Unit Schedule: Modules

Module	Week	Content	Ross
1.	1	introduction to modelling	1,2
2.	2	probability refresher	3
	3	random vars & expected values	4
	4	special distributions	5
3.	5	statistical inference	6&7
	6	confidence intervals	7
	7	hypothesis testing	8
4.	8	dependence & linear regression	9
	9	classification, clustering & mixtures	
5.	10	**random numbers & simulation**	15(bits)
	11	basic machine learning	
6.	12	modelling, validation & review	

Revision at *https://flux.qa/43FMK4*

FIT5197 Statistical Data Modelling

Module 5

Simulation

2020 Lecture 10

Monash University

Simulation and Sampling

(ePub sections 5.5)

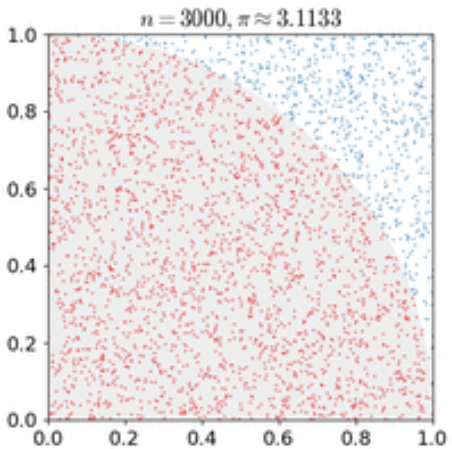
Outline

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

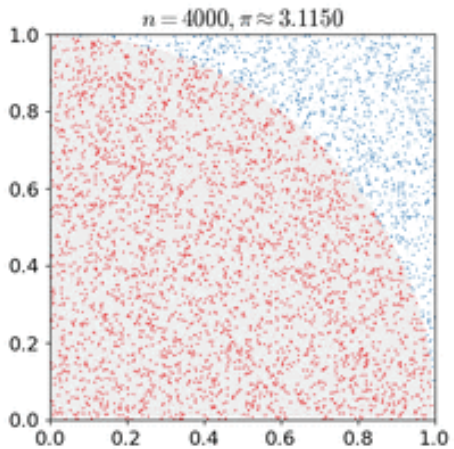
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

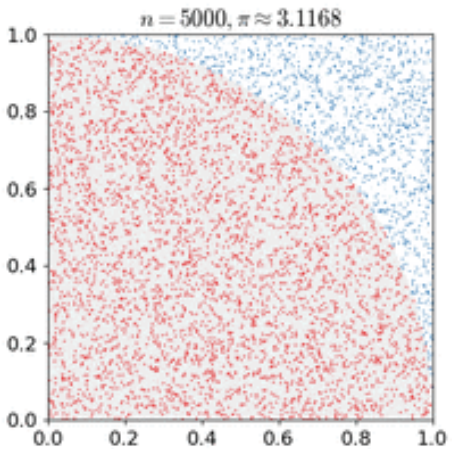
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

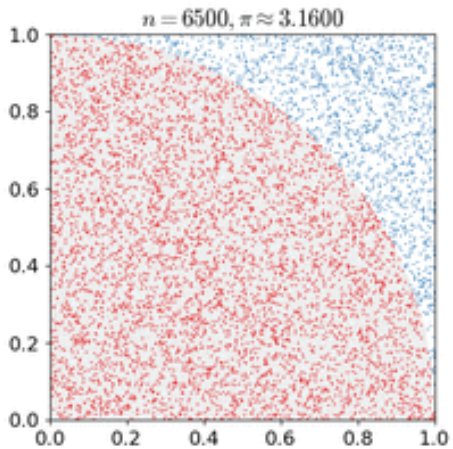
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

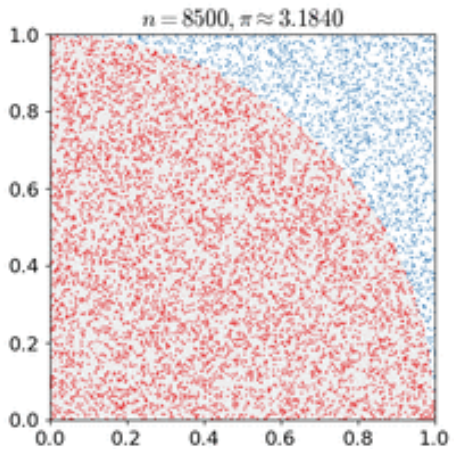
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

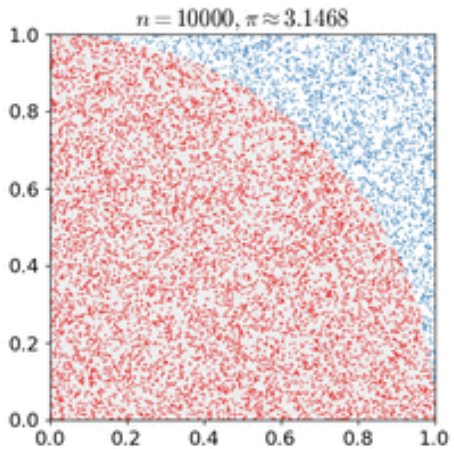
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

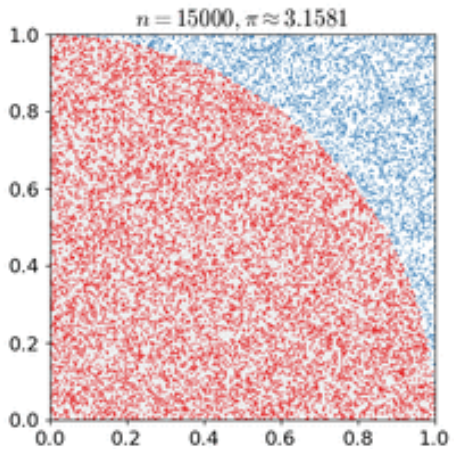
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

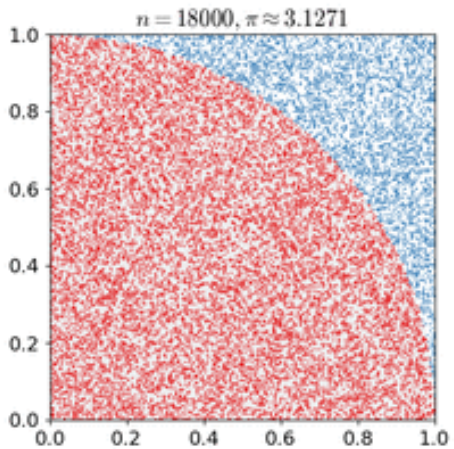
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

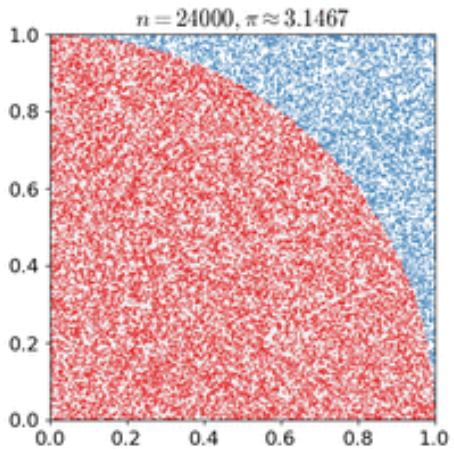
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

