Drawing & Summarizing Random Numbers

My Favorite Letters are r,p,q and d

Paul E. Johnson[1,2]

[1]Department of Political Science
University of Kansas

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

2016

Outline

What R Can Do

- draw random samples from a distribution
- draw a graph representing the probability density function
- draw a graph representing the cumulative distribution function
- easily explore sampling distributions

Potpourri of Probability Possibilities Protudes Prodigiously

- Distributions in the base of R.
  normal, beta, gamma, binomial, negative binomial, exponential, chi-square, t, F, logistic, poisson, cauchy, geometric, weibull
- Many other distributions easily available in packages
  tweedie, multivariate normal, multivariate t

For starters, a confession

People often pop up with a model based on some random process that I've never heard of before

- Sure, I could go read a book. But I'd rather get a quick picture
- For example, I don't know what the "Cauchy" distribution is.
- Without even reading a manual, I'll explore.

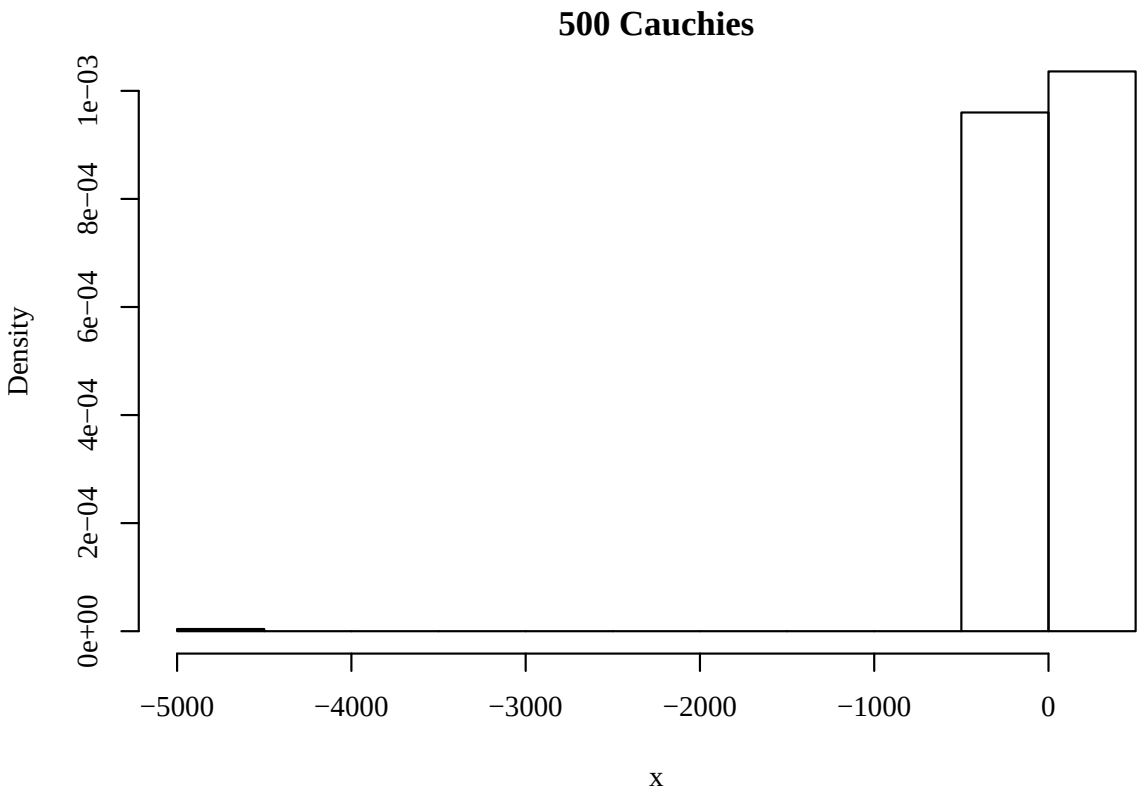
```
set.seed(234234)
rcauchy(6)
```

```
[1]  6.06146448 -0.01975683  0.99342088 -0.97462749
    -5.51630207 -4.49156565
```

- Hm. Well, they are real numbers, that's all I can see.

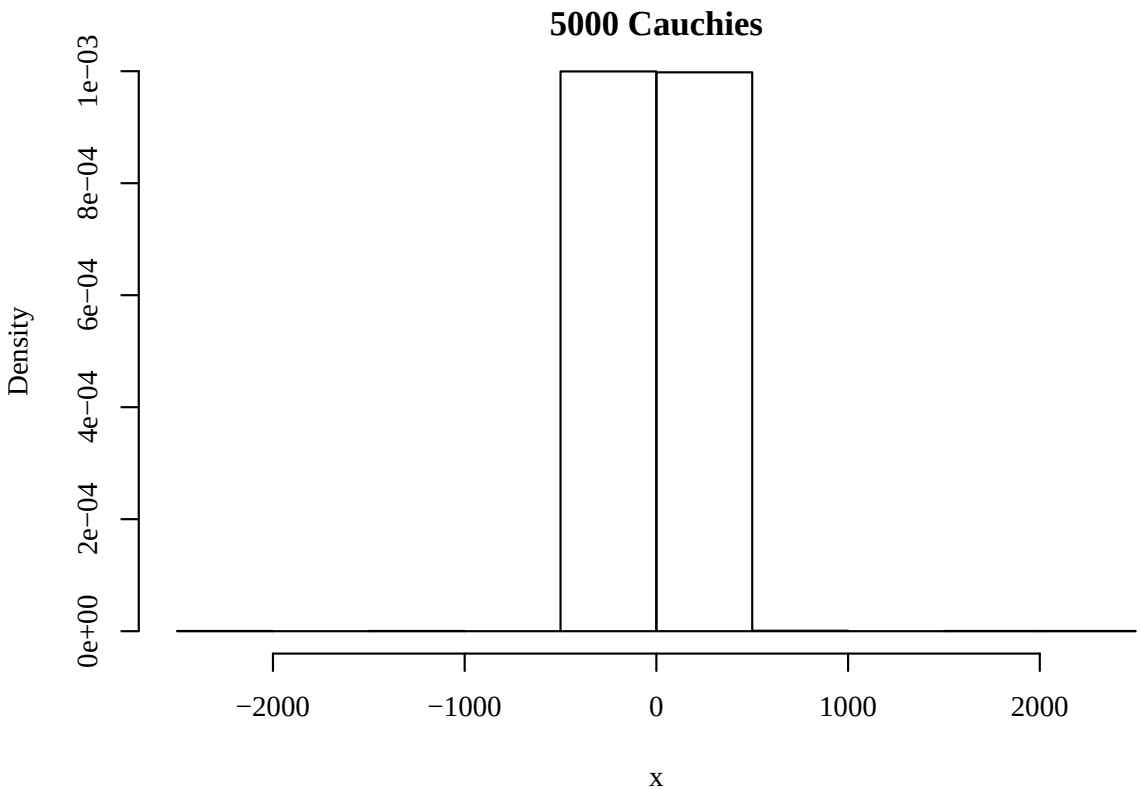
I'll draw a larger sample and make a histogram

```
x <- rcauchy(500)
hist(x, prob= TRUE, main="500 Cauchies")
```



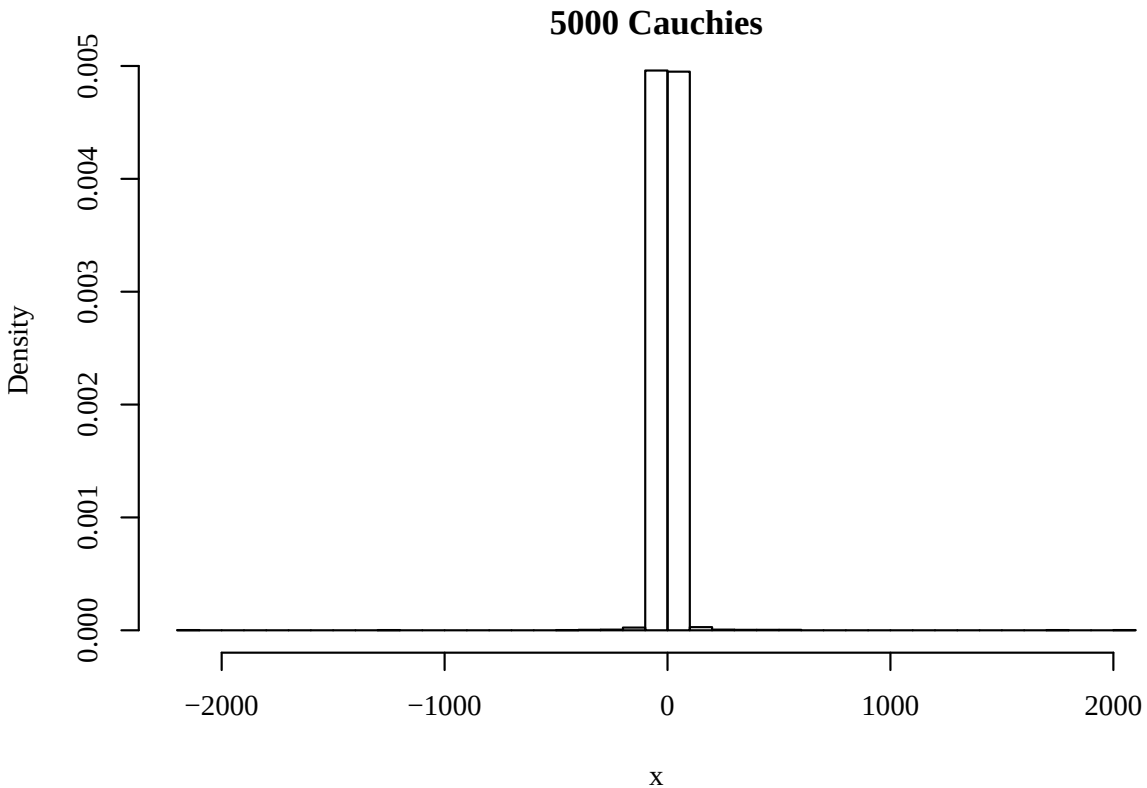
Maybe I need more observations?

```
x <- rcauchy(5000)
hist(x, prob= TRUE, main="5000 Cauchies")
```



What if I force more break points into the histogram?

