Numerical Methods

JEFF GILL
Distinguished Professor
Departments of Government and Mathematics and Statis
American University

Motivation

- Sampling, randomization, and permutation are important for survey research, genetics analysis, and epidemiology.
- These tools allow us to control random processes with counting and drawing.
- Most of these ideas are mathematically simple and easy to do in `R`.

Using R to Generate Normals

```
pnorm(1.5, mean=0, sd=1)
[1] 0.9331928

pnorm(4.75, mean=4, sd=1/2)
[1] 0.9331928

qnorm(c(0.25,0.5,0.75))
[1] -0.6744898  0.0000000  0.6744898

pnorm(1) - pnorm(-1)
[1] 0.6826895

1 - 2*pnorm(-2)
[1] 0.9544997

diff(pnorm(c(-3,3)))
[1] 0.9973002
```

Generating Distributional Quantities in General

- Four different queries:

`d dist (x,<parameters>)`	density at x
`p dist (x,<parameters>)`	cumulative distribution funtion for x
`q dist (p,<parameters>)`	inverse cdf
`r dist (n,<parameters>)`	generates n random numbers from distribution

- Common distributions included:

`<dist>`	Distribution	Parameters	Defaults
`beta`	beta	shape1, shape2	-, -
`cauchy`	Cauchy	location, scale	0, 1
`chisq`	chi-square	df	-
`exp`	exponential	rate	-
`f`	F	df1, df2	-, -
`gamma`	Gamma	shape, rate, scale = 1/rate	-
`lnorm`	log-normal	mean, sd (of log)	0, 1
`logis`	logistic	location, scale	0, 1
`norm`	normal	mean, sd	0, 1
`t`	Student's t	df	-
`unif`	uniform	min, max	0, 1

Uniform Random Variables

- As noted in the table the uniform generator in `R` is the function `runif`.
- The only necessary entry is the number of values to be generated.
- The other optional parameters are `min` and `max`, used according to:

  ```
  runif(100, min=2, max=5)
  ```

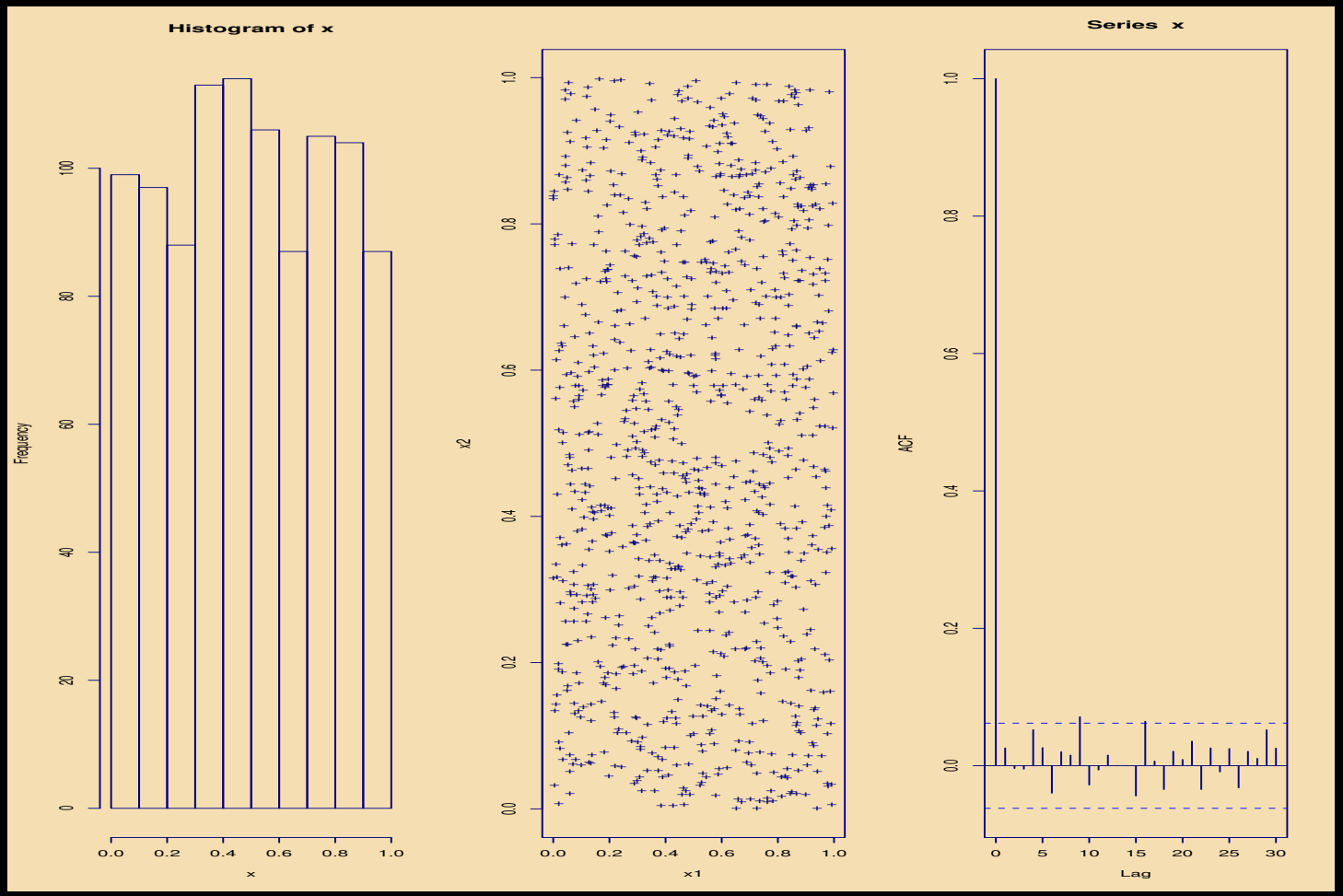
  which will produce 100 random variables $U(2, 5)$.