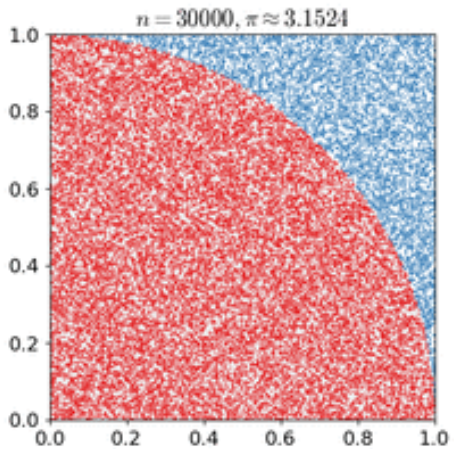
- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

Monte Carlo Algorithms

Monte Carlo Casino

A. Prévot, Tumblr 2013 (CC BY-SA 2.0)

Estimating Pi with MC



nicoguaro, Wikipedia 2017 (CC BY 3.0)

- *Monte Carlo (method)* is the use of randomness to solve problems, referring to the Monte Carlo casino
- often done with simulation, encoding physical laws with some randomness
- Examples:
  - weather prediction
  - approximate integrals
  - test game-playing
  - evaluate theories in biology
  - test systems in engineering (automotive, construction, electronics, etc.)

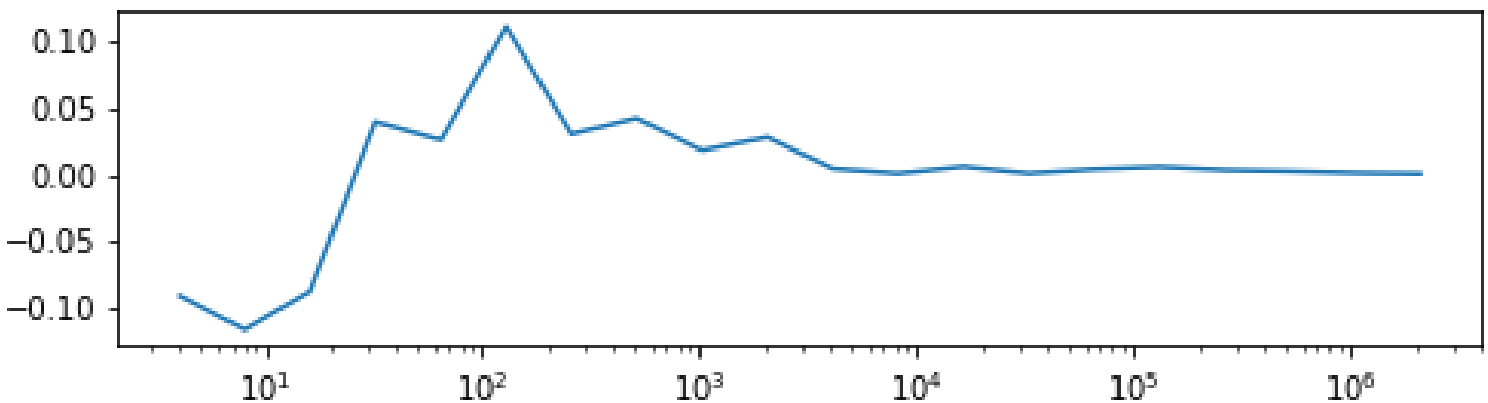
Monte Carlo Integration

- have a function $f(x)$ in $\mathcal{R}$ on domain A, and wish to approximate $\int_A f(x)\mathrm{d}\,x$
  - similar for the multivariate case, but quality/convergence deteriorates for higher dimensions
- approximate by

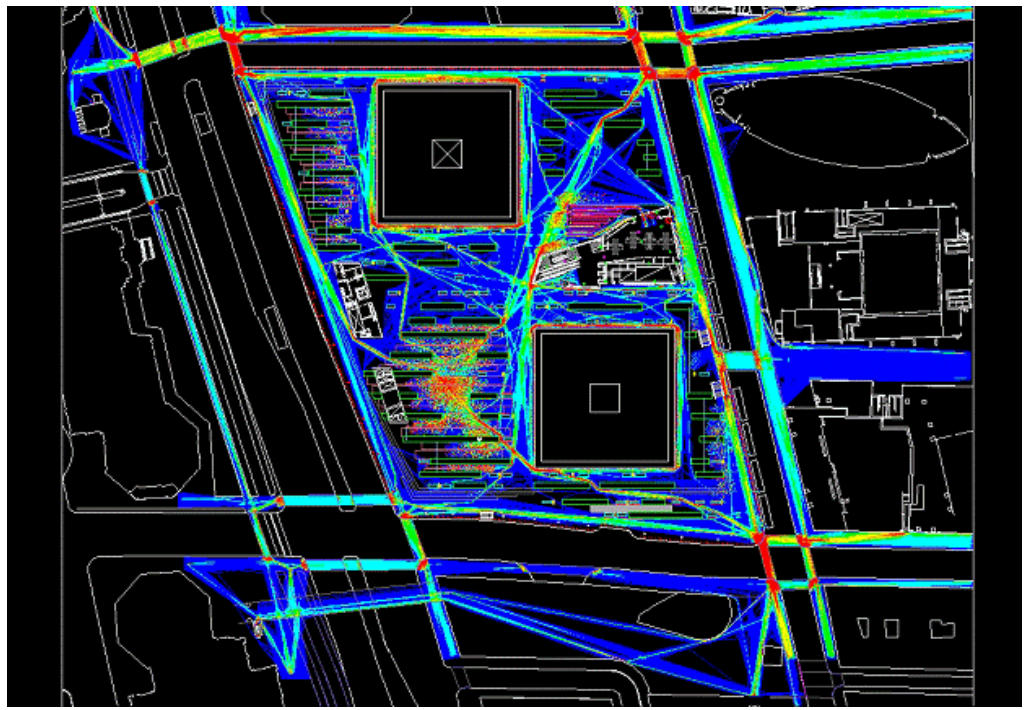
$$\int_A f(x)\mathrm{d}\,x \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

where x_i are N uniform random samples from A

e.g. use $f(x) = \sin x$ for $A = [-\pi, \pi]$, N to log scale on X-axis



Modelling Pedestrian Traffic



By M.F. Monteleone [Public domain], via Wikimedia Commons

- have individual models of pedestrians with different objectives
- what does overall traffic look like?

Pedestrian Traffic, cont.

- cannot just create a pedestrian traffic snapshot from the behavioural models
- because **behaviours interact** (avoiding blockages, gathering around interest areas, etc.)
  - same for automobile traffic simulation
  - same for species mating/growth prediction
  - same for neighbouring neurons when training neural networks
- therefore must run system for some time and watch range of behaviours
- this is called a simulation

Markov Chain Model

a *Markov chain* model uses randomness and a state-change (i.e., behaviour) model to generate a chain of states:

- $S_1, S_2, ...S_n, ...$
- usually the state at time $n+1$ is dependent on just the state at time n
- the model is specified as $p(S_{n+1}|S_n)$

e.g. watch this video modelling *"bacterial growth in a Petri dish"*

Markov Chain Model

combining Markov chains with Monto Carlo methods gives us *Markov chain Monte Carlo* (MCMC)

- general class of algorithms used in machine learning and statistics
- handles the case, especially deep learning, where older statistical theory doesn't yield "clean" solutions
  - Gaussian or Poisson data have simple to compute MLE solutions
  - deep neural networks and recommender systems models don't have simple to compute MLE solutions

Outline

Random Numbers

pseudorandom number generator (PRNG)

