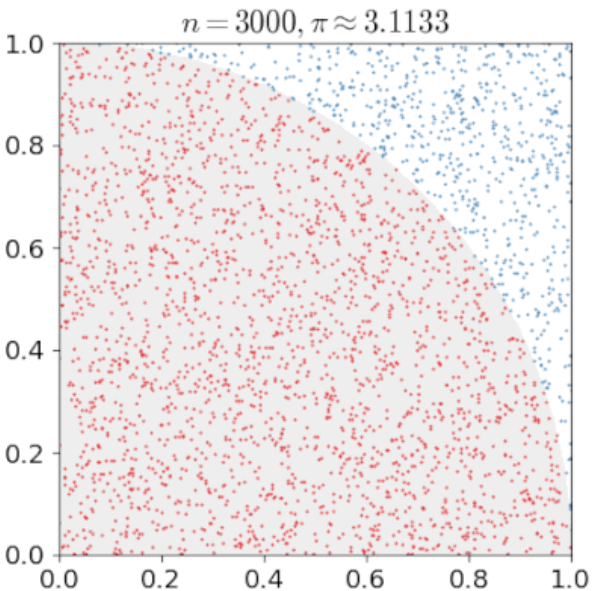
Approximate computation: Monte Carlo

Roger Levy
9.S918: Quantitative inference in brain and cognitive sciences

24 February 2025

Monte Carlo methods, or "probabilistic simulation"

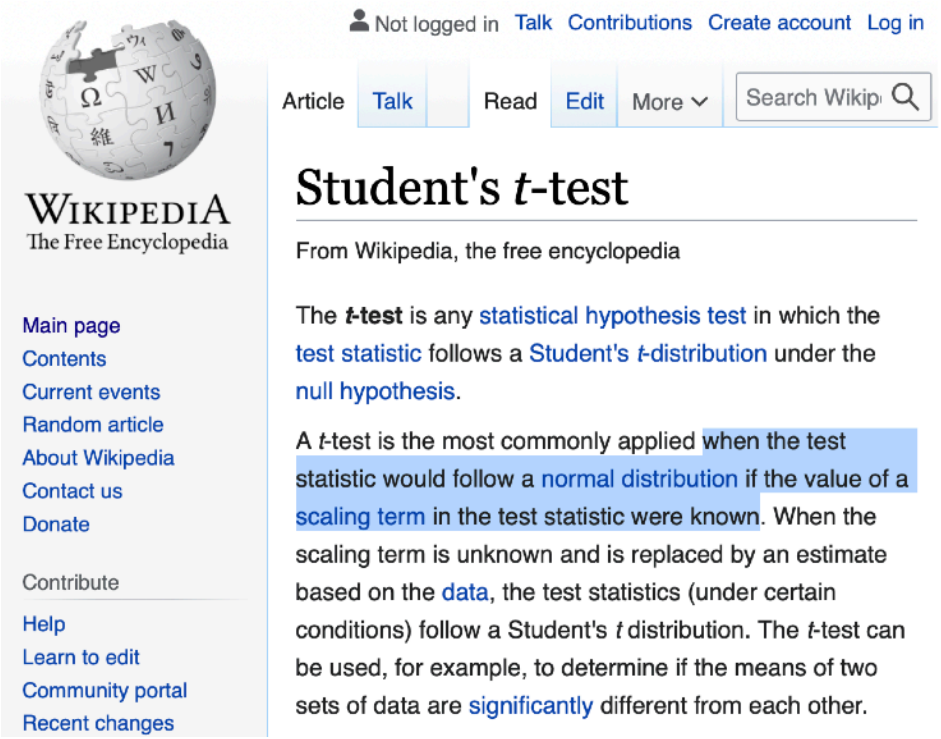
- Generally speaking:
  1. Define a domain of possible inputs
  2. Generate n iid random inputs from a probability distribution on the domain
  3. Perform a deterministic computation on each randomly generated input
  4. Aggregate the results of the deterministic computation
- As n grows larger, the simulated result approaches the true value



https://en.wikipedia.org/wiki/Monte_Carlo_method

Simple example of Monte Carlo

- Suppose I want to do a two-sample *t*-test but my data aren't normally distributed



