Drawing Random Samples From Statistical Distributions

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2017

Where do random samplings come from?

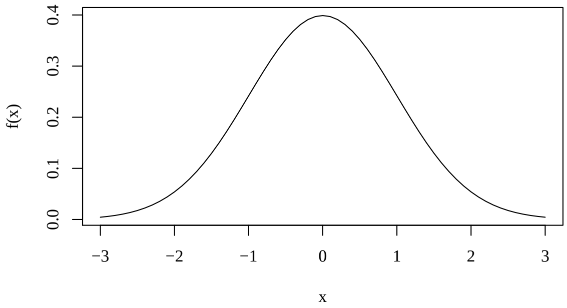
- Analytical solutions for a few distributions (ones that have invertible CDF)
- Approximate computational solutions for the rest

Where Do We Start?

- Assume we have formulae for distributions from which we want to draw samples
- Assume we have a random generator that can give us random integers on [0, MAX]
- Assume that the random generator is a good one, either MT19937 or one of L'Ecuyer's parallel stream generators.
- Of course, do whatever book keeping necessary to assure perfect replication

PDF

- Probability Density Function, $f(x)$
- PDF
- Note small letter used for PDF

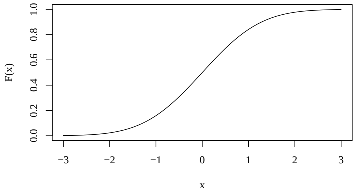


$F(x)$ Cumulative Distribution Function (CDF) is S-Shaped

- CDF: Area on left of point x

$$F(k) = \int_{-\infty}^{k} f(x)dx \;\; or \;\; F(x) = \int_{-\infty}^{x} f(e)de$$

- Used e for dummy variable of integration.
- Note CAPITAL letter used for CDF
- CDF is always "S shaped"



- Some people may be confused about usage of x in $f(x)$ and $F(x)$. Sometimes I write $F(x_{upper})$ or $F(k)$ to clear that up

The U[0,1] is Obtained Easily from Generator

- A Uniform draw on $0, 1$ is obtained:

$$\begin{aligned} x &\sim \textit{random integer from}\,[0, MAX] && (1) \\ y &= \frac{x}{MAX} && (2) \end{aligned}$$

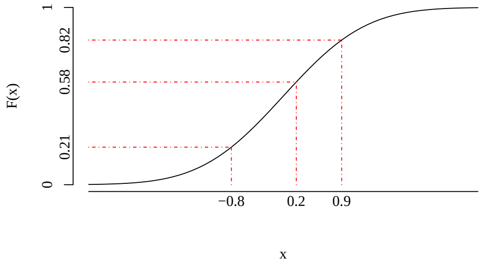
- From that, can get Bernoulli *Heads*, *Tails* sample. If $y > 0.5$, then *Heads*

Other Distributions Require More Thought

- Inversion method:
  - Works easily if we can calculate "quantiles" (meaning the CDF is invertible).
  - If CDF can be closely approximated, an approximate "look-up table" can be created (R's Normal)
- Rejection Sampling
  - Find some other similar PDF that is easier to calculate
  - Use algorithm to test "candidates" and keep ones that fit
- Composition, MCMC, and other methods are not worked out in these notes.

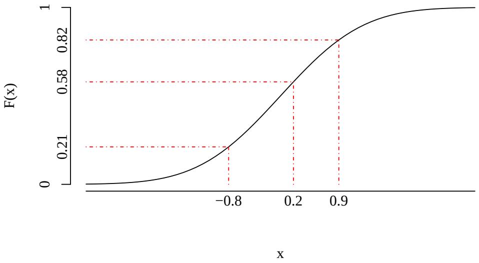
Inversion

- Consider a CDF.
- What does the left hand side mean?
  - Fraction of cases smaller than that point.
  - Think "backwards" to find x that corresponds.



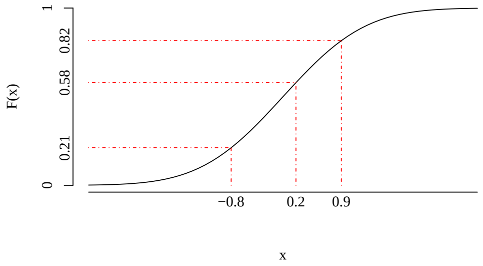
Concept behind Inversion

- An "equally likely" draw from $f(x)$ would have this property:
  - All points on the vertical axis between [0,1] are going to be equally likely. Right?
- (Otherwise, a randomly drawn x wouldn't really be random, eh?)