- the use of a computer function to generate a sequence of numbers "that appear to be random" in simulation and cryptography
- PRNGs are computer functions, they are deterministic
- after a (very) long sequence the same sequence will repeat
- a *seed* is used to initialise a PRNG to duplicate exactly the same sequence of random numbers on a single threaded process
  - see `set.seed()` in R or `random.seed()` in Python
  - lets you duplicate an entire simulation if used right
- the digits of
  π *might look random but it's full of hidden patterns*

PRNGs, Examples in R

```
> rbinom(100,1,0.5)
  [1] 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 0
 [38] 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0
 [75] 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1
> rbinom(100,10,0.5)
  [1] 4 3 6 5 6 6 4 3 4 6 6 7 3 4 4 5 7 4 5 3 7 4 5 4 5 8 2
 [38] 4 7 3 4 5 4 5 7 3 5 7 6 2 7 8 6 8 5 4 4 6 4 8 5 7 4 3
 [75] 2 2 5 4 5 7 7 5 7 6 5 6 5 3 6 8 7 6 5 1 5 6 6 7 8 5
> runif(20,0,1)
 [1] 0.26992842 0.87449218 0.83997401 0.49518187 0.70944773
 [7] 0.38181808 0.78576018 0.33326881 0.27482388 0.43720669
[13] 0.74526744 0.04861203 0.89959392 0.27030836 0.40687984
[19] 0.47453052 0.85392008
```

Linear Congruential Generator

- the *linear congruential generator* is a PRNG built using number theory (i.e., the theory of integers)
  - widely used in earlier computer systems
- generate uniform random integer variates in the range $[0, m-1]$
- use $X_{n+1} = (aX_n + b) \bmod m$, where m is the module and a the multiplier
  - see Wikipedia page above for examples

e.g. $a = 5$, $b = 1$, $m = 16$, (seed) $X_0 = 1$

- $X_1 = (5 \times 1 + 1) \bmod 16 = 6$
- $X_2 = (5 \times 6 + 1) \bmod 16 = 15$
- 12, 13,2,11,8,9,14,7,4,5,10,3,0,**1,6,15,...**

- but it has patterns in its generation, see Hyperplanes figure at Wikipedia page
- now generally replaced by the *Mersenne Twister*

Outline

Simple Samplers

- PRNGs give us uniform samplers for integers in $[0, m-1]$ for choice of $m \in \mathcal{N}$
- can use these to uniformly sample a real valued number in $[0, 1)$ represented in floating point by computing r/m where r is the PRNG
- but what about other one dimensional distributions:
  - Gaussian
  - Poisson
  - Students t
  - etc.

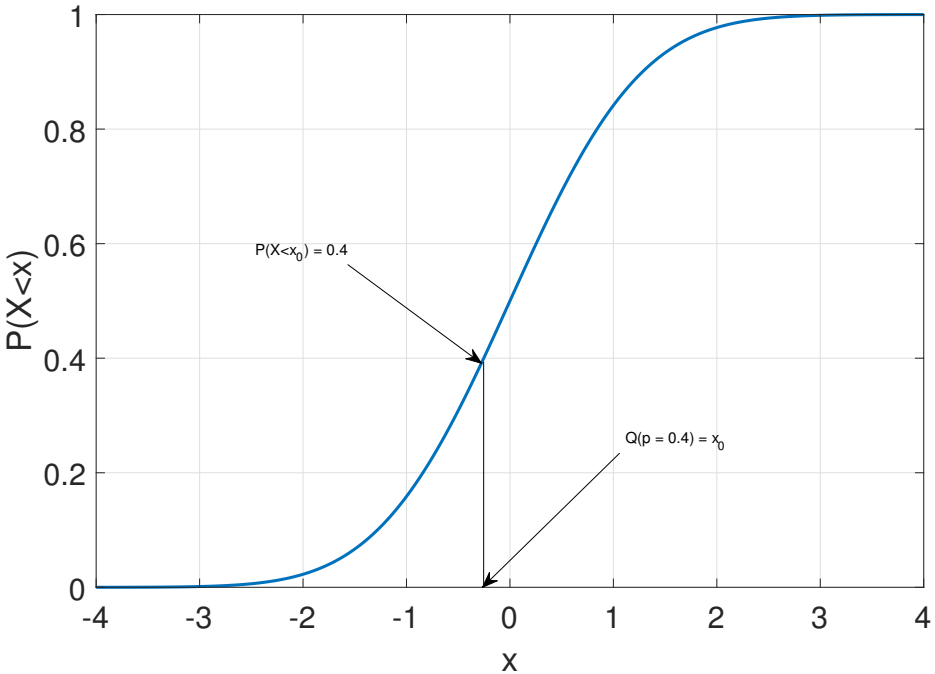
CDF and Quantile Function

The **cumulative distribution function** (CDF) of a continuous RV X is:

$$P(x) = p(X \leq x) = \int_{-\infty}^{x} p(x')dx'$$

that is, the probability that X is less than some value x. The **quantile function** is

$$Q(p) = \{x \in \mathcal{X} : p(X \leq x) = p\}$$



- plot shows CDF for Gaussian
- quantile obtained by flipping the axes

The Inverse Transform Sampler

Inverse Transform Sampler: Let RV X have the CDF $P(X)$, and its quantile function be $Q(p)$ (usually, $= P^{-1}(p)$). To sample X, do the following:

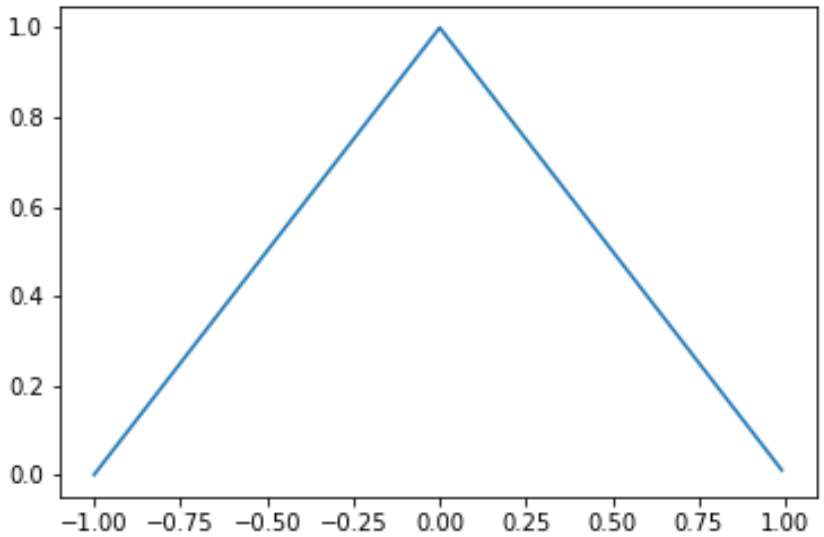
1. sample U as a uniform variable in $(0, 1)$
2. return $Q(U)$

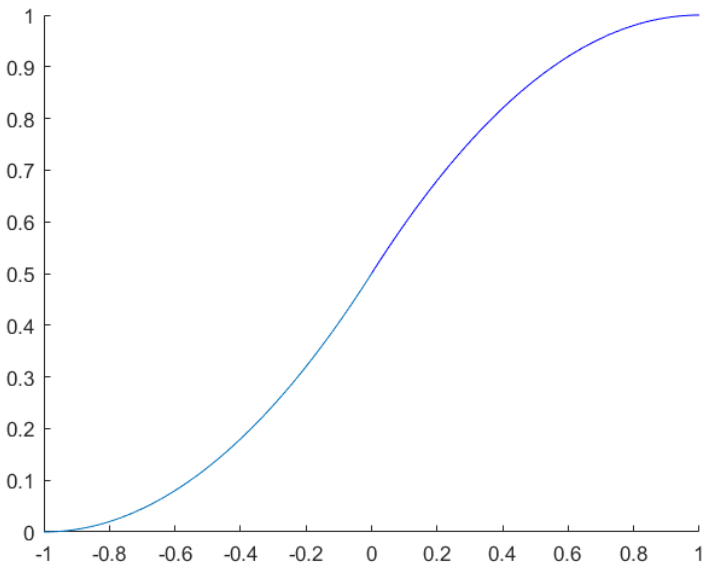
- the inverse transform sampler only works when you have an easily computable expression for the quantile function
  - not the case for the Gaussian!

