344HW3

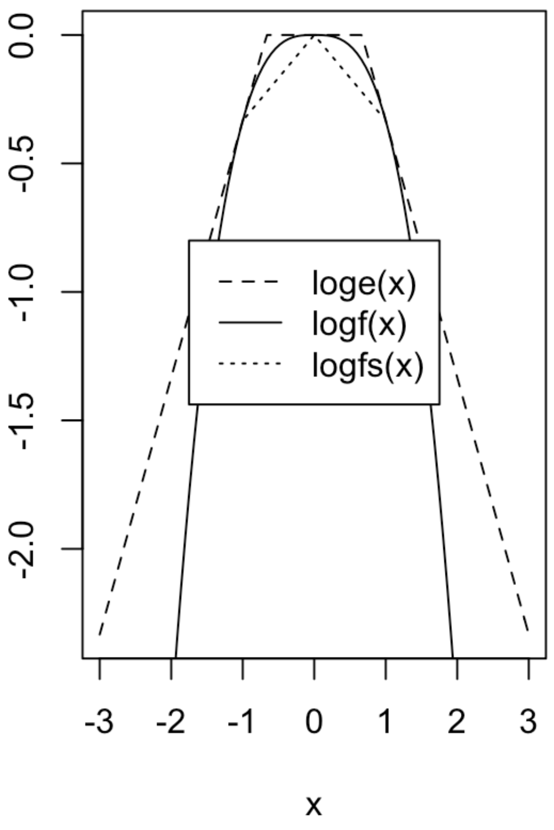
Shay Lebovitz

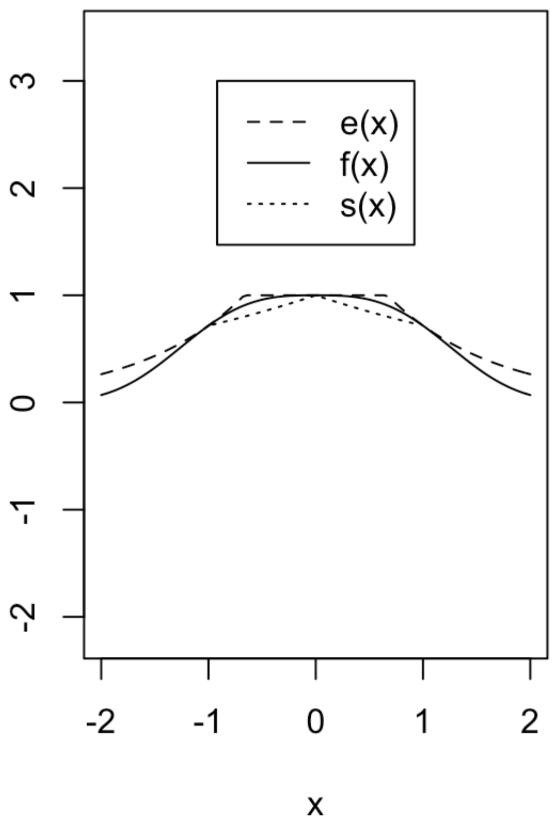
4/25/2020

We want to use adaptive squeezed rejection sampling to estimate S = E[X^2] where X has a density proportional to q(x) = exp{-|x|^3/3}. We will set k = 3 and x = -1, 0, 1.

First, we'll define all the functions and plot them, showing both the envelope and squeezing functions

```
q <- function (x) {
    exp (-abs(x)^3/3)
}

logq <- function (x) {
    -abs(x)^3/3
}

e_star_x = function (x) {
  ifelse (abs(x) > 2/3, -abs(x) + (2/3), 0)
}

e_x = function (x) {
  ifelse (abs (x) > 2/3, exp(-abs(x) + (2/3)), 1)
}

s_star_x = function (x) {
  ifelse (abs(x) < 1, -abs(x)/3, NA)
}

s_x = function (x) {
  ifelse (abs(x)<1, exp(-abs(x)/3), NA)
}

par(mfcol=c(1, 2))
plot(e_star_x, -3, 3, lty=2, xlab="x", ylab="")
plot(logq, -3, 3, lty=1, add=T)
plot(s_star_x, -3, 3, lty=3, add=T)
legend(0, -0.8, legend=c("loge(x)","logf(x)","logfs(x)"), lty=c(2,1,3), xjust = 0.5)

plot(e_x, -2, 2, lty=2, asp = 1, xlab="x", ylab="")
plot(q, -2, 2, lty=1, add=T)
plot(s_x, -2, 2, lty=3, add=T)
legend(0, 3, legend=c("e(x)","f(x)","s(x)"), lty=c(2,1,3), xjust = 0.5)
```





Next, we'll need to define G^{-1} (x), based on the inverse CDF method

```
g_inv = function (y) {
  if ((y < 0.3)||(y == 0.3)) {
    log ((10/3)*y) - 2/3
    }
  else if ((y > 0.3)&&((y<0.7)||(y==0.7))) {
    (10/3)*(y-1/2)
    }
  else {
    (2/3)-log(-(10/3)*(y-1))
    }
}
```