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit More Search Wikip

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

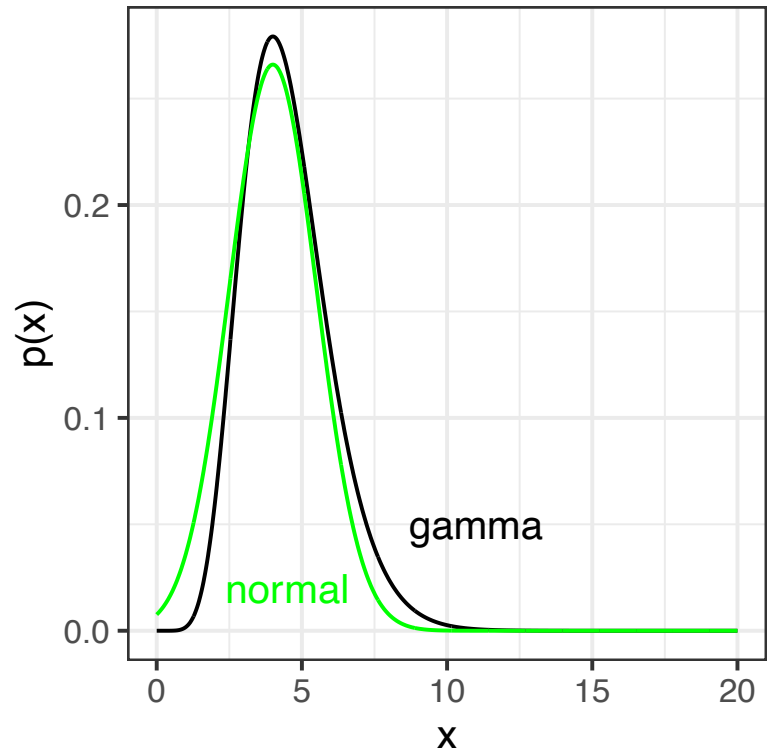
Contribute
Help
Learn to edit
Community portal
Recent changes

Student's *t*-test

From Wikipedia, the free encyclopedia

The ***t*-test** is any statistical hypothesis test in which the test statistic follows a Student's *t*-distribution under the null hypothesis.

A *t*-test is the most commonly applied when the test statistic would follow a normal distribution if the value of a scaling term in the test statistic were known. When the scaling term is unknown and is replaced by an estimate based on the data, the test statistics (under certain conditions) follow a Student's *t* distribution. The *t*-test can be used, for example, to determine if the means of two sets of data are significantly different from each other.



- **How bad will this be for my *t*-test????**

Monte Carlo, in action

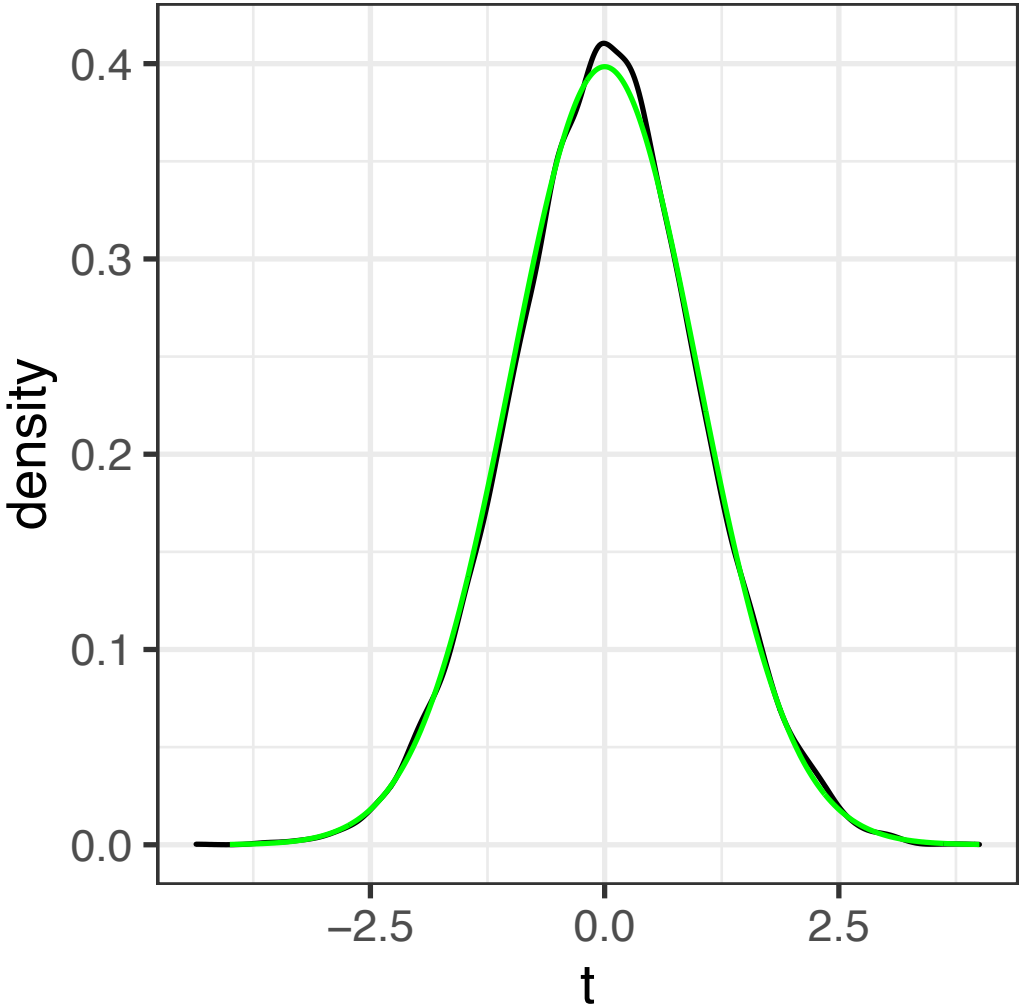
```r
library(ggplot2)
library(tidyverse)

# Manually compute Student t-statistic
f <- function(seed,N=100,shape=9,scale=0.5) {
  set.seed(seed)
  y1 <- rgamma(N,shape=shape,scale=scale)
  y2 <- rgamma(N,shape=shape,scale=scale)
  s_p <- sqrt( (var(y1) + var(y2)) / 2 )
  t_statistic <- ( mean(y1) - mean(y2) ) / ( s_p*sqrt(2/N ) )
  return(t_statistic)
}

N <- 100
Ts <- sapply(1:10000,f)

t_reference <- tibble(x=seq(-4,4,by=0.01),t=dt(x,df=2*(N-1)))

ggplot(data=tibble(t=Ts),aes(x=t)) +
  geom_density() +
  geom_line(data=t_reference,aes(x=x,y=t),color="green",linetype="dashed")
```