Example

consider the distribution shown at right,

$$p(x) = 1 - |x| \quad \text{for } |x| < 1$$
$$= \begin{cases} 1 + x & : \quad -1 < x < 0 \\ 1 - x & : \quad 0 \leq x < 1 \end{cases}$$

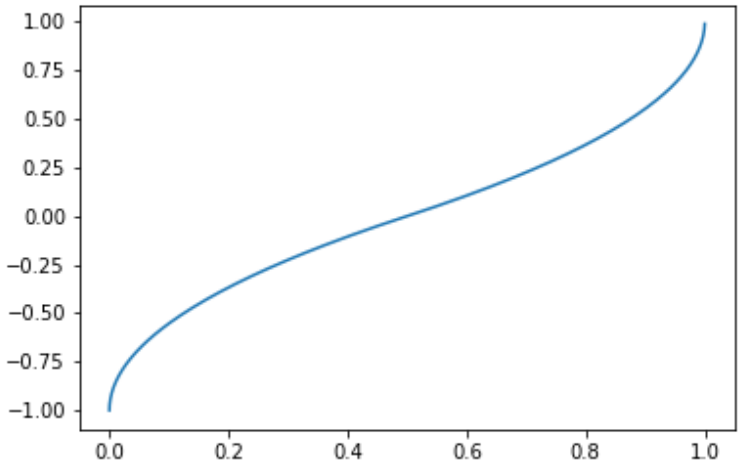




it has the CDF $P(x)$ plotted on the left

$$\begin{cases} \int_{-1}^{x} (1 + y) \mathrm{d}\, y & : \quad -1 < x < 0 \\ \frac{1}{2} + \int_{0}^{x} (1 - y) \mathrm{d}\, y & : \quad 0 \leq x < 1 \end{cases}$$
$$= \begin{cases} \frac{1}{2} + x + \frac{1}{2} x^2 & : \quad -1 < x < 0 \\ \frac{1}{2} + x - \frac{1}{2} x^2 & : \quad 0 \leq x < 1 \end{cases}$$

Example, cont.

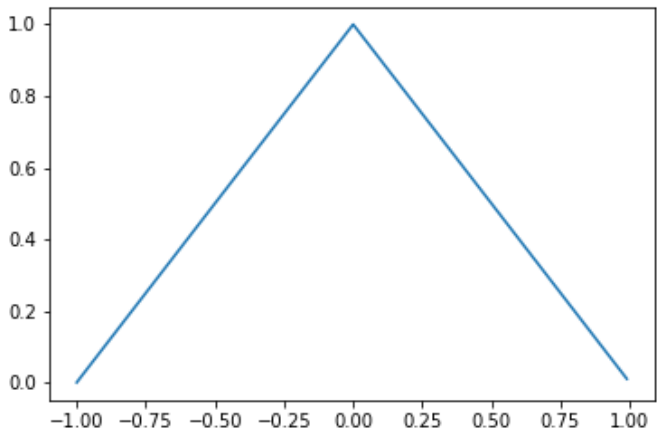


flip the plot for $P(x)$ to get the inverse CDF $Q(p)$ plotted on the left

$$\begin{cases} -1+\sqrt{2p} & : \quad 0<p<\frac{1}{2} \\ 1-\sqrt{2-2p} & : \quad \frac{1}{2}\leq p<1 \end{cases}$$

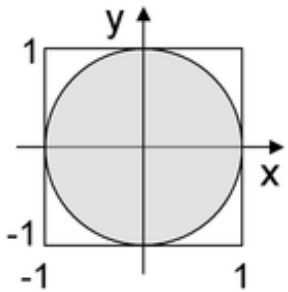
To sample from the original $p(x)$ (on right)

1. sample U in uniform $(0,1)$
2. return $Q(U)$



Rejection Sampling

How do we generate a point inside a circle easily?



Method:

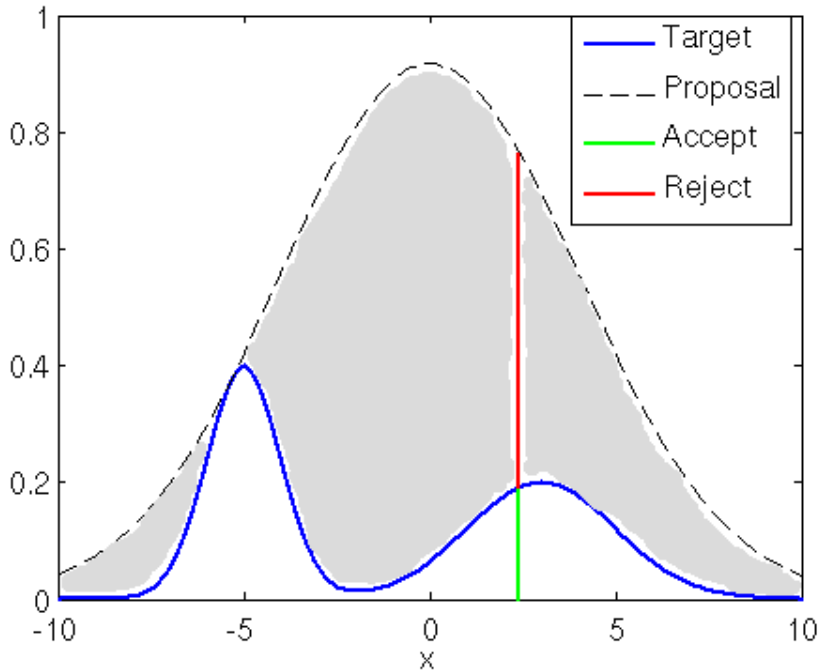
1. sample both $x, y \sim \text{uniform}(0, 1)$,
2. if $x^2 + y^2 < 1$ return (x, y), otherwise go back to (1).

- Step (1) proposes a sample point (x, y) with a proposal distribution $g(x, y)$, inside the square
- Step (2) does an acceptance test using an acceptance probability $a(x, y)$.
- Their combination $g(x, y)a(x, y)$ yields a random point in a circle.

The resulting general technique called *Rejection sampling*.

Rejection Sampling, cont.

Sample x for proposal distribution (dashed line) is easily generated with a known sampler (a Gaussian in the figure). Value of the blue line at a point is easily computed.



1. **sample** x according to proposal distribution
2. **reject** x with probability of red/(red+green) line, and return to 1
3. if accepted, x distributed proportionally to the blue curve

In the process you have thrown away samples equal in area to the grey area:

- smaller is better!

