Statistical Inference

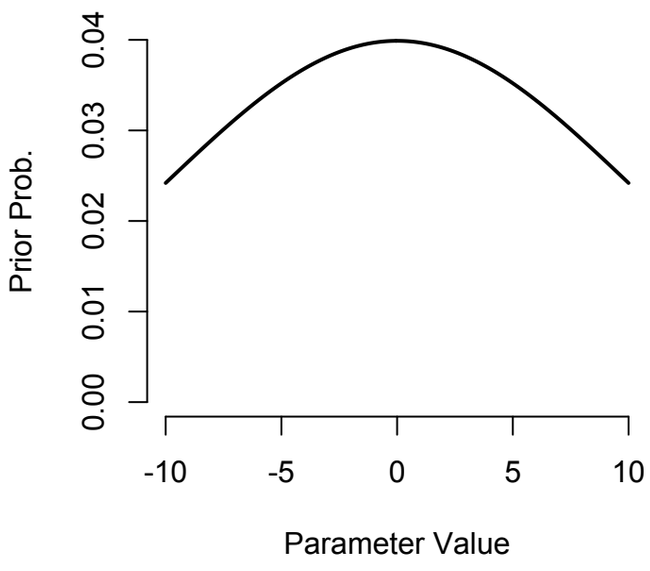
Gene Hunt
NMNH, Smithsonian Institution
Analytical Paleobiology Workshop, July 2018

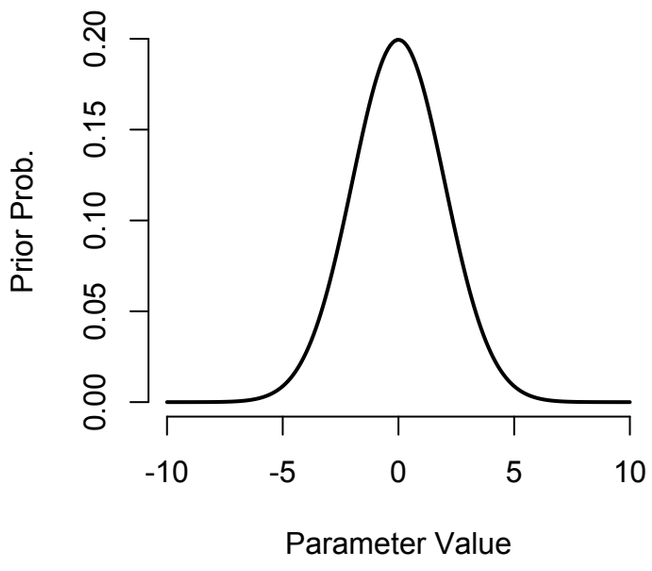
Outline

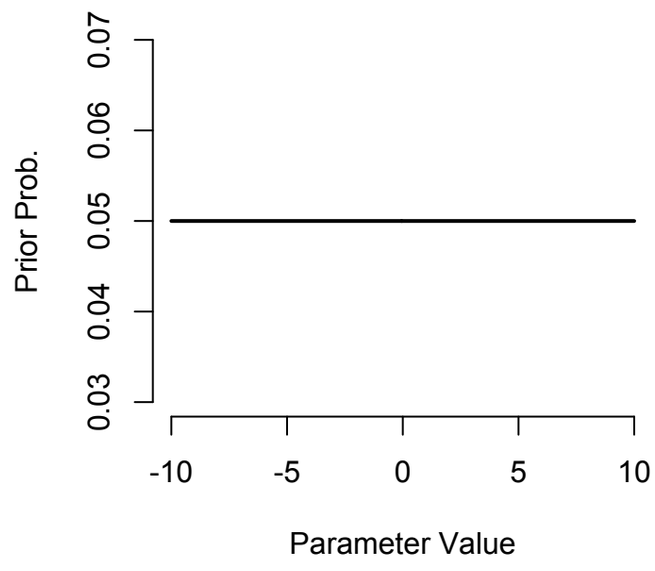
- Preliminaries: Probability and Samples
- Classical (frequentist) approaches
- Likelihood
- Bayesian Approaches
- P-hacking and Replication Crises

Bayesian Approaches

- Formalism for updating prior belief in the face of new evidence
- It is the analytical use of prior belief that sets it apart from frequentist and likelihood approaches — and made it controversial







Conceptions of Probability

- Frequentist: event probabilities defined by their frequency in an infinitely long sequence of trials
    - obviously not practical
    - some events really can't be repeated (e.g., probability France wins next World Cup)
- Bayesian: **subjective probability**, reflects one's uncertainty about events
    - may differ from person to person
    - can be treated the same as data?
    - arguably, frequentist also are swayed by prior belief, but only haphazardly

Bayesian Approaches

- Many practical forms of reasoning are inherently Bayesian, e.g. doctor's diagnosis

ailment	symptom
none	none
migraine	headache
brain tumor	headache

Migraine and Brain Tumor are equally consistent with the evidence (same likelihood), but have different priors.

