Partially Recursive Acceptance Rejection

Mark Huber
Department of Mathematical Sciences
Claremont McKenna College

13 Oct, 2011

Perfect Simulation

Monte Carlo Methods

For $w \geq 0$ over discrete space Ω:

Generate random variable X so that

$$\mathbb{P}(X = i) = \frac{w(i)}{\sum_{j \in \Omega} w(j)}$$

Use random draws of X to approximate

$$\sum_{j \in \Omega} w(j)$$

Monte Carlo Methods Continuous Version

For $f \geq 0$ over $\mathbb{R}^d$:

Generate random variable X so that

$$\mathbb{P}(X \in A) = \frac{\int_A f(x)\, dx}{\int_{\mathbb{R}^d} f(x)\, dx}$$

Use random draws of X to approximate

$$\int_{\mathbb{R}^d} f(x)\, dx$$

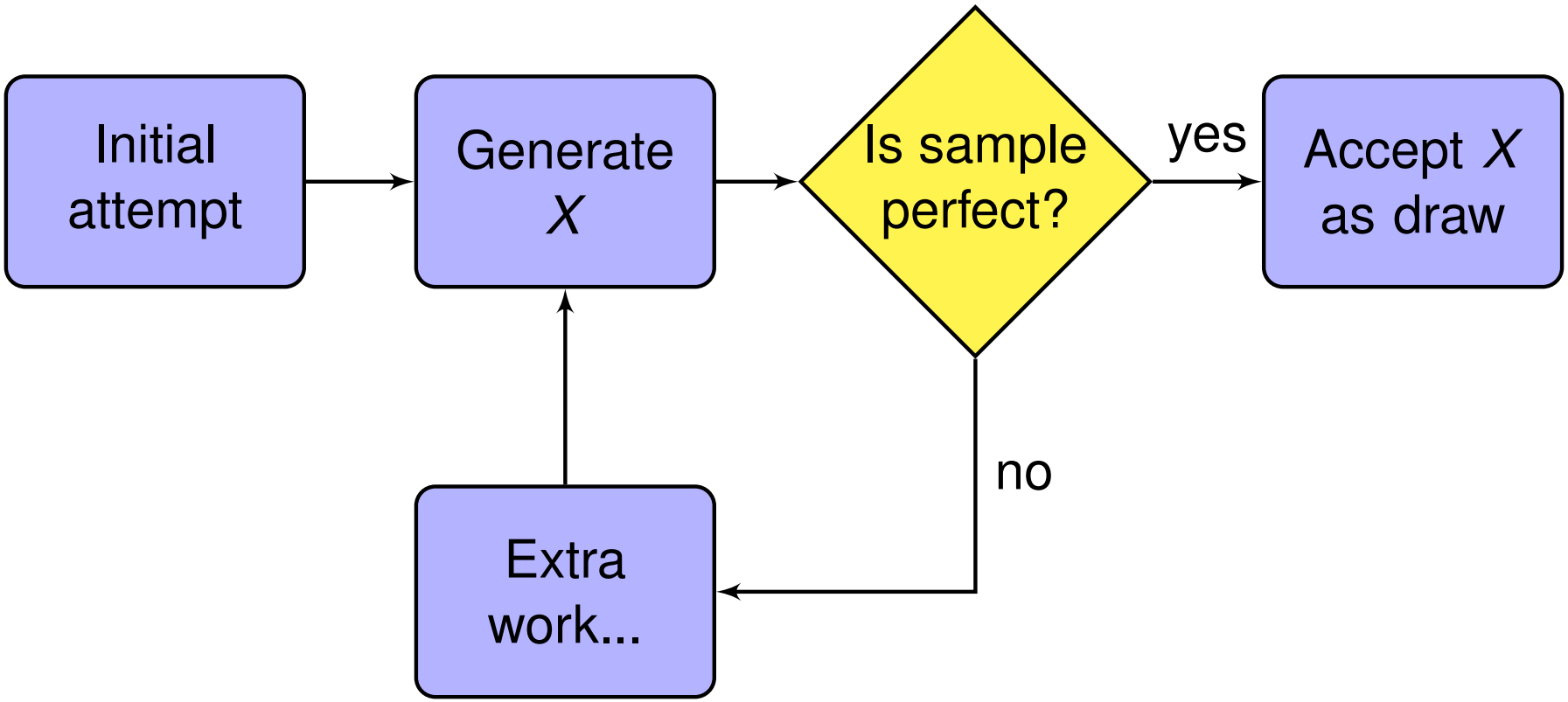
Applications

- Exact p values [Frequentist Statistics]
- Model selection [Bayesian Statistics]
- Phase Transitions [Statistical Physics]
- Approximation Algorithms for #P complete problems [Theoretical Computer Science]

Perfect simulation

Definition (Perfect simulation)

A *perfect simulation* algorithm draws variates exactly from a target distribution in a random number of steps.



Ways to design perfect simulation algorithms

Acceptance/Rejection [Buffon 1777, von Neumann 1933]
Works well for single dimension, not so good in high dimensions

Coupling from the past [Propp, Wilson 1996]
Uses a proxy Markov chain, works well in high dimensions

Popping [Wilson 1996]
Random spanning trees, sink free orientations

Retrospective Sampling [Beskos, Papspilopoulos, Roberts 2006]
Sampling stochastic differential equations perfectly

My work in perfect sampling

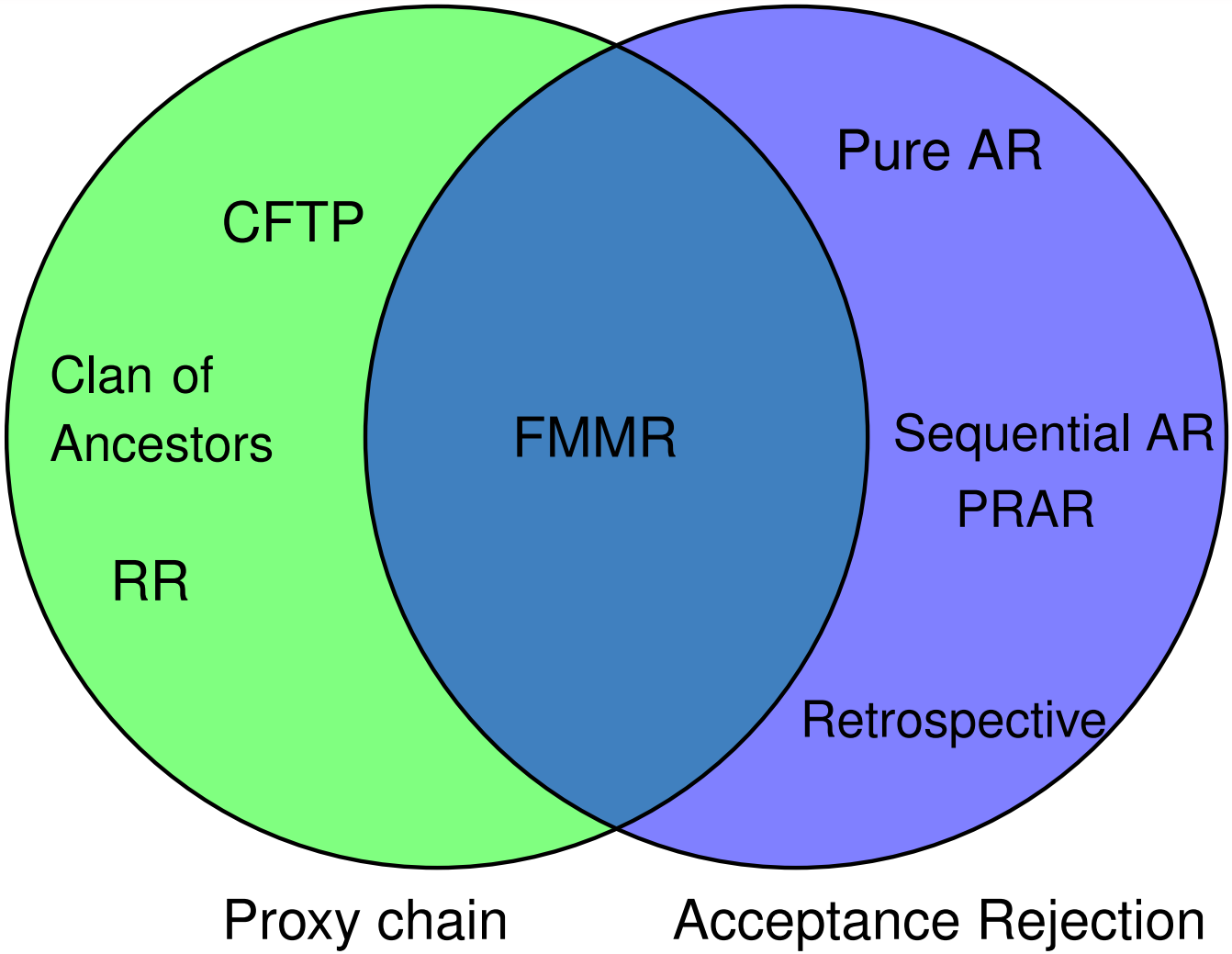
Bounding chains [H. 1999,2004]
Extending CFTP to a wider set of problems

Randomness Recycler [Fill, H. 2000]
Avoiding the need for a proxy Markov chain as in CFTP, FMMR

Scaling Sequential Acceptance Rejection [H. 2004]
Perfect sampling for matchings, permanent

Partially Recursive Acceptance Rejection [H. present]
Pure acceptance rejection for weakly coupled spin systems

Proxy Markov chains and Acceptance/Rejection



Acceptance Rejection

Basic Acceptance/Rejection

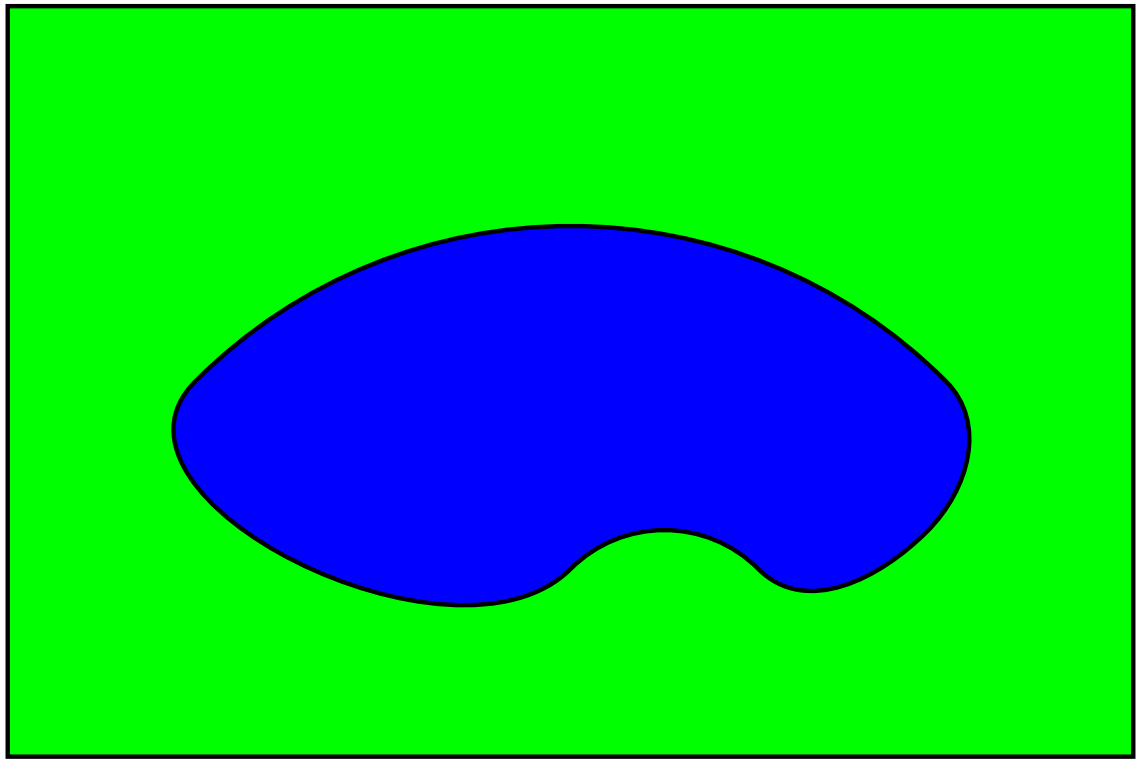
Before running the algorithm

- Goal: Generate X uniformly from B
- Find A that is easier to uniformly sample from where $B \subset A$

Basic acceptance rejection algorithm

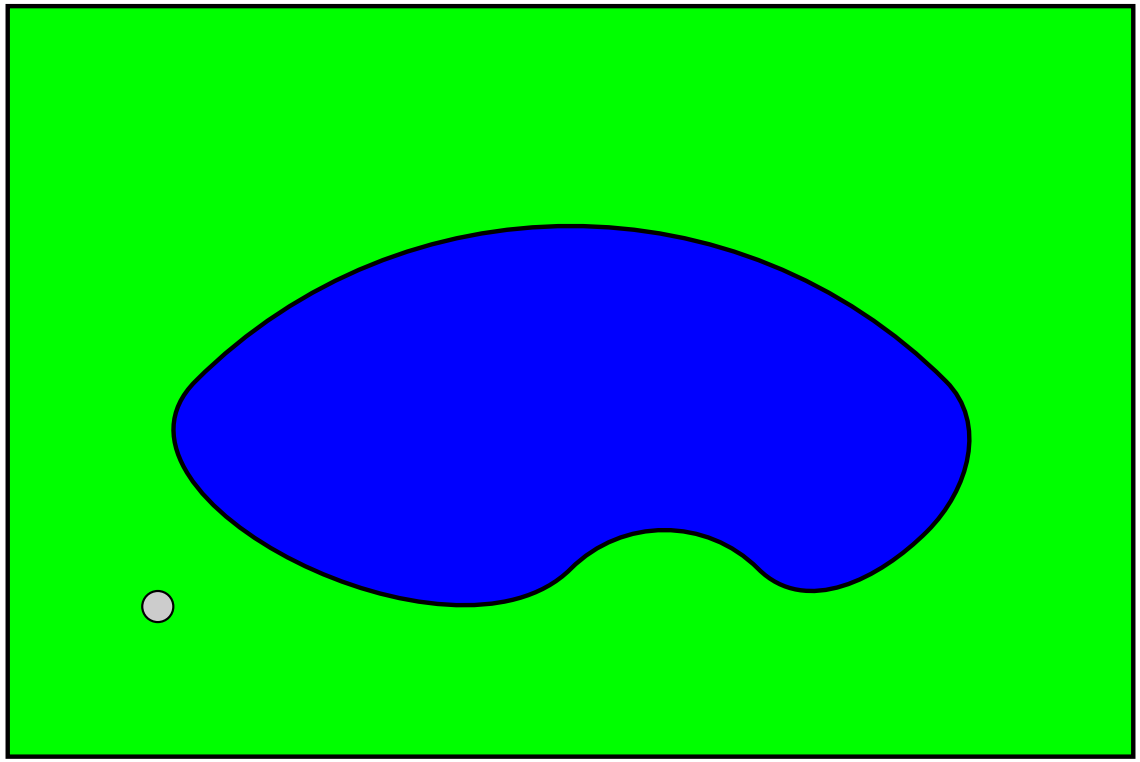
- Step 1: Sample X uniformly from A
- Step 2a: If X is in A output as uniform sample from B
- Step 2b: Else go to Step 1