Reproducibility!

Monte Carlo simulation

Compare against Student's *t* distribution

Monte Carlo, in action



- The *t* distribution is still a pretty good approximation of the distribution of the *t* statistic, even when the underlying distribution is gamma!
- This exemplifies what is meant when people say that the *t* test is **robust to deviations from normality**

Unnormalizable posteriors

- Our motivation: Bayesian posterior inference

Observed data

$$P(\theta \mid \mathbf{y}, I) = \frac{P(\mathbf{y} \mid \theta, I) P(\theta \mid I)}{P(\mathbf{y} \mid I)}$$

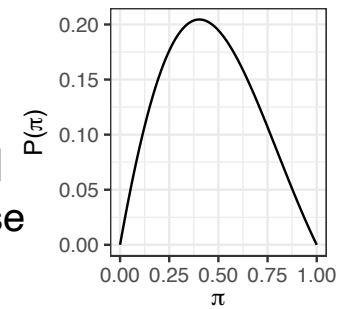
Model parameters

Background knowledge

- Sometimes $P(\mathbf{y} \mid I)$ can't be calculated exactly. Example

Bernoulli data with non-conjugate prior:

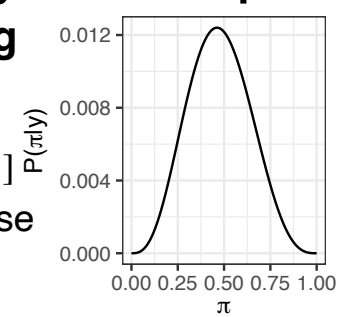
$$P(\pi) \propto \begin{cases} \pi(1-\pi)e^{-\pi^2} & \pi \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$



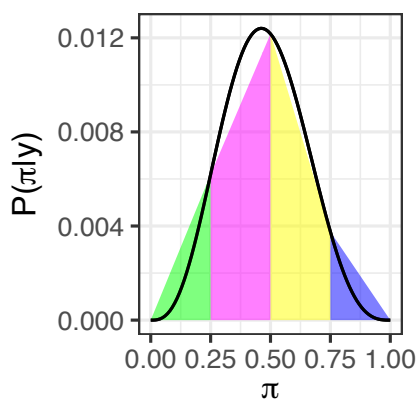
Posterior after observing 2 heads, 2 tails:

$$P(\pi) \propto \begin{cases} \pi^3(1-\pi)^3 e^{-\pi^2} & \pi \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

No closed form!



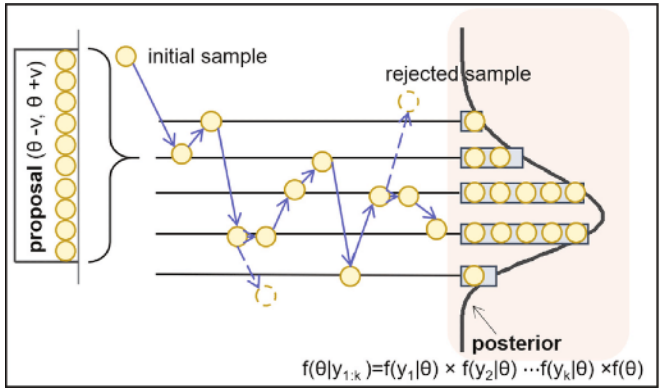
- In simple cases like this, we can numerically approximate the integral:



- But in high dimension and/or unbounded ranges, difficult or even impossible!