Bayes Theorem

Prior Distribution **[+]** **Updated with Evidence from Data** **[=]** **Posterior Distribution**

Posterior **Prior** **Likelihood**

$$p(\theta \mid x) = \frac{p(\theta)p(x \mid \theta)}{p(x)}$$

Probability of data

Unnormalized posterior density

$$p(\theta \mid x) \propto p(\theta)p(x \mid \theta)$$

Bayes Theorem

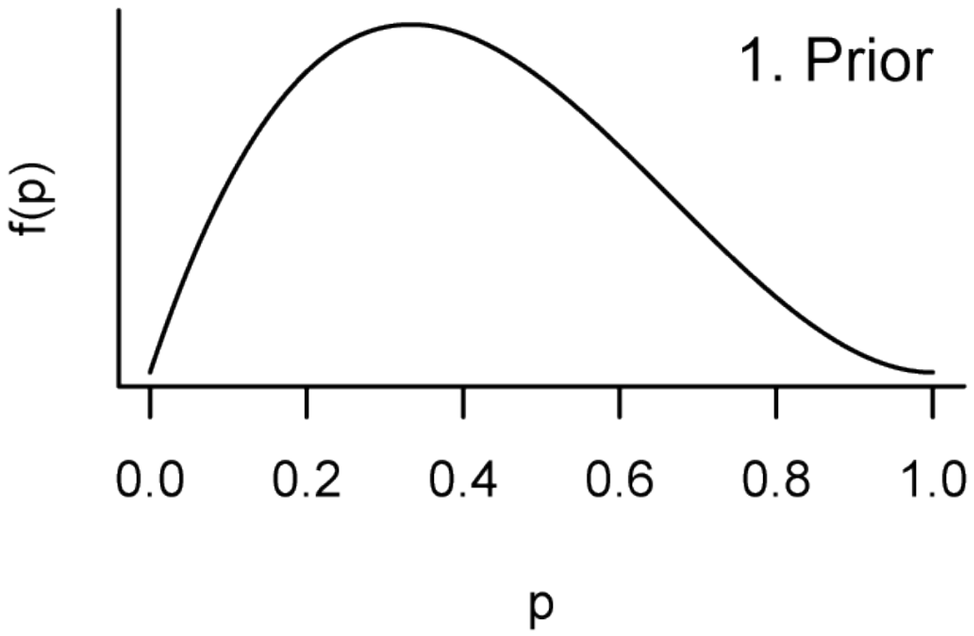
$$p(\theta \mid x) = \frac{p(\theta)p(x \mid \theta)}{p(x)}$$

- Posterior probability is what we want — probability hypothesis is true!
- With increasing data, likelihood term usually overwhelms the priors

Example: Lost Wallets

- Worked example from Wang (2010)
- Television program *Prime Time* did a hidden camera experiment to investigate whether police officers would return a supposedly lost wallet filled with cash.
- Parameter: p, the probability of **dishonest** officers

Wallet Prior



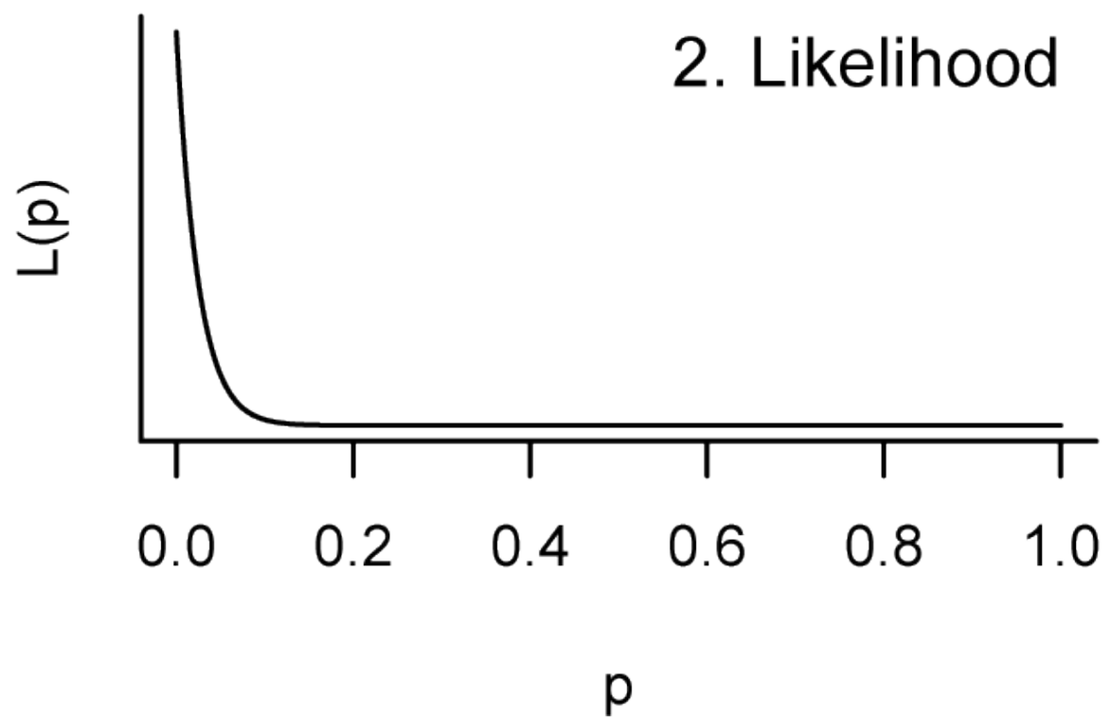
- Beta(2,3) for prior
- informative; could have chosen uniform