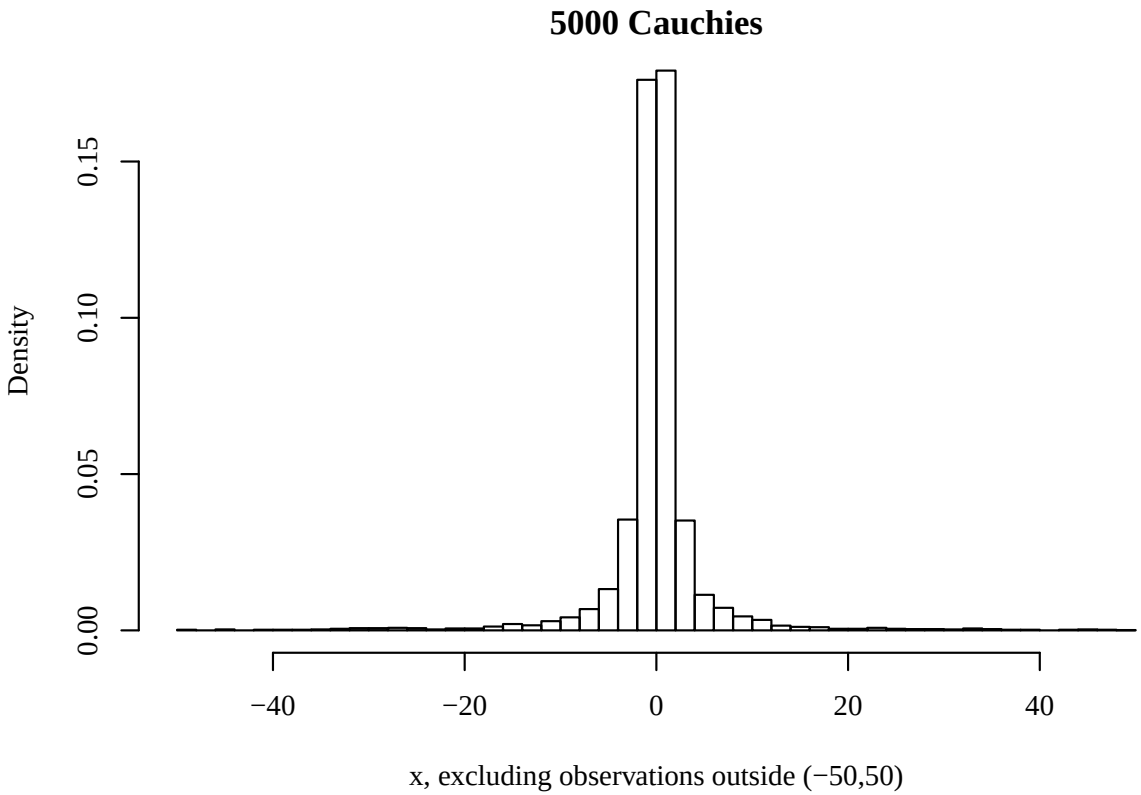
```
hist(x, prob= TRUE, breaks = 50, main="5000 Cauchies")
```



No good

Hm. Focus on a smaller domain?

```
hist(x[ x > -50 & x < 50], xlab="x, excluding observations
     outside (-50,50)", prob= TRUE, breaks = 50, main="
    5000 Cauchies")
```



Hm. Focus on a smaller domain? ...

The excluded observations were:

```
excludedX <- x[ x < -50 | x > 50]
sort(excludedX)
```

```
 [1] -2143.76510 -1212.56570  -495.93430  -358.92679
     -323.25928  -293.65362  -223.63214  -216.08593
     -186.35147  -177.87912  -163.75993  -162.37585
     -142.00162  -135.69665
[15]  -133.84333  -122.74492  -120.03737  -119.96264
    -117.64502  -102.39544   -99.85192   -93.06852
    -92.24122   -84.46856   -84.21631   -81.01521
    -77.92960   -75.33751
[29]   -66.58324   -65.38211   -60.57836   -60.33039
    -60.10383   -58.97226   -56.38789   -54.58040
    -51.65760   -50.35593   -50.03025    50.78813    51
    .06803    51.78402
[43]    55.42471    56.68128    61.68350    64.04770    67
    .16140    69.42834    69.68492    72.67026    75.91167
       78.37429    97.36832   102.47457   107.95593   117
    .63511
```

Hm. Focus on a smaller domain? ...

```
[57]   120.12563   120.46632   121.92550   127.23775   142
    .63761   156.35327   160.00410   173.72400   178.47843
     188.60917   198.14579   226.02443   241.90805   285
    .97483
[71]   335.98579   359.62339   420.67198   470.93352   515
    .39648   566.43654  1716.44174  2038.52465
```

Get Descriptive stats

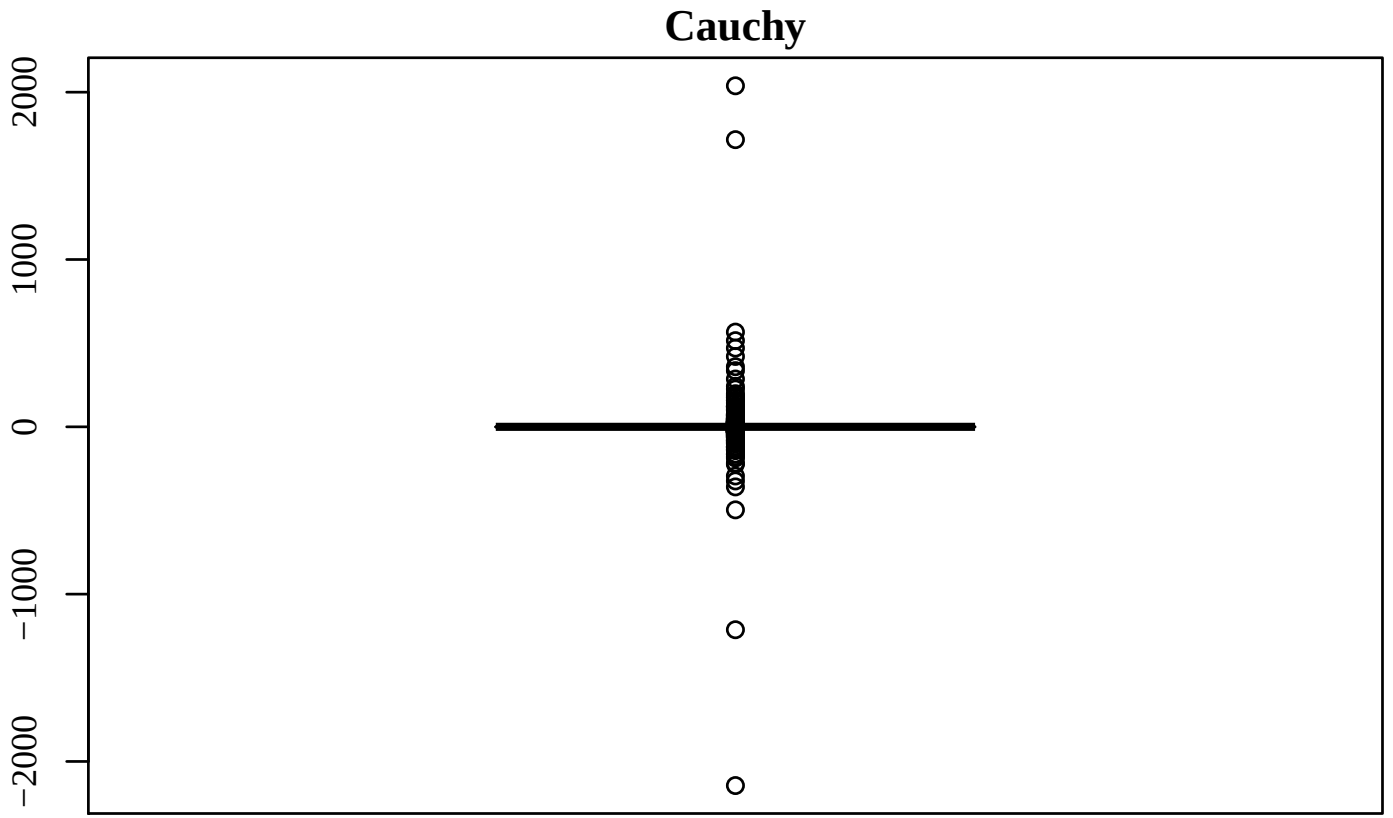
Confirms fact that this distribution emits some extreme observations

```
library(rockchalk)
mydf <- data.frame(x = x)
summarizeNumerics(mydf)
```

```
                   x
0%       -2143.7651
25%         -1.0349
50%         -0.0002
75%          0.9759
100%      2038.5247
mean         0.2970
sd          56.7890
var       3224.9875
skewness     2.4781
kurtosis   951.5539
NA's         0.0000
N         5000.0000
```

Duh! Why didn't I think of a boxplot before?

```
boxplot(x, main="Cauchy")
```



What are we trying to find out with this?

Are the random values

- real numbers or integers?
- bounded below or above?
- symmetric?
- pleasant? Are there extreme outcomes?

Caution : Students often tempted call the extreme outcomes "outliers", meaning they are somehow mistaken or drawn from a different probability process. But these are not outliers–they are from the SAME random process.