Uniform Random Variables

- A easy check on the properties of this uniform generator is to:
  - look at a histogram of the X_i's,
  - plot the pairs $(X_i, X_i + 1)$,
  - look at the estimate autocorrelation function.
- Consider:

```
Nsim=10^3
x=runif(Nsim)
x1 <- x[-Nsim]     # THESE TWO LINES 'OFFSET' THE TWO
x2 <- x[-1]        # VECTORS BY ONE INDICE
par(mfrow=c(1,3),bg="wheat",fg="navy")
hist(x)
plot(x1,x2,pch="+")
acf(x)
```

Uniform Random Variables



Uniform Random Variables

- Note that `runif` does not involve randomness per se.
- It is a deterministic sequence based on a random starting point.
- The `R` function set.seed can produce the same sequence:

```
set.seed(1)
runif(5)
[1] 0.2655087 0.3721239 0.5728534 0.9082078 0.2016819
set.seed(1)
runif(5)
[1] 0.2655087 0.3721239 0.5728534 0.9082078 0.2016819
set.seed(2)
runif(5)
[1] 0.0693609 0.8177752 0.9426217 0.2693818 0.1693481
```

- Setting the seed determines all the subsequent values.

Make the Computer Do the Work

- Often you can get the right answer faster with `R` than by analytics.
- For instance, what is the 90% confidence interval for $N(6,3)$?

```
n.data <- rnorm(1000000,6,3)
sort(n.data)[c(50000,950000)]
[1]  1.0697 10.9266
```

or

```
mean(n.data) + c(-1,1)*qnorm(0.95)*sd(n.data)
[1]  1.0662 10.9299
```

or

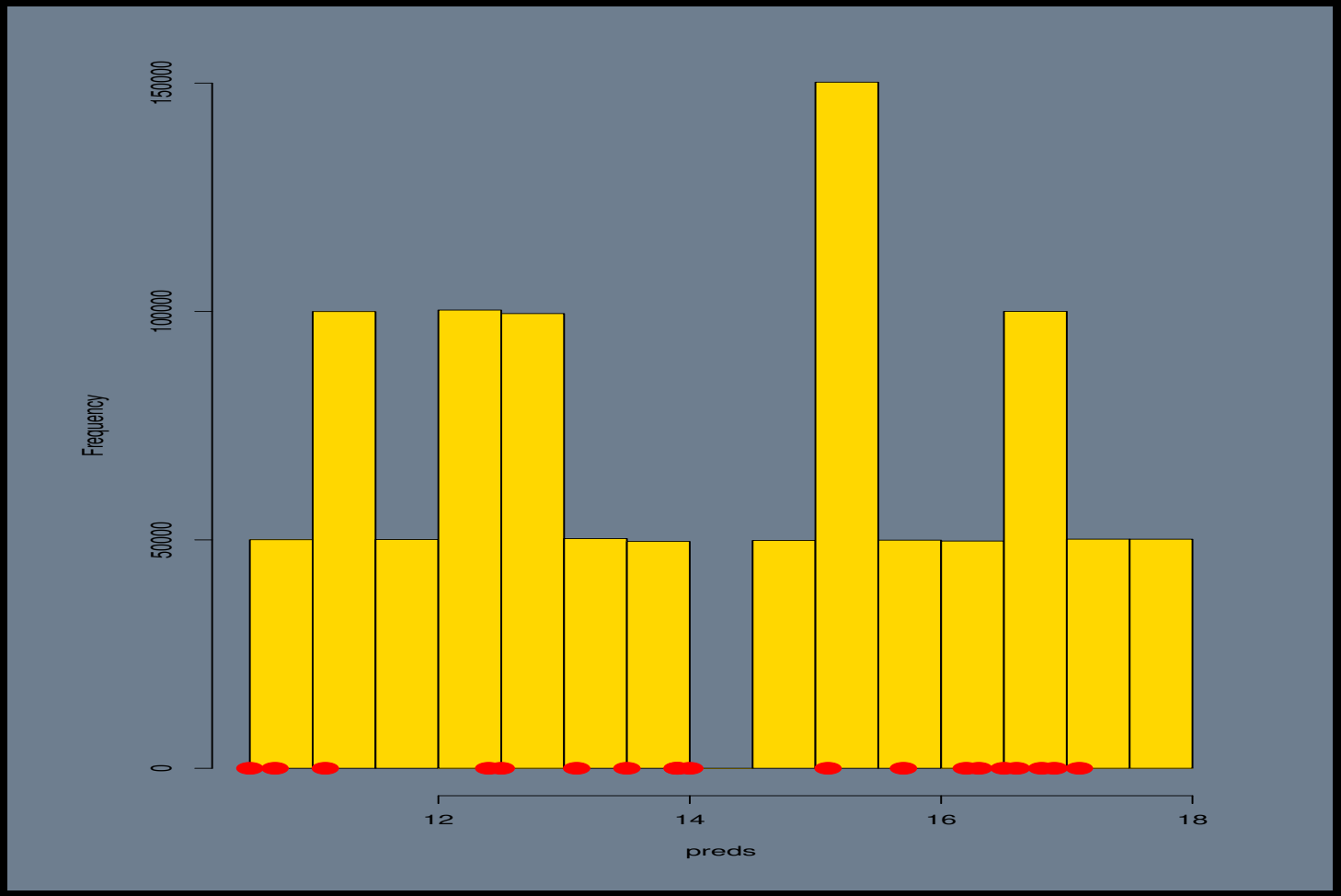
```
qnorm(c(0.05,0.95),6,3)
[1]  1.0654 10.9346
```

- Notice the slight simulation difference.