Wallet Likelihood

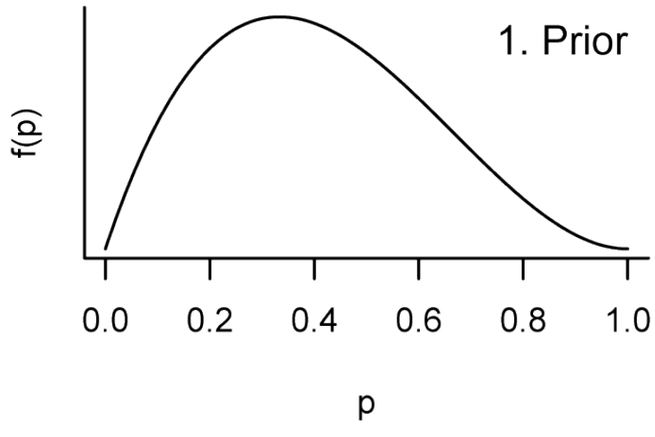
- The data: $x = 0$ dishonest cops out of $n = 40$

$$p(x \mid p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

- Likelihood computed from the binomial density function

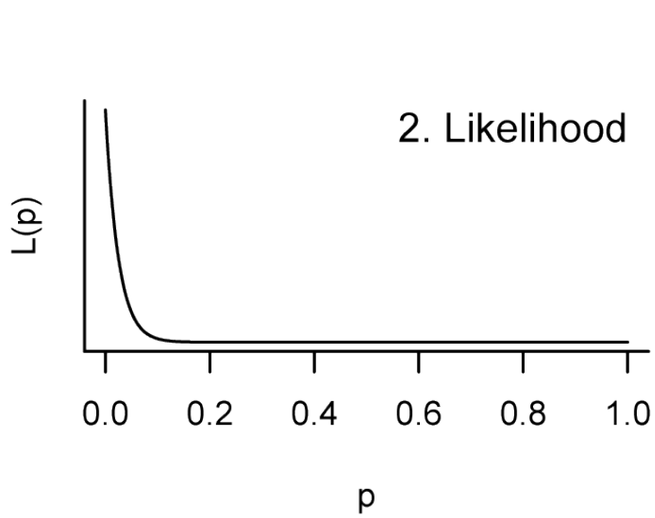


Wallet Posterior



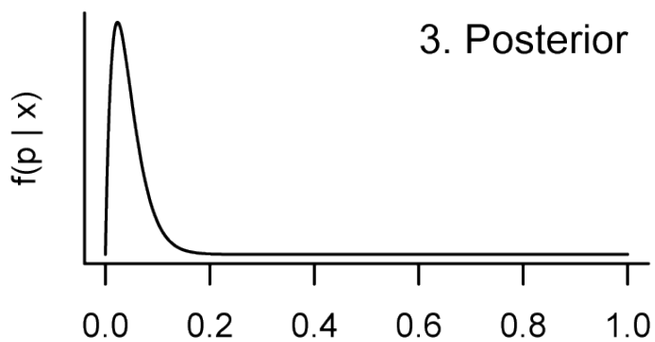
Beta(a, b)

Beta(2, 3)



Binom(p)

Binom(x = 0, n = 40)



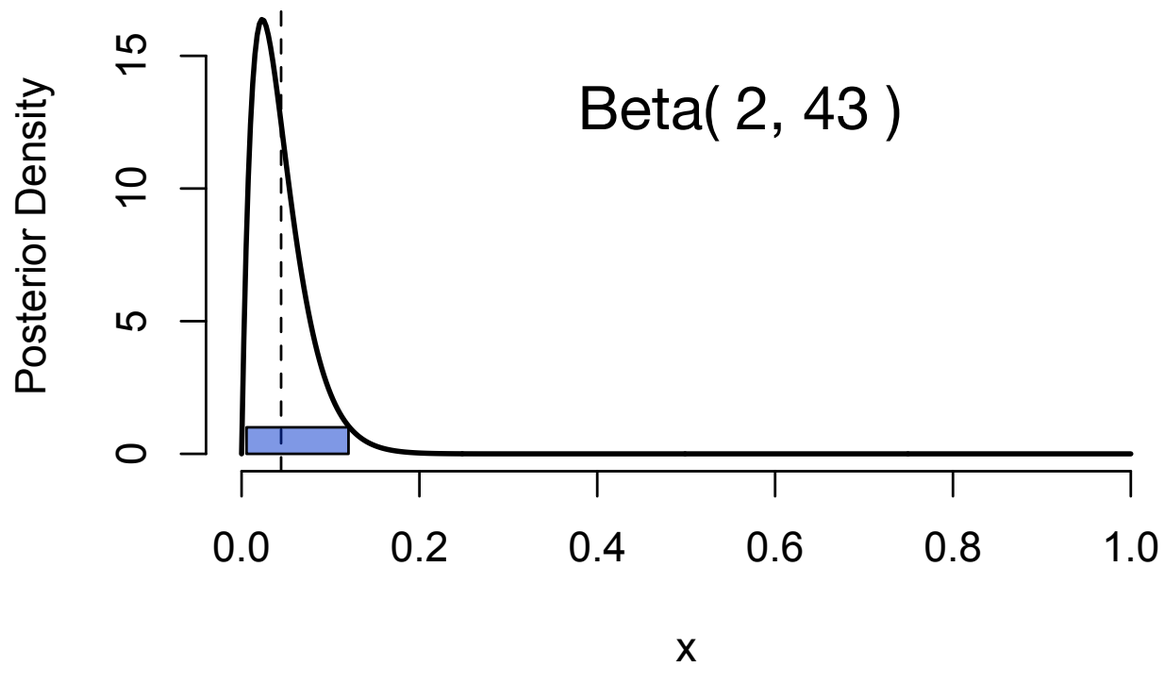
Beta(a + x, b + n - x)

Beta(2, 43)

Beta is the conjugate prior for Binomial data

Summarizing the Posterior

- Because known analytical form, we can compute useful summaries of the posterior
- Mean of posterior is taken as **point estimate** for parameter
- Mean of Beta(a, b) is a/(a+b) = 2 / (2 + 43) = 0.044
- Can compute 95% **credible intervals** from quantiles of Beta distribution




```
qbeta(p = c(0.025, 0.975), 2, 43)  # (0.005, 0.12)
```

