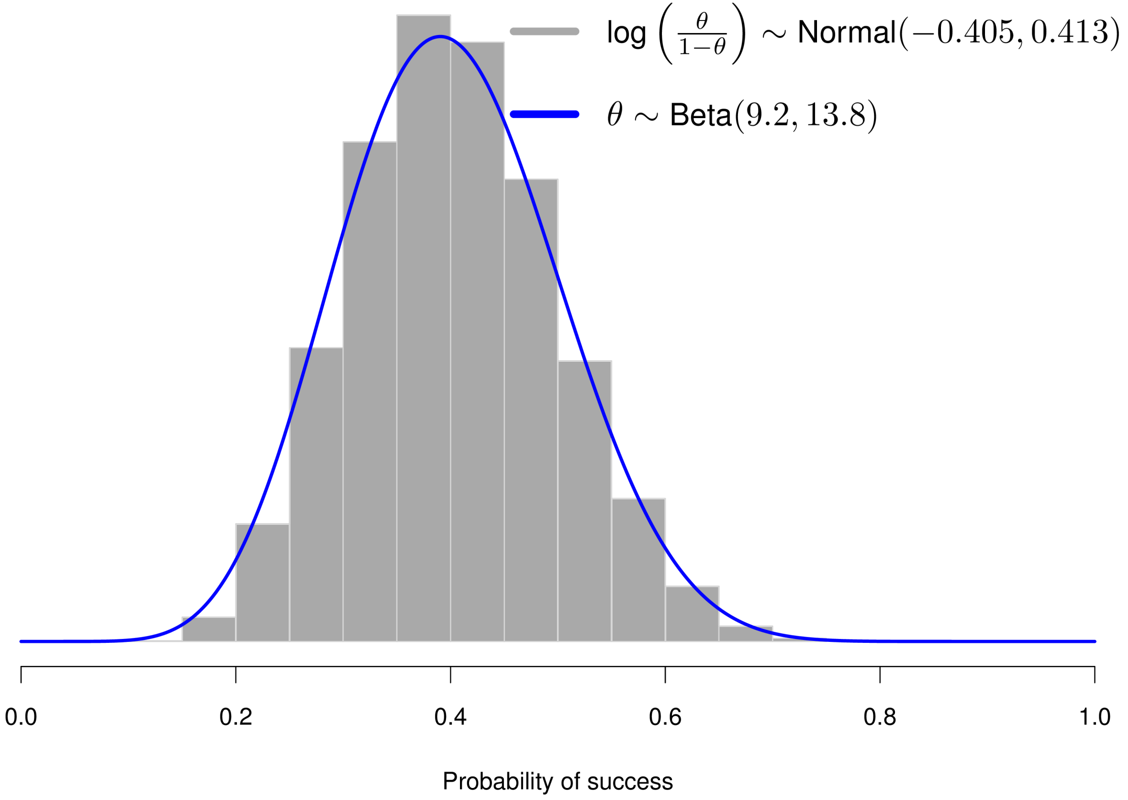
- Consider a drug to be given for relief of chronic pain
- Experience with similar compounds has suggested that annual response rates between 0.2 and 0.6 could be feasible
- Interpret this as a distribution with mean = 0.4 and standard deviation = 0.1



A **Beta(9.2,13.8)** distribution has these properties

Prior *knowledge* vs prior *distribution*

- Consider a drug to be given for relief of chronic pain
- Experience with similar compounds has suggested that annual response rates between 0.2 and 0.6 could be feasible
- Interpret this as a distribution with mean = 0.4 and standard deviation = 0.1



But: there are many possible ways to encode this prior information. For example, one could use a **Normal distribution on the logit scale**!

Making predictions

1. Model sampling variability of $y \mid \theta$ and uncertainty on θ
2. **Propagate** uncertainty in the success rate θ
3. Compute the *predictive* distribution of y: averaged over uncertainty about θ

$$p(y) = \int p(y \mid \theta)p(\theta)d\theta$$

Making predictions

1. Model sampling variability of $y \mid \theta$ and uncertainty on θ
2. **Propagate** uncertainty in the success rate θ
3. Compute the *predictive* distribution of y: averaged over uncertainty about θ

$$p(y) = \int p(y \mid \theta)p(\theta)d\theta$$

For instance, we can model

- $y \mid \theta \sim$ Binomial(θ, n): "natural" model for sampling variability
- $\theta \sim$ Beta(a, b): convenient model for epistemic uncertainty