Looking at Predictions from Regression Models

```
anaemia <- read.table("http://jeffgill.org/data/anaemia.dat",
                      header=TRUE,row.names=1)
anaemia.lm <- lm(Hb ~ PCV + Age + Menapause, data=anaemia)
preds <- sample(anaemia.lm$fitted.values,10^6,replace=TRUE)
par(mar=c(6,6,3,3),bg="slategrey",fg="black")
hist(preds,breaks=12,main="",col="gold")
points(anaemia$Hb,rep(1,length(anaemia$Hb)),col="red",cex=2,pch=19)
```

Looking at Predictions from Regression Models

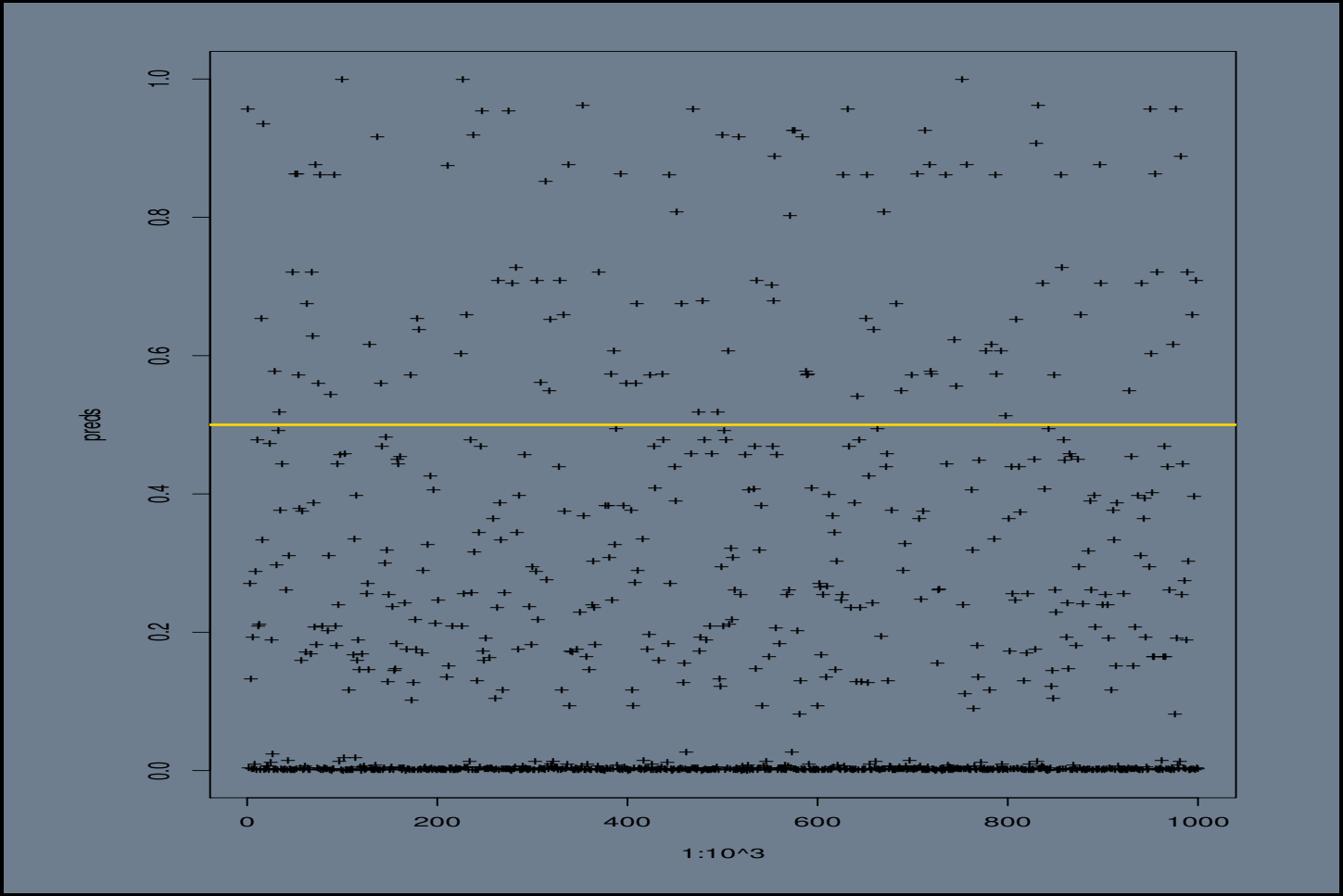


Looking at Predictions from Regression Models

```
library(mgcv)
prostate.df <- read.table("http://jeffgill.org/data/prostate.full.dat",header=TRUE)
prostate.gam1 <- gam(bm ~ stage + pf + log1p(sz) + s(ap) + s(hg),
    family=binomial(link=logit), data=prostate.df)

preds <- sample(prostate.gam1$fitted.values,10^3,replace=TRUE)
par(mar=c(6,6,3,3),bg="slategrey",fg="black")
plot(1:10^3,preds,pch="+")
abline(h=0.5,lwd=3,col="gold")
```

Looking at Predictions from Regression Models



More On Generating Samples in R

- Sample values, e.g. $\bar{X}$ are used to make claims about population values, μ (also s^2 for σ^2).
- We have a very convenient way to generate samples from a desired (known) distribution.
- The `sample` command takes three arguments: source, size, replace, prob...
- Generating a sample with a discrete probability vector:

```
prob.vec <- c(0.2,0.3,0.5)
sample(1:3,1,replace=TRUE,prob=prob.vec)
[1] 3
sample(1:20,5,replace=TRUE,prob=rep(1/20,20))
[1]  8  2  2 11 20
```

- If you do not specify a probability statement R will use the uniform distribution.
- The default is sampling without replacement.
- For more complex, multistage etc., sampling plans use the `sampling` package.