Markov chain Monte Carlo

- However, we can often **take samples from the posterior** even when we can't compute normalized probabilities
- One general and widely used approach: **Markov chain Monte Carlo (MCMC)**
- MCMC is a mathematically principled random walk on a non-negative function, directed toward regions where the function takes on a larger value



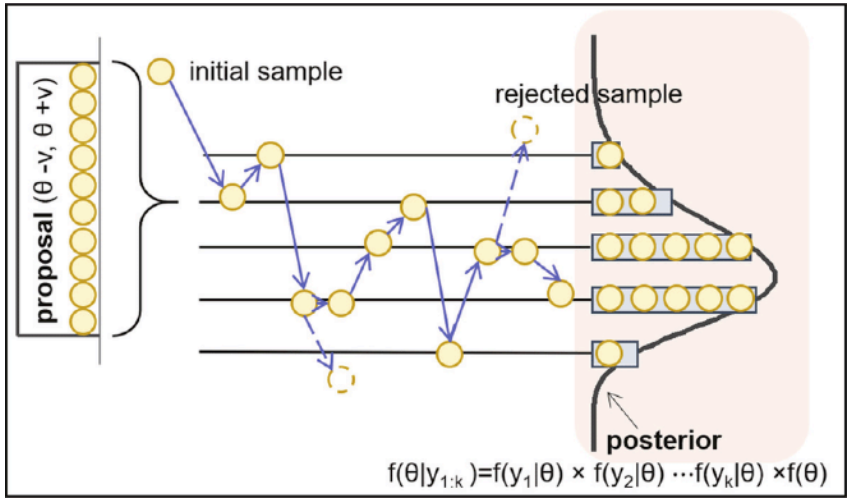
- Asymptotically, the random walk gives us samples from in proportion to the height of the function

Image credit: Dong et al., 2019, http://dx.doi.org/10.5772/intechopen.82781

MCMC for posterior sampling

- We use the *unnormalized* form of the posterior:

$$P(\theta \mid \mathbf{y}, I) \propto P(\mathbf{y} \mid \theta, I) P(\theta \mid I)$$

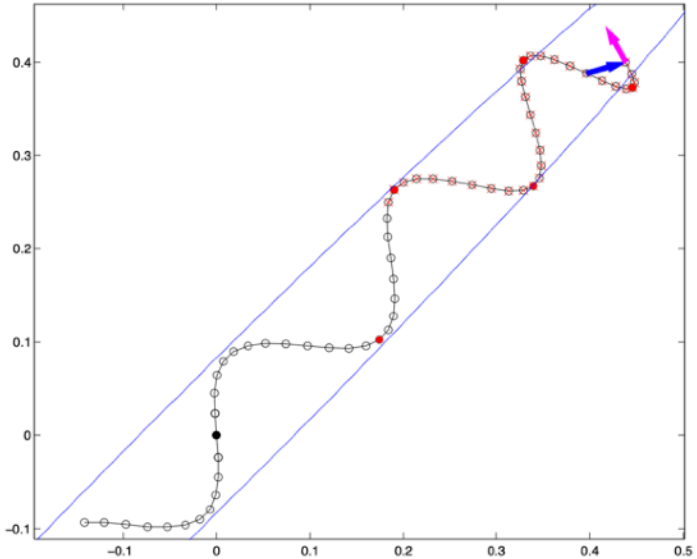


- We run MCMC and then treat the chain of values as samples from the posterior
- The full set of samples is not iid (nearby values on the chain are correlated), but methods exist for estimating "effectively" how many independent samples we have

Image credit: Dong et al., 2019, http://dx.doi.org/10.5772/intechopen.82781

Stan, HMC, and NUTS

- There are many different MCMC algorithms (e.g., Metropolis, Gibbs Sampling)
- We will use the probabilistic programming language **Stan** for Bayesian inference about model parameters
- Stan uses an algorithm called **Hamiltonian Monte Carlo** (**HMC**) with the **No U-Turn Sampler (NUTS)**



(Hoffman & Gelman, 2014)

- This algorithm tends to be particularly efficient for many problems we'll face

Bayesian posterior inference with Stan

1. Define the generative model you assume underlies the data you want to analyze
2. Choose a prior distribution for your model parameters θ
3. Encode the model structure and prior in a Stan program
4. Provide the data Y you want to analyze, and ask Stan to sample from the posterior $P(\theta \mid Y, I) \propto P(Y \mid \theta) P(\theta \mid I)$ (often written as $P(\theta \mid Y) \propto P(Y \mid \theta) P(\theta)$, i.e. eliding I)