Markov Chain Monte Carlo (MCMC)

- Wallet example was particularly simple; we know the analytical form of the posterior
- Usually, there are many variables and the posterior is analytically intractable.
- MCMC algorithms approximate the posterior distribution by generating samples from it.
- There are multiple MCMC algorithms available: Metropolis, Metropolis-Hastings, Gibbs sampler, Hamiltonian Monte Carlo…

Metropolis Algorithm

Simplified description of the Metropolis algorithm:

1. Start from an initial parameter value, θ_0

2. Propose to jump to a nearby position in parameter space, θ^*

3. Calculate the ratio, *r*, of the proposal posterior densities

$$r = \frac{p(\theta^* \mid x)}{p(\theta^t \mid x)}$$

4. If $r > 1$, accept the proposed jump. Otherwise accept with a probability *r*

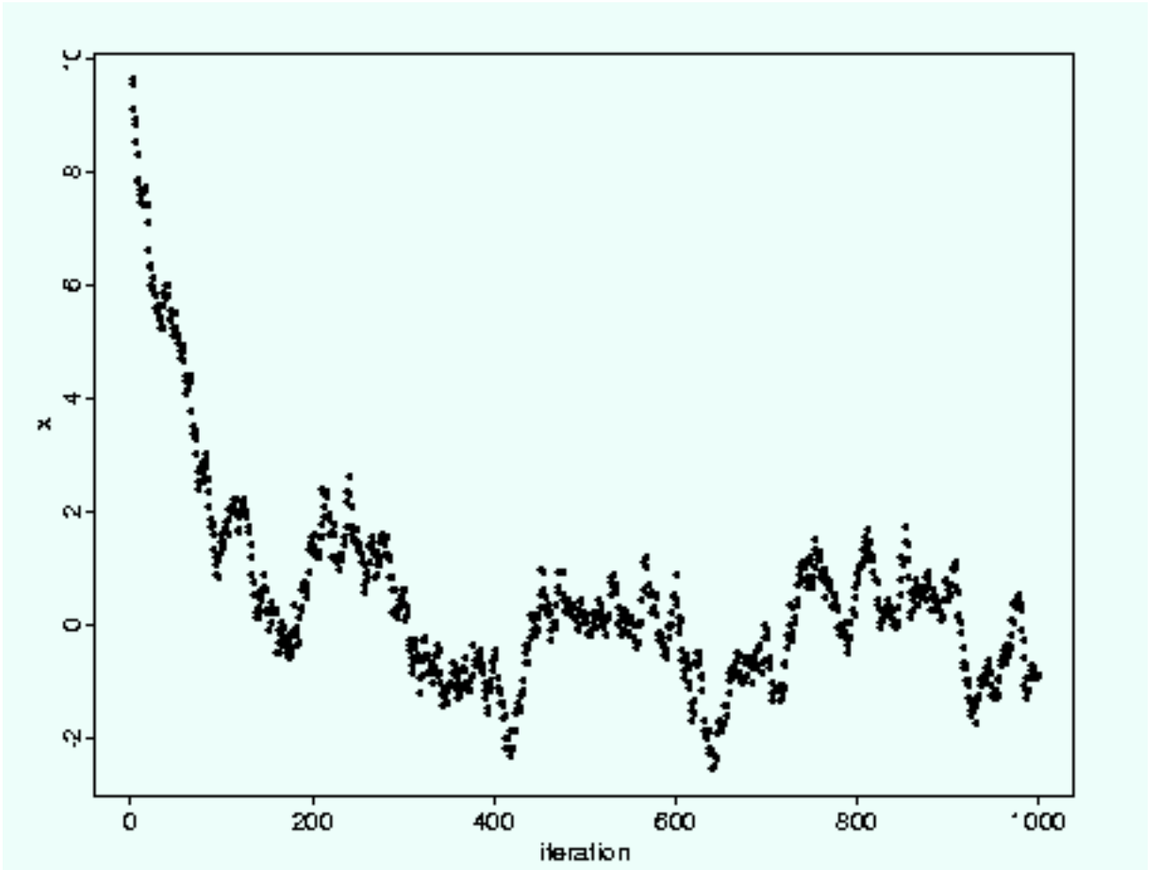
5. Go back to Step 2 and continue…

MCMC Algorithms

- Each sequence, called a **chain**, meanders through parameter space.
- It can be shown that these algorithms eventually converge to a stationary distribution that matches the posterior
- Some processing of the chain is required and there are diagnostics to assess convergence