The Rejection Sampler

Rejection Sampler: Given a PDF proportional to $q(x)$, and have a distribution $p_{prop}(x)$ whose PDF satisfies $Cq(X) \leq p_{prop}(x)$ for some constant C. The algorithm below

1. **sample** $x \sim p_{prop}(x)$
2. **sample** U as uniform in $(0, 1)$
3. **reject** x if $U > \frac{Cq(X)}{p_{prop}(x)}$ then return to step 1
4. return x

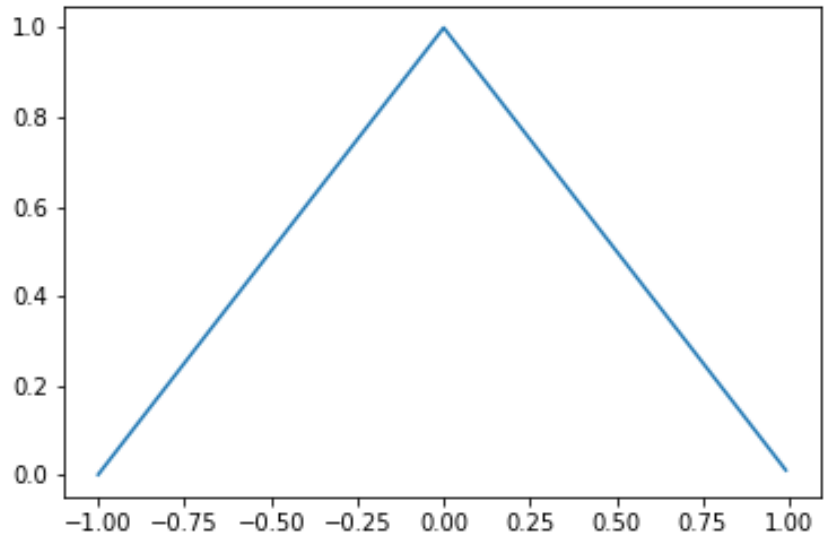
samples according to the PDF proportional to $q(x)$, and it rejects $\left(1 - C \int q(X) \mathrm{d}\, x\right)$ of its samples.

- need to guess an appropriate $p_{prop}(x)$ and find a C
- don't need to normalise required PDF

Example

consider the distribution shown at right,

$$p(x) = \quad 1 - |x| \qquad \text{for } |x| < 1$$



Use $q(x) = p(x)$ above, and try $p_{prop}(x) = \frac{1}{2}$ for $|x| < 1$, and $C = \frac{1}{2}$.

1. **sample** x as uniform in $(-1, 1)$
2. **sample** U as uniform in $(0, 1)$
3. **reject** x if $U > \frac{\frac{1}{2}(1-|x|)}{\frac{1}{2}} = 1 - |x|$, then return to step 1
4. return x

much simpler than inverse sampler, but throw away 1/2 samples

Outline

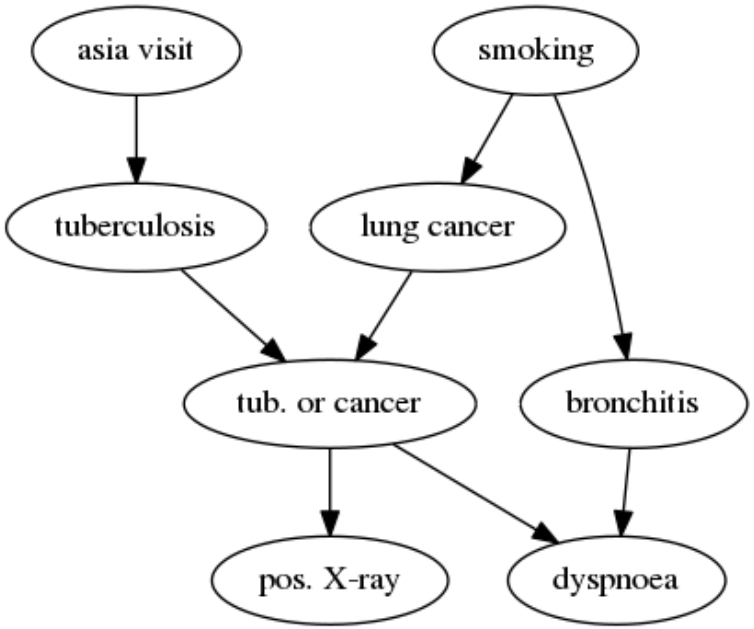
Sampling Groups of Variables

Revision at *https://flux.qa/43FMK4*

- so far we have looked at sampling univariate distributions
- what if you wanted to sample a full group of variables
- we will look at Gibbs sampling which does this by sampling one variable at a time, using an MCMC sampler

- **first**, we will show how to represent a multivariate probability distribution
- **then**, as a simplification of sampling, we will consider local search on the graph colouring problem
- **finally**, we will review Gibbs sampling

Example Bayesian Network



original directed graph with full names

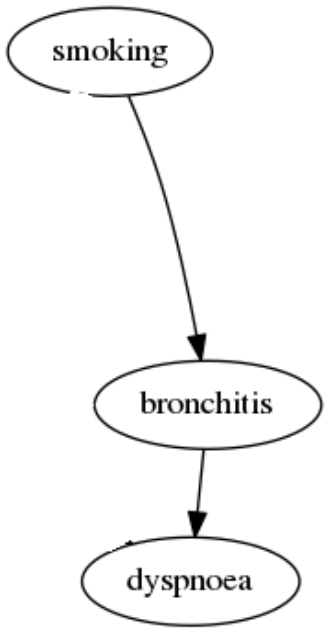
full joint probability takes the probability form:

$$p(\textit{asia visit})p(\textit{tuberculosis}|\textit{asia visit})$$
$$p(\textit{smoking})p(\textit{lung cancer}|\textit{smoking})$$
$$p(\textit{bronchitis}|\textit{smoking})$$
$$p(\textit{tub. or cancer}|\textit{tuberculosis}, \textit{lung cancer}$$
$$p(\textit{pos. Xray}|\textit{tub. or cancer})$$
$$p(\textit{dyspnoea}|\textit{tub. or cancer}, \textit{bronchitis})$$

For conciseness, let us represent this as

$$p(av)p(t|av)p(s)p(lc|s)p(b|s)p(toc|t, lc)p(x|toc)p(d|toc, b)$$

Simpler Example of Bayesian Network

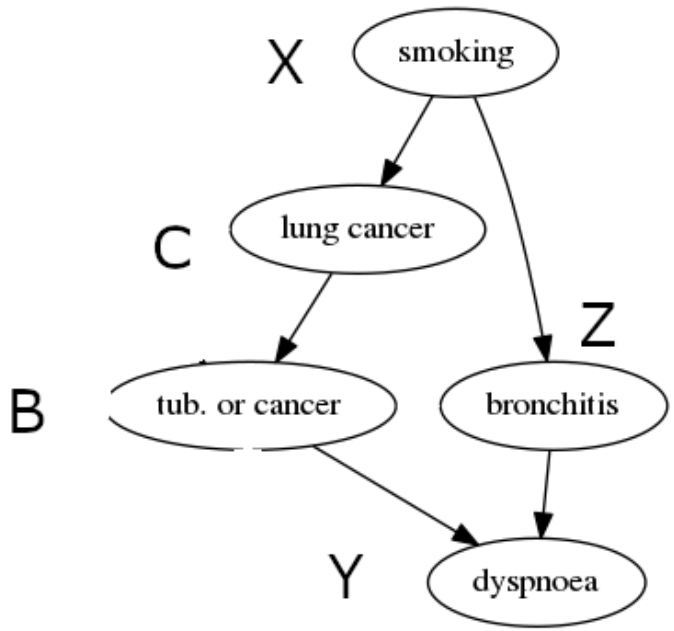


Model asserts that smoking causes bronchitis directly, and that bronchitis causes dyspnoea directly.