Acceptance Rejection



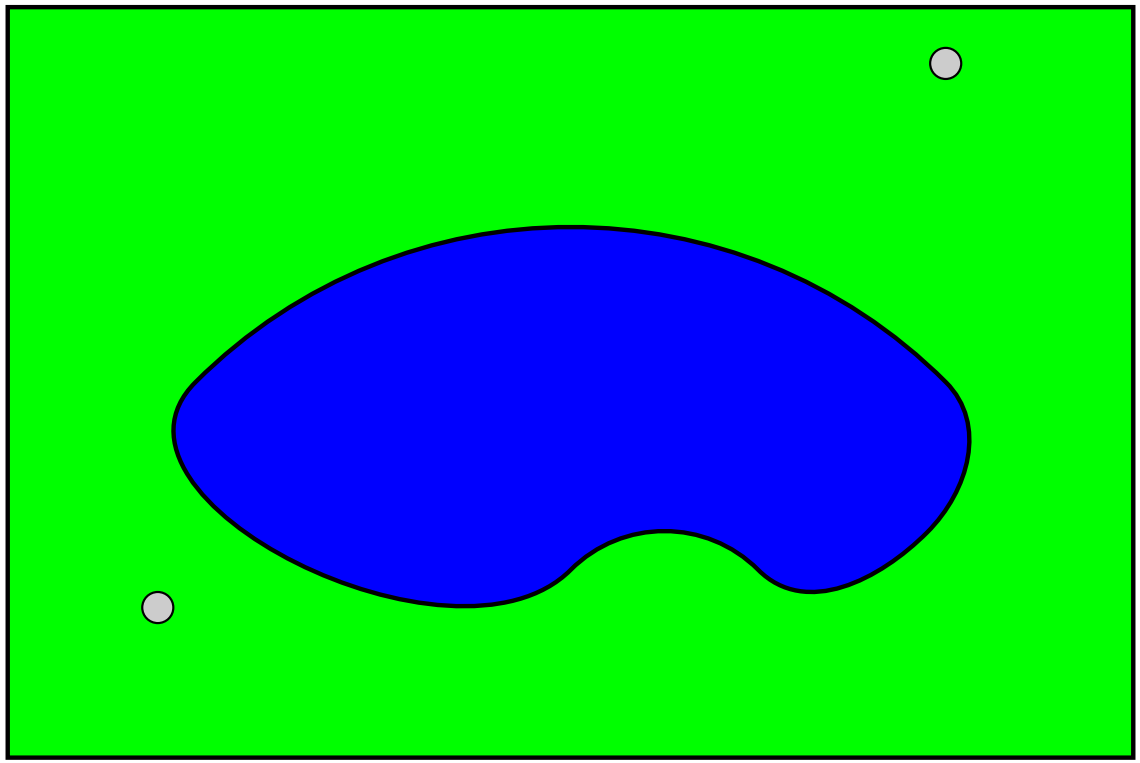
Fire uniformly at green region
Until land in blue region

Acceptance Rejection



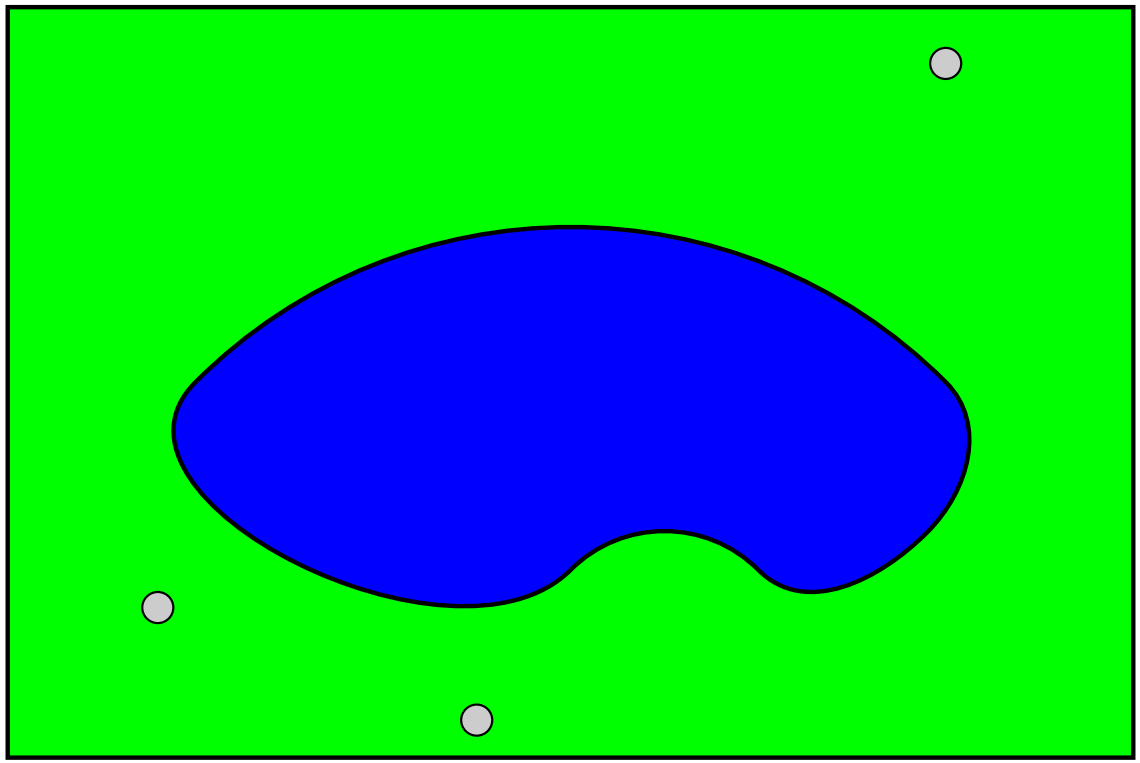
Fire uniformly at green region
Until land in blue region

Acceptance Rejection



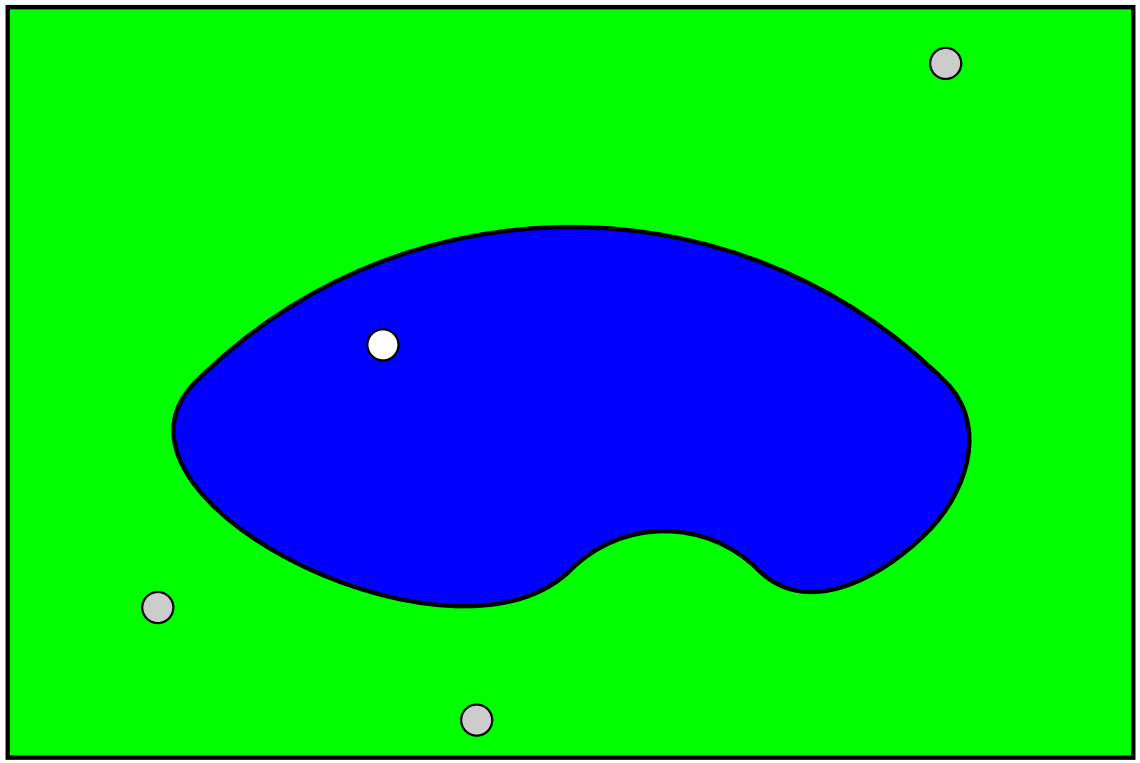
Fire uniformly at green region
Until land in blue region

Acceptance Rejection



Fire uniformly at green region
Until land in blue region

Acceptance Rejection



Fire uniformly at green region
Until land in blue region

A word about uniformity

Fundamental Theorem of Simulation

Can make any problem a uniform one by adding an auxiliary random variable.

Example: to draw a normal random variable

- To draw a standard normal random variable
- Draw (X, Y) uniformly from $\{(x, y) : 0 \leq y \leq (2\pi)^{-1/2} \exp(-x^2/2)\}$
- Keep X

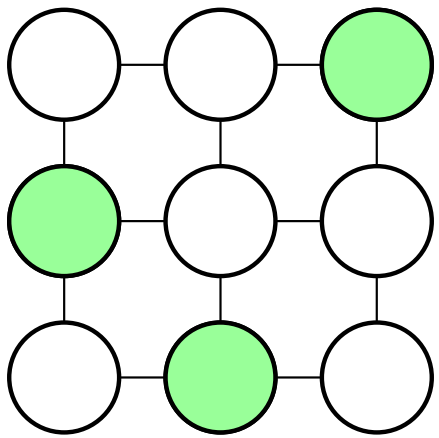
▸ Animation

Coloring nodes with AR

A representative problem

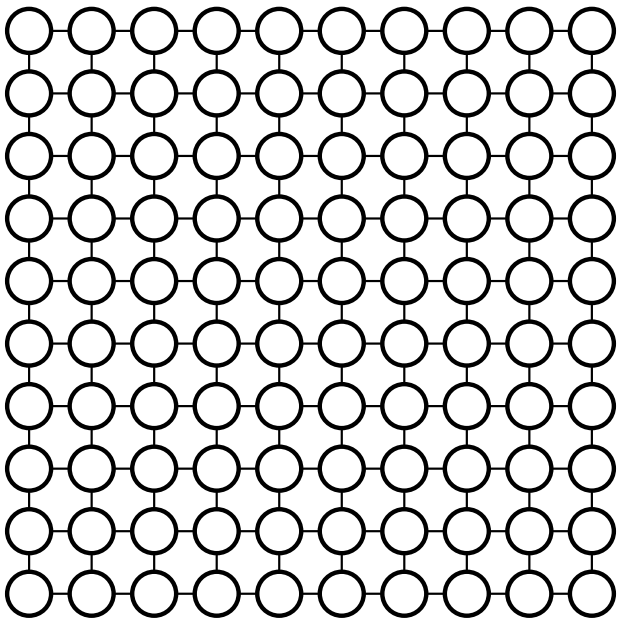
Definition

An *independent set* of a graph is a subset of nodes where no edge has both endpoints in the subset.



Basic AR for independent sets

- Draw X uniformly from subsets of nodes
- If X is not an independent set, go to first step

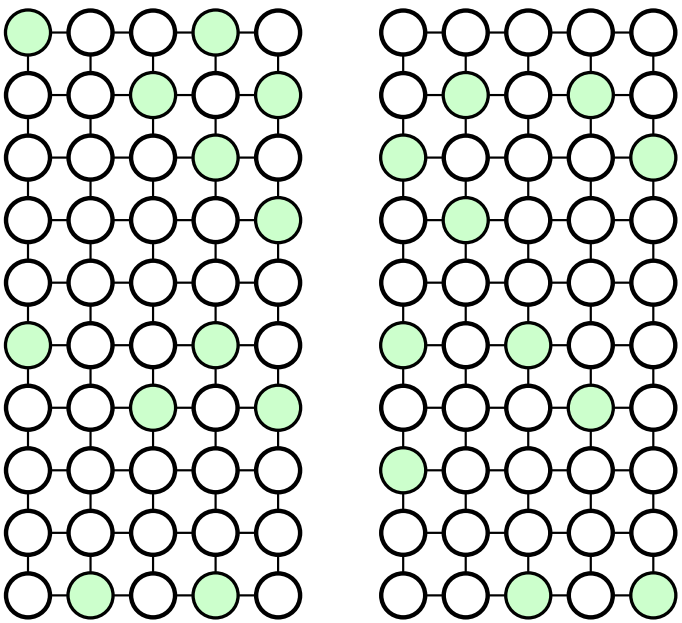


Chance of accepting for 10 by 10 grid $< 10^{-10}$.

Recursive AR on nodes

To sample from π_V over $\Omega = C^V$

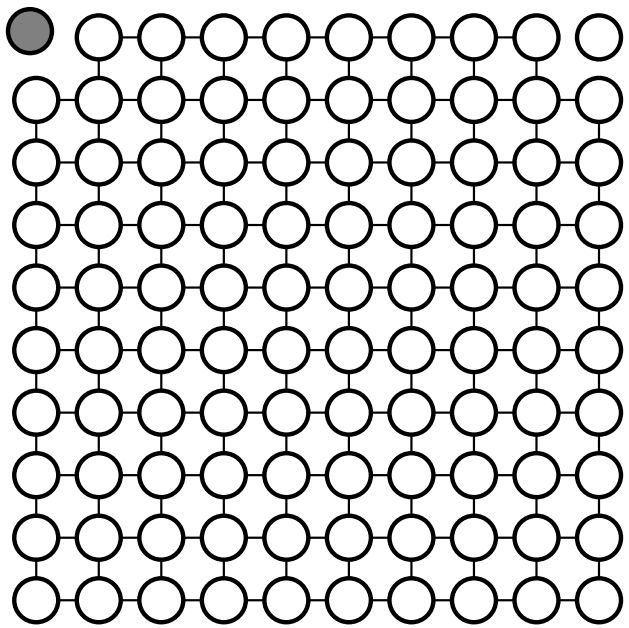
- Partition set of nodes V into V_1 and V_2
- Recursively generate X_1 from π_{V_1} and X_2 from π_{V_2}
- Find probability X_1 with X_2 form configuration on V
- Accept with that probability, otherwise start over



Better idea

Split set of nodes into $\{v\}$ and $V \setminus \{v\}$

- Easy to draw uniformly from independent sets on v
- With probability 1/2 v in set, otherwise it is out



Key observation

Standard recursive AR

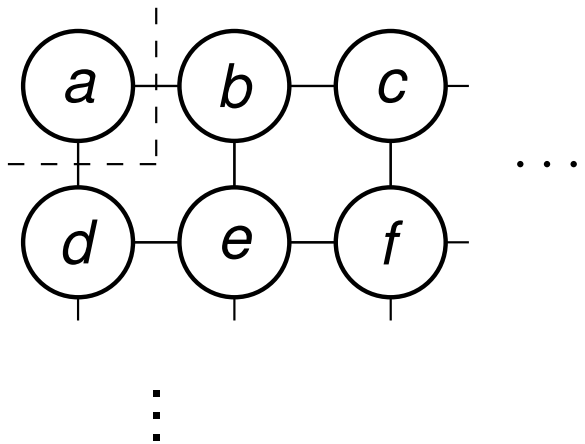
- Draw X_1 on v, X_2 on $V \setminus \{v\}$ first
- If combined form independent set, accept, otherwise reject

Mixing up the order

- Draw X_1 on v first
- If v not in independent set, always accept, don't need X_2!
- If v in ind. set, don't need all of X_2...
- ...only need to know about neighbors of v

One at a time

Build up the sample one node at a time



- First make a in set or out each with probability $1/2$;
- If a out, always accept
- Else find out if b and d are in set, that determines acceptance for a