In BUGS:

```
model {
  theta ~ dbeta(a, b)
  y ~ dbin(theta, n)
}
```

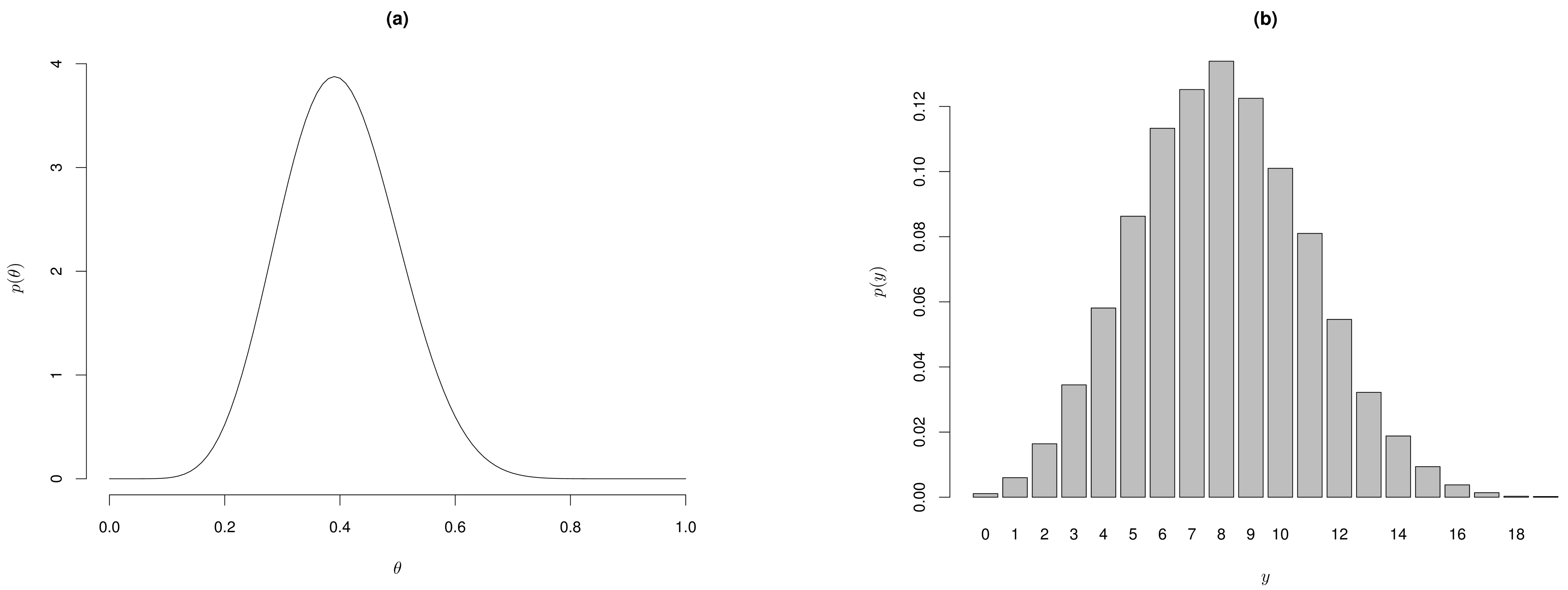
Forward sampling

R Code | Output

```
> # Sets up the parameters for the Beta prior
> a=9.2
> b=13.8
>
> # Then simulates 1000 random samples from the prior
> theta=rbeta(n=10000,a,b)
>
> # Now "mixes" **uncertainty** in the value of the parameter
> # with **variability** in the data sampling distribution
> y=rbinom(n=10000,size=20,prob=theta)
```

Forward sampling

R Code | Output



(a) is the Beta (prior) distribution

(b) is the predictive **Beta-Binomial** distribution of the number of successes in the next 20 trials

Using MC to estimate tail area probabilities

- What is the chance of getting 15 or more responders?
  - $\theta \sim$ Beta$(9.2, 13.8)$: prior distribution
  - $y \sim$ Binomial$(\theta, 20)$: sampling distribution
  - $P_{\text{crit}} =$ Pr$(y \geq 15)$: probability of exceeding critical threshold

Using MC to estimate tail area probabilities

- What is the chance of getting 15 or more responders?
  - $\theta \sim$ Beta$(9.2, 13.8)$: prior distribution
  - $y \sim$ Binomial$(\theta, 20)$: sampling distribution
  - $P_{\text{crit}} =$ Pr$(y \geq 15)$: probability of exceeding critical threshold

- In BUGS can code by translating the equations

```
# In BUGS syntax
 model {
   theta ~ dbeta(9.2, 13.8)    # prior distribution
   y ~ dbin(theta, 20)         # sampling distribution
   P.crit <- step(y - 14.5)    # = 1 if y >=15, 0 otherwise
 }
```

- **NB**: in BUGS, statements can be given in any order!