R distributions have functions “r”,”d”,”p”,and “q”

- d stands for “density”, as in probability density.
- p stands for “probability”, as in cumulative distribution
- q stands for “quantile”, returns values of x corresponding to requested quantiles.
- r stands for “random sample”

Draw a PDF using the "d" functions

1. Check the help page, ?dnorm, ?runif, ?dpois, ?dgamma, ?dbeta, ?dt, etc.
2. Decide on values of parameters for the desired distribution

   normal: mean, sd
   uniform: min, max
   poisson: lambda
   gamma: shape, scale
   beta: shape1, shape2
   t: df, ncp
3. Choose the range of possible values in which we are interested and draw the plot!

Calculate the Normal Density at a Particular Point

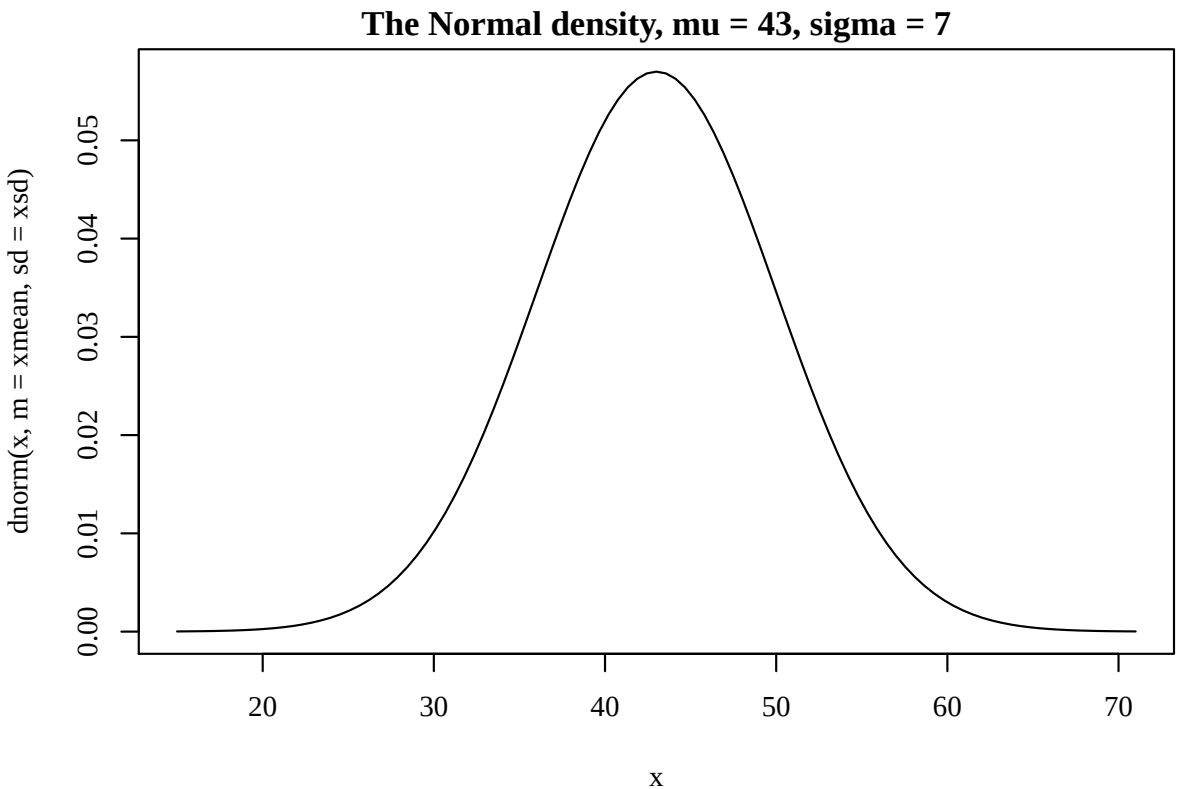
- use dnorm

```
dnorm( 0.323, m = 43, sd = 7 )
```

- the density (chance of drawing) the value 0.323 from a normal with
  - the "mu" parameter: the expected value 43
  - the sigma parameter: the "true" standard deviation 7.

Plot the PDF with the curve() function

```
xmean <- 43; xsd <- 7
xrange <- c(xmean-4*xsd, xmean+4*xsd)
curve(dnorm(x, m = xmean, sd= xsd), from = xrange[1], to =
      xrange[2])
title("The Normal density, mu = 43, sigma = 7")
```



Here's how I usually *really* do this

The curve() function's output not so easy to customize (for me, at least).

- Step 1. Choose the range of x for which probabilities are required.

```
xrange <- c(xmean - 4*xsd, xmean + 4*xsd)
```

- Step 2. Create the Plotting Sequence across the range.

```
xseq <- seq(from = xrange[1], to = xrange[2],
    length.out = 100)
```

- Step 3. Calculate the density for each point in xseq

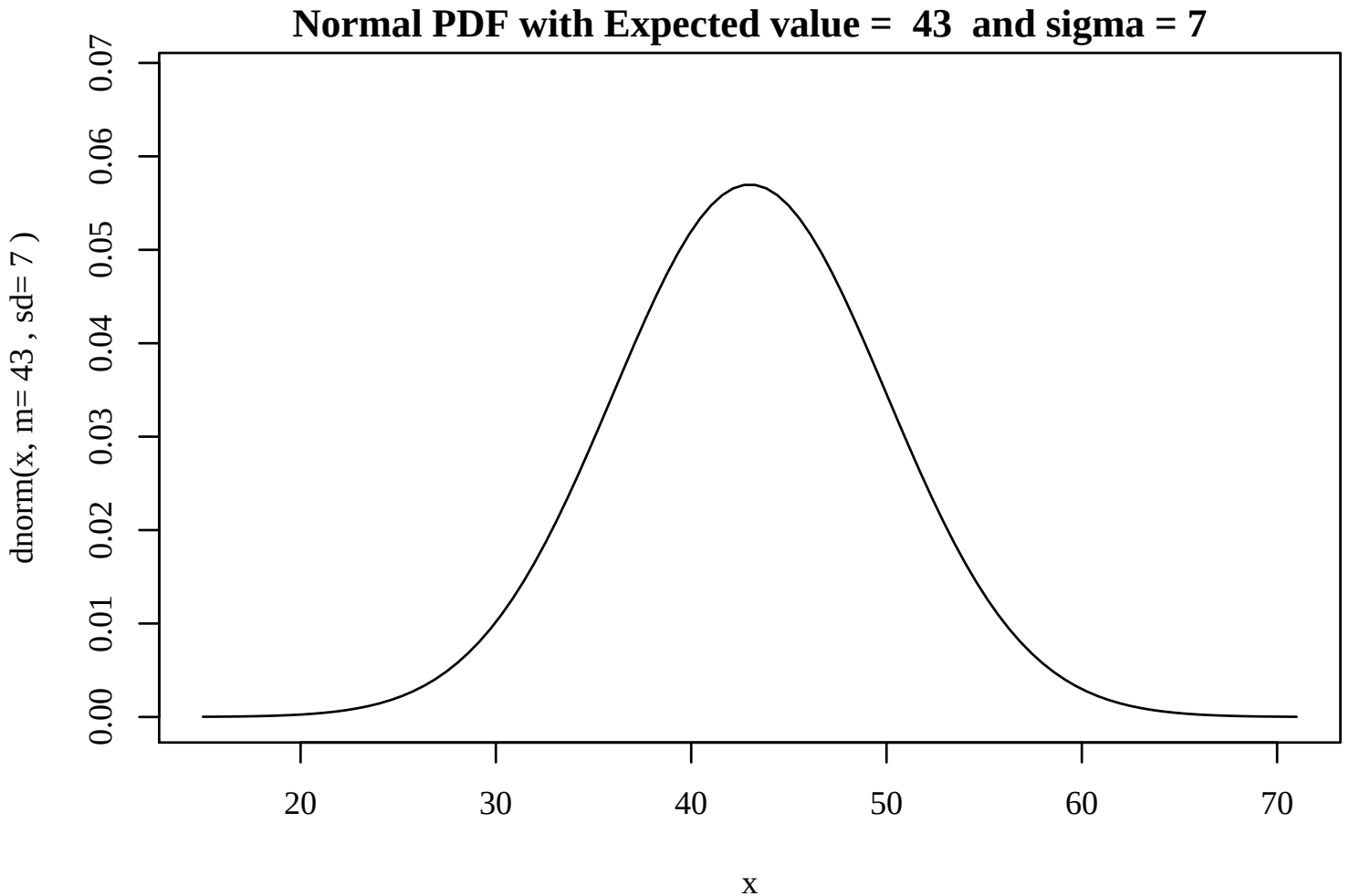
```
dxseq <- dnorm(xseq, m=xmean, sd=xsd)
```

- Step 4. Draw the plot

Here's how I usually *really* do this ...

```
plot(dxseq ~ xseq, type = "l", ylim = c(0,1.2* max(
    dxseq)), xlab = "x", ylab = paste("dnorm(x, m=",
    xmean,", sd=", xsd,")"), main = paste("Normal PDF
     with Expected value = ",xmean," and sigma =",
    xsd))
```