Sampling From the Poisson Distribution

```
(rand.pois <- rpois(n=300, lambda=3))
  [1] 3 1 2 2 4 2 2 3 1 3 2 1 1 6 2 4 7 2 3 2 5 3 8 4 2 3 3 1 2 1 2 5 2 0 3 0 2
 [38] 1 4 3 3 0 3 1 2 1 1 3 5 4 2 0 1 4 4 6 1 2 2 6 5 1 1 1 4 4 1 0 4 3 5 3 6 4
 [75] 1 3 3 5 5 4 5 4 2 2 7 3 9 2 1 3 1 6 1 3 3 6 3 0 2 1 5 4 2 1 3 2 3 3 2 1 7
[112] 2 2 2 4 3 5 4 4 5 4 1 1 4 5 1 3 3 5 2 8 4 5 5 2 1 2 1 3 1 2 2 3 1 4 3 3 3
[149] 3 3 5 6 0 3 5 1 1 3 1 4 4 2 2 3 2 2 2 5 4 6 3 7 6 1 4 3 0 1 3 2 1 2 2 3 1
[186] 2 6 1 3 4 2 2 4 4 2 2 6 4 4 4 3 3 4 4 4 2 9 3 2 1 4 2 4 3 5 2 4 3 2 2 2 5
[223] 7 4 5 4 3 5 6 5 2 4 1 2 6 5 4 2 6 4 2 2 4 4 4 4 1 3 2 4 3 4 4 4 2 4 4 5 4
[260] 3 3 4 4 6 7 3 7 3 2 2 3 4 4 4 4 7 4 4 3 4 7 3 1 2 1 4 3 2 8 1 1 1 4 3 2 3
[297] 1 3 3 3
sample(rand.pois,10)
[1] 1 4 2 4 4 4 4 3 3 1
sample(rand.pois,10)
[1] 4 3 6 3 5 4 3 3 6 4
sample(rand.pois,10)
[1] 2 2 3 4 3 1 3 6 3 1
sample(rand.pois,10)
[1] 2 3 2 4 4 4 4 4 1 5
```

Monte Carlo Introduction

- Simulation work in applied statistics replaces analytical work with repetitious, low-level effort by the computer.
- If a distribution is difficult or impossible to manipulate analytically, then it is often possible to create a set of simulated values that share the same distributional properties, and describe the posterior by using empirical summaries of these simulated values.
- This process was termed *Monte Carlo simulation* by von Neumann and/or Ulam (although some accounts attribute the naming to Metropolis) because it uses randomly generated values to perform calculations interest, perhaps reminiscent of expected value calculations in gambling.

Basic Monte Carlo Integration

- Suppose $g(x)$ is difficult to express or manipulate but for which we could generate samples on an arbitrary support of interest: $[a{:}b]$.
- A common quantity of interest is

$$I[a,b] = \int_a^b g(x)h(x)dx,$$

that is, the expected value of some function, $h(x)$, of x distributed $g(x)$.

- A substitute for analytically calculating this is to randomly generate n values of x from $g(x)$ and calculate:

$$\hat{I}[a,b] = \frac{1}{n}\sum_{i=1}^{n} h(x_i).$$

- The idea is to replace analytical integration with summation from a large number of simulated values, rejecting (ignoring) values outside of $[a{:}b]$

Basic Monte Carlo Integration (cont).

- By the strong law of large numbers, $\hat{I}[a,b]$ converges with probability one to the desired value, $I[a,b]$.
- Although $\hat{I}[a,b]$ now has "simulation error," this error is measured by the empirical variance of the simulation estimate:

$$\texttt{Var}(\hat{I}[a,b]) = \frac{1}{n(n-1)} \sum_{i=1}^{n} (h(x_i) - \hat{I}[a,b])^2$$

- Note that the researcher fully controls the simulation size, n, and therefore the simulation accuracy of the estimate.

Basic Monte Carlo Integration (cont).

- Because the central limit theorem applies here as long as $\texttt{Var}(I[a,b])$ is finite, credible intervals can be easily calculated by

$$[95\%_{lower}, 95\%_{upper}] = \left[\hat{I}[a,b] - 1.96\sqrt{\texttt{Var}(\hat{I}[a,b])},\ \hat{I}[a,b] + 1.96\sqrt{\texttt{Var}(\hat{I}[a,b])}\right],$$

  or by reporting the 0.025 and 0.075 quantiles of the set of $h(x_i)$.

- Bayesian context: replace $g(x)$ with a posterior statement $\pi(\theta|x)$ and $h(x)$ with $h(\theta)$. So $I[a,b]$ is really the posterior expectation of $h(\theta)$:

$$E[h(\theta)|x)] = \int \pi(\theta|x)h(\theta)d\theta \approx \frac{1}{n}\sum_{i=1}^{n} h(\theta).$$

Basic Monte Carlo Integration (cont).

- Example, standard normal PDF over a specific region, suppose we want:

$$I[-2,1] = \int_{-2}^{1} x^2 \phi(x) dx,$$
$$\hat{I}[-2,1] = \frac{1}{n} \sum_{i=1}^{n} x_i^2.$$

- in R

```
norm.sample <- rnorm(100000)
mean(norm.sample[norm.sample>-2 & norm.sample <1]^2)
[1] 0.5747261
```

- we can also "check" the algorithm with a related problem:

```
length(norm.sample[norm.sample>-2 & norm.sample <1])/100000
[1] 0.81989
(pnorm(2)-pnorm(0))+(pnorm(1)-pnorm(0))
[1] 0.8185946
```

Basic Monte Carlo Integration (cont).