OpenBUGS output and exact answer

```
node      mean     sd      MC error     2.5%     median   97.5% start  sample
theta     0.4008   0.09999 9.415E-4     0.2174   0.3981   0.6044   1    10000
y         8.058    2.917   0.03035      3.0      8.0      14.0     1    10000
P.crit    0.0151   0.122   0.001275     0.0      0.0      0.0      1    10000
```

NB: Mean of the 0-1 indicator `P.crit`: estimated tail-area probability

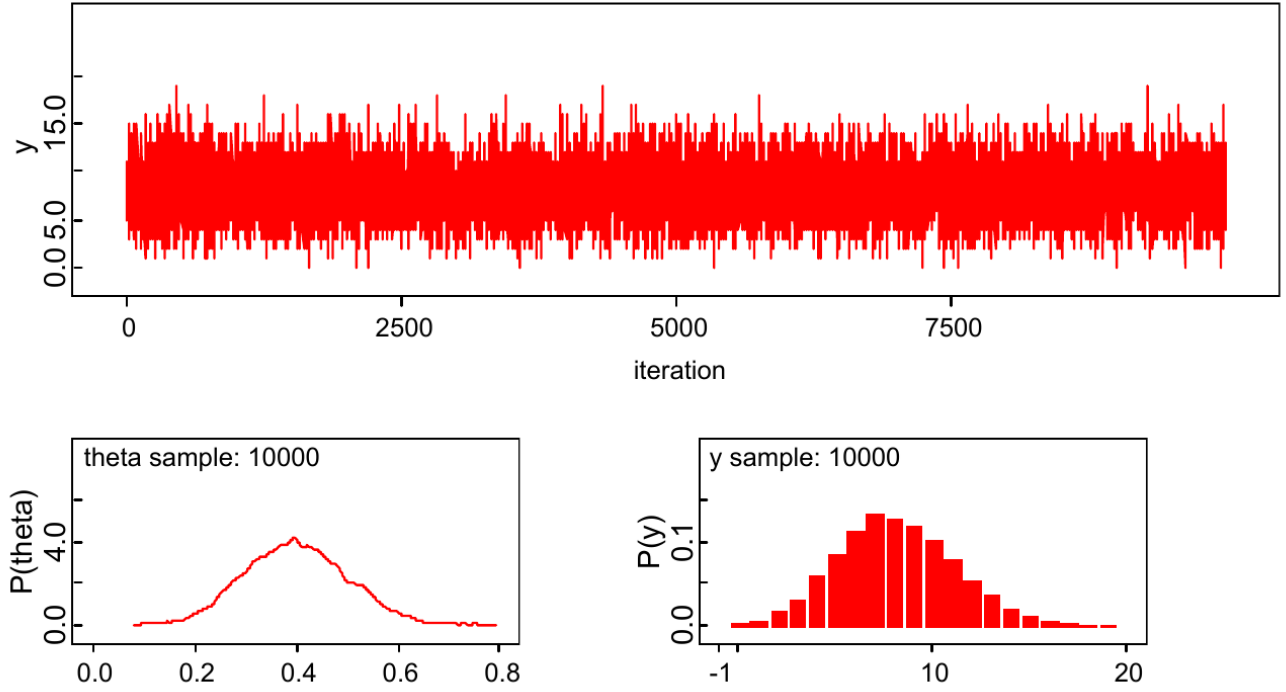
Exact answers from closed-form algebra:

- θ: mean 0.4 and standard deviation 0.1
- y: mean 8 and standard deviation 2.92
- Probability of at least 15: 0.015

MC error $\approx$ sd$/\sqrt{\text{No. iterations}}$ = std. error for *estimate of* mean

Can achieve arbitrary accuracy by running the simulation for longer

OpenBUGS output



Independent samples, so no concern with convergence (More on this later...)

Drug: Monte Carlo example

Run from `Model/Script` menu

```
modelDisplay('log')                       # set up log file
modelCheck('c:/bugscourse/drug-MC')       # check syntax of model
#  modelData('c:/bugscourse/drug-data')   # load data file if there is one
modelCompile(1)                           # generate code for 1 simulations
# modelInits('c:/bugscourse/drug-in1',1)  # load initial values if necessary
modelGenInits()                           # generate initial values for all unknown
                                          # quantities not given initial values
samplesSet(theta)                         # monitor the true response rate
samplesSet(y)                             # monitor predicted number of successes
samplesSet(P.crit)                        # monitor whether a critical number of successes
samplesTrace("*")                         # watch some simulated values (NB: slows simulation!)
modelUpdate(10000)                        # perform 10000 simulations
samplesHistory(theta)                     # Trace plot of samples for theta
samplesStats("*")                         # Summary statistics for all monitored quantities
samplesDensity(theta)                     # Plot distribution of theta
samplesDensity(y)                         # Plot distribution of y
```

Automates mouse clicks: important analyses should be *repeatable* (**But** effectively superseded by more effective tools, eg R2OpenBUGS!)

(warning: script commands different between `OpenBUGS` and `WinBUGS`)