Detecting Convergence

Trace Plot of a chain



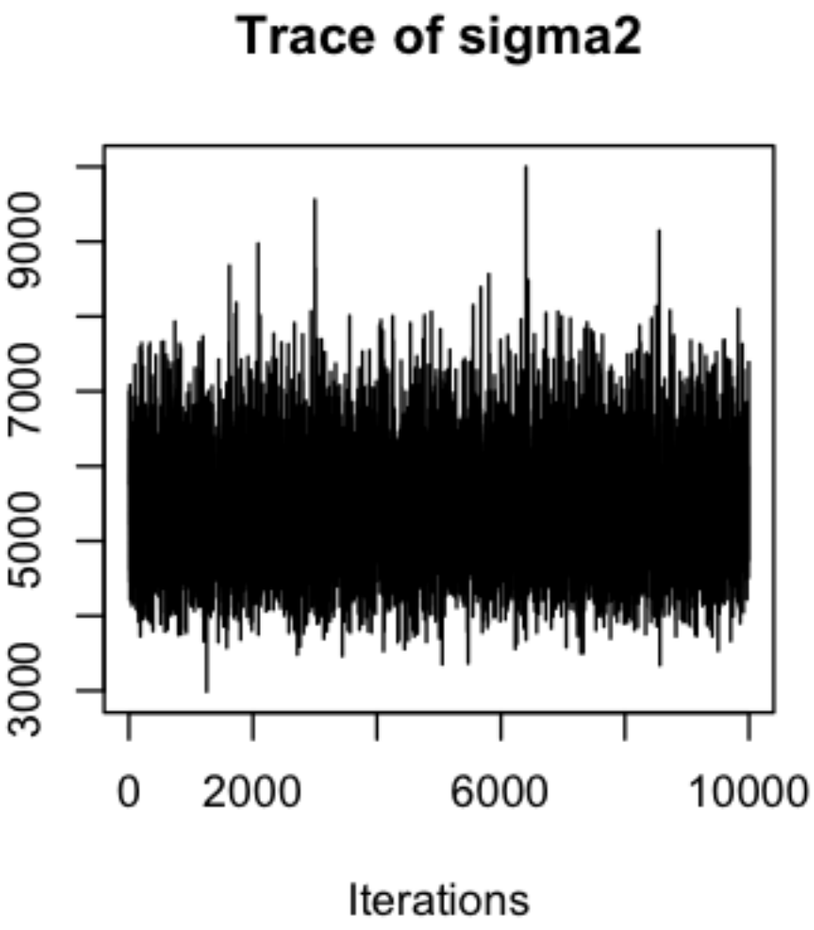
generation

Two issues

- it takes a while to get to the high probability area of parameter space: **burn-in** - discard some of the initial generations

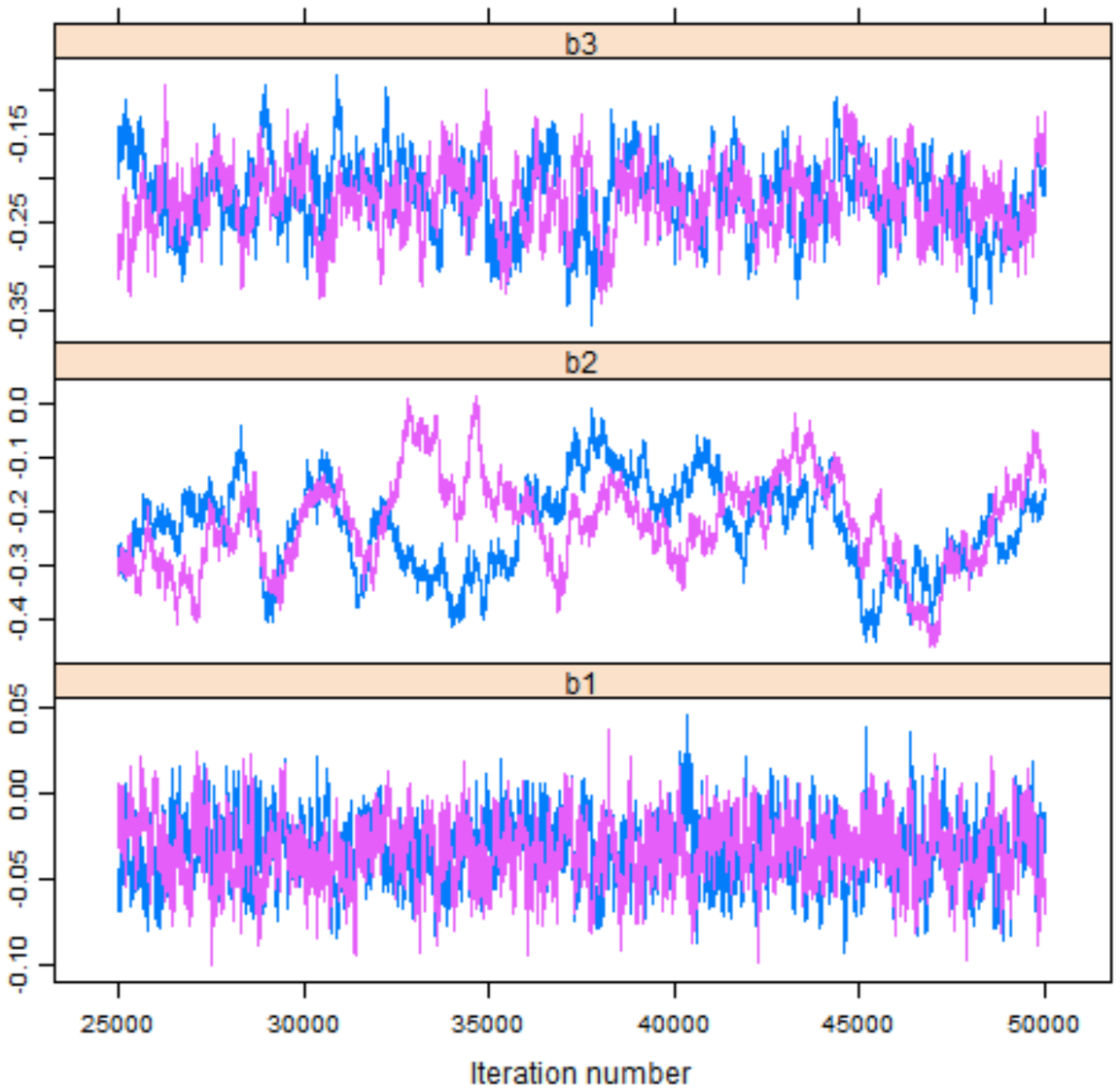
- Parameter values are serially correlated in the chain: **thinning** - sample every 1000 samples