Inversion Algorithm

- Inversion method
  - draw a random $u \sim Uniform[0,1]$
  - "Think Backwards" to get corresponding $x = F^{-1}(u)$
- Collect a lot of those, and you've got a sample from $f(x)$



Logistic: Easy Inversion

Recall the Logistic PDF, with “location” parameter μ and “scale” parameter σ:

$$f(x) = \frac{1}{\sigma}\frac{exp(-\frac{x-\mu}{\sigma})}{(1+exp(-\frac{x-\mu}{\sigma}))^2} \tag{3}$$

In the usual cases we discuss, $\mu = 0$ and $\sigma = 1.0$, so the pdf in question is simpler.
And CDF:

$$F(x) = \frac{exp(\frac{x-\mu}{\sigma})}{1+exp(\frac{x-\mu}{\sigma})} = \frac{1}{1+e^{-(x-\mu)/\sigma}}. \tag{4}$$

Calculate Logistic Inverse CDF

- Figure for each probability density output value y, find the x that corresponds to it via F:

$$y = \frac{1}{1 + e^{-(x-\mu)/\sigma}}$$

- Through the simple algebra used in derivation of Logistic Regression

$$ln\left[\frac{y}{1-y}\right] = (x-\mu)/\sigma$$

- So

$$x* = \mu + \sigma \cdot ln\left[\frac{y}{1-y}\right]$$

Use That For Inversion Sampling

- Draw $u \sim U[0,1]$
- Calculate $x* = \mu + \sigma \cdot ln\left[\frac{u}{1-u}\right]$
- And, as they say on Shampoo instructions, Repeat.

Limitations of Inversion

- Inversion is only practical when we have a formula for F^{-1} that is easy to calculate.
- Logistic distribution, for example, has an easy formula.
- Normal, and many others, DO NOT.
- So we must either
  - Approximate F^{-1} in order to conduct inversion
  - Find some other algorithm.

Rejection Sampling

- $f(x)$: The pdf from which we want to draw a sample
- $f(x)$: is some complicated formula, we can't calculate its CDF or inverse CDF. That means we have no obvious way to sample from $f(x)$
- But we can calculate the PDF at any given point, and that turns out to be the magic bullet.

General Idea behind Rejection Sampling

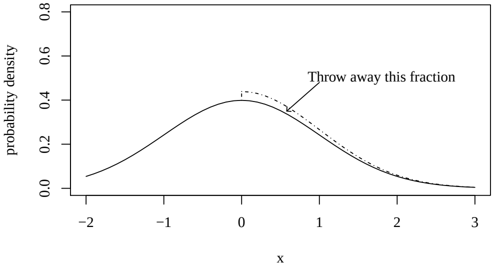
- $r(x)$ is different from $f(x)$ in some understandable way.
- So, draw samples from $r(x)$
- then keep the ones you need.
- How do you know when to throw away a draw from $r(x)$? That's the trick!

Start with an Easy Case

- Suppose
  - When $x < 0$, $r(x) = f(x)$.
  - When $x \geq 0$, $r(x) = 1.1 \cdot f(x)$
- For now, don't worry if such an $r(x)$ exists, because it doesn't. But it really makes the point clear.
- Draw a "candidate" random number $x*$ from r. Should we keep it?

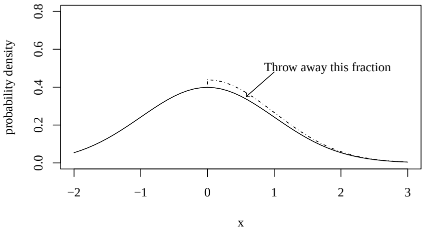
Illustration

- If $x* < 0$, accept it as a representation of $f(x)$
- When $x < 0$, r and f coincide, so we can keep all of those draws.



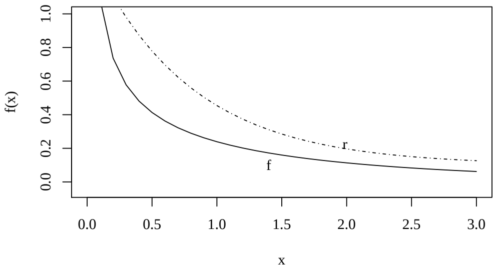
Illustration, if $x* \geq 0$

- Most of the time we want to keep that observation, since φ and r coincide most of the time.
- Where r and f "overlap", we want to keep $x*$
- That happens with probability $f(x*)/r(x*) = f(x*)/(1.1 * f(x*)) = 1/1.1$
- So, with probability $1/1.1 = 0.9090909\ldots$, we keep $x*$
- Otherwise, throw it away and draw another.