Examples

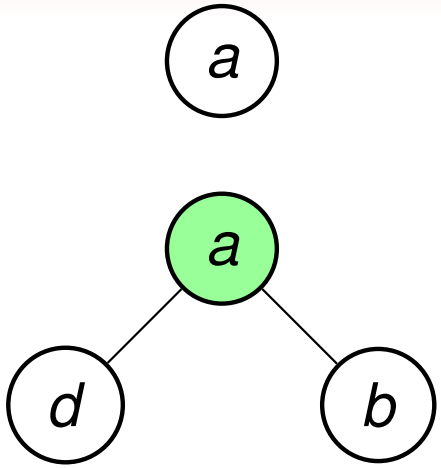
a out
Result: *a* is out of set

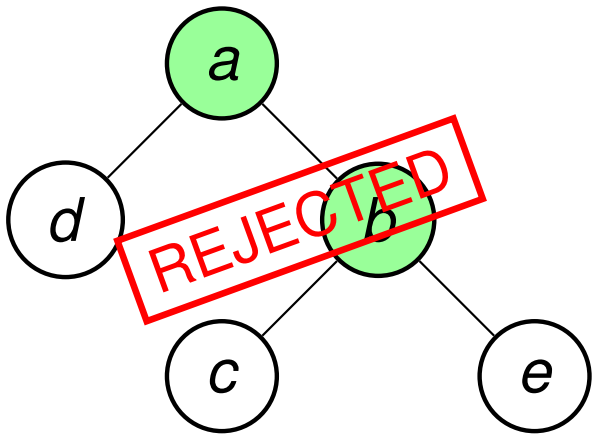
---

a in: put *b* and *d* on queue
b out
d out
Result: *a* is in, *b* is out, *d* is out

---

a in: put *b* and *d* on queue
b in: put *c* and *e* on queue
c out
e out
Result: with *c* and *e* out, *b* is also in, so reject *a* and everything removed, start over with *a*





The algorithm

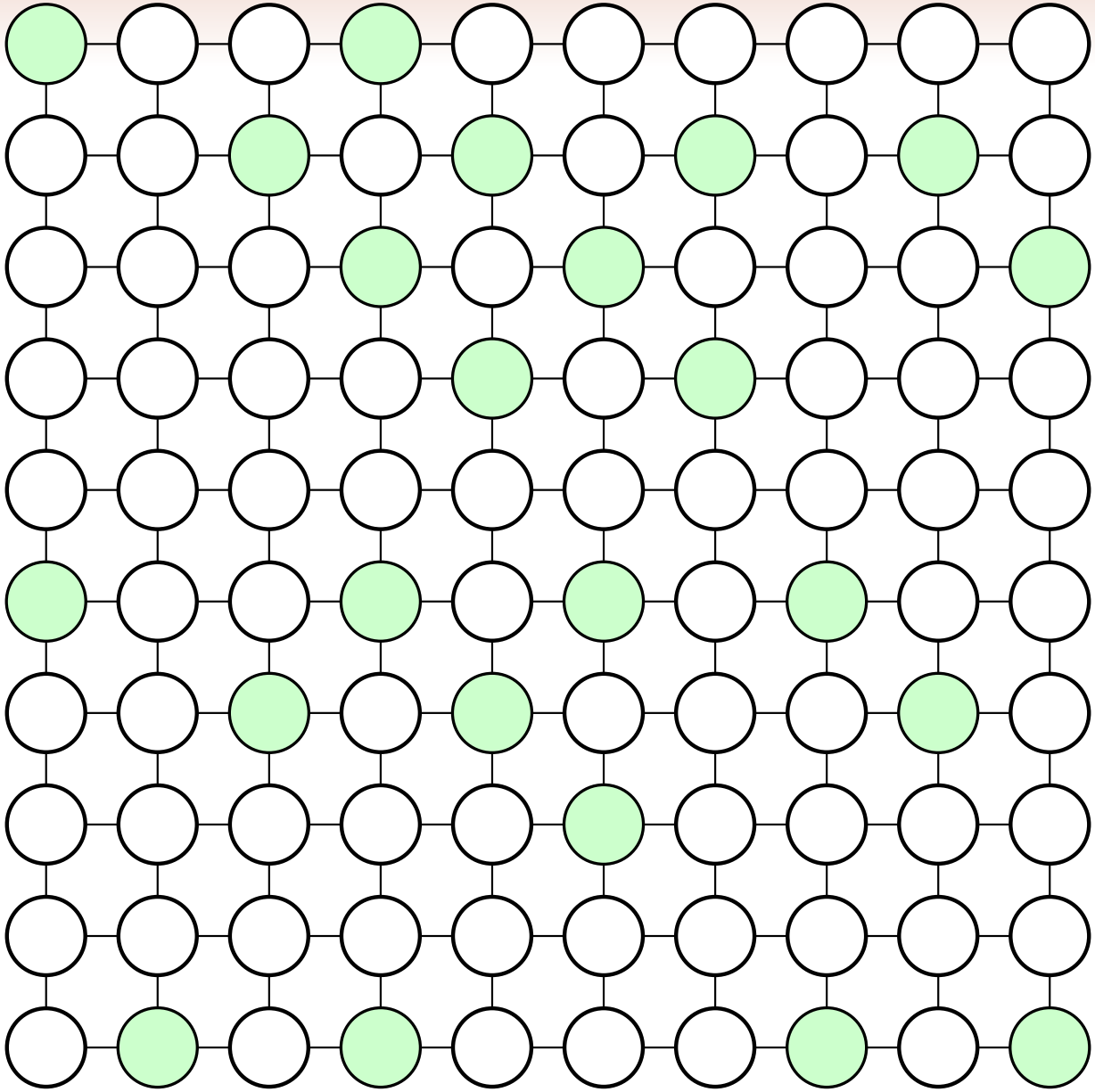
Begin with all nodes in list

- Get a node v off of list
- form its recursion tree
- Fix the values of all nodes in tree
- Continue until list empty

To form recursion tree

- With probability 1/2, node is out
- Otherwise node could be in: add neighbors to tree as children
- If all neighbors/children are out, then node is in...
- ...so reject and remove subtree formed by parent of node

A sample run on 10 by 10 grid



Analysis

Nodes that try to be “in” form node percolation process

- Since node in with probability 1/2, looks like at critical temperature
- But nodes that are “in” remove their siblings...
- ...which leads to subcritical behavior

More generally

- If interactions weak, tree shrinks in size on average
- Artificial phase transition

Method works for Markov random fields

Not just for independent sets...

$$\Omega = C^V, \quad f(x) = \left[\prod_{v \in V} a(x(v))\right] \left[\prod_{\{v,w\} \in E} b(x(v), x(w))\right]$$

This includes

- The Ising model
- Spin glasses
- Strauss model
- Kelly-Ripley interaction models
- Dimensions only interact locally

Strauss process

Like independent set, configuration is subset of nodes

- $x(v) = 1$ means in subset, $x(v) = 0$ means out
- $a(x(v)) = \lambda^{x(v)}$
- $b(x(v), x(w))$ is $\gamma < 1$ if $x(v) = x(w) = 1$
- Strauss density:

$$\lambda^{\#\text{ of occupied nodes}}\gamma^{\#\text{ of edges with both ends occupied}}$$

Using PRAR with Strauss

Changes to PRAR

- Initial value for v is 1 with probability $\lambda/(1+\lambda)$
- If $x(v) = 1$ proposed...
- ...only check neighbor with probability $1 - \gamma$

Running time polynomial

- Let Δ be the maximum degree of the graph
- Expected linear time algorithm if

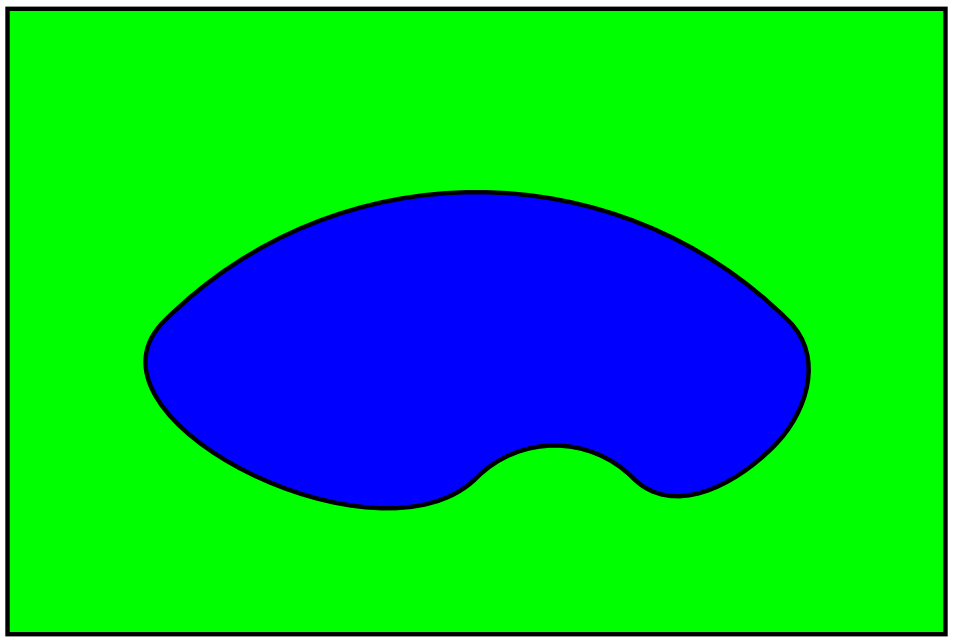
$$\frac{\Delta(1-\gamma)\lambda}{1+\lambda} < 1$$

Sequential Acceptance Rejection

Recursive Acceptance Rejection: Geometric view

Goal: sample from $B \subset A$

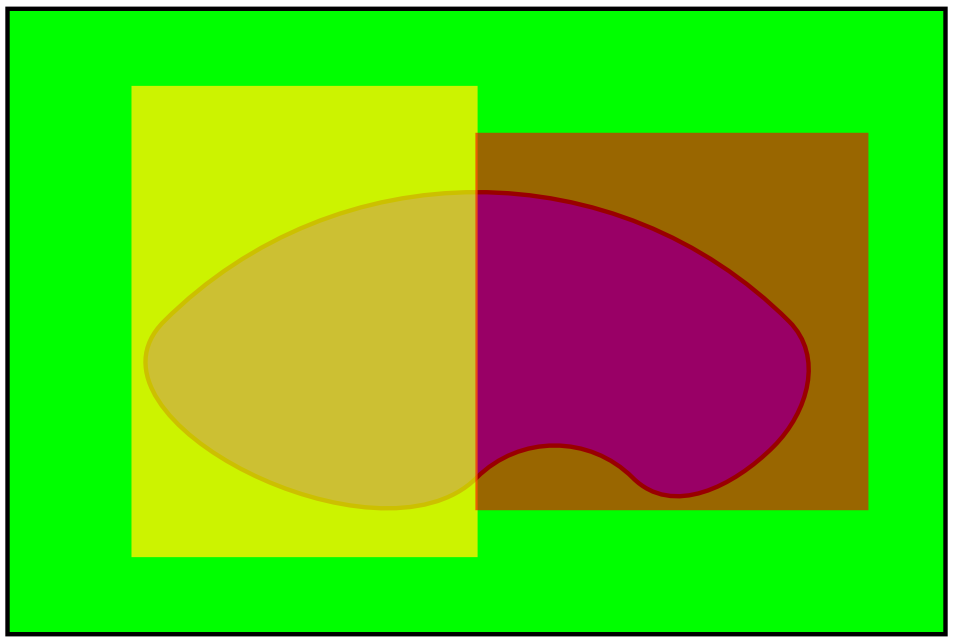
- Find disjoint $A_1, \ldots, A_n$ in A that cover B
- Let I be a random variable with $\mathbb{P}(I = i)$ to $\mu(A_i)/\mu(A)$ (otherwise $I = 0$)
- If $I = 0$ start over, else draw X recursively from A_I



Recursive Acceptance Rejection: Geometric view

Goal: sample from $B \subset A$

- Find disjoint $A_1, \ldots, A_n$ in A that cover B
- Let I be a random variable with $\mathbb{P}(I = i)$ to $\mu(A_i)/\mu(A)$ (otherwise $I = 0$)
- If $I = 0$ start over, else draw X recursively from A_I



Acceptance Rejection without bounding sets

Do not need the sets A_i!

- Just need the *sizes* of A_i to find $\mathbb{P}(I = i)$
- Partition B into $B_1, \ldots, B_n$
- Enough to have an upper bound on size of B_i

For Sequential Acceptance Rejection

- Need partition $B_1, \ldots, B_n$ of B
- Upper bound function f such that $\mu(B) \leq f(B)$
- f must respect partition: $f(B_1) + f(B_2) + \cdots f(B_n) \leq f(B)$

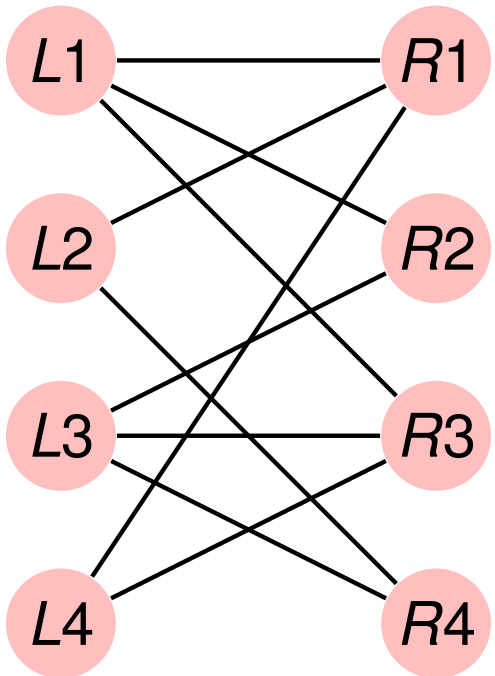
Result

Sequential Acceptance Rejection

- Gives method for class of problems where CFTP/RR/PRAR fail
- CFTP/RR/PRAR work well when Markov chain mixing provable by coupling
- Some Markov chains whose mixing time proved by conductance
- Example: Perfect matchings in graphs

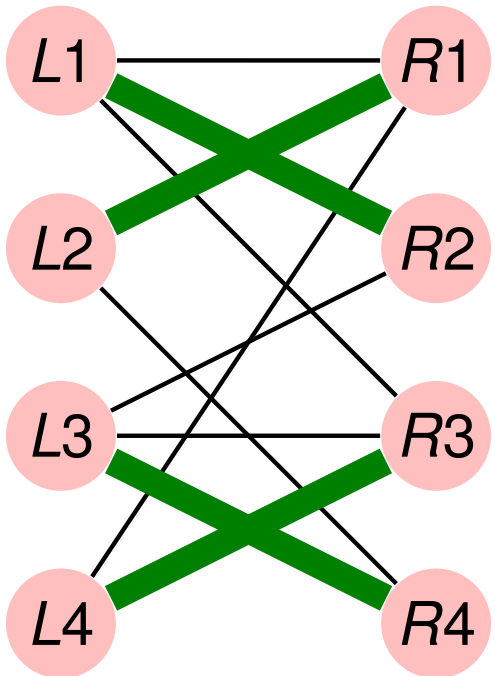
Perfect Matchings

A perfect matching in a graph is a collection of edges such that every node is adjacent to exactly one edge



Perfect Matchings

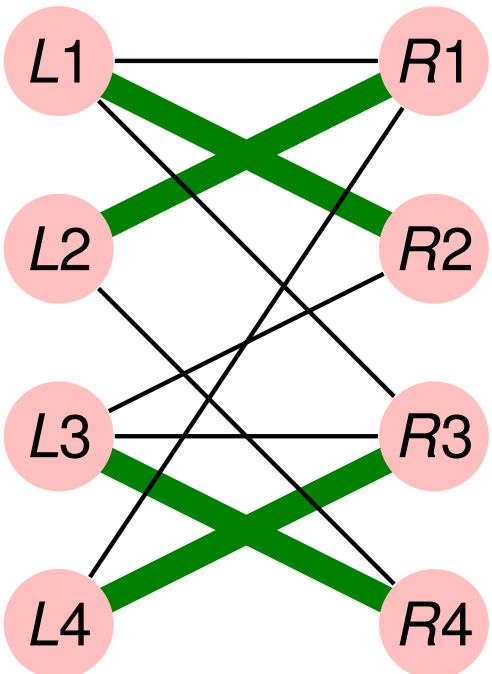
A perfect matching in a graph is a collection of edges such that every node is adjacent to exactly one edge



Encodings

More than one way to encode matching...