Detecting Convergence



- example of good convergence ("fuzzy caterpillar")

Detecting Convergence



- good to run more than one chain to assess mixing — how well chains are exploring all of the space
- diagnostics
  - Effective sample size - reflects serial autocorrelation
  - Potential scale reduction factor (R) (Gelman & Rubin 1992) - reflects variation between and within chains

Summarizing MCMC Posteriors

- With convergence, can easily compute natural summaries of the MCMC samples
  - point estimates: mean or median
  - credible intervals (also called Highest Posterior Density [HPD] intervals): from the quantiles, i.e., middle 95% of parameter values from the chain(s)

Bayes Factors

- **Bayes Factors** are sometimes used to compare support for two models. It is the ratio of the **marginal likelihoods** of each model:

$$BF = \frac{p(x \mid H_2)}{p(x \mid H_1)}$$

- Similar to AIC scores, these measure support on a continuous scale, but there are rules of thumb for interpretation

Features of Bayesian Approaches

- Bayesian approaches are asymptotically consistent: they converge to truth with increasing *n*
- Bayesian approaches are particularly strong with complex models with many parameters
- They can naturally handle many sorts of uncertainty that are ignored in other analyses.
- If one goes along with subjective probability, Bayesian interpretations are more straightforward, e.g., credible intervals versus confidence intervals

$$p(\theta \mid x) = \frac{p(\theta)p(x \mid \theta)}{p(x)}$$

How To Do Bayesian Analysis

- MCMC methods often have to be tailored to specific problems to get good performance of the MCMC
- Specialized-purpose programs, such as BEAST, MrBayes, and PyRate, are therefore common
- General purpose Bayesian tools, where users can specify models broadly, often use R as a interface and glue (manage data, make plots, etc.) but use a more specialized language for the MCMC (BUGS, JAGS, Stan, etc.)
- There are some pure R implementations; we'll use one, the `MCMCpack` library, for the exercises

Exercise 1. Bayesian Analysis using MCMCpack

1. Install MCMCpack and load it as `library(MCMCpack)`. This package supports several kinds of models, but we will focus only on regression models as fit by `MCMCregress()`. It relies on the `coda` package for functions analyzing the Markov chain. For data, we'll return to the `cope` dataset we analyzed using regression on the first day, so go ahead and import those data again.
2. We'll start with the simple bivariate model `valve.length ~ mg.temp`. Start with a simple analysis, using mostly the defaults: `m1a <- MCMCregress(valve.length ~ mg.temp, data = cope)`. This object (`m1a`) contains the set of parameter values from when the Markov chain meanders through parameter space. To see what this looks like, view the beginning of this object with `head(m1a)`. Next look at trace plots with `plot(m1a)` - you may need to resize your plot window to get a good look. Are these trace plots consistent with convergence?
3. Make another chain for this very same model; call it `m1b` and then combine it with the first chain as `m1ab<- mcmc.list(m1a, m1b)`. Plot this combined pair of chains and check if the chains look well-mixed.
4. To keep things simple, go back to the first chain and do `summary(m1a)`. Focus for now on the information at the top - can you tell how many MCMC samples are in the chain, and how much burn-in was used? Now, use the `effectiveSize` function from `coda` to assess autocorrelation. Strong autocorrelation can make the effective sample size much lower than the number of MCMC samples. Does autocorrelation seem to be a problem here?

5. Finally, let's look at the information about the coefficients from `summary(m1a)`. From this output, identify point estimates for the parameters and their 95% credible intervals. Go back and do the same regression using `lm()` and then use `confint()` to compute 95% confidence intervals. Compare these to their counterparts you just got via `MCMCregress`.

6. We want to use Bayes Factors to compare the support for the two-predictor model, `valve.length ~ mg.temp + depth`, to that of the one-predictor model we have been using. However, we need to specify different priors because the default priors are *improper* (they are not a true probability density function because they do not integrate to 1). I've worked out some reasonable weakly informative priors by examining plots of the priors. The details are too involved to warrant getting into, but they are listed in the answers to this exercise. Check them out and run the code you see there. Then, do `summary(BF)` to see what interpretation these Bayes Factors lead to. Are interpretations based on classical statistics, AIC, and Bayesian approaches all consistent?

(generated from group discussion)

Approach	Benefits	Drawbacks
Frequentist	familiar, easy little specialized skill required less model dependence?	can only reject, can't support hypothesis over-focus on p-values alpha threshold problem
Likelihood + AIC	likelihood = evidence can compare models on equal footing focus on parameters	model dependence can fall into threshold trap AIC sometimes overfits
Bayesian	likelihood = evidence can compare models on equal footing good with complex models good with uncertainty focus on parameters straightforward interpretation	analyses often takes longer hard for non-specialists to understand model dependent complex sometimes no basis for priors some dislike subjective probability