The model implies that smoking and dyspnoea are correlated, but that the partial correlation of smoking and dyspnoea controlling for bronchitis does vanish.

Method dealing with (partial) correlations called d-separation.

Simpler Example of Bayesian Network

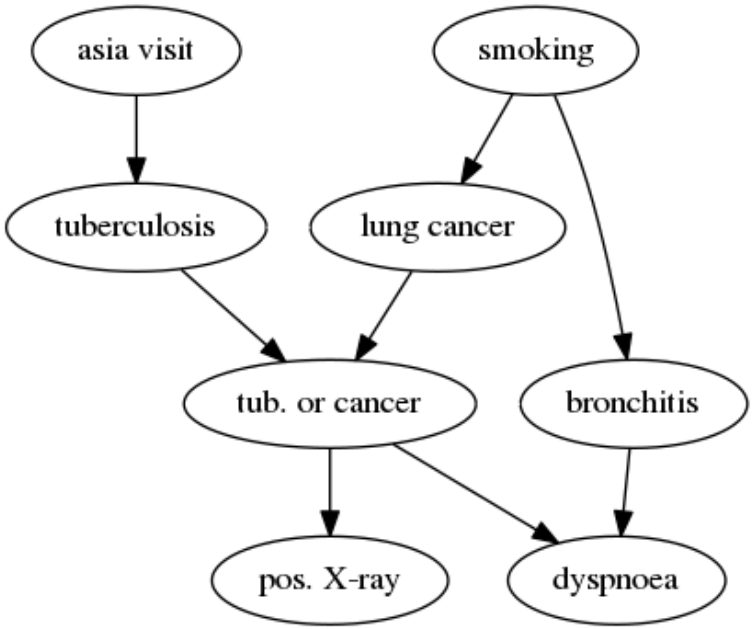


Basic idea of d-separation: two variables X and Y are independent conditional on Z if knowledge about X gives you no extra information about Y once you have knowledge of Z. In other words, once you know Z, X adds nothing to what you know about Y.

X and Y are d-separated if all the paths that connect them carry no dependence

d-separation algorithm

Sampling Bayesian Network



sampling from a Bayesian network can easily be done in any variable order consistent with the parent-child relationships of the network. e.g.,

- asia visit
- smoking
- tuberculosis
- lung cancer
- etc.

As long as the order **obeys the parent-child relationship**, for any probability such as $p(\textit{lung cancer}|\textit{smoking})$ the RHS will already be sampled before attempting the LHS.

Bayesian Network, cont.

Have $p(av, t, s, lc, b, toc, x, d)$ given by

$$p(av)p(t|av)p(s)p(lc|s)p(b|s)p(toc|t, lc)p(x|toc)p(d|toc, b)$$

To **sample** $(av, t, s, lc, b, toc, x, d)$ can do it sequentially, like so:

1. sample $av \sim p(av)$
2. sample $t \sim p(t|av)$
3. sample $s \sim p(s)$
4. sample $lc \sim p(lc|s)$
5. sample $b \sim p(b|s)$
6. sample $toc \sim p(toc|t, lc)$
7. etc.

Bayesian Network, cont.

Suppose, however, *toc* was fixed

$$p(av)p(t|av)p(s)p(lc|s)p(b|s)p(toc|t,lc)p(x|toc)p(d|toc,b)$$

The previous sequentially sampler cannot be used:

- the initial $p(av)$ should really be $p(av|toc)$
- and so forth for others

Need another way of sampling.

Bayesian Network, cont.

Consider:

$$p(av)p(t|av)p(s)p(lc|s)p(b|s)p(toc|t,lc)p(x|toc)p(d|toc,b)$$

So we drop the variable ordering and represent it in a functional form. i.e. forget that terms in the product are probabilities.

Replace each conditional probability by a function:

- $p(av) \longrightarrow f_1(av)$
- $p(lc|s) \longrightarrow f_4(lc,s)$
- etc.

Resulting in:

$$f_1(av)f_2(t,av)f_3(s)f_4(lc,s)f_5(b,s)f_6(toc,t,lc)f_7(x,toc)f_8(d,toc,b)$$

Bayesian Network, cont.

Univariate conditional distributions are easily obtained. So

$$p(av, t, s, lc, b, toc, x, d)$$
$$= f_1(av)f_2(t, av)f_3(s)f_4(lc, s)f_5(b, s)f_6(toc, t, lc)f_7(x, toc)f_8(d, toc, b)$$

Therefore:

$$p(b|t, s, lc, av, toc, x, d)$$
$$\propto f_1(av)f_2(t, av)f_3(s)f_4(lc, s)f_5(b, s)f_6(toc, t, lc)f_7(x, toc)f_8(d, toc, b)$$
$$\propto f_5(b, s)f_8(d, toc, b)$$

Likewise

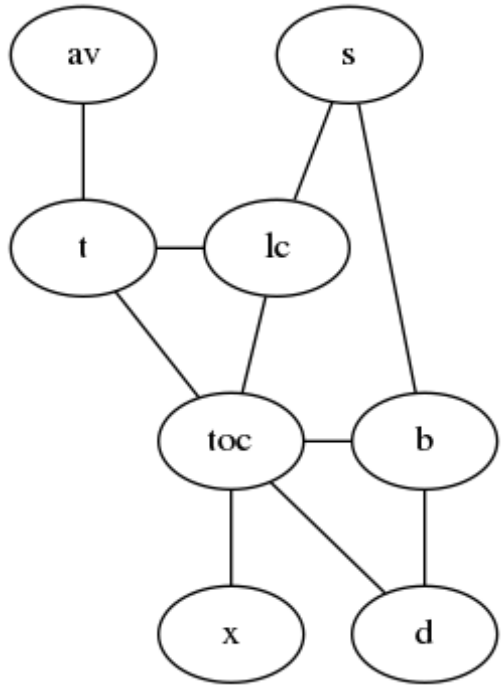
- $p(s|av, t, lc, b, toc, x, d) \propto f_3(s)f_4(lc, s)f_5(b, s)$
- $p(lc|av, t, s, b, toc, x, d) \propto f_4(lc, s)f_6(toc, t, lc)$
- etc.

Bayesian Network, Summary

full joint probability takes the functional form:

$$f_1(av)f_2(t,av)f_3(s)f_4(lc,s)f_5(b,s)$$
$$f_6(toc,t,lc)f_7(x,toc)f_8(d,toc,b)$$

plus individual condition probabilities, e.g., $p(s|av,t,lc,b,toc,x,d)$, easily obtained

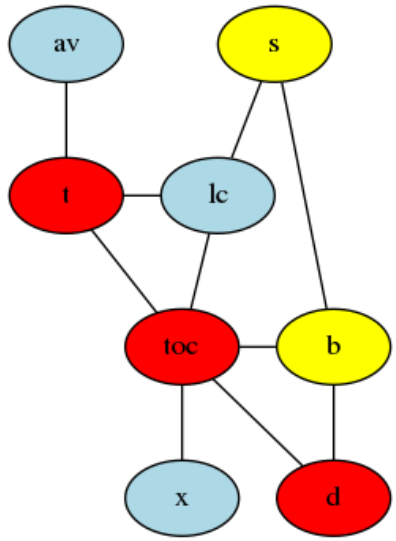


derived **undirected graph**

Outline

Graph Colouring

Graph colouring: try to find a colouring so no two neighbours have the same colour.