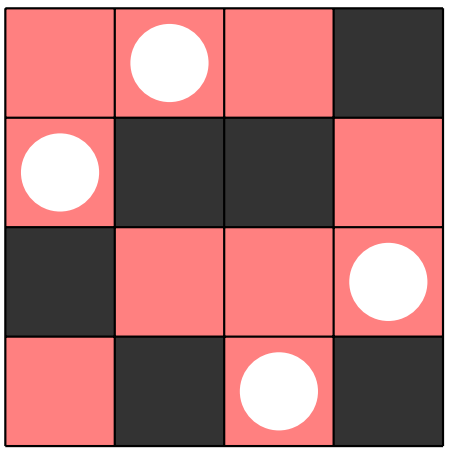
Perfect Matchings



Permutation

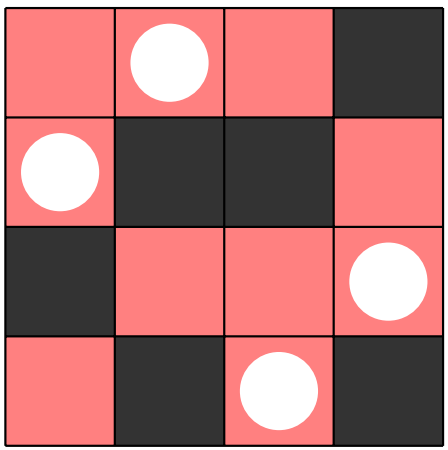
i	$\sigma(i)$
1	2
2	1
3	4
4	3

Rook Placement



More encodings....

Rook Placement



Matrix Encoding

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Number of valid rook placements = permanent of matrix A

The Permanent

Definition (Permanent)

The permanent of a matrix $A = (A(i,j))_{i,j=1}^{n}$ is

$$\text{per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A(i, \sigma(i))$$

Similar in form to determinant, but much harder to compute!

Applications

The permanent of nonnegative matrix arises in...

- Nonparametric statistics (permutation statistical models)
- Contingency tables
- Graphs with given degree sequence
- One of first #P-complete problems

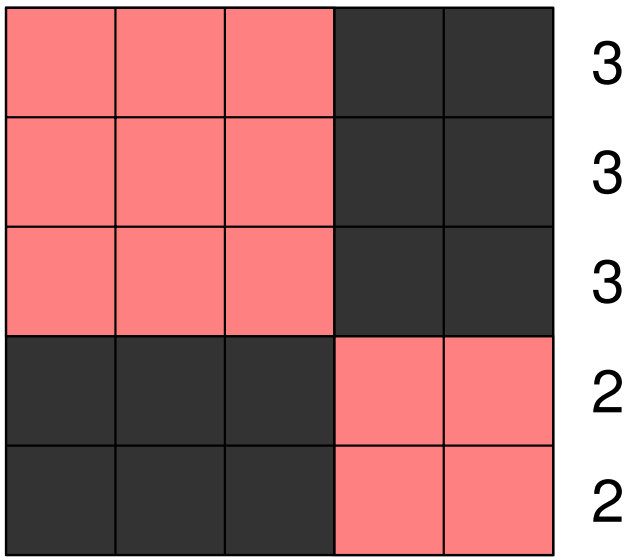
Goal

Basic Problem

- Count number of perfect matchings/restricted permutations/rook placements
- #P-complete problem, best can do is approximation
- Jerrum-Sinclair-Vigoda used Markov chain approach
- Even with improvements, still an $O(n^{10}(\log n)^4)$ algorithm
- With sequential AR get $O(n^4 \log n)$ but only for dense problems

To use SAR, need upper bounds on permanent

Consider the following example:



$$\text{per}(A) = 3!2!$$

Minc's Conjecture (1963)

For matrix A with fixed row sums $r(i)$ for i from one to n, this block structure maximizes the permanent

Of course, not all matrices in blocks

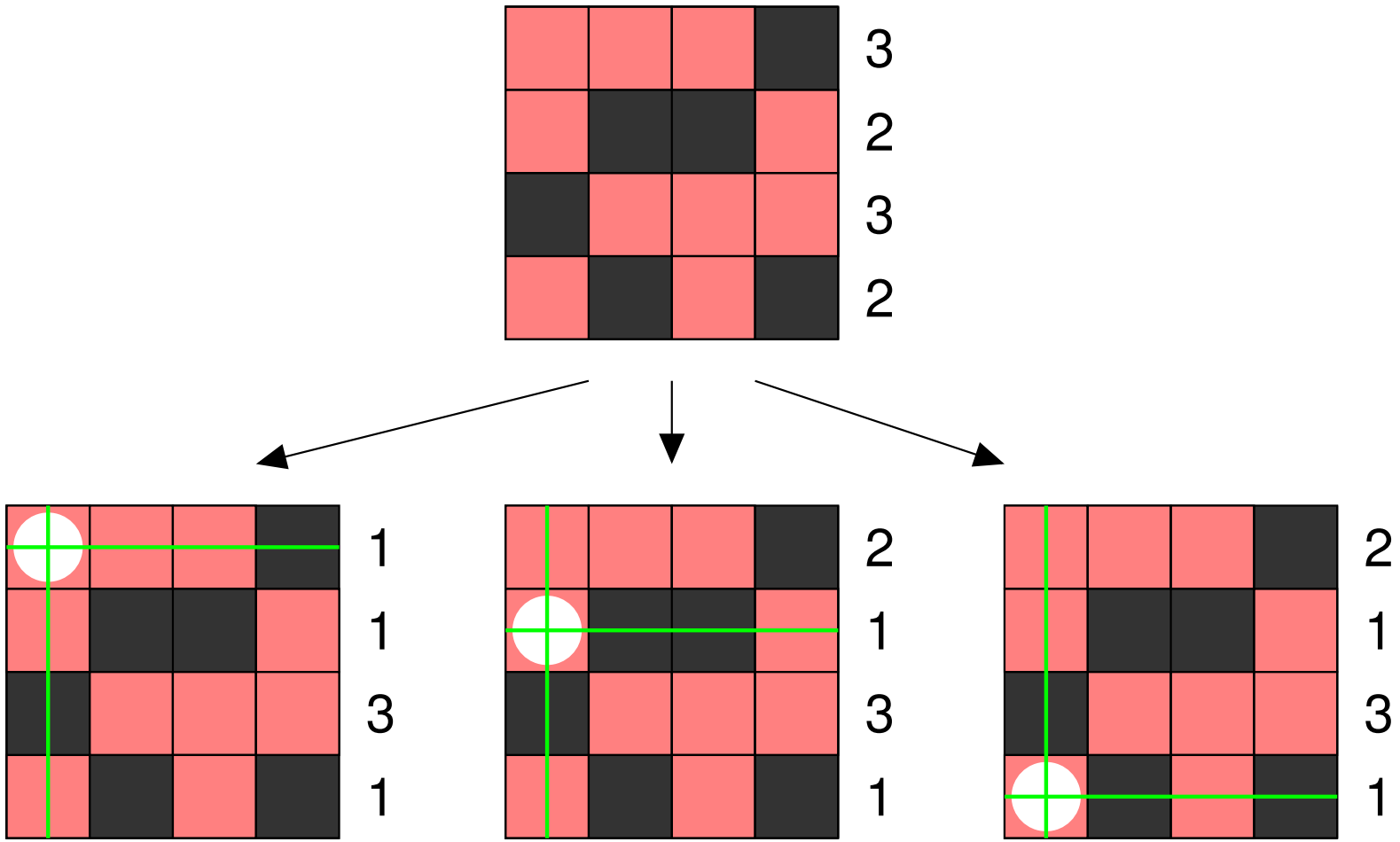
Bregman's Inequality (1973)

For matrices whose entries are 0 or 1:

$$\text{per}(A) \leq \prod_{i=1}^{n} [r(i)!]^{1/r(i)},$$

where $r(i)$ is the sum of row i of matrix A.

Problem: cannot prove via induction on minors



To work for upper bound...

Need

$$\text{bound}\begin{pmatrix}3\\2\\3\\2\end{pmatrix} \geq \text{bound}\begin{pmatrix}1\\1\\3\\1\end{pmatrix} + \text{bound}\begin{pmatrix}2\\1\\3\\1\end{pmatrix} + \text{bound}\begin{pmatrix}2\\1\\3\\1\end{pmatrix}$$

but for Minc/Bregman:

$$6.6038... \leq 1.81... + 2.57... + 2.57...$$

Understanding Minc's conjecture

Upper bound is:

$$f(r) = \prod_i g(r(i)), \text{ where } g(r) = (r!)^{1/r}$$

Use Stirling's inequality to understand $g(r)$:

$$g(r) \approx \frac{1}{e}\left[r + \frac{1}{2}\ln(r) + \frac{1}{2}\ln(2\pi)\right]$$

New upper bound

Theorem

For a matrix A with entries either 0 or 1 and row sums $r(i)$:

$$\mathrm{per}(A) \leq \prod_{i=1}^{n} h(r(i)),$$

where

$$h(r) = \frac{1}{e}\left[r + \frac{1}{2}\ln r + e - 1\right]$$

Moreover, this bound can be proved by induction on minors.

Example from earlier

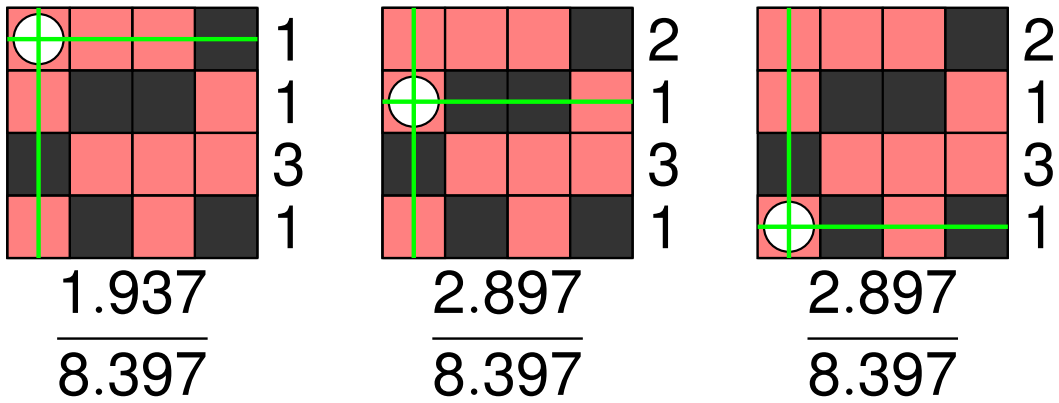
Need

$$\text{bound}\begin{pmatrix}3\\2\\3\\2\end{pmatrix} \geq \text{bound}\begin{pmatrix}1\\1\\3\\1\end{pmatrix} + \text{bound}\begin{pmatrix}2\\1\\3\\1\end{pmatrix} + \text{bound}\begin{pmatrix}2\\1\\3\\1\end{pmatrix}$$

and for new bound:

$$8.397... \geq 1.937... + 2.897... + 2.897 = 7.733...$$

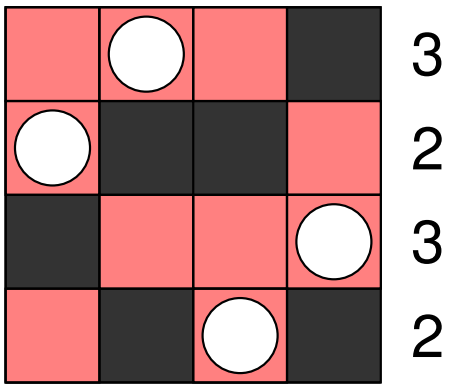
Sampling algorithm



Procedure

- Choose rook placement for first column...
- ...with probability bound of minor over original bound
- By Theorem, these ratios add to less than one
- If no placement chosen, reject and start over
- Otherwise continue until all rooks placed

Why does this work?



Probability of making this choice:

$$\frac{h(2,1,3,1)}{h(3,2,3,2)} \cdot \frac{h(1,1,2,1)}{h(2,1,3,1)} \cdot \frac{h(1,1,1,1)}{h(1,1,2,1)} \cdot \frac{h(1,1,1,1)}{h(1,1,1,1)} = \frac{1}{h(3,2,3,2)}$$

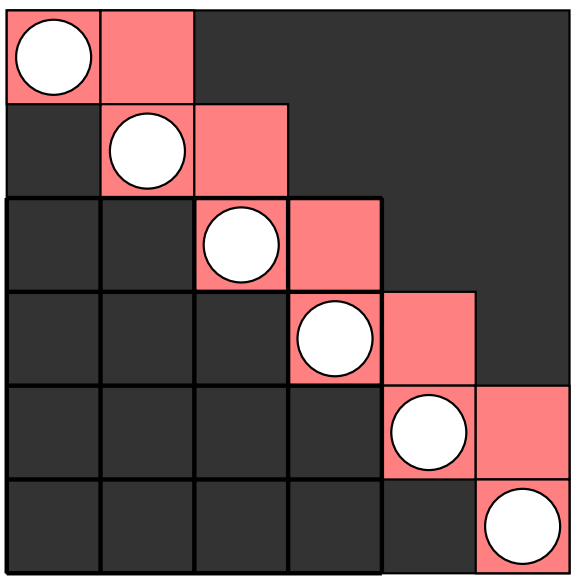
where $(3,2,3,2)$ are original row sums

How long does in take to run?