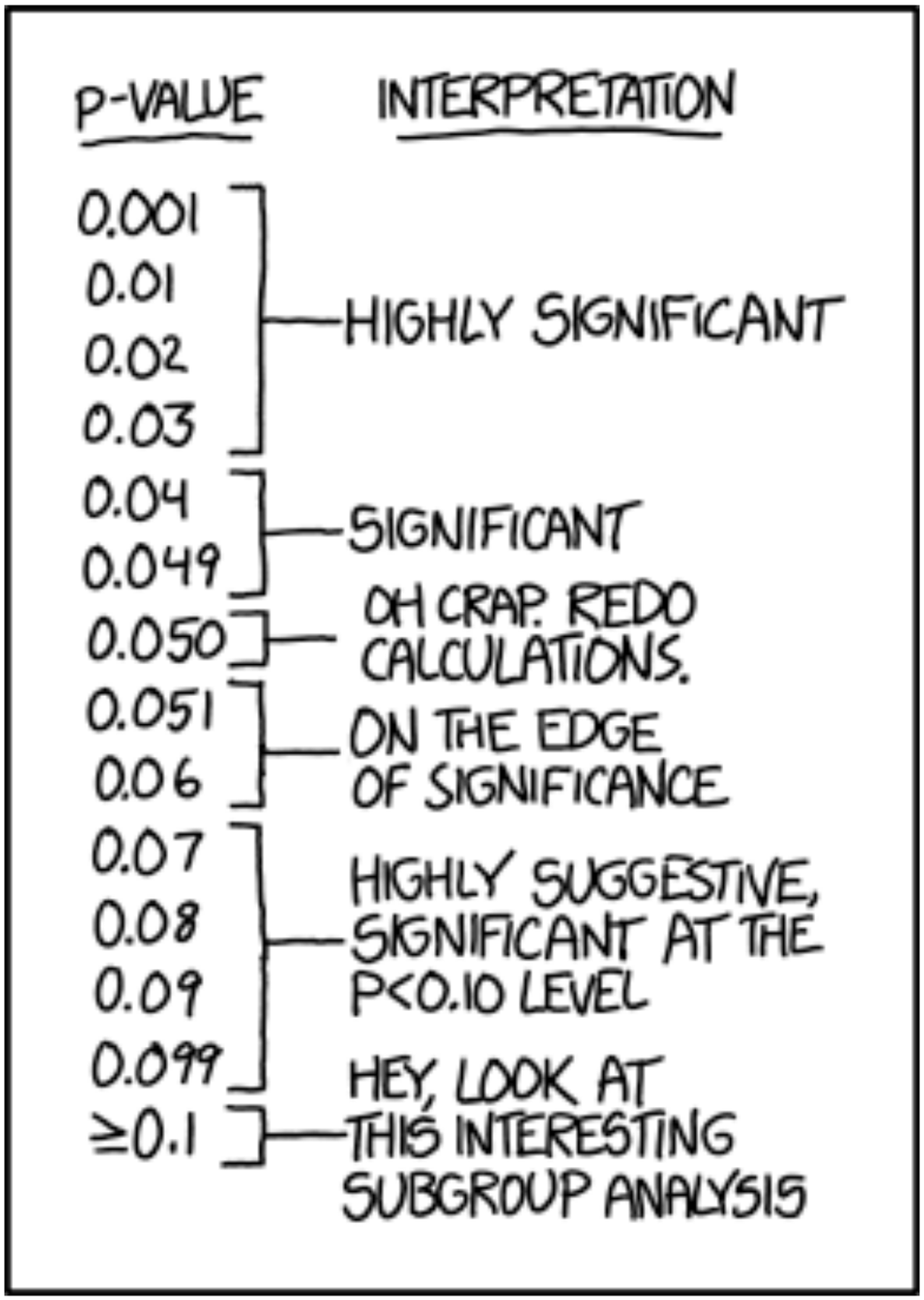
Outline

- Preliminaries: Probability and Samples
- Classical (frequentist) approaches
- Likelihood
- Bayesian Approaches
- P-hacking and Replication Crises

Discuss P-hacking papers

Head et al. (2015) The extent and consequences of P-hacking in science. PLoS Biol 13(3).

Baker (2015). Over half of psychology studies fail reproducibility test. Nature (News).


P-VALUE
INTERPRETATION
0.001
0.01
0.02
0.03
HIGHLY SIGNIFICANT
0.04
0.049
SIGNIFICANT
0.050
OH CRAP. REDO CALCULATIONS.
0.051
0.06
ON THE EDGE OF SIGNIFICANCE
0.07
0.08
0.09
HIGHLY SUGGESTIVE, SIGNIFICANT AT THE P<0.10 LEVEL
0.099
HEY, LOOK AT THIS INTERESTING SUBGROUP ANALYSIS
≥0.1

xkcd.com

Causes of P-hacking

Statistical hypothesis tests assume that everything about the test is determined ahead of time

- 1 vs. 2 tails, α
- which variables to analyze, what transformations, if any, to use
- what groupings to apply, subgroups to test
- when to stop collecting data
- which cases to include (no post-test dropping outliers)

If you have lots of separate tests, should apply a correction for multiple comparisons (Bonferroni) - remember, with $\alpha = 0.05$, expect a false positive every 20 tests.

Theory is your friend! Focus on few variables that have sound scientific reasons for considering; these also probably more likely to have real effects

Exploratory Analyses are Fine!

It is totally fine to do all those things on the previous slide as part of **exploratory analysis** — as long as it is reported openly.

This is important for hypothesis generation, but one should then collect new data to test these hypotheses in a **confirmatory analysis**

Exercise 2. P-hacking

1. Let's make some fake data with *n* = 100. Use `rnorm()` to generate a y variable and at least three x variables. Then, make two or more grouping variables by sampling with replacement a vector of labels, such as `g1 <- sample(c("A", "B"), 100, replace = T)` and `g2 <- sample(c("D", "E", "F", "G"), 100, replace = T)`. Combine these all into a data frame.
2. Your P-hacking goal: find significant relationships between y and any of the x or grouping variables. You can just start doing random tests, but it is probably helpful to make some plots first. Go ahead and make plots of y versus each x, with colors and plotting symbols varying according to grouping variables. This should help you find subsets of the data that are particularly "interesting."
3. Another strategy is to start with a full regression model, `lm (y ~ x1 + x2 +x3 + g1 + g2)`, and then probe the variables that have the most significant coefficients.
4. Allow yourself to do any of the P-hacking approaches we talked about, including doing lots of tests, omitting "outliers", and especially, excluding, combining or splitting data according to groups.
5. In some ways, P-hacking these data will be harder than real data because these simulated data are so homogenous. Real datasets will have more heterogeneity from unmeasured variables. In the answers, I implemented a simple way to add heterogeneity to the y variable that mimics the data being drawn from two subgroups, such as clades. You can check out my P-hacking results there, and you can replicate them because I used `set.seed()` to make the random parts repeatable.