The Probability Density Function



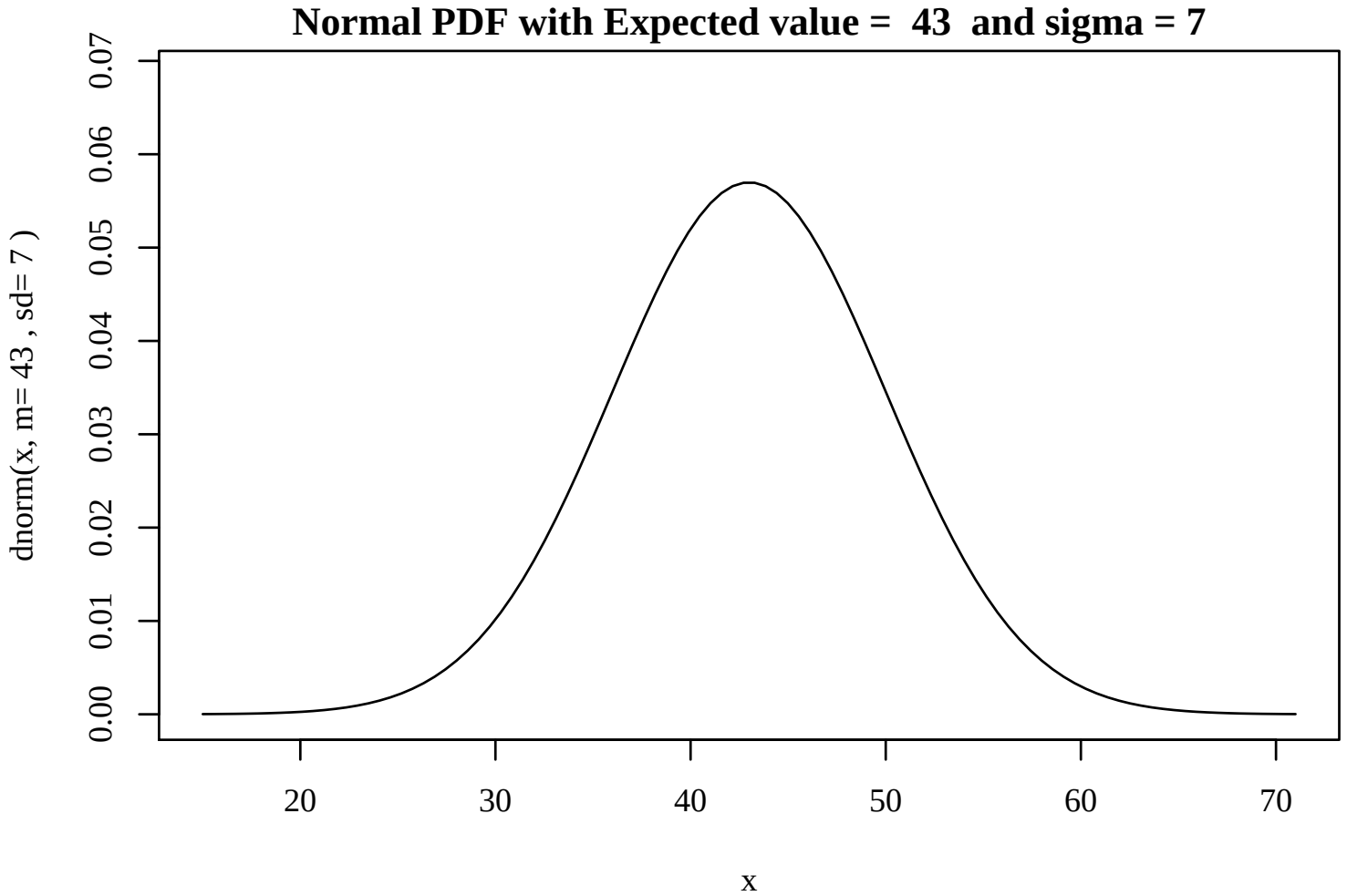
Break the plot command into 3 smaller steps

```
plot(dxseq ~ xseq, type="n", ylim=c(0,1.2* max(dxseq)),
    xlab="x",ylab=paste("dnorm(x, m=", xmean,", sd=", xsd,
    ")"))
lines(xseq, dxseq)
title(paste("Normal PDF with Expected value = ", xmean,"
    and sigma =", xsd))
```

- type="n" draws axes, but an empty plot inside
- lines function is very customizable

Advantage of this approach is easier customization of all features

See? Resulting Plot Same



Why take small steps? Sometimes, the "one giant command" approach frustrates customization of labels and line types

Calculate the Gamma Density at a Particular Point

- use dgamma.
- dgamma usage example

```
dgamma(3.232, shape=2.1, scale = 1.7)
```

asks a gamma for the density of 3.232 when the shape and scale parameters are 2.1 and 1.7.

Calculate the Gamma Density for a Range of Points

- Gamma requires 2 parameters, called “shape” and “scale”

```
shape <- 1.1; scale <- 2.2
xrange <- c(0.0001, 5*shape*scale)
xseq <- seq( xrange[1], xrange[2], length.out = 200)
```

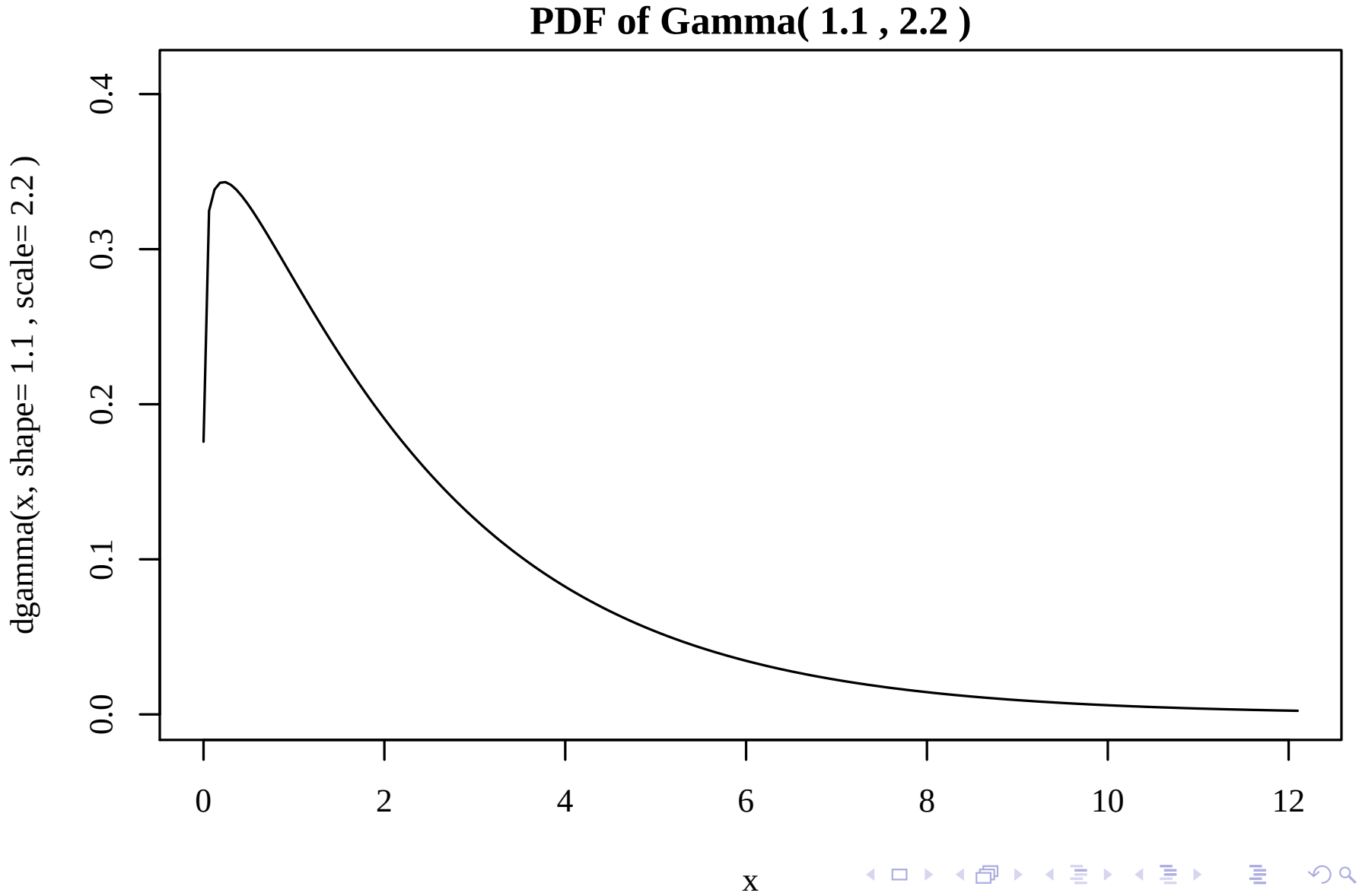
- Create the sequence of density estimates

```
dxseq <- dgamma(xseq, shape=shape, scale=scale)
```