Average number of steps:

$$\frac{\text{bound}(\text{per}(A))}{\text{per}(A)}$$

Can be exponential in n
Example: A diagonal plus superdiagonal:



$$\text{per}(A) = 1$$

$$\text{bound}(\text{per}(A)) = (1.495\ldots)^{n-1}$$

Speeding things up

Upper and lower bounds

Regular graphs

- Minc: permanent large when 1's are clumped
- Van der Waerden: permanent small when entries spread out as much as possible

Lower bounding the permanent

Regular graphs

- Regular graphs have all degrees of all nodes are same
- Matrices all row and column sums are same, say r
- Van der Waerden lower bound for regular matrices

$$\operatorname{per}(A) \geq \operatorname{per} \begin{pmatrix} r/n & r/n & \cdots & r/n \\ r/n & r/n & \cdots & r/n \\ \vdots & \vdots & \cdots & \vdots \\ r/n & r/n & \cdots & r/n \end{pmatrix} = r^n \frac{n!}{n^n}$$

- Proved indep. by Egorychev (1981), Falikman (1981)

Use Stirling as before

For regular problems:

$$\sqrt{2\pi n}\left(\frac{r}{e}\right)^n \leq \text{per}(A) \leq \left(\frac{r+(1/2)\ln r + e - 1}{e}\right)^n$$

Looking at ratio of upper to lower bound:

$$\frac{\text{upper}}{\text{lower}} = \frac{1}{\sqrt{2\pi n}}\left(1 + \frac{1}{2r}\ln r + e - 1\right)^n \approx \frac{1}{\sqrt{2\pi n}} r^{n/2r}(e-1)^{n/r}$$

So if problem is dense:

$$r \geq \gamma n, \quad \gamma \in [0,1]$$

then runtime polynomial

Achieving regularity

Sinkhorn step

1. Divide each row by its row sum
2. Divide each column by its column sum

Repeat until nearly regular

- Row and col sums in $[1 - n^{-2}, 1 + n^{-2}]$ in $O(n^{4.5})$ steps
- Can be done in $O(n^4)$ using ellipsoid method

Sinkhorn in action

$$
\begin{array}{cccc|c}
1 & 1 & 1 & 0 & 3 \\
1 & 0 & 0 & 1 & 2 \\
0 & 1 & 1 & 1 & 3 \\
1 & 0 & 1 & 0 & 2 \\
\hline
3 & 2 & 3 & 2 &
\end{array}
\quad \rightarrow \quad
\begin{array}{cccc|c}
1/3 & 1/3 & 1/3 & 0 & 1 \\
1/2 & 0 & 0 & 1/2 & 1 \\
0 & 1/3 & 1/3 & 1/3 & 1 \\
1/2 & 0 & 1/2 & 0 & 1 \\
\hline
4/3 & 2/3 & 7/6 & 3/6 &
\end{array}
$$

$$
\rightarrow \quad
\begin{array}{cccc|c}
1/4 & 1/2 & 2/7 & 0 & 29/28 \\
3/8 & 0 & 0 & 3/5 & 39/40 \\
0 & 1/2 & 2/7 & 2/5 & 73/70 \\
3/8 & 0 & 3/4 & 0 & 45/56 \\
\hline
1 & 1 & 1 & 1 &
\end{array}
$$

After many Sinkhorn steps

.2069	.5450	.2479	0	≈ 1
.3381	0	0	.6618	≈ 1
0	.4549	.2069	.3381	≈ 1
.4549	0	.5450	0	≈ 1
1	1	1	1	

This has permanent about .1359

Making rows as large as possible

Last scaling of rows

- Want row sums as large as possible
- Divide each row through by its maximum entry
- All entries still in $[0, 1]$

.3797	1	.4549	0	1.8346
.5108	0	0	1	1.5108
0	1	.4549	.7432	2.1982
.8346	0	1	0	1.8346

The Theorem/Method

Theorem

For a matrix A with entries in $[0,1]$ *and row sums* $r(i)$*:*

$$\text{per}(A) \leq \prod_{i=1}^{n} h(r(i)),$$

where

$$h(r) = \begin{cases} e^{-1}[r + \frac{1}{2}\ln r + e - 1] & r > 1 \\ e^{-1}[1 + (e-1)r] & r \leq 1 \end{cases}$$

Moreover, this bound can be proved by induction on minors.

Summary: Speeding things up

What to do

- Regularize matrix using Sinkhorn steps or ellipsoid method
- Van der Waerden's inequality lower bounds permanent

Theorem (H.,Law 2008)

If original row sums at least γn *for* $\gamma \in (.5, 1]$*, then*

$$\frac{\text{bound}(\text{per}(A))}{\text{per}(A)} \leq n^{-.5+.5/(2\gamma-1)}.$$

Conclusions

Perfect simulation

- Many different protocols now exist for designing algorithms
- Partial Recursive Acceptance Rejection fast on graphs
- Sequential Acceptance Rejection fast for scalable problems

Open questions

- Is PRAR faster or slower than CFTP for, say, Ising?
- Can the permanent be reduced to dense problem?

References

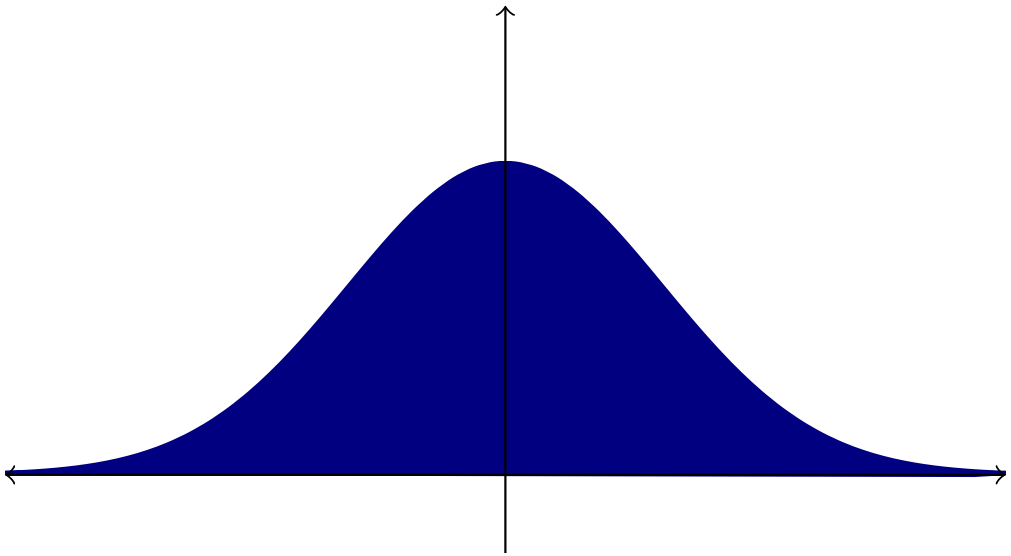
M. Jerrum, L. Valiant, and V. Vazirani.
Random generation of combinatorial structures from a uniform distribution
Theoret. Comput. Sci., **43**, 169–188, 1986

M. Huber.
Exact Sampling from perfect matchings from dense regular bipartite graphs
Algorithmica, **44**, 183–193, 2006

M. Huber and J. Law.
Fast approximation of the permanent for very dense problems
Proc. of Syposium on Discrete Algorithms, 681–689, 2008