Apply What You've Learned

Your Challenge: take any aspect of what we've covered and apply it to a question, problem, or dataset of interest to you.

Possibilities:

- Do a resampling-based test in place of a standard parametric test that you have done
- Do simulations to assess power of some test of interest to you. For example, you could explore the sample sizes you will need to have reasonable power in your own work
- Explore MCMCpack, for example, by playing around with different priors, or just applying it to do regression on a dataset of your own.
- Others…

Exercise answers follow

Exercise Answers

```
## MCMCpack exercises (Ex. 1)

 ## Ex. 1
 library(MCMCpack)
 cope <- read.table(file = "cope.txt", header = T)

 # Ex. 2
 # simple regression
 m1a <- MCMCregress(valve.length ~ mg.temp, data = cope)
 head(m1a)
 plot(m1a)    # yes, consistent with convergence


 # Ex. 3
 m1b<- MCMCregress(valve.length ~ mg.temp, data = cope)
 m1ab<- mcmc.list(mod1, mod1b)
 plot(m1ab)

 # Ex. 4
 summary(m1a) # 10,000 samples, 1,000 discarded as burn-in
 effectiveSize(m1a)    # all near 10,000 so basically no autocorrelation

 # Ex. 5
 summary(m1a)
 w1 <- lm(valve.length ~ mg.temp, data = cope)
 summary(w1)
 confint(w1)  # point estimates and CIs are nearly identical

 # Ex. 6
   b0 <- c(700, 0)
   B0 <- matrix(c(1e-6, 0, 0, 1e-4), 2,2)
   c0 <- 2
   d0 <- var(cope$valve.length)
   mbf1 <- MCMCregress(valve.length ~ mg.temp, data = cope, b0 = b0,
                       B0 = B0, c0 = c0, d0 = d0, marginal.likelihood = "Chib95")

   b0 <- c(700, 0, 0)
   B0 <- matrix(c(1e-6, 0, 0, 0, 1e-4, 0, 0, 0, 1e-4), 3,3)
   c0 <- 2
   d0 <- var(cope$valve.length)
   mbf2 <- MCMCregress(valve.length ~ mg.temp + depth, data = cope, b0 = b0,
                       B0 = B0, c0 = c0, d0 = d0, marginal.likelihood = "Chib95")
   BF <- BayesFactor(mbf1, mbf2)
```

Exercise Answers

```
## P-hacking exercises (Ex. 2)

 # 1 generate data set n = 100, rnorm() with several grouping variables, Use set.seed
 # to make repeatable
  set.seed(1)
  N1 <- 30   # say, y data come from two clades -- adds heterogeneity to data (but no relationship to x vars)
  N2 <- 70
  NN <- 100
  y <- c(rnorm(N1, 50, 1), rnorm(N2, 52, 1))
  x1 <- rnorm(NN, 25, 1)
  x2 <- rnorm(NN, 100, 1)
  x3 <- rnorm(NN, 100, 1)
  g1 <- sample(c("A", "B"), size = NN, replace = T)
  g2 <- sample(c("D", "E", "F"), size = NN, replace = T)
  X <- data.frame(y, x1, x2, x3, f1 = factor(g1), f2 = factor(g2))

 # 2 P-hack away -- find significant relationships!
 # first look visually
 fn1 <- as.numeric(X$f1)  # useful for plotting colors by group
 fn2 <- as.numeric(X$f2)

 plot(y ~ x1, data = X, pch = g1, col=fn1)  # looks like group B has relation
 w <- lm(y ~ x1, data = X, subset = f1 == "B")  ## P = 0.00079
 plot(y ~ x1, data = X, pch = g2, col = fn2)
 plot(y ~ x2, data = X, pch = g1, col = fn1)
 plot(y ~ x2, data = X, pch = g2, col = fn2)
 plot(y ~ x3, data = X, pch = g1, col = fn1)
 plot(y ~ x3, data = X, pch = g2, col = fn2)

 # another approach, start with full model, explore variables with low P
 wfull <- lm(y ~ x1 + x2 + x3 + f1 + f2, data = X)  # x1 and x3 have P < 0.05
 w <- lm(y ~ x1)  # P = 0.02
 w <- lm(y ~ x3)  # P = 0.04 - can we get this even lower?
 plot(y ~ x3, data = X, pch = g1, col=fn1)  # maybe stronger in group A?
 w <- lm(y ~ x3, data = X, subset = f1=="B")  # hmm, no, P = 0.07

 # Well, maybe x2 is not important on its own, but it is relative to x1 (x2/x1)
 X$x2.x1 <- X$x2 / X$x1  # x2/x1
 w <- lm(y ~ x2.x1, data = X)  # just what I thought - P = 0.01
 plot(y ~ x2.x1, data = X, pch = g1, col = fn1) # you know, pattern looks even stronger in grp B
 w <- lm(y ~ x2.x1, data = X, subset = f1 == "B")  # P = 0.00029; wow, this must be a robust finding
```