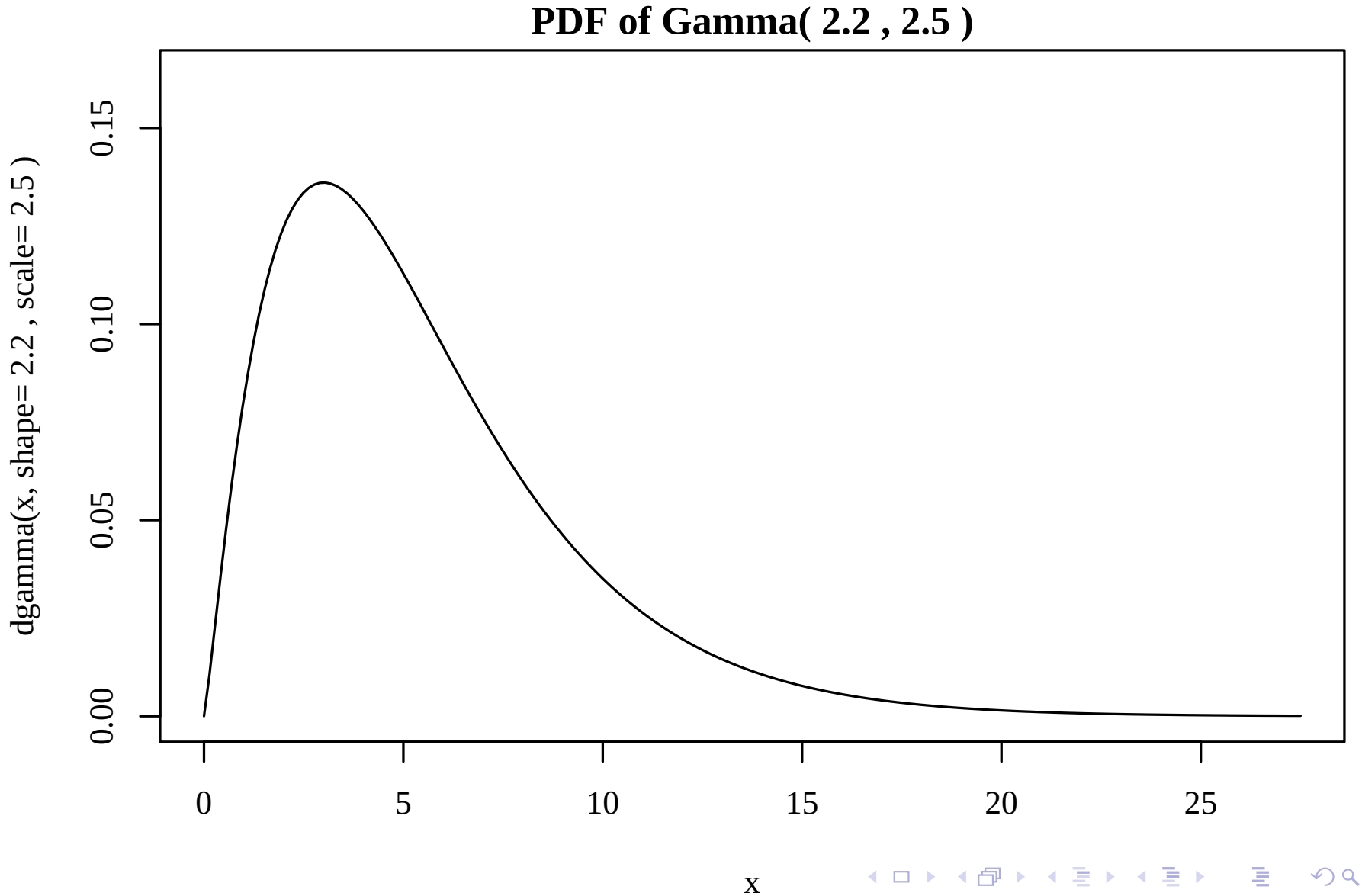
- Draw the plot

```
plot(dxseq ~ xseq, type="l", ylim=c(0,1.2* max(dxseq)
    ), xlab="x",ylab=paste("dgamma(x, shape=",shape,"
    , scale=",scale,")"), main=paste("PDF of Gamma(",
    shape,",",scale,")"))
```

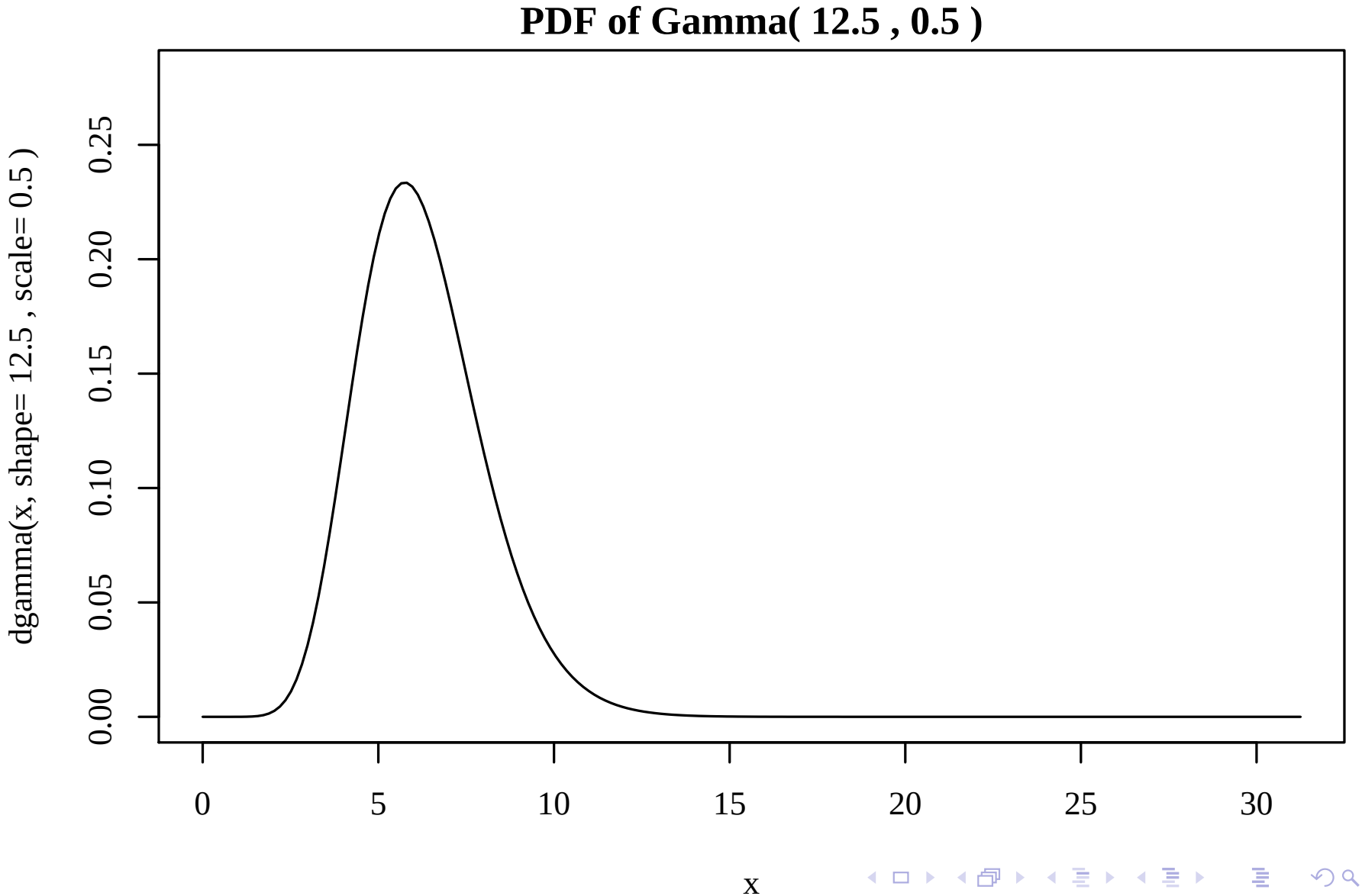
The Gamma Density is defined for $x > 0$



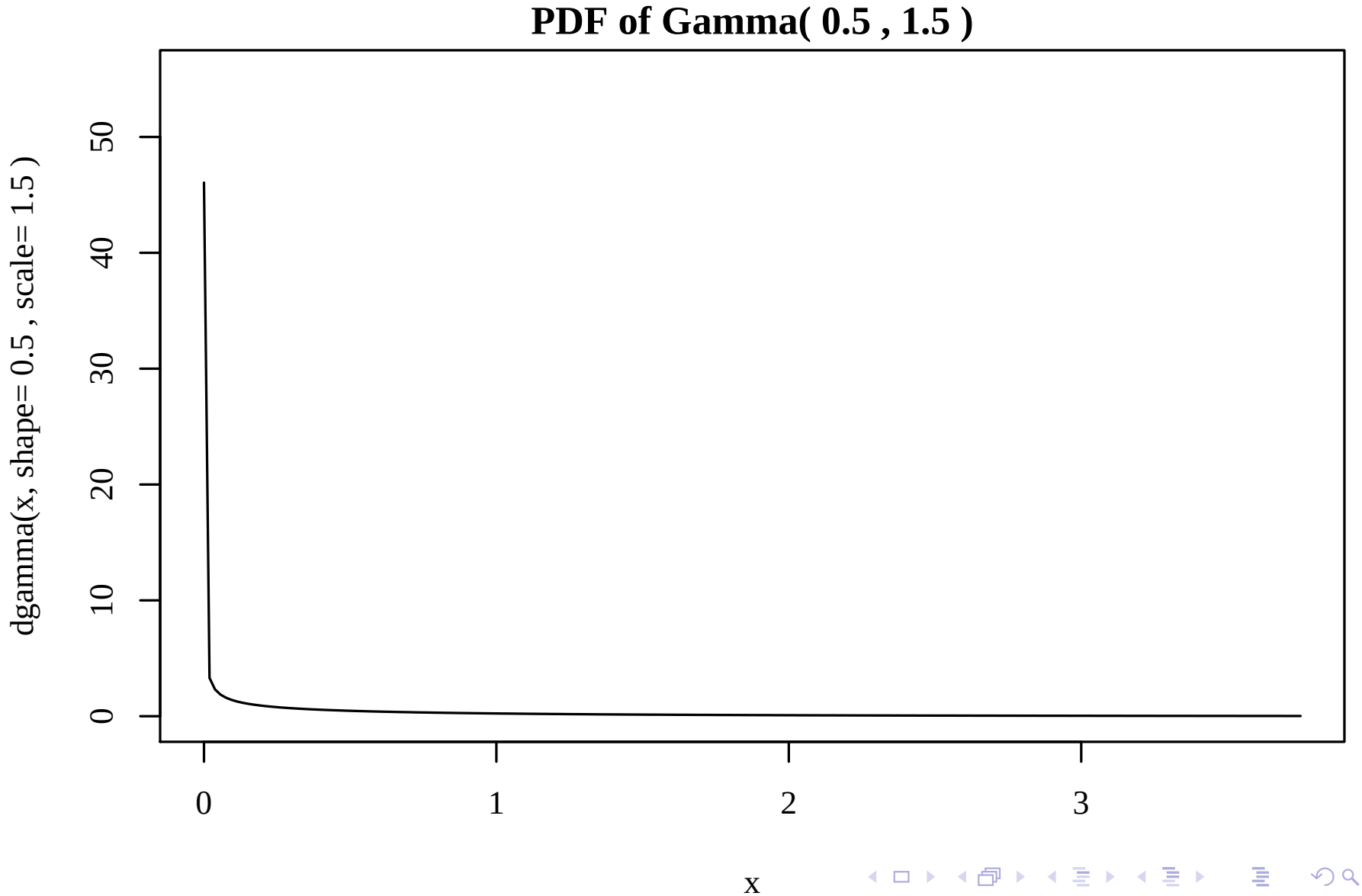
The Gamma shape = 2.2, scale = 2.5



The Gamma shape = 12.5 scale = 0.5



The Gamma shape = 0.5 scale = 1.5



Summary: What is "d" good for?