Standard normal random variables

X

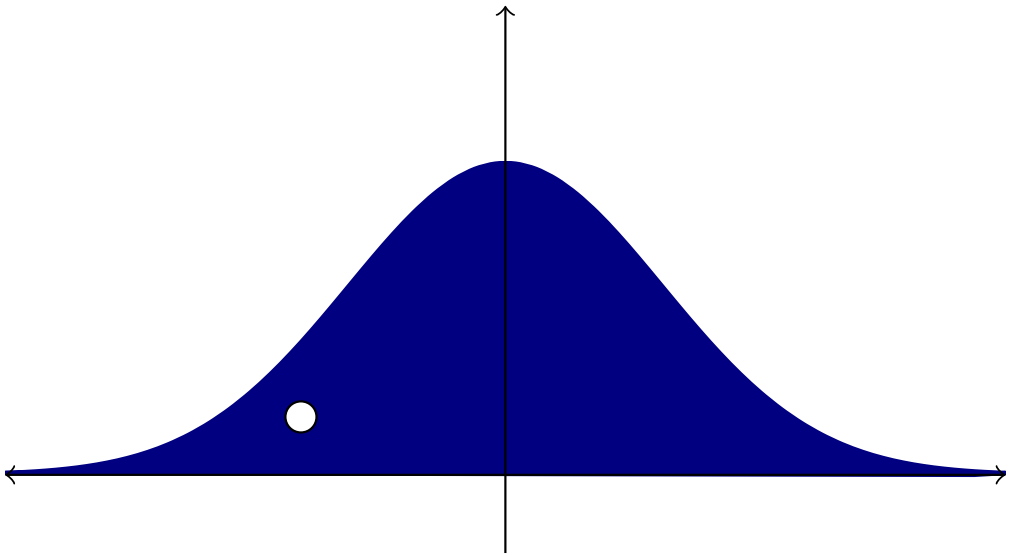


Restart

Finish

Standard normal random variables

X

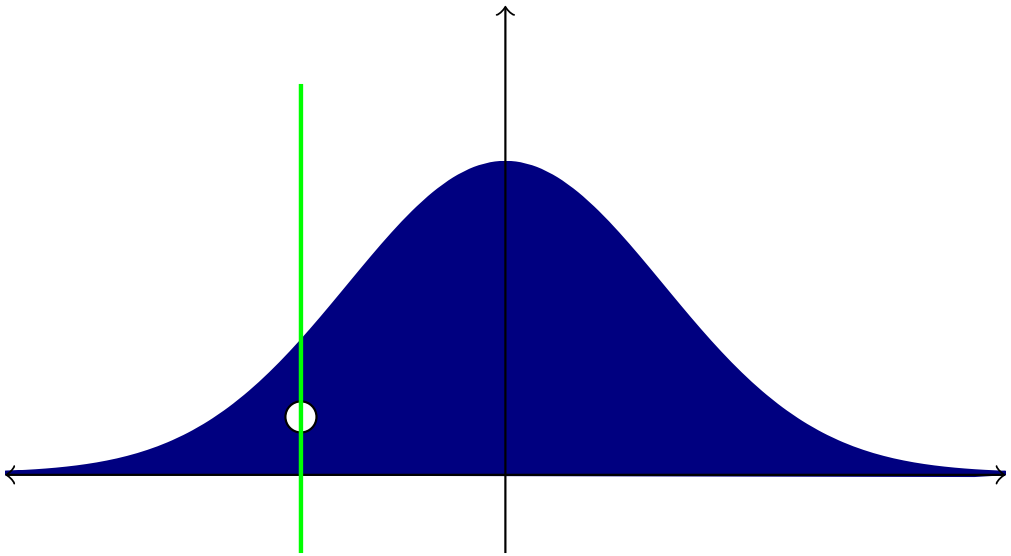


▸ Restart

▸ Finish

Standard normal random variables

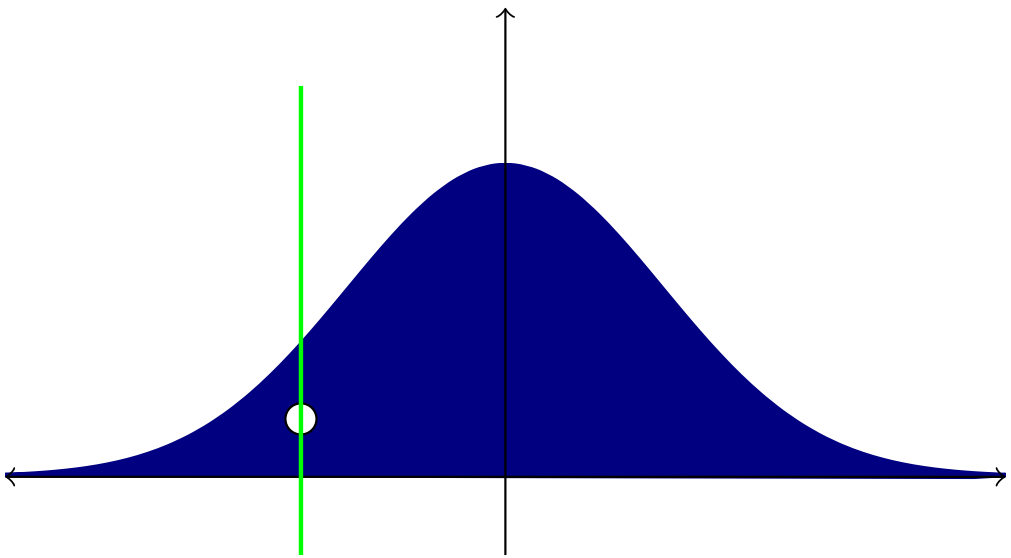
X



Restart

Finish

Standard normal random variables

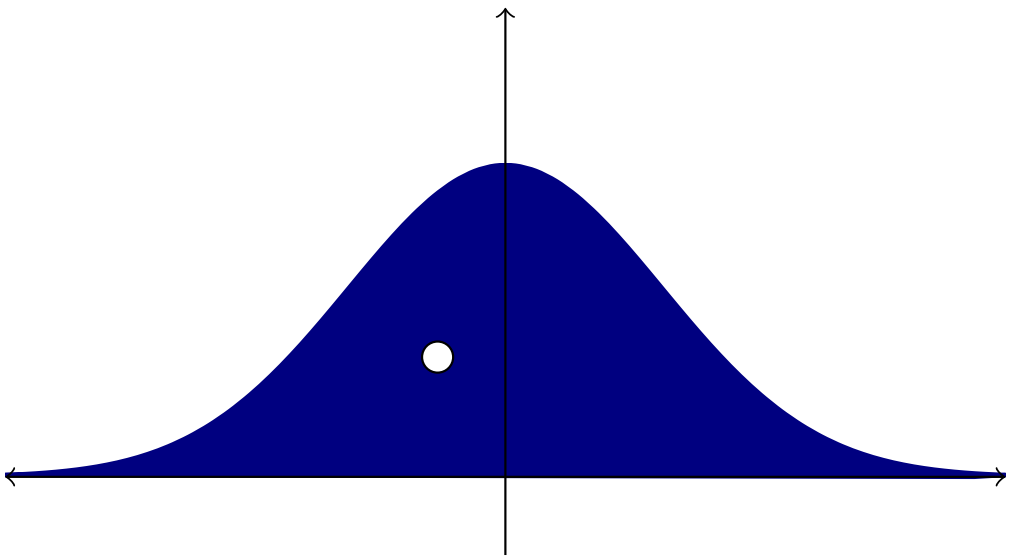


X
-1.3077

▸ Restart

▸ Finish

Standard normal random variables

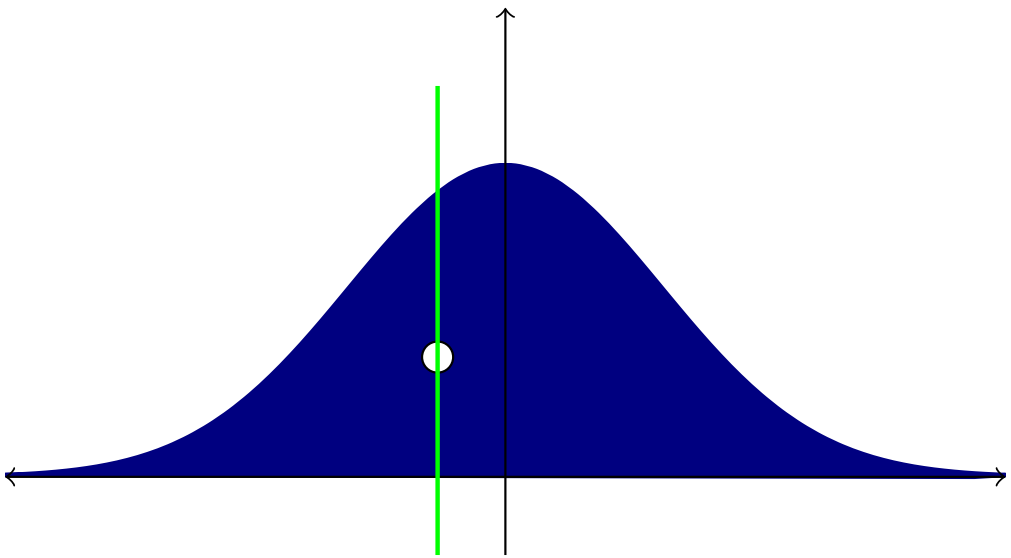


X
-1.3077

▸ Restart

▸ Finish

Standard normal random variables

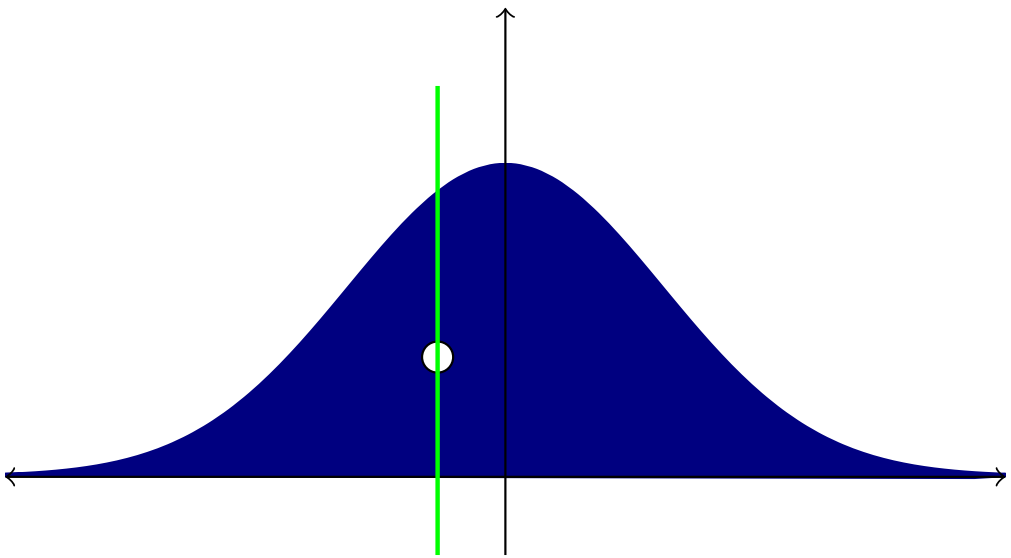


X
-1.3077

▸ Restart

▸ Finish

Standard normal random variables

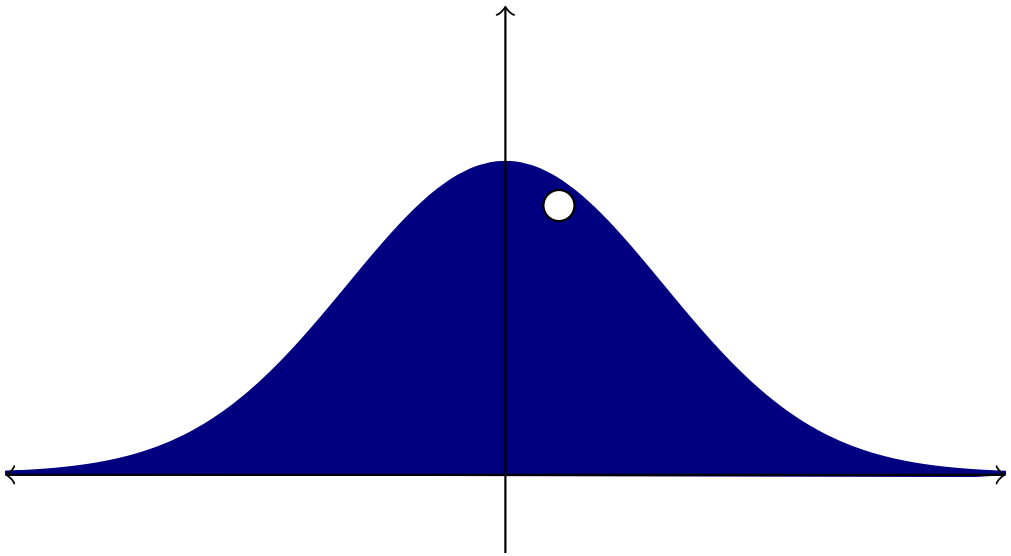


X
-1.3077
-0.4336

▸ Restart

▸ Finish

Standard normal random variables

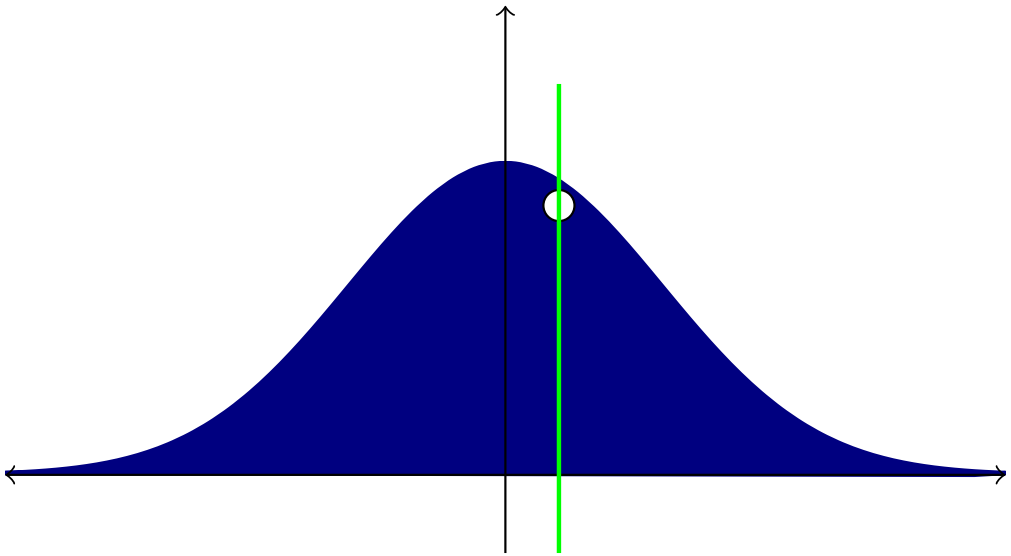


X
-1.3077
-0.4336

▸ Restart

▸ Finish

Standard normal random variables

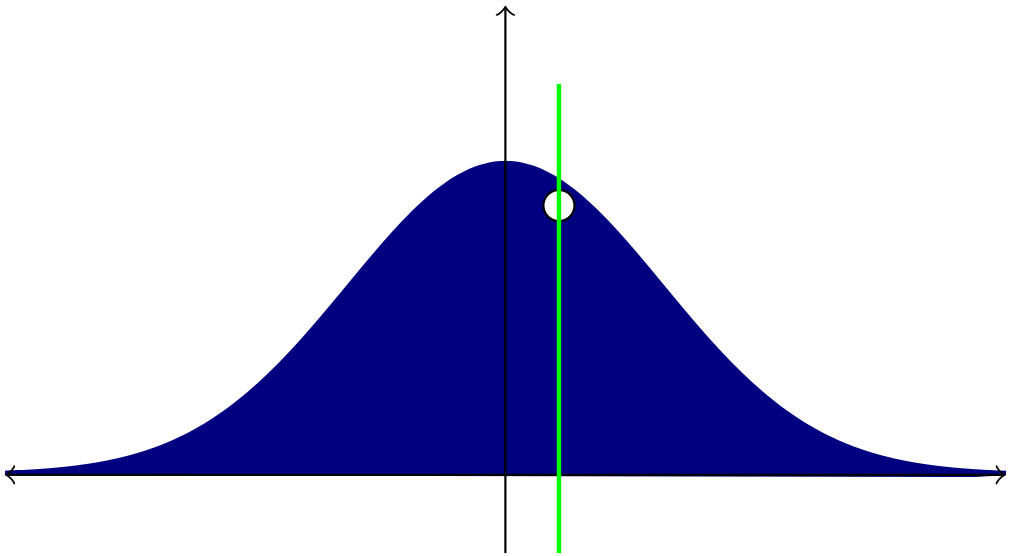


X
-1.3077
-0.4336

▸ Restart

▸ Finish

Standard normal random variables

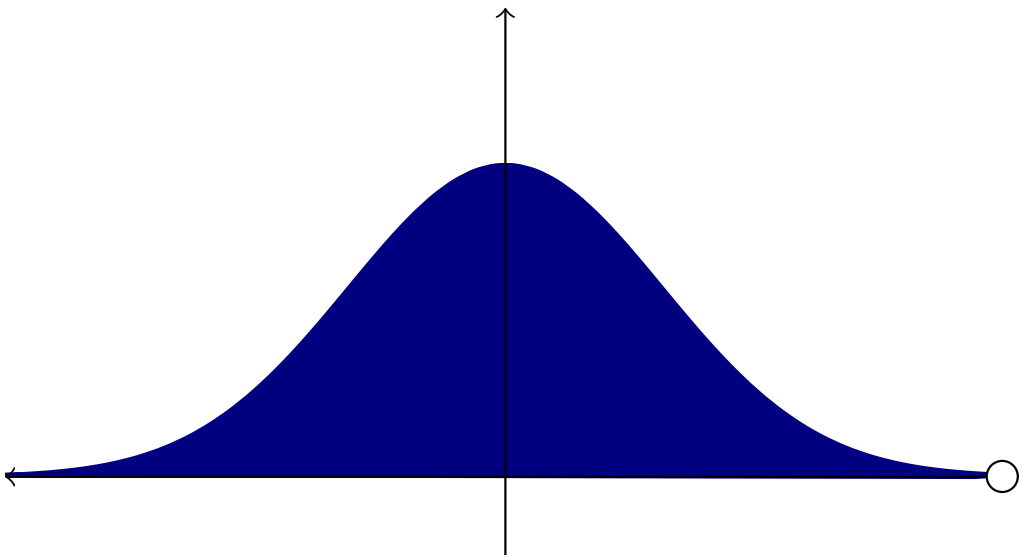


X
-1.3077
-0.4336
0.3426

▸ Restart

▸ Finish

Standard normal random variables

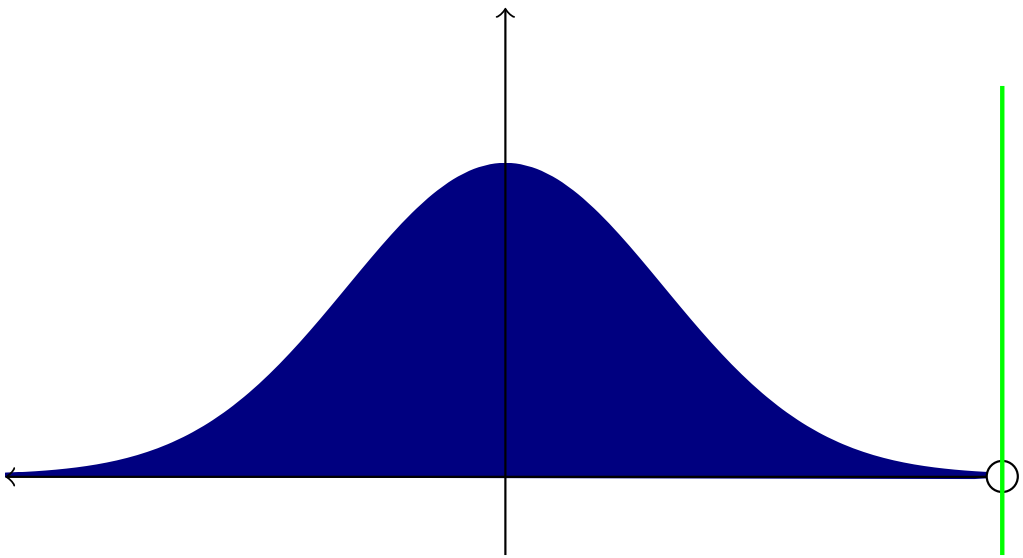


X
-1.3077
-0.4336
0.3426

▸ Restart

▸ Finish

Standard normal random variables

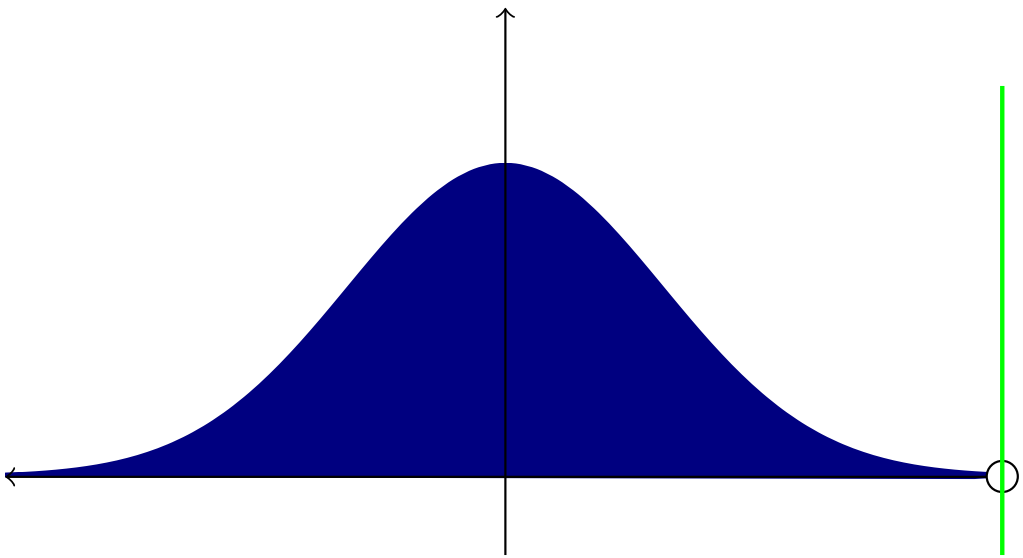


X
-1.3077
-0.4336
0.3426

▸ Restart

▸ Finish

Standard normal random variables