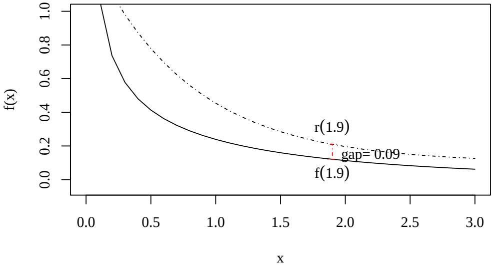
More Realistic Case

- Assume $r(x)$ is always bigger than $f(x)$ (by definition)
- A draw from $r(x)$ might be something like a draw from $f(x)$.



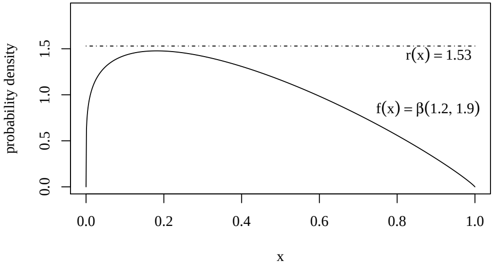
Check Out The Size of That Gap!

- The probability of drawing $x* = 1.9$ can be calculated from $r(x)$ and $f(x)$
- Keep $x*$ with probability $f(1.9)/r(1.9)$.
- Amounts to "throwing away" a "gap sized fraction" of candidate draws equal to 1.9



Realism and Rejection

- This procedure wastes computational effort
- "Works" even if r is not like f at all, but is just more wasteful



- If we draw a candidate $x* = 0.2$, we are likely to keep it
- If we draw a candidate $x* = 0.9$, we are almost always going to throw it away because $r(0.9)/f(0.9)$ is very large.

The rgamma distribution uses rejection sampling I

- Rejection uses the random number stream unpredictably.
- Sometimes, it takes just a few pulls on the stream to get a gamma draw, sometimes can take a lot more.
- Discussed in vignette with portableParallelSeeds "PRNG-basics.pdf"
- Look at row 2, which is the position in the random stream at which we are positioned after a draw.

```
RNGkind ("Mersenne-Twister")
set.seed(12345)
invisible(rgamma(1, shape = 1)); v1 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v2 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v3 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v4 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v5 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v6 <- .Random.seed[1:4]
cbind(v1, v2, v3, v4, v5, v6)
```

The rgamma distribution uses rejection sampling II

```
                 v1          v2          v3          v4
                             v5          v6
[1,]            403         403         403         403
                403         403
[2,]              2           4           7           9
                 11          16
[3,] −1346850345 −1346850345 −1346850345 −1346850345
     −1346850345 −1346850345
[4,]   656028621   656028621   656028621   656028621
     656028621   656028621
```

Repeat

```
invisible(rgamma(1, shape = 1)); v1 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v2 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v3 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v4 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v5 <- .Random.seed[1:4]
invisible(rgamma(1, shape = 1)); v6 <- .Random.seed[1:4]
cbind(v1, v2, v3, v4, v5, v6)
```

The rgamma distribution uses rejection sampling III

```
                v1          v2          v3          v4
                              v5          v6
[1,]           403         403         403         403
             403         403
[2,]            19          21          24          26
              28          31
[3,] -1346850345 -1346850345 -1346850345 -1346850345
  -1346850345 -1346850345
[4,]   656028621   656028621   656028621   656028621
  656028621   656028621
```

```
v <- vector(mode = "integer", length = 1000)
for(i in 1:10000){
  invisible(rgamma(1, shape = 1)); v[i] <- .Random.seed[2]
 }
vd <- diff(v)
table(vd)
```

The rgamma distribution uses rejection sampling IV

```
vd
-622 -621 -619 -617 -615 -614 -613 -611 -609 -607 -605 -603
    -601 -599    2    3    5    7    8    9
  10   25    2    2    3    1    1    1    3    1    1    1
          1    1 4937 3836  316  267   32  138
  10   11   12   13   14   15   16   17   18   19   20   21
          22   23   24   25   26   27   29   30
  25  111   13   54   10   46   23   28    8   24   12   19
           5   12    1    7    1    5    4    1
  32   33   34   35   36   38   39
   2    2    1    1    3    1    1
```

There's some distracting wrap around when the counter hits 624 and goes back to 1. But the point is clear enough. Often, gamma takes 2 or 3 draws from the stream, while we see 20 or 30 draws sometimes.