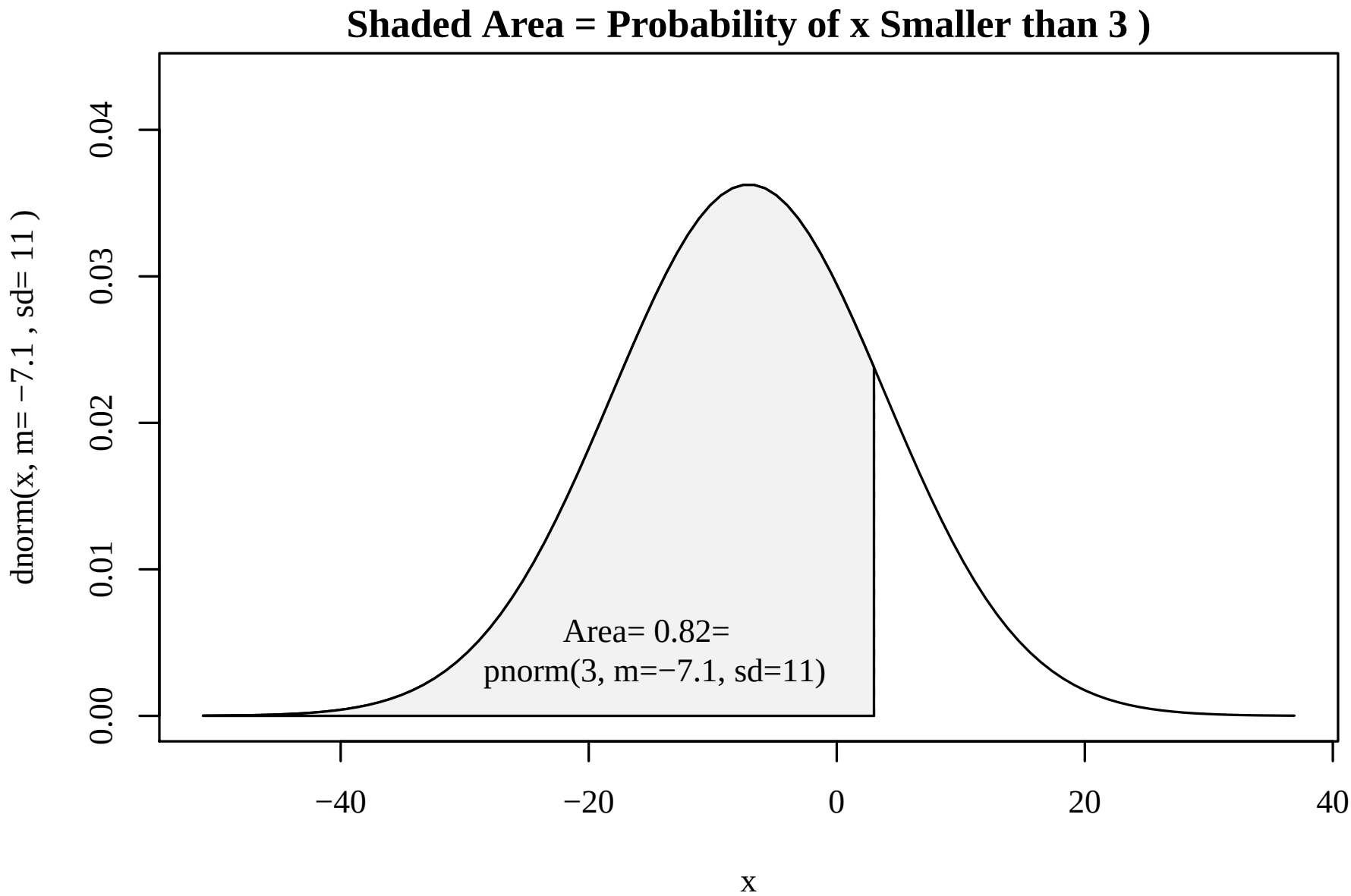
- "d" functions are handy for making illustrations of probability models.
- When we wonder "how likely is each outcome," the PDF is the most natural answer

Calculate the CDF with the "p" functions

- "p" functions are cumulative probability distribution functions (CDF).
- Remember, the CDF is the Area up to a Point Under the PDF
- CDF F(x) is "chance of outcome smaller than x"
- Ever wonder, "What is the chance that a score from $N(-7.1, 11^2)$ will be smaller than -14.434 ?" No problem!

```
pnorm(-14.434, -7.1, 11)
```

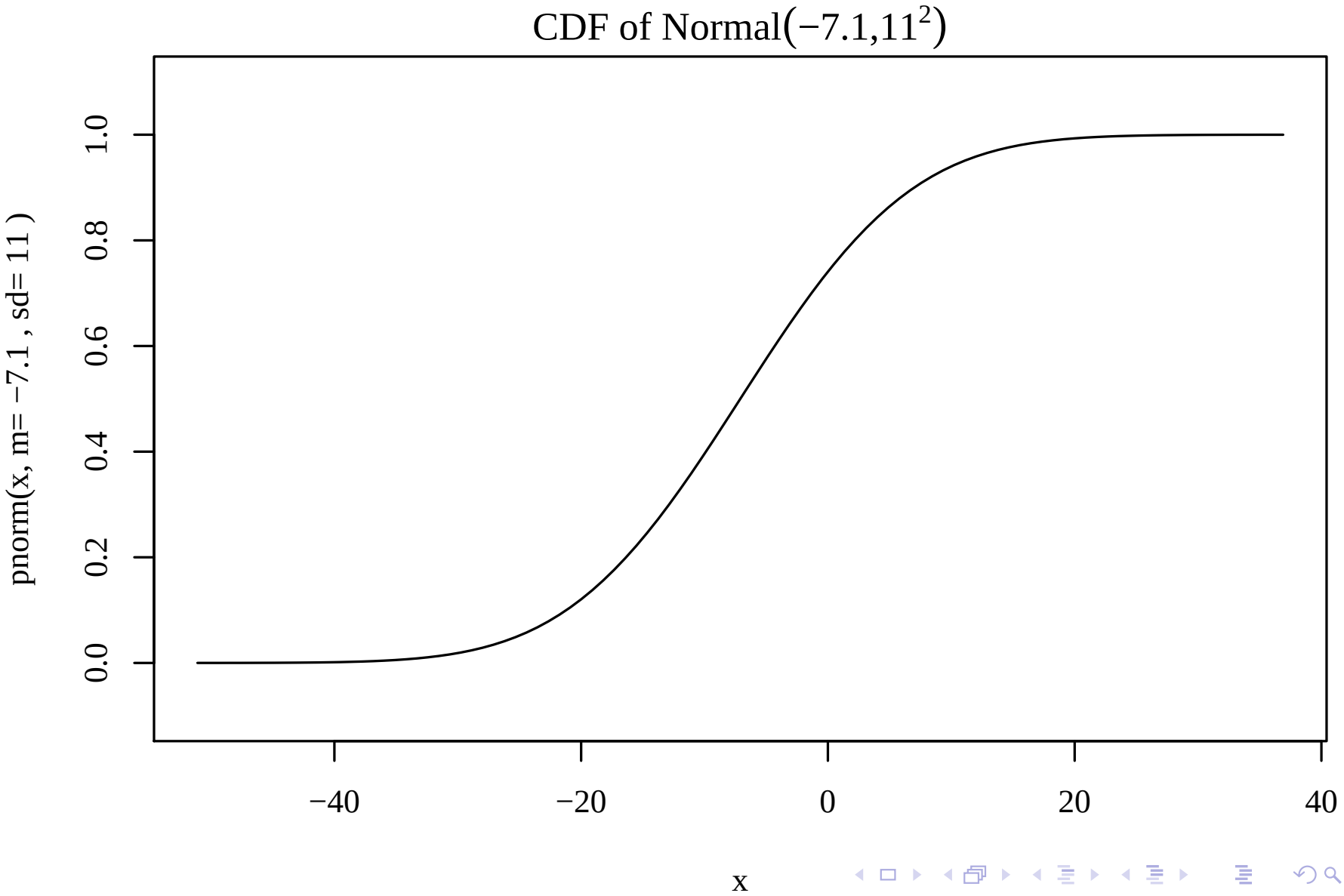
```
[1] 0.2524732
```


Shaded Area = Probability of x Smaller than 3)
dnorm(x, m= −7.1 , sd= 11)
0.00
0.01
0.02
0.03
0.04
Area= 0.82=
pnorm(3, m=−7.1, sd=11)
−40
−20
0
20
40
x

An Illustration of the CDF

```
m= -7.1; sd=11
xrange <- c(m-4*sd, m+4*sd)
xseq <- seq(xrange[1], xrange[2], length.out = 100)
pxseq <- pnorm(xseq, m = m, sd = sd)
plot(pxseq ~ xseq, type="l", ylim = c(-0.1,1.1), xlab =
     "x", ylab = paste("pnorm(x, m=",m,",","sd=",sd,")"
    ), main = bquote("CDF of Normal"(.(m) * "," * .(sd)
    ^2)))
```

About that Illustration



Draw random samples with the "r" function

- There's no such thing as a truly "random number" from a computer
- Computers can create pseudo-random number streams (values are unpredictable to an outside observer)
- We use the set.seed() function in R to put the random stream at the same starting point (replication)

Check on set.seed()

```
set.seed(2343234)
runif(3, min = 0, max = 1)
```

```
[1] 0.06155516 0.61827393 0.56189637
```

```
runif(3, min = 0, max = 1)
```

```
[1] 0.7723577 0.2458070 0.5049224
```