Let conflicts(x) be the number of neighbours of x with the same colour: conflicts(lc) $= 0$, conflicts(t) $= 1$.

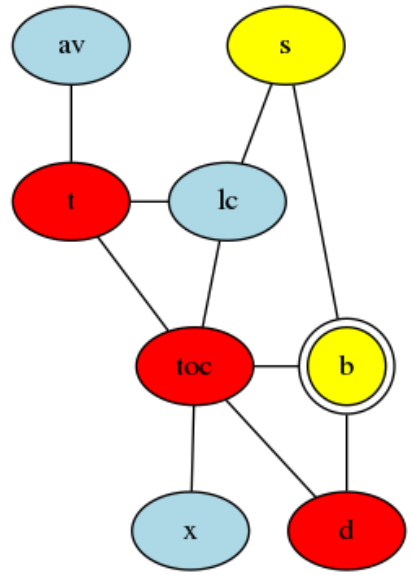
Local Search for Graph Colouring

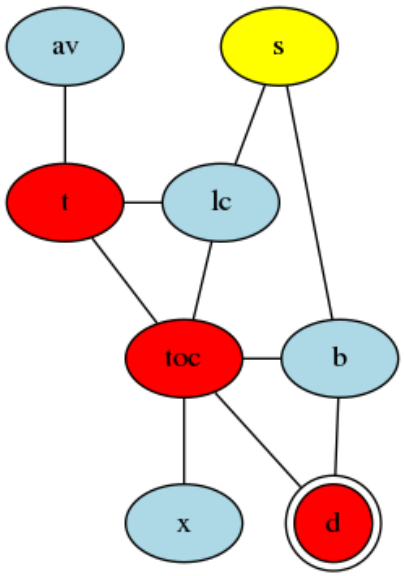
Repeat until bored.

1. *Choose* dimension k randomly or by cycling through.
2. *Reassign* x_k to a colour such that conflicts(x_k) is locally minimum.

Graph Colouring (1)

change b**:** conflicts(b) $=$ 1,
change to color light blue

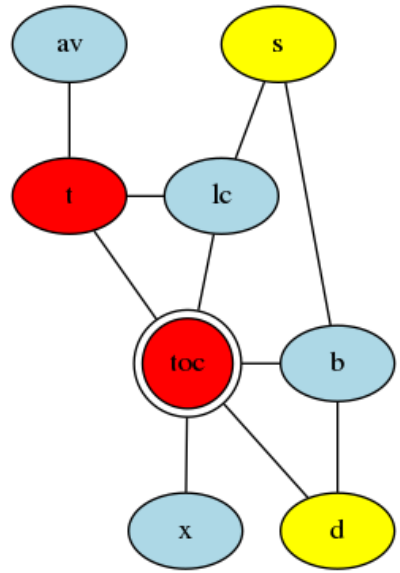


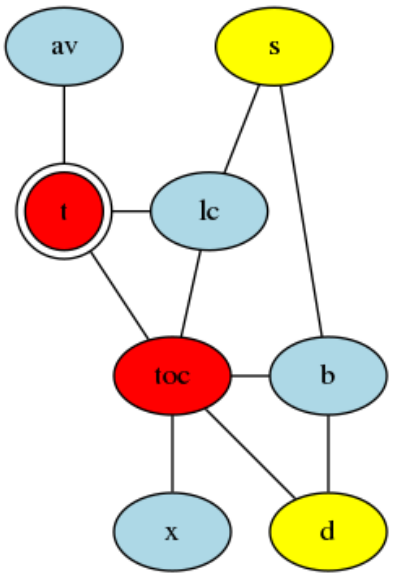


change d**:** conflicts(d) $=$ 1,
change to color yellow

Graph Colouring (2)

change *toc*: conflicts(toc) $= 1$, no change as nothing makes it better

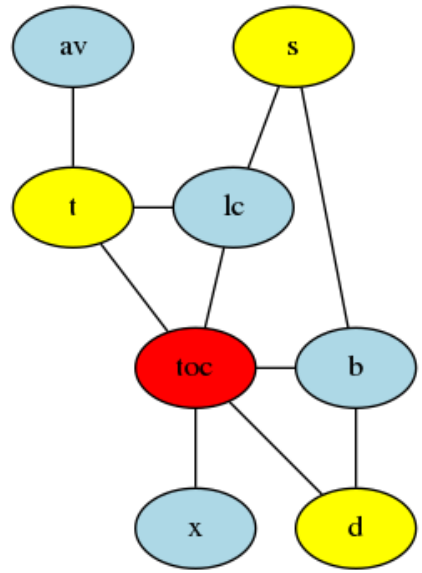




change *t*: conflicts(t) $= 1$, change to color yellow

Graph Colouring (3)

no more conflicts so we have
converged to a solution



- *local search* changes one node at a time based on its neighbours
- it is simple to implement
- works moderately well with smaller (<5000) variables on not too hard problems

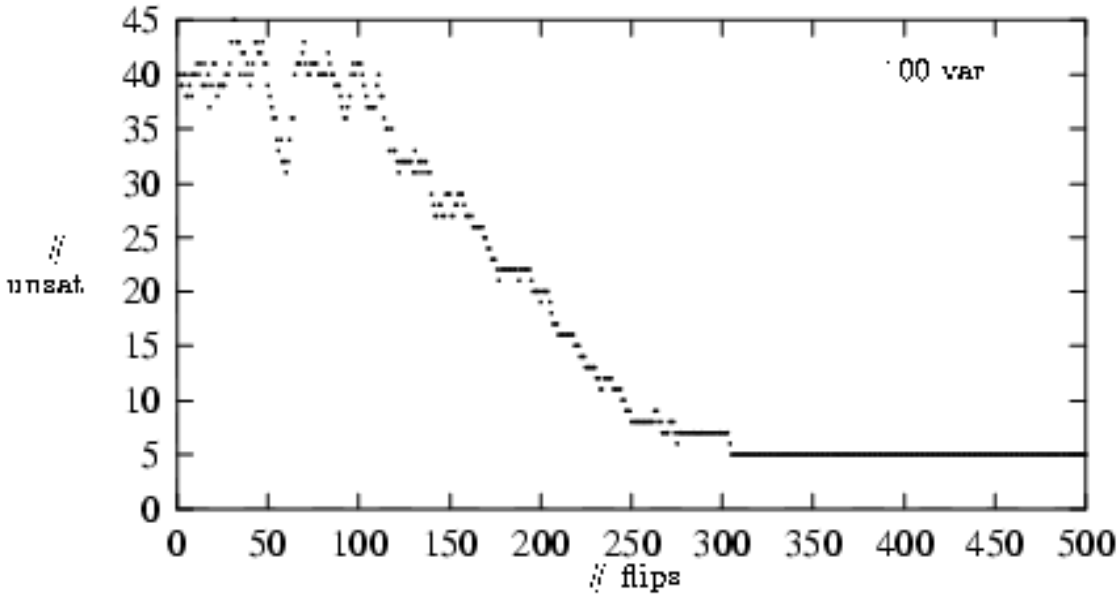
Outline

Probabilistic Local Search

The Grand Canyon (from Wikipedia.org)

To find the bottom (1) mostly walk down hill (2) but sometimes follow flat ridges (3) and maybe occasionally go up hill to get out of a local valley.