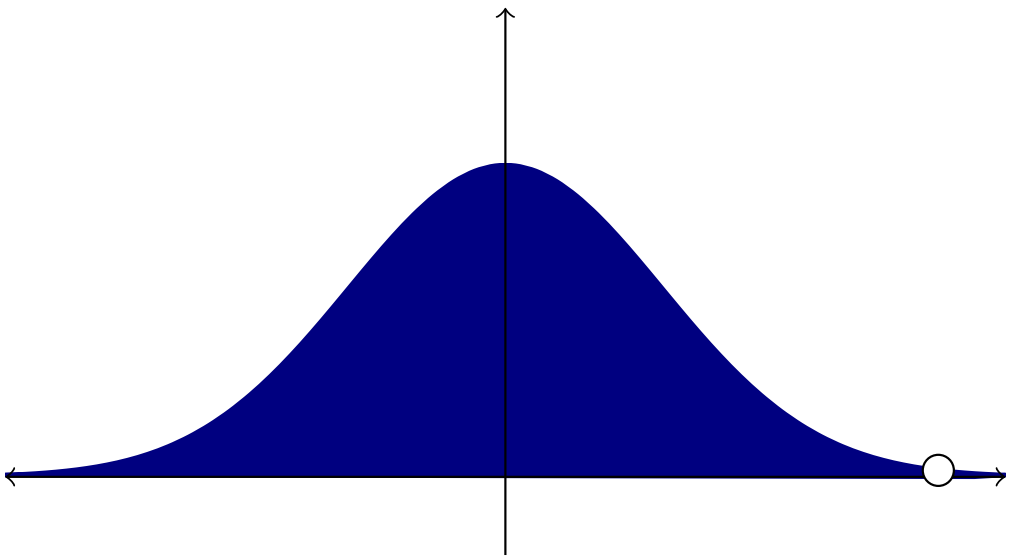


X
-1.3077
-0.4336
0.3426
3.1784

▸ Restart

▸ Finish

Standard normal random variables

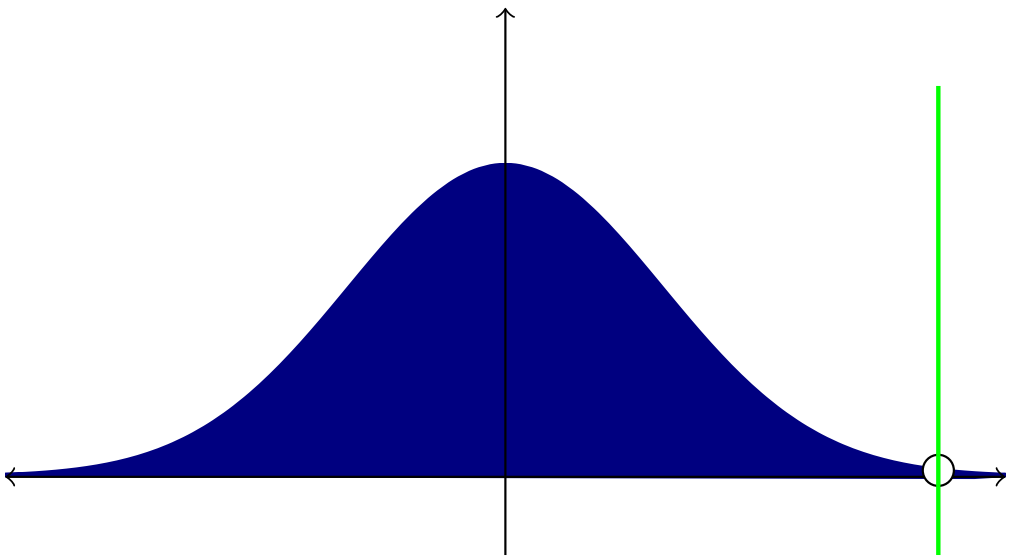


X
-1.3077
-0.4336
0.3426
3.1784

▸ Restart

▸ Finish

Standard normal random variables

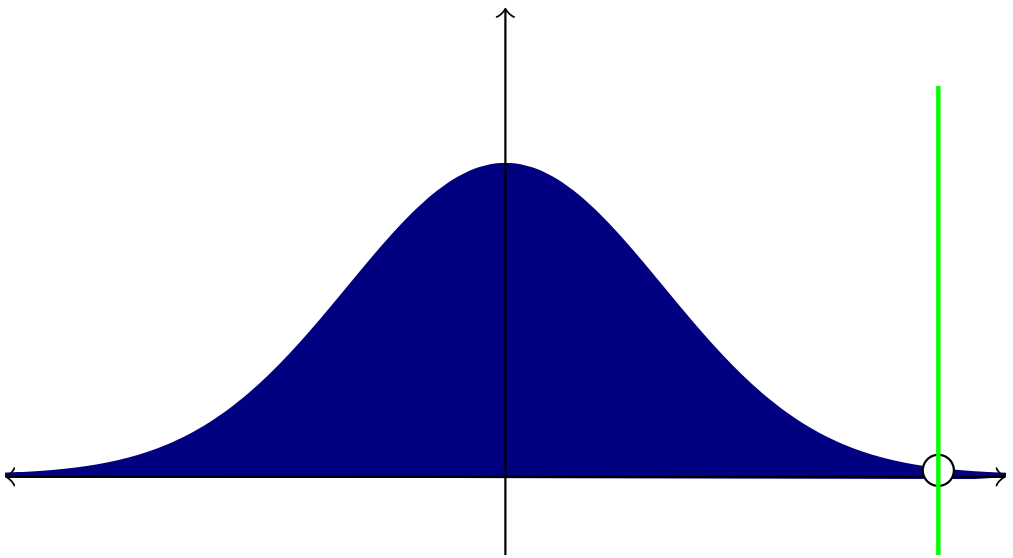


X
-1.3077
-0.4336
0.3426
3.1784

▸ Restart

▸ Finish

Standard normal random variables

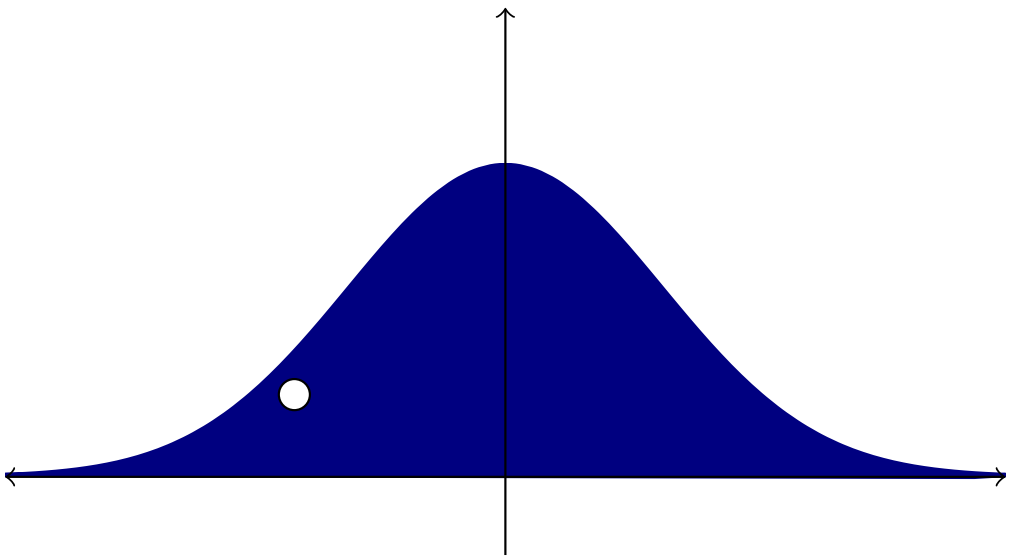


X
-1.3077
-0.4336
0.3426
3.1784
2.7794

Restart

Finish

Standard normal random variables

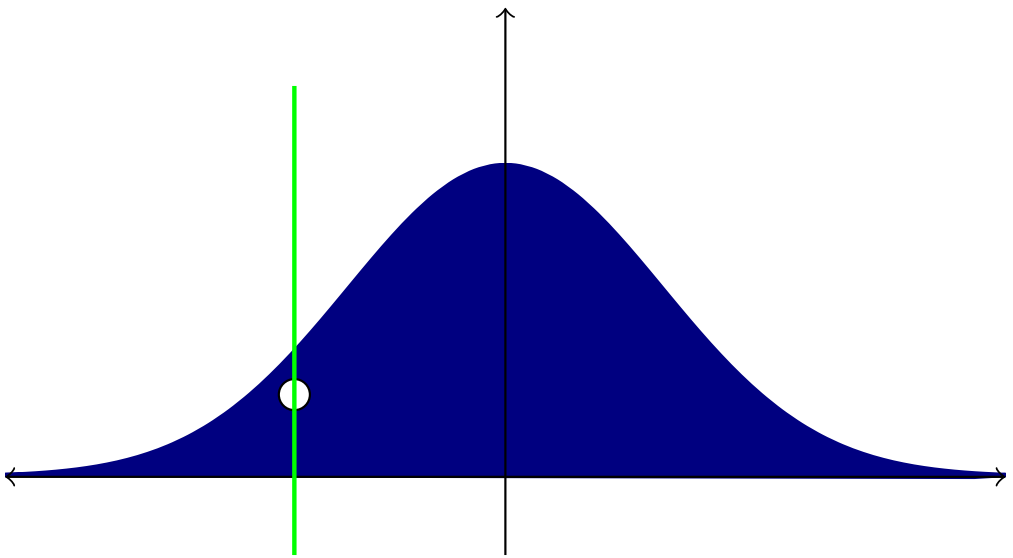


X
-1.3077
-0.4336
0.3426
3.1784
2.7794

▸ Restart

▸ Finish

Standard normal random variables

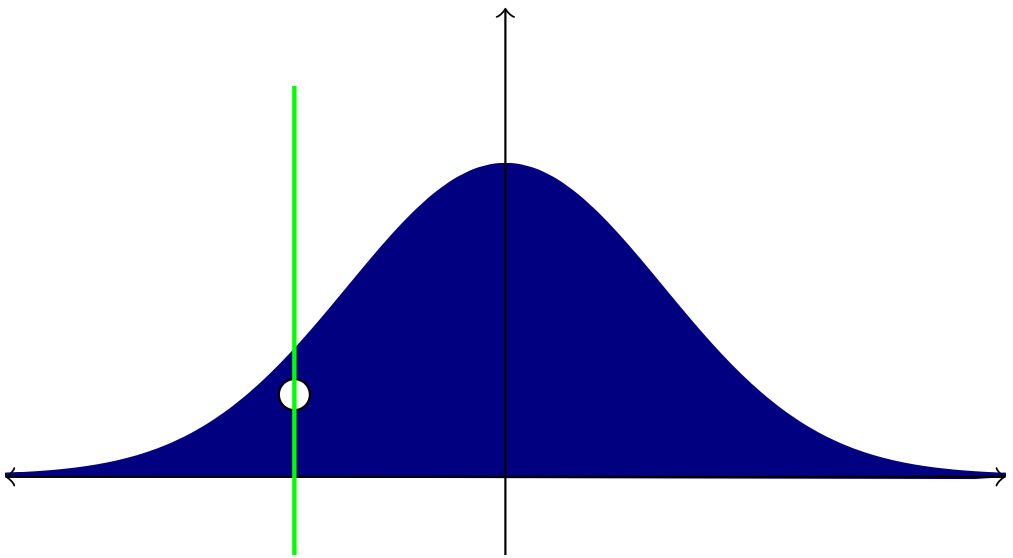


X
-1.3077
-0.4336
0.3426
3.1784
2.7794

Restart

Finish

Standard normal random variables

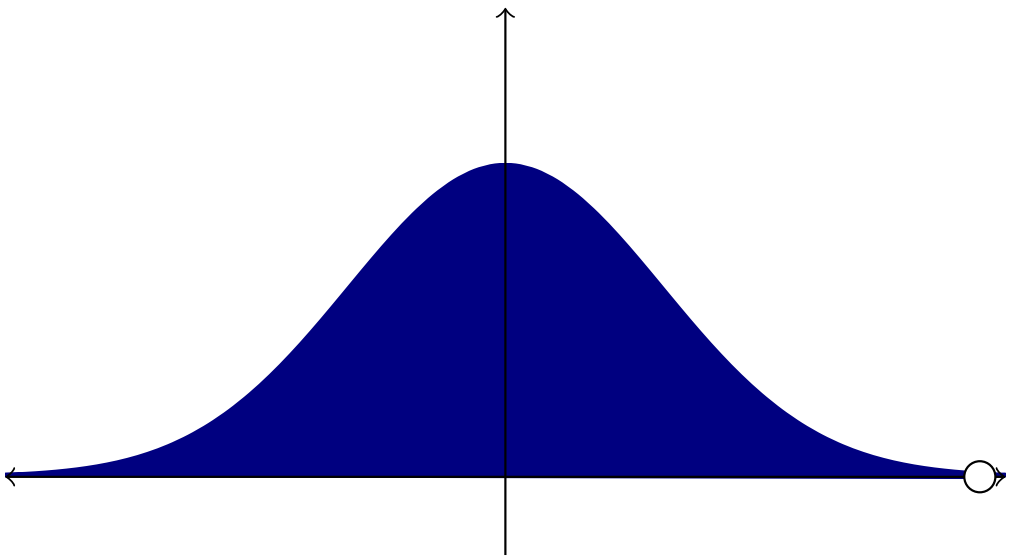


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499

Restart

Finish

Standard normal random variables

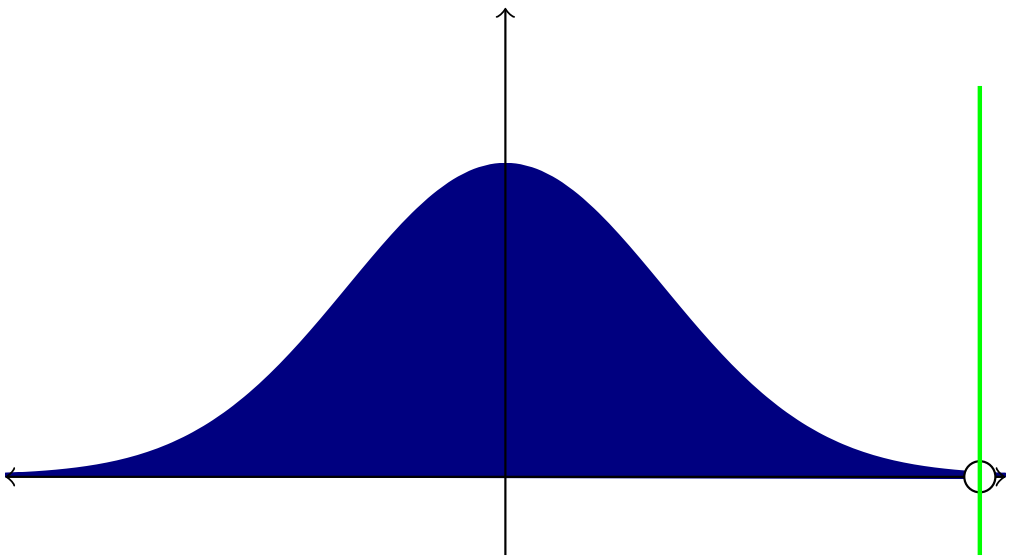


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499

▸ Restart

▸ Finish

Standard normal random variables

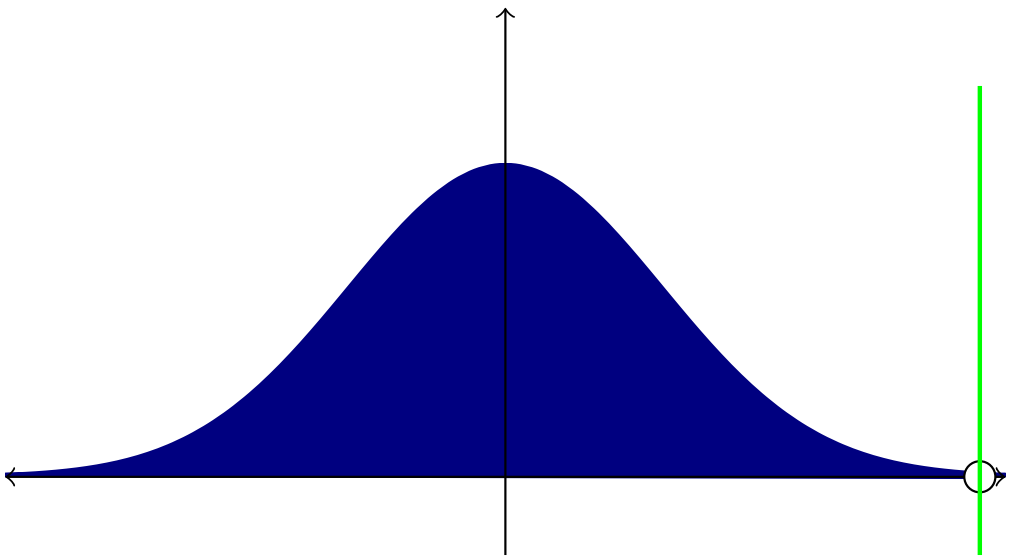


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499

Restart

Finish

Standard normal random variables

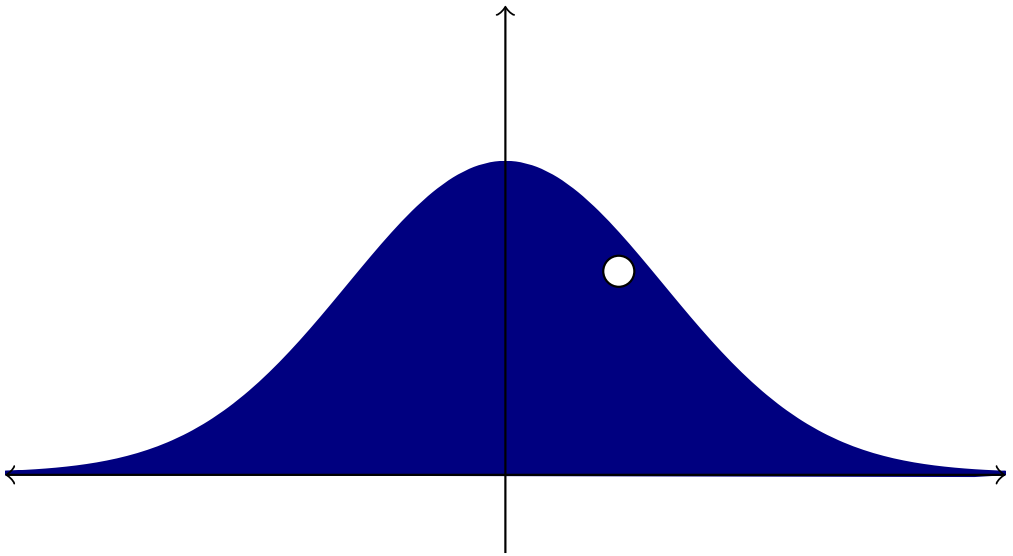


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349

▸ Restart

▸ Finish