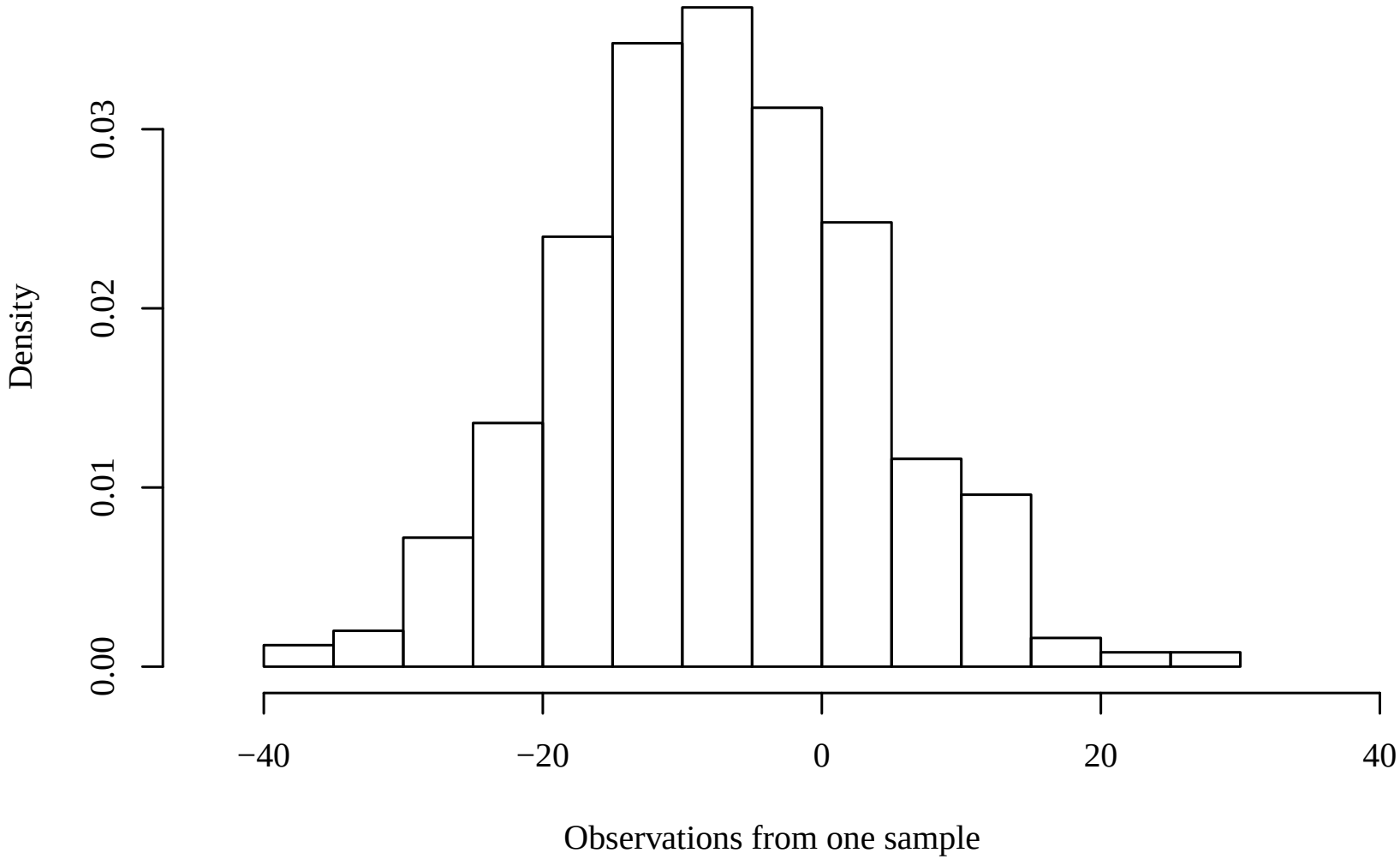
```
set.seed(2343234)
runif(3, min = 0, max = 1)
```

```
[1] 0.06155516 0.61827393 0.56189637
```

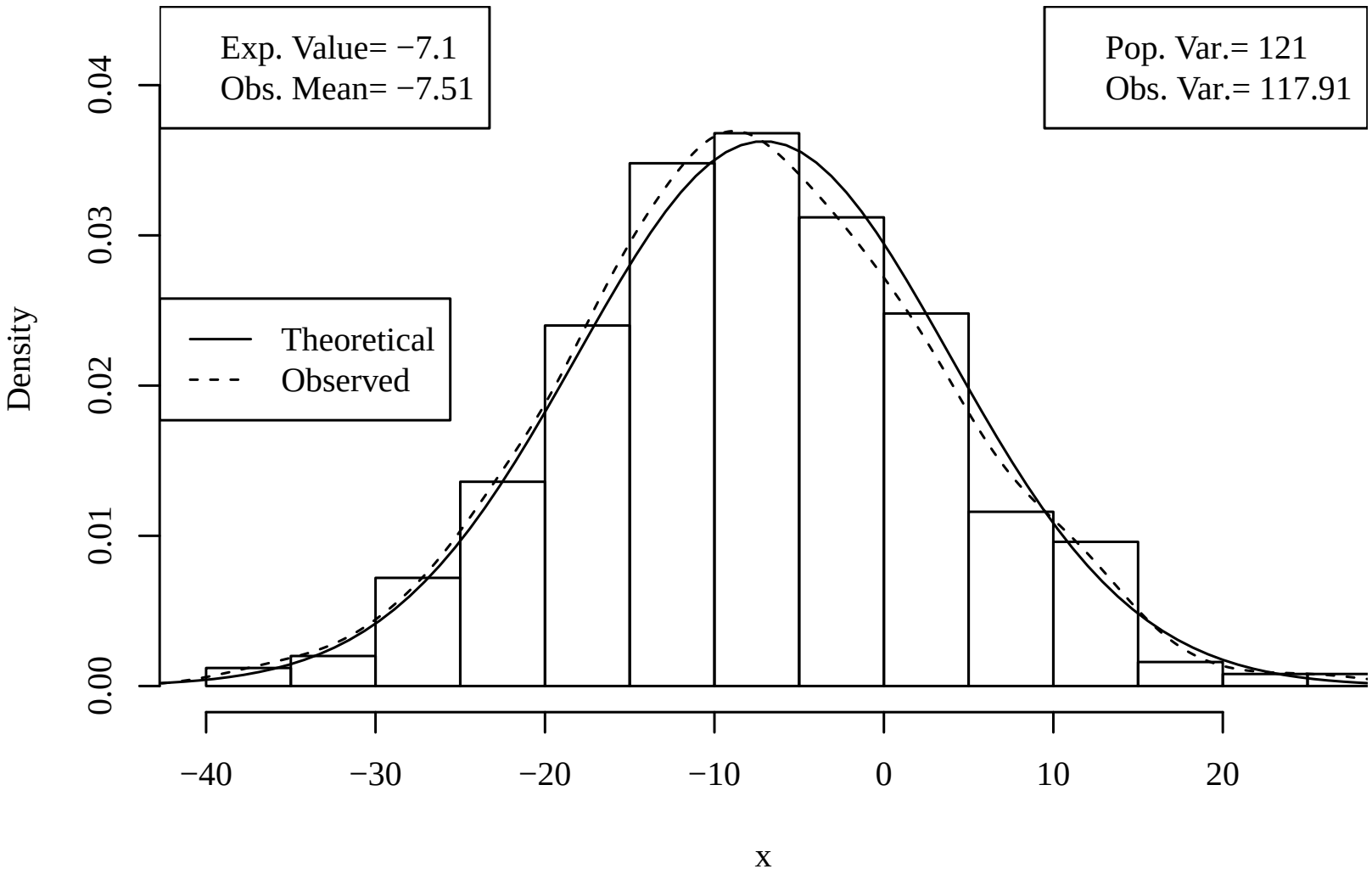
Each distribution has its own "r" function

- Normal: rnorm(N, m = m, sd = sd) draws N observations from a normal with expected value m and standard deviation sd
- Poisson: rpois(N, lambda = 4) draws N observations from a Poisson distribution with the lambda 4
- Gamma: rgamma(N, shape = 2.1, scale = 1.7) draws N observations from a gamma distribution with indicated shape and scale
- Read the help page, you'll usually find a statement about how the expected value and variance (or standard deviation) relate to the parameters.

Histogram for one draw from $N(-7.1, 11^2)$



Compare the Observed and Theoretical Density



Using R to explore probabilty models

- We often find unfamiliar distributions in research
- I often use R to find out about them.
- . Here's my plan.
  1. Draw some random samples and make histograms
  2. Draw some pdfs

Let's expore the binomial distribution

- Binomial. How many "successes" do we expect out of "N" experiments, when the chance of a success on each one is π
- The chance of success is assumed to be the same across all experiments.