Probabilistic Local Search, e.g.



Here we have a 100 variable 3SAT problem (rather like 3 colouring) done with probabilistic local search: we sample variables at each stage instead of assigning to the value maximising conflicts.

Probabilistic Local Search

Problem: wish to sample from $p(av, t, s, lc, b, toc, x, d)$ given by

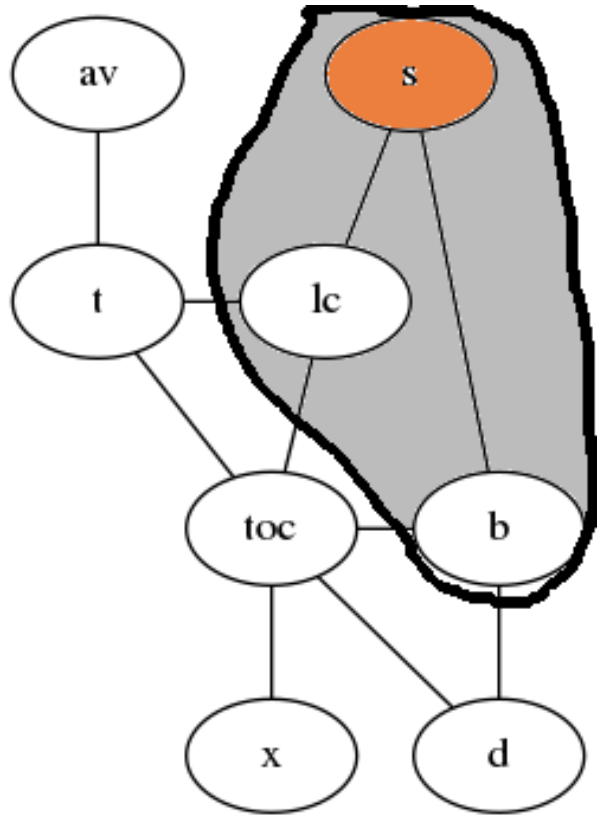
$$f_1(av)f_2(t, av)f_3(s)f_4(lc, s)f_5(b, s)f_6(toc, t, lc)f_7(x, toc)f_8(d, toc, b)$$

Solution:

- sample each single variable in turn, $av|t, s, lc, b, toc, x, d$; then $t|av, s, lc, b, toc, x, d$; the $s|av, t, lc, b, toc, x, d$; etc.
  - use a sequence of univariate samplers
- this is an MCMC algorithm for sampling $av, t, s, lc, b, toc, x, d$
- a similar property to the weak law of large numbers holds!

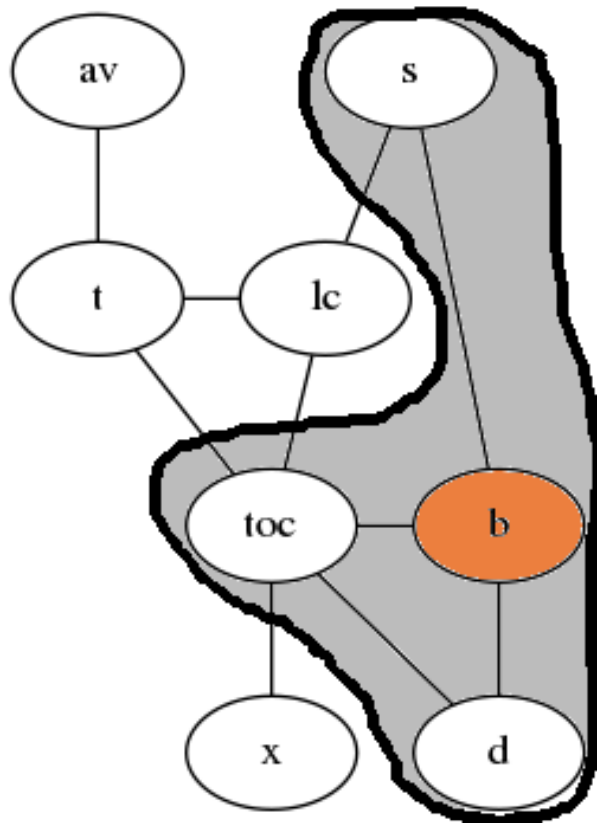
Probabilistic Local Search (1)

Sample s using $p(s|b, lc)$



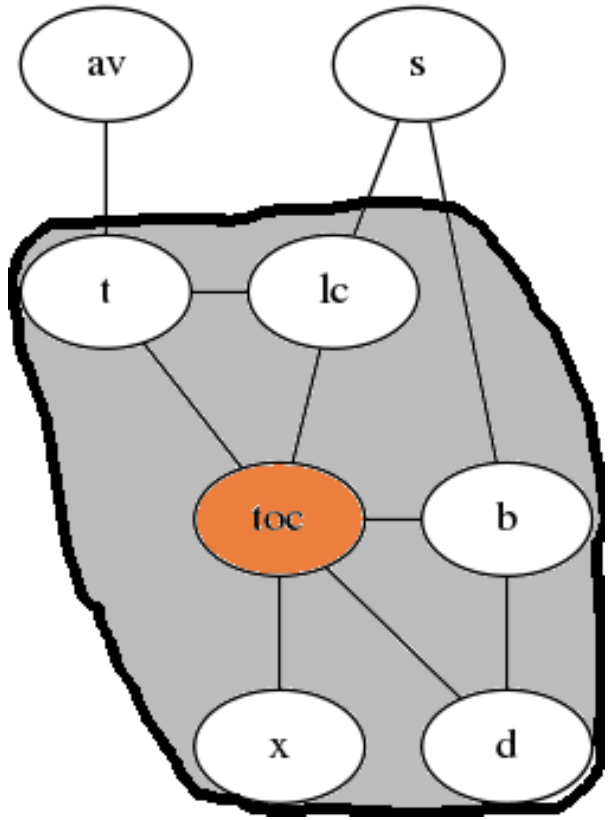
Probabilistic Local Search (2)

Sample b using $p(b|s, toc, d)$



Probabilistic Local Search (3)

Sample *toc* using $p(toc|t, lc, b, x, d)$



continue, many times!

The Gibbs Sampler

Gibbs Sampler: Have RVs $\vec{x} = (x_1, x_2, ..., x_k)$ with joint distribution $p(x_1, x_2, ..., x_k) > 0$. Repeat n times:

1. for $i = 1, ..., k$, re-sample x_i according to the conditional $p(x_i | x_1, ..., x_{i-1}, x_{i+1}, ..., x_k)$
2. record $f_j = f(x_1, x_2, ..., x_k)$

Then for any $\epsilon > 0$

$$p\left(\left|\frac{f_1 + ... + f_n}{n} - \mathbb{E}_{\vec{x}}\left[f(x_1, x_2, ..., x_k)\right]\right| > \epsilon\right) \to 0 \text{ as } n \to \infty$$

- the Gibbs sampler generates MCMC samples that can be used to form estimates of population means
- can be very slow
- need to be able to sample from required univariate conditional distributions efficienctly
- is a version of the weak law of large numbers for MCMC

End of Week 10