Standard normal random variables

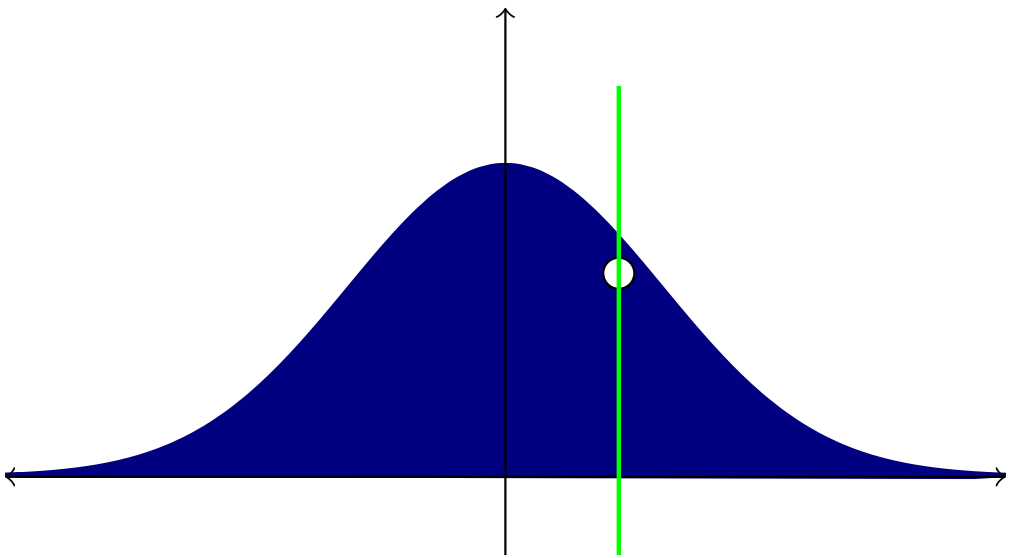


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349

▸ Restart

▸ Finish

Standard normal random variables

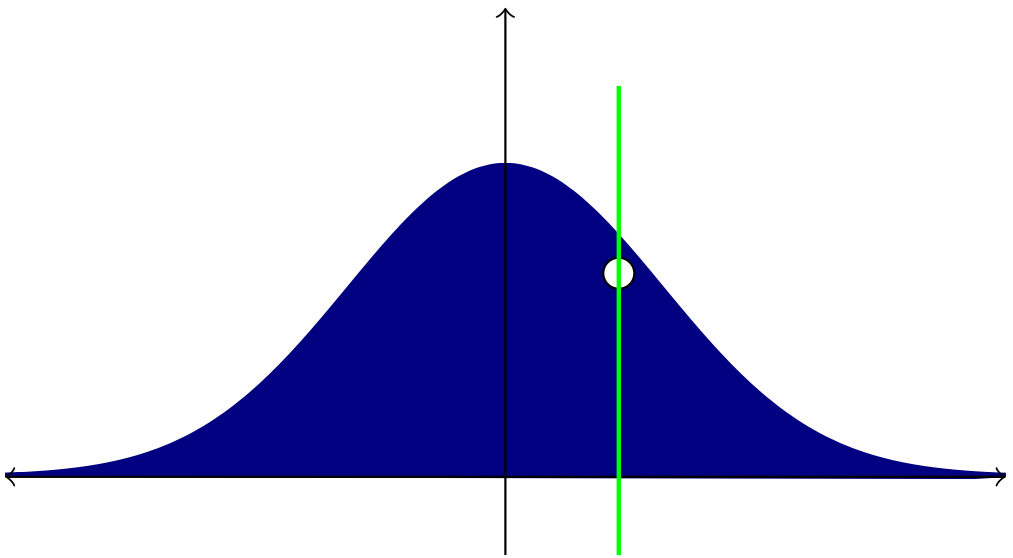


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349

▸ Restart

▸ Finish

Standard normal random variables

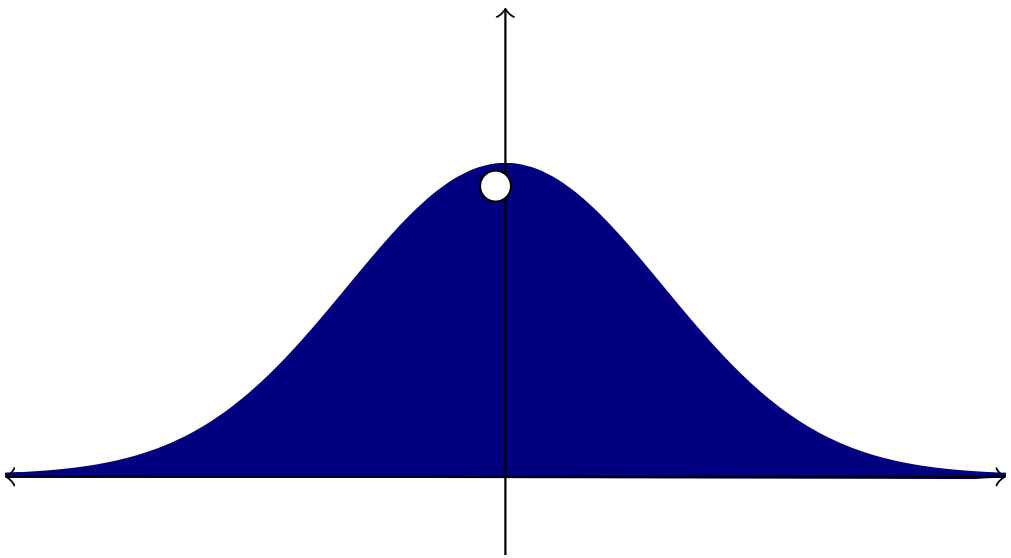


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254

▸ Restart

▸ Finish

Standard normal random variables

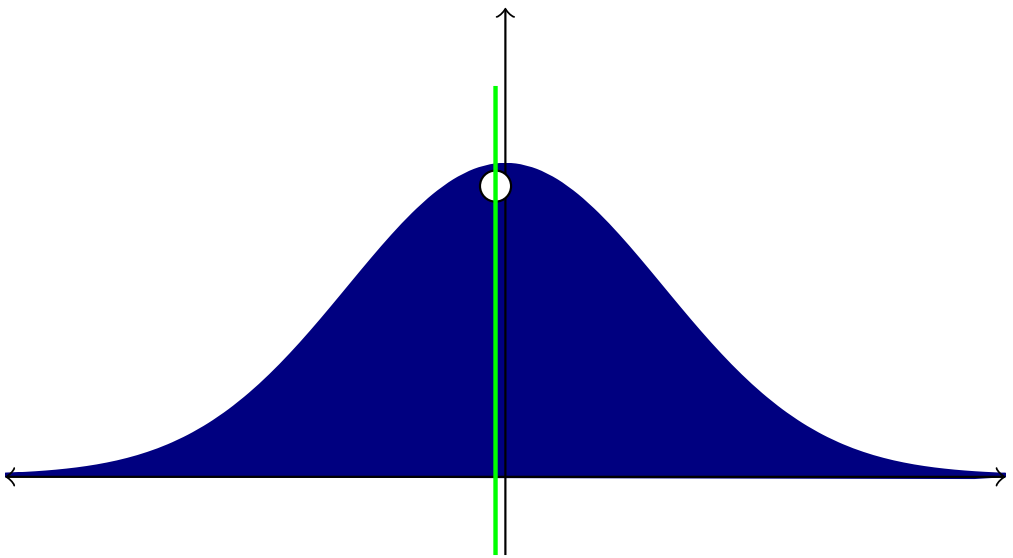


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254

▸ Restart

▸ Finish

Standard normal random variables

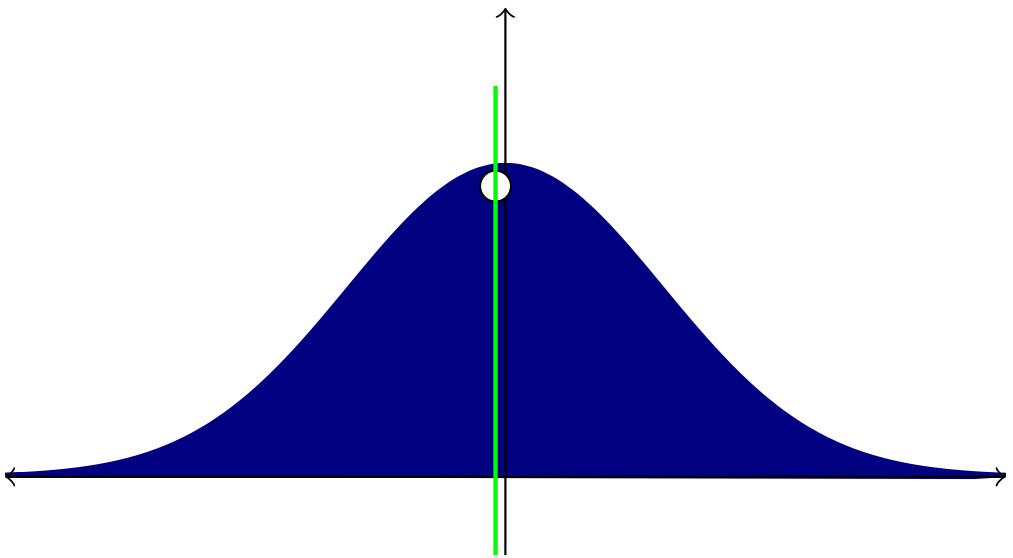


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254

Restart

Finish

Standard normal random variables

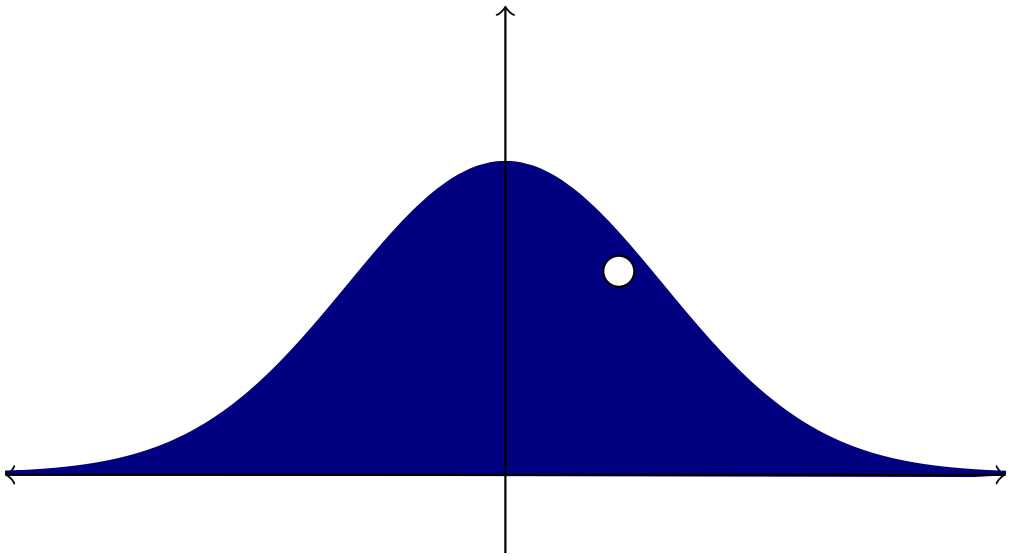


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254
-0.0631

▸ Restart

▸ Finish

Standard normal random variables

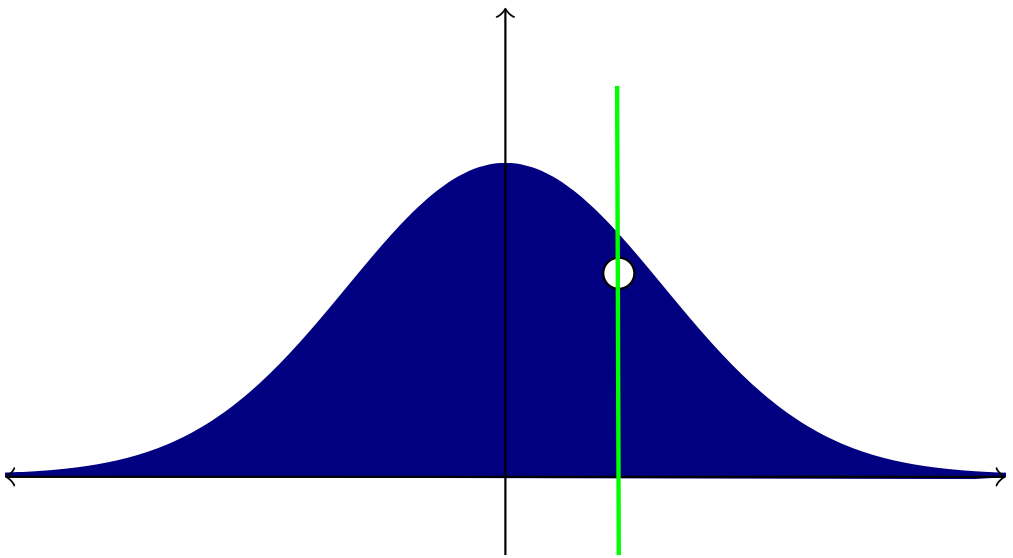


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254
-0.0631

▸ Restart

▸ Finish

Standard normal random variables

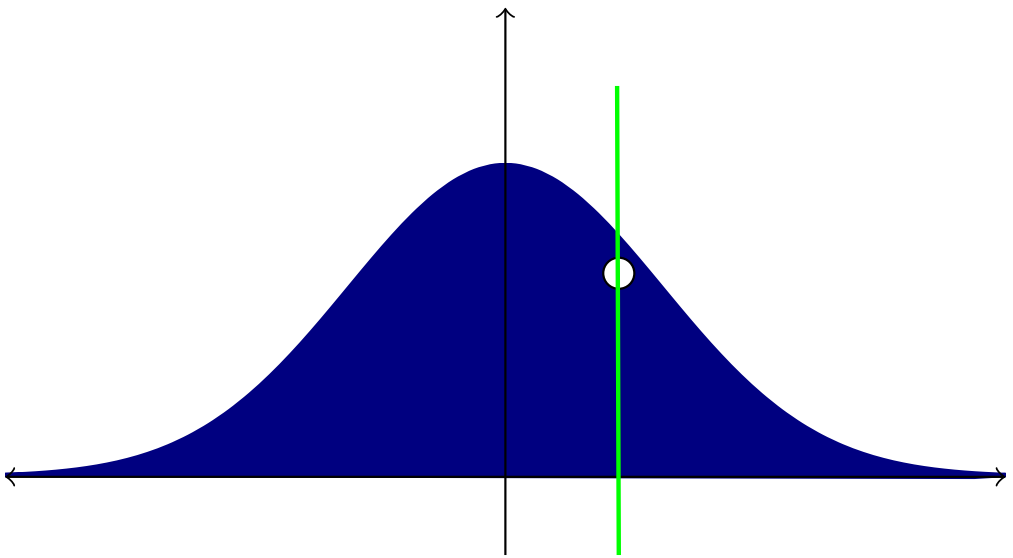


X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254
-0.0631

▸ Restart

▸ Finish

Standard normal random variables



X
-1.3077
-0.4336
0.3426
3.1784
2.7794
-1.3499
3.0349
0.7254
-0.0631
0.7147

Restart

Finish