- How about a "harder" problem:

$$I[e,\pi] = \int_e^{\pi} \arctan(x^{\frac{1}{3}})\mathcal{C}(x|\theta = 3, \sigma = 2)dx$$

- Recall that:

$$\mathcal{C}(x|\theta,\sigma) = \frac{1}{\pi\sigma}\frac{1}{1+\left(\frac{x-\theta}{\sigma}\right)^2}, \qquad -\infty < x, \theta < \infty, 0 < \sigma.$$

- R code:

```
c.sample <- rcauchy(100000,3,2)
mean(atan(c.sample[c.sample > exp(1) & c.sample < pi]^(1/3)))
[1] 1.058232
```

Rejection Sampling

- If we cannot produce random values from the posterior of interest, we cannot do MC integration as just described.
- New idea: rejection sampling according to...
  1. find some other distribution that is convenient to sample from,
  2. initially it must enclose the target distribution,
  3. compare generated values with PDF form and decide which values to keep and which to reject,
  4. integral of interest is the normalized ratio of these values.
- "Rejection sampling" used in two contexts: obtaining dimensional integral quantities, and generating random variables.
- Rejection sampling can be particularly useful in determining the normalizing factor for nonnormalized posterior distributions.

Continuous Form with Bounded Support

- Bounded support: defined, non-infinite range of some PDF or other function.
- Posit a function $f(x)$ such that $f(x)$ can be evaluated for any value over the support of x: $S_x = [A, B]$.
- Determine the maximum *density* value by either:
  1. analytical root finding,
  2. numerical techniques such as gridding, Newton-Raphson, EM, etc.
- Now we define a rectangle (or hyper-rectangle) that bounds all possible values for the pair $(x, f(x))$. Other geometric forms work too.
- Sample the two-dimensional random variable in this rectangle by independently sampling uniformly over $[A, B]$ and $[0, \tilde{x}]$ and pairing.
- Count the proportion of points that fall in the region of interest.

Continuous Form with Bounded Support (cont.)

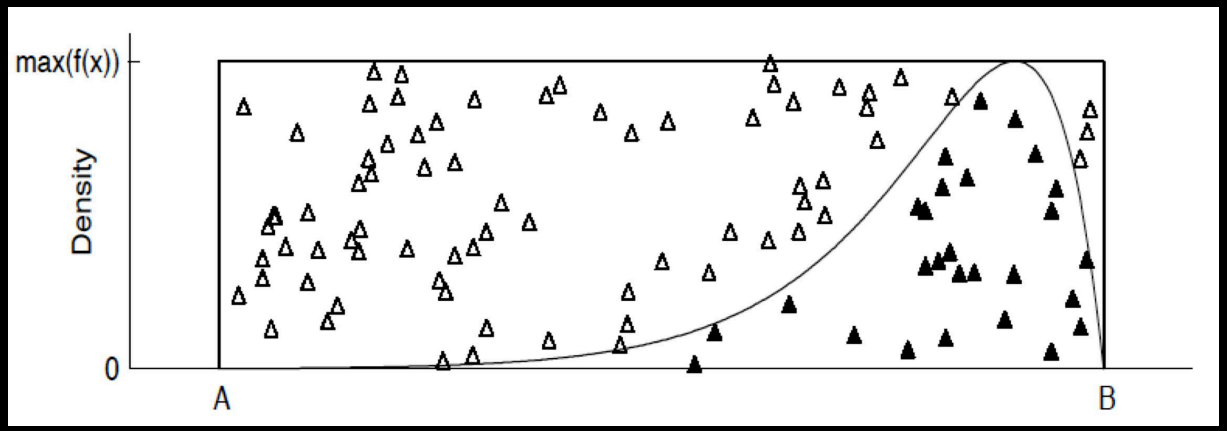
- The value of the integral, the area under the curve is just the ratio of points under the curve to the total number of points scaled by the size of the box:

$$\frac{\text{number of points under curve}}{\text{total number of points}} \times \text{size of box} \underset{n\to\infty}{\longrightarrow} \int_A^B f(x)dx.$$

- Discrete problems turn out to be much more straightforward as it is usually a matter of counting bin heights and taking a weighted sum.
- The degree of accuracy is entirely controlled by the researcher through the number of points generated.

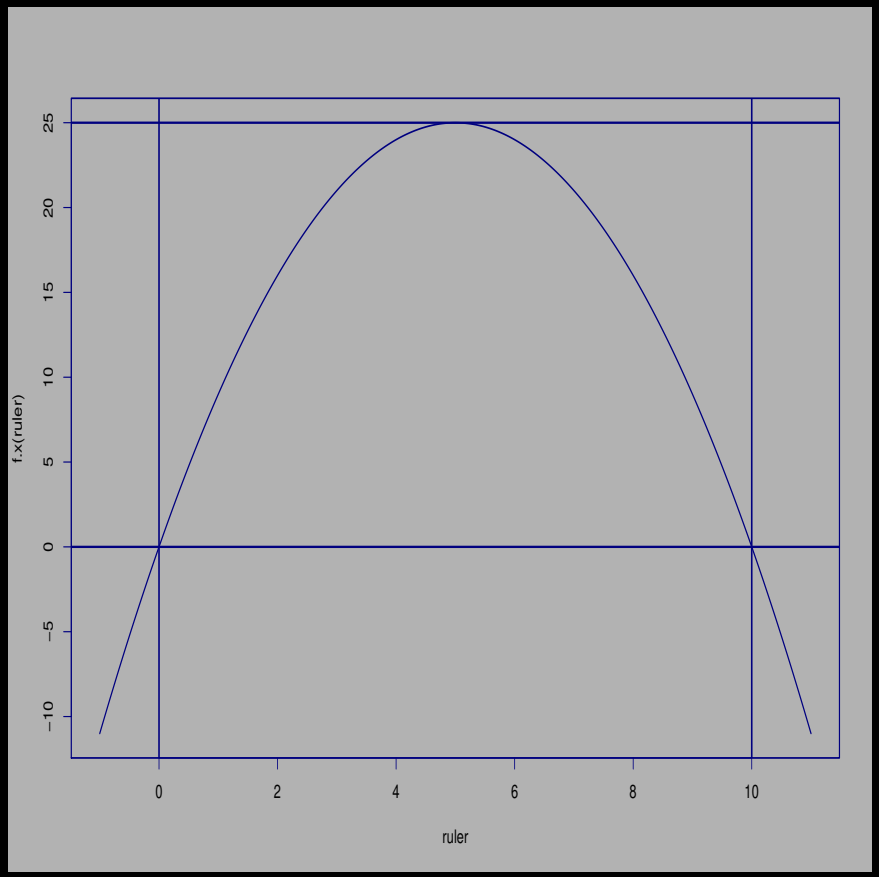
Continuous Form with Bounded Support (cont.)