Next, we define the main function, which tests if the uniform value is less than or equal to s(x)/e(x). If not, then we check if U < q(x)/e(x). If not, we reject it.

```
ars = function (n, LIMIT=Inf) {
  y <- rep(NA,n);
  i <- 0;                    # index of y
  j <- 0;                    # index of g
  i1 <- 0;                   # index of acratio.sx

  while((i<n)&&(j<LIMIT)) {
    g <- g_inv(runif(1));
    u <- runif(1);
    if( (g>-1)&&(g<1) ) {  #if in the range of s(x) function (-1, 1)
      if( u < s_x(g)/e_x(g) ){  #if u < s(y)/e(y), keep g for y
          i<-i+1;
          y[i]<-g;
          i1<-i1+1;
      } else {  # that is, if u > s(x)/e(x), we still need to test if u < q(x)/e(x)
        if( u < q(g)/e_x(g) ) {
          i<-i+1;
          y[i]<-g;
        }
      }
    } else if( u < q(g)/e_x(g) ) {  #if g falls outside range of s(x) (-1, 1) but sti
ll <f(y)/e(y)
      i<-i+1;
      y[i]<-g;
    }
    j <- j + 1;
  }
  if(i<n) cat("\n No enough random numbers! ", round(i/n*100), "% complete.\n");
  list(acratio.sx=i1/j, acratio=i/j, y=y);
}

temp <- ars(100000)
temp$acratio
```

```
## [1] 0.7723499
```

```
temp$acratio.sx
```

```
## [1] 0.5098359
```

Here, “acratio” is the total acceptance ratio, and “acratio.sx” is the acceptance ratio of the squeeze function.

We can compare these to the theoretical acceptance ratios, found by integrating the functions

```
# theoretical acceptance ratios
integrate(q, lower=-Inf,upper=Inf)$value/integrate(e_x, lower=-Inf,upper=Inf)$value #
0.7715454
```

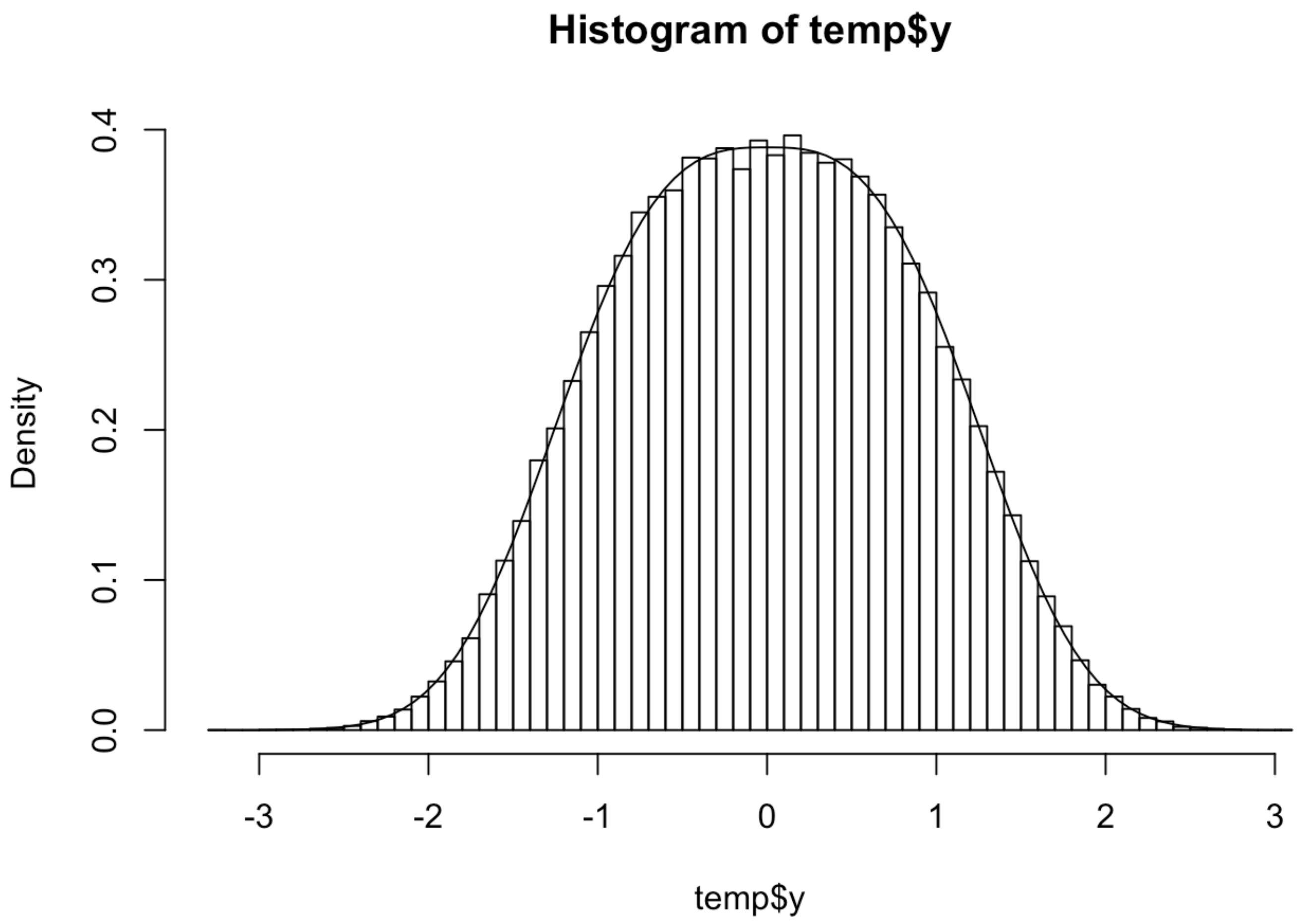
```
## [1] 0.7727395
```

```
integrate(s_x, lower=-1,upper=1)$value/integrate(e_x, lower=-Inf,upper=Inf)$value #0.
5071136
```

```
## [1] 0.5102436
```

This shows that our acceptance rates are accurate. Finally, we can draw a histogram and estimate S = E[X^2]

```
par (mfrow = c(1,1))
hist (temp$y, breaks = 50, freq=FALSE)
curve (q(x)/2.5758, add = T)
```



```
mean(temp$y^2)
```

```
## [1] 0.7772555
```