- 100 points sampled uniformly from the two-dimensional rectangle over: $[(A, B), (0, \max(f(x)))] = [(0, 10), (0, 0.4)]$. In total 27 values fell into the area we wish to integrate, so we obtain the size of the interval from: $(27/100)(10 \times 0.4) = 1.04$.



Measuring the Area Under a Quadratic

- Question: what is the area under $f(x) = -x^2 + 3x + 7x$?
- This is not a difficult integral, but let's do it computationally.



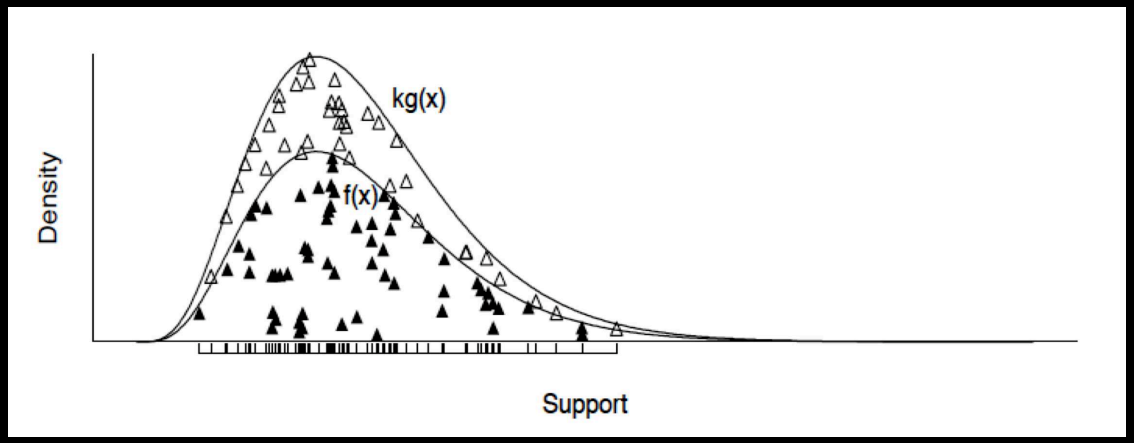
Measuring the Area Under a Quadratic

```
f.x <- function(x) -x^2 +3*x +7*x
ruler <- seq(-1,11,length=300)
par(bg="grey70",fg="navyblue",lwd=1.5)
plot(ruler, f.x(ruler),type="l")
abline(h=c(0,25),lwd=2)
abline(v=c(0,10),lwd=2)

n <- 100000
samples <- cbind(runif(n,0,10), runif(n,0,25))
10*25*sum(samples[,2]<f.x(samples[,1]))/n
[1] 166.0825
```

Continuous Form with Unbounded Support

- Now address the integral of some function $f(x)$ in which the analytical solution is difficult or impossible, and the form of $f(x)$ has unbounded tails.
- Specify a "majorizing function," $g(x)$, which for every value of x in the support of $f(x)$ has the property that $g(x) \geq f(x)$.



Continuous Form with Unbounded Support (cont.)

- If the target distribution has unbounded tails, then obviously the majorizing function must also have this property, and simply picking a PDF for $g(x)$ which has heavier tails than $f(x)$ will not work since it will then have other regions where it is not uniformly greater than the target.
- The solution is to use a multiplication factor so that:

$$f(x) \leq kg(x), \qquad \forall x, k > 1$$

- Sample x_i from $g(x)$ and then make an accept/reject decision based on $f(x_i)$.

Continuous Form with Unbounded Support (cont.)

- Steps:
  1. calculate $f(x_i)/kg(x_i)$,
  2. randomly draw a uniform(0,1) variate,
  3. accept x_i if this uniform is less than $f(x_i)/kg(x_i)$,
  4. repeat many times.
- If the majorizing distribution is very dissimilar from the target distribution, then the sampling procedure would be less efficient (more values rejected).

Example: Enveloping with an Exponential Density

- Start with nonnormalized folded normal kernel:

$$f(x) \propto \exp[-x^2/2], \qquad x \geq 0.$$

- In order to normalize this distribution, we require the integral quantity in the denominator of the right-hand side of Bayes' law:

$$I(x) = \int_0^\infty \exp[-x^2/2]dx$$

- Use an exponential enveloping distribution:

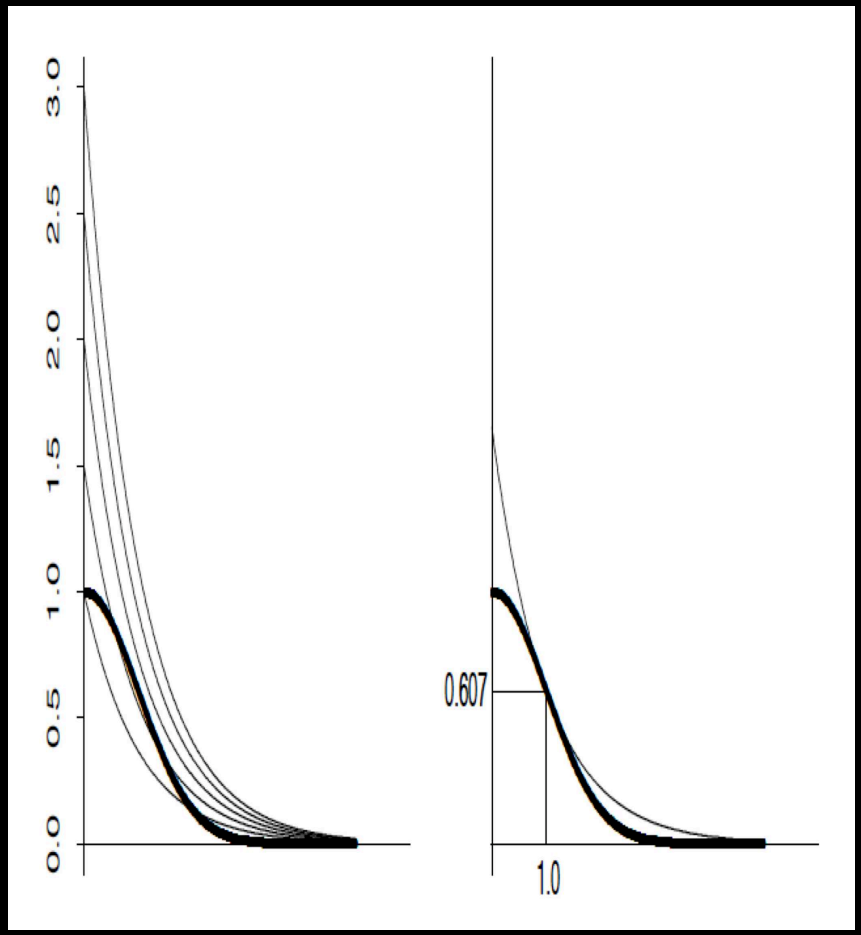
$$g(x) = k\exp[-x]$$

but what value of k is best to ensure coverage but maximize algorithmic efficiency by minimizing rejected values?

Example: Enveloping with an Exponential Density (cont.)

The left panel shows prospective exponential function along with the folded normal kernel (in bold). The exponential functions displayed are for $k = 1.0, 1.5, 2.0, 2.5, 3.0$ as indicated by the $x = 0$ point: $k\exp[0] = k$.

The right panel shows the optimal exponential enveloping function.



Example: Enveloping with an Exponential Density (cont.)

- To find the optimal coverage exponential, equate the two functions and solve for the k that gives the roots to the resulting quadratic equation:

$$\begin{aligned} g(x) \equiv k\exp[-x] &= \exp[-x^2/2] \equiv f(x) \\ k &= \exp[x - x^2/2] \\ 0 &= x^2 - 2x + 2\log k. \end{aligned}$$

  Equate the two possible quadratic solutions, one of which must be 1:

$$\begin{aligned} 1 &= 2\log k \\ \therefore k &= \exp(1/2). \end{aligned}$$

- Using this value, $k = 1.649$, in $g(x) = k\exp[-x]$, we get the enveloping function where the single point of intersection occurs at $[1, 0.607]$.

Example: Enveloping with an Exponential Density Steps

1. Draw n random variables from $\mathcal{EX}(1)$: $x_1, x_2, \ldots, x_n$.
2. For each x_i, calculate the corresponding value of the majorizing function: $kg(x_i) = (1.649)\exp[-x_i]$
3. Draw a random uniform value for each x_i over the interval from 0 to $kg(x_i)$: u_{x_i}.
4. Accept this draw as being from $f(x)$ if u_{x_i} is less than $f(x_i)$.
5. The number of accepted values relative to the total number of draws is proportional to the ratio of the area under the target function to the area under the majorizing function. So:

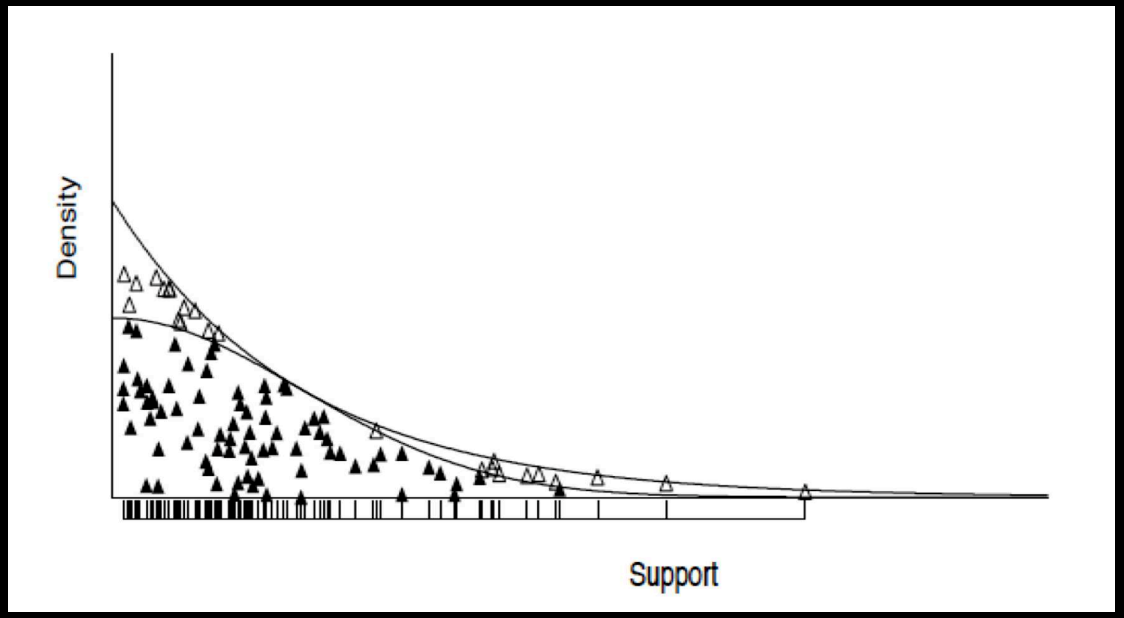
$$I(x) = \int_0^\infty f(x)dx = k \times \frac{\text{number of accepted points}}{\text{total number of draws}}$$

Example: Enveloping with an Exponential Density R Code

```
n <- 1000000
x.expo <- rexp(n, rate=1)
x.k.expo <- 1.649*exp(-x.expo)
u.vals <- rep(NA,n)
for (i in 1:n)  u.vals[i] <- runif(1,0,x.k.expo[i])
f.x <- exp(-x.expo^2/2)
accepted <- 0
for (i in 1:n) if (u.vals[i] < f.x[i]) accepted <- accepted+1
1.649*accepted/n
[1] 1.252678
```

Example: Enveloping with an Exponential Density (cont.)

For only $n = 100$ draws, we get the reasonably accurate result of 1.2546, which is very close to the true value of 1.25314.



Bootstrapping for Standard Errors

- **Big Idea**: sometimes it is difficult to get the sampling properties of an estimator, even a commonly used one.
- Some statistics have known variance properties for finite samples and some do not. Does this mean we should only use the former unless we have population data?
- Definitive citations: Efron (1979), Efron and Tibshirani (1993).
- **Case Study:** suppose we have a dataset on leukemia, ignoring a whole host of things and condensing our analysis down to two variables: *CD4 Count/10* a dichotomous outcome indicating that there was a relapse from a remission stage:

Relapse	94	197	16	38	99	141	23		
No Relapse	52	104	146	10	50	31	40	27	46

- Note that these data are *imbalanced*.

Bootstrapping for Standard Errors

- The question is whether there is a difference by CD4 count, and the natural choice of test is the difference of means: $\bar{x}_{\text{relapse}} = 86.86,\ \bar{x}_{\text{no relapse}} = 56.22$.
- This is easy since we know that:

$$SE(\bar{x}_{\text{relapse}}) = \sqrt{(s^2_{\text{relapse}}/n_{\text{relapse}})} = 25.24,$$
$$SE(\bar{x}_{\text{no relapse}}) = \sqrt{(s^2_{\text{no relapse}}/n_{\text{no relapse}})} = 14.14$$

- But we also know that the mean is not very resistant to outliers and it could be that a notable case, and one could be driving the subsequent findings.
- So what about using the median instead of the mean? This is obvious choice in one sense, but it leaves us with no closed form solution for the standard error.

Bootstrapping for Standard Errors

- So consider the following algorithm, for some statistic of interest, θ:

  1. Draw B "bootstrap" samples of size n, independently, **with replacement** from the sample $\mathbf{x}$ of size n:
  $$\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*B}$$
  (note the notation to differentiate the bootstrap sample from the original sample).
  2. Calculate the sample statistic of interest, θ^{*b} for each bootstrap sample, and the mean of these statistics:
  $$\bar{\theta}^* = \frac{1}{B}\sum_{b=1}^{B}\theta^{*b}$$
  3. Estimate the bootstrap standard error of the statistic by:
  $$\texttt{Var}(\theta) = \frac{1}{B-1}\sum_{b=1}^{B}\left(\theta^{*b} - \bar{\theta}^*\right)^2$$
  where obviously $SE(\theta) = \sqrt{\texttt{Var}(\theta)}$.

- We call the limit of this standard error as B goes to infinity is called the *ideal* bootstrap estimate, and this procedure is called the *nonparametric* bootstrap estimate.

Bootstrapping for Standard Errors

```
relapse <- c(94,197,16,38,99,141,23)
no.relapse <- c(52,104,146,10,50,31,40,27,46)
B <- 1000
no.relapse.mat <- relapse.mat <- NULL
for (i in 1:B)  {
    relapse.mat    <- rbind(relapse.mat,   sample(relapse,length(relapse),
                            replace=TRUE))
    no.relapse.mat <- rbind(no.relapse.mat,sample(no.relapse,length(no.relapse),
                            replace=TRUE))
}

relapse.mean <- mean(apply(relapse.mat,1,mean))
relapse.se <- sqrt(var(apply(relapse.mat,1,mean)))
no.relapse.mean <- mean(apply(no.relapse.mat,1,mean))
no.relapse.se <- sqrt(var(apply(no.relapse.mat,1,mean)))
relapse.median <- mean(apply(relapse.mat,1,median))
relapse.median.se <- sqrt(var(apply(relapse.mat,1,median)))
no.relapse.median <- mean(apply(no.relapse.mat,1,median))
no.relapse.median.se <- sqrt(var(apply(no.relapse.mat,1,median)))
```

Bootstrapping for Standard Errors

```
final.relapse.mat <- rbind( c(relapse.mean, relapse.se,
                              no.relapse.mean, no.relapse.se),
    c(relapse.median, relapse.median.se, no.relapse.median, no.relapse.median.se) )
dimnames(final.relapse.mat) <-
    list( c("Mean","Median"), c("Relapse Est","Relapse SE",
                                "No Relapse Est","No Relapse SE") )

final.relapse.mat
    Relapse Est  Relapse SE  No Relapse Est  No Relapse SE
Mean   88.19143    21.57597        54.86444       12.45360
Median 84.37000    36.40226        44.07000       12.65810
```