Run "?rbinom"

Description:
Density, distribution function, quantile function and random generation for the binomial distribution with parameters 'size' and 'prob'.
Usage:
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)

- n the number of observations I want to draw
- size is the number of experiments, the thing I called N
- prob is the parameter π

Draw 481 From Binomial(197, 0.2)

- I wondered, "what if I ran 197 experiments, counting Yes and No in the results, and the chance of a Yes is 0.2?"
- If I conduct just one "run" (batch of 197 experiments), in R I want

```
set.seed(2324)
oneBinomial <- rbinom(1, size = 197, prob = 0.2)
oneBinomial
```

```
[1] 37
```

- Again!

```
oneBinomial <- rbinom(1, size = 197, prob = 0.2)
oneBinomial
```

```
[1] 42
```

Repeat that

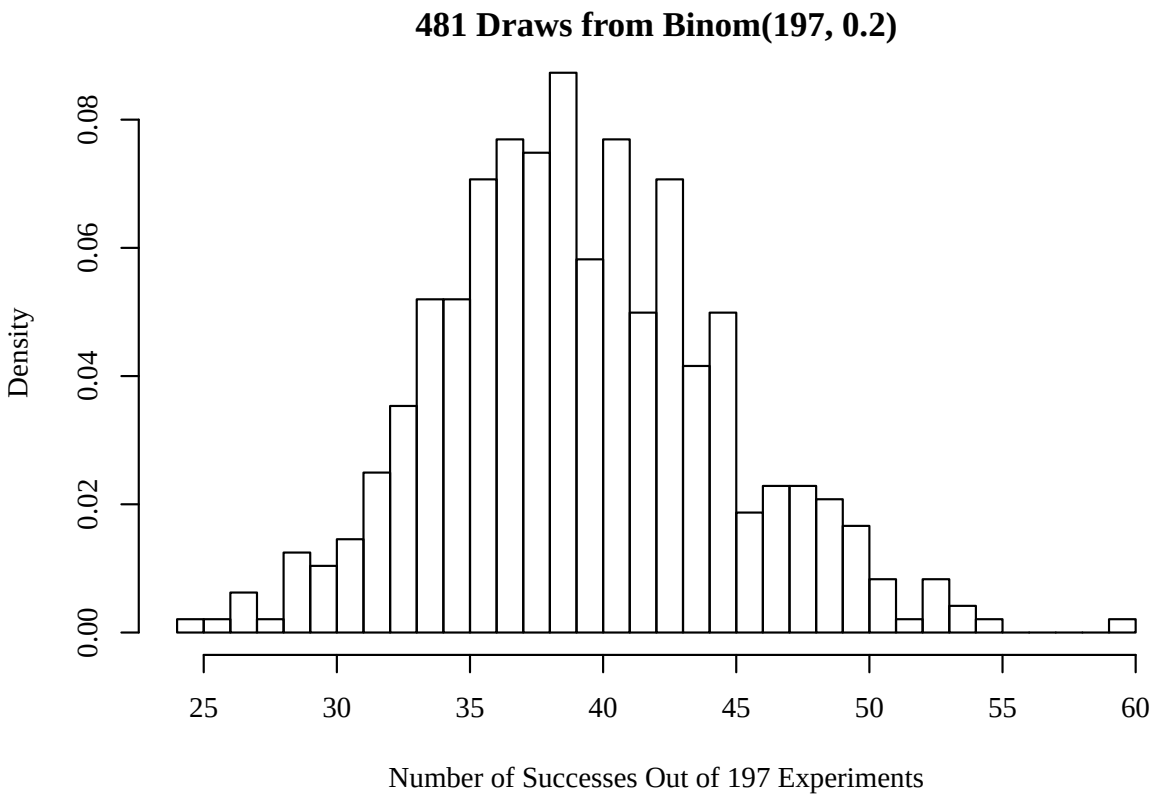
- I need 481 batches like that (each has 197 draws)!

```
lotsOfBinomial <- rbinom(481, size = 197, prob = 0.2)
head(lotsOfBinomial)
```

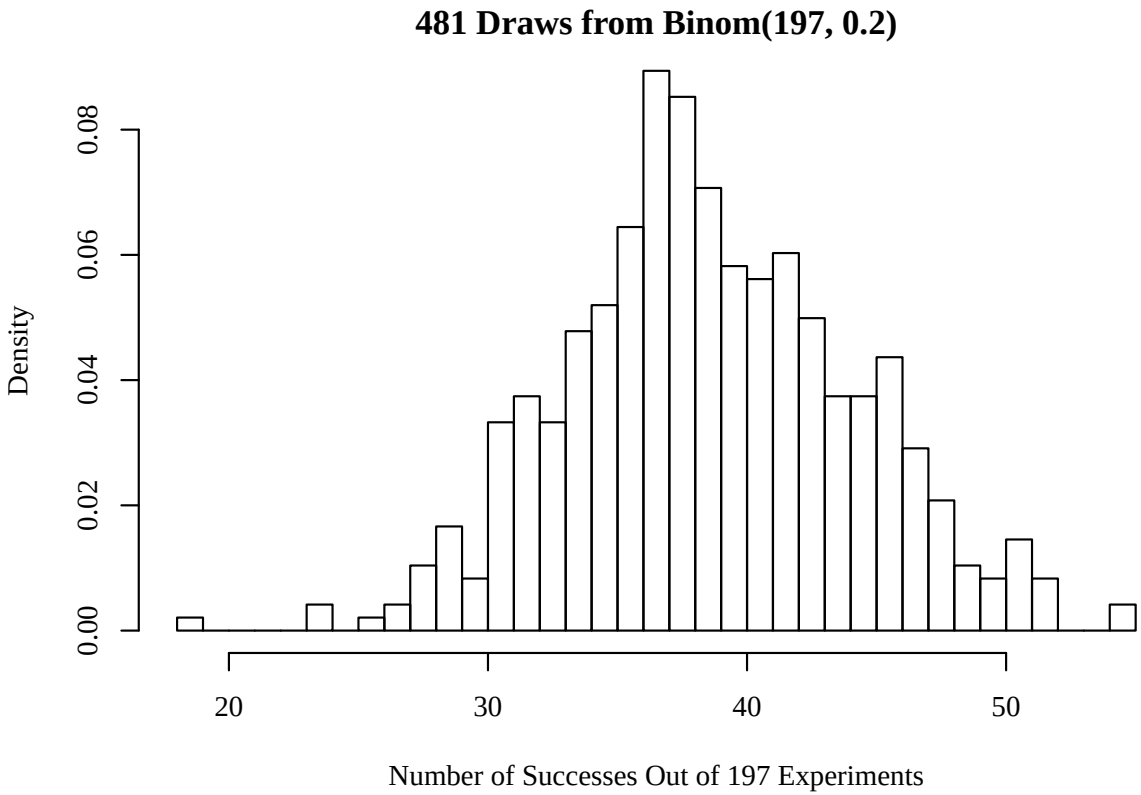
```
[1] 34 41 50 45 32 37
```

Draw The Histogram

```
hist(lotsOfBinomial, prob = TRUE, breaks=40, xlab="Number
    of Successes Out of 197 Experiments", main="481 Draws
    from Binom(197, 0.2)")
```



Draw another Sample, Another Histogram



Can calculate summary stats

```
mean(lotsOfBinomial)
```

```
[1] 39.65904
```

```
sd(lotsOfBinomial)
```

```
[1] 5.3786
```

```
mean(lotsOfBinomial2)
```

```
[1] 39.06237
```

```
sd(lotsOfBinomial2)
```

```
[1] 5.556177
```

Check out the PMF

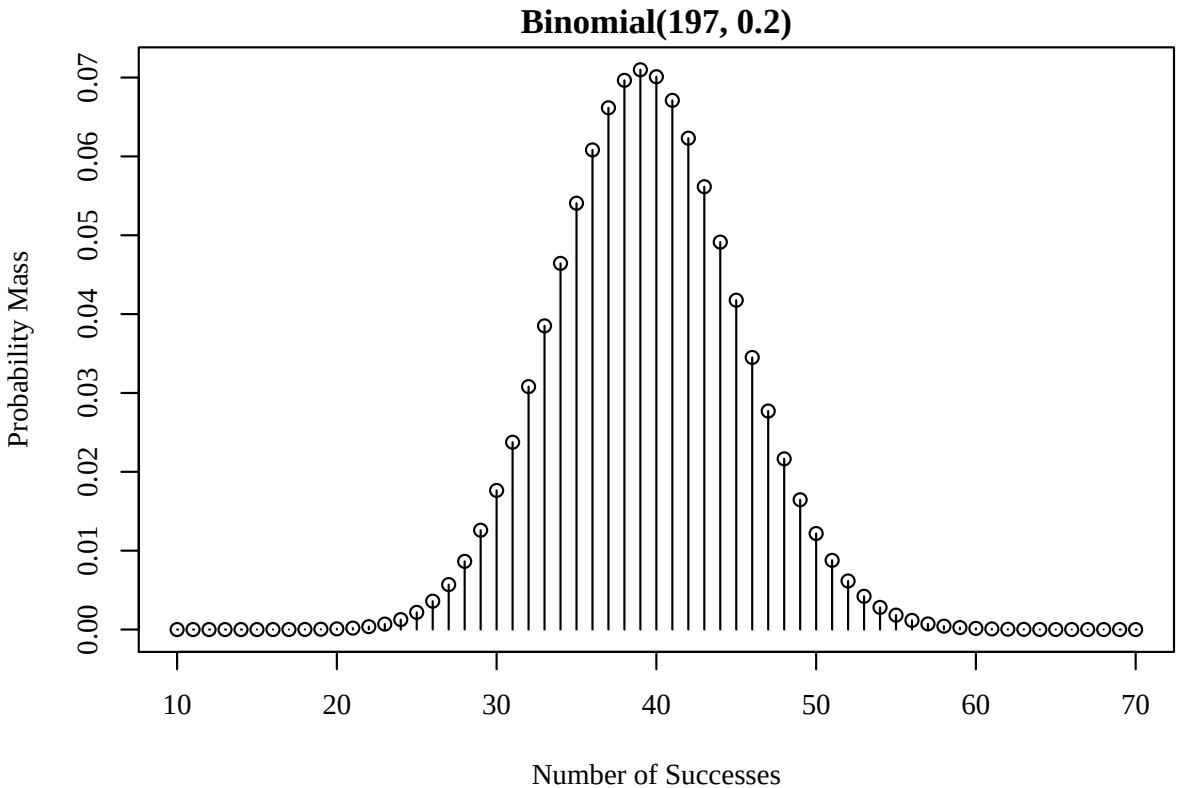
- From histograms, I discern the range is betwen 10 and 70
- I create a plotting sequence, then calculate the "true" chance of each point.

```
xseq <- 10:70
xprob <- dbinom(xseq, size = 197, prob = 0.2)
```

- a plot of type "h" provides "spikes" above each value of xseq
- use the points command to add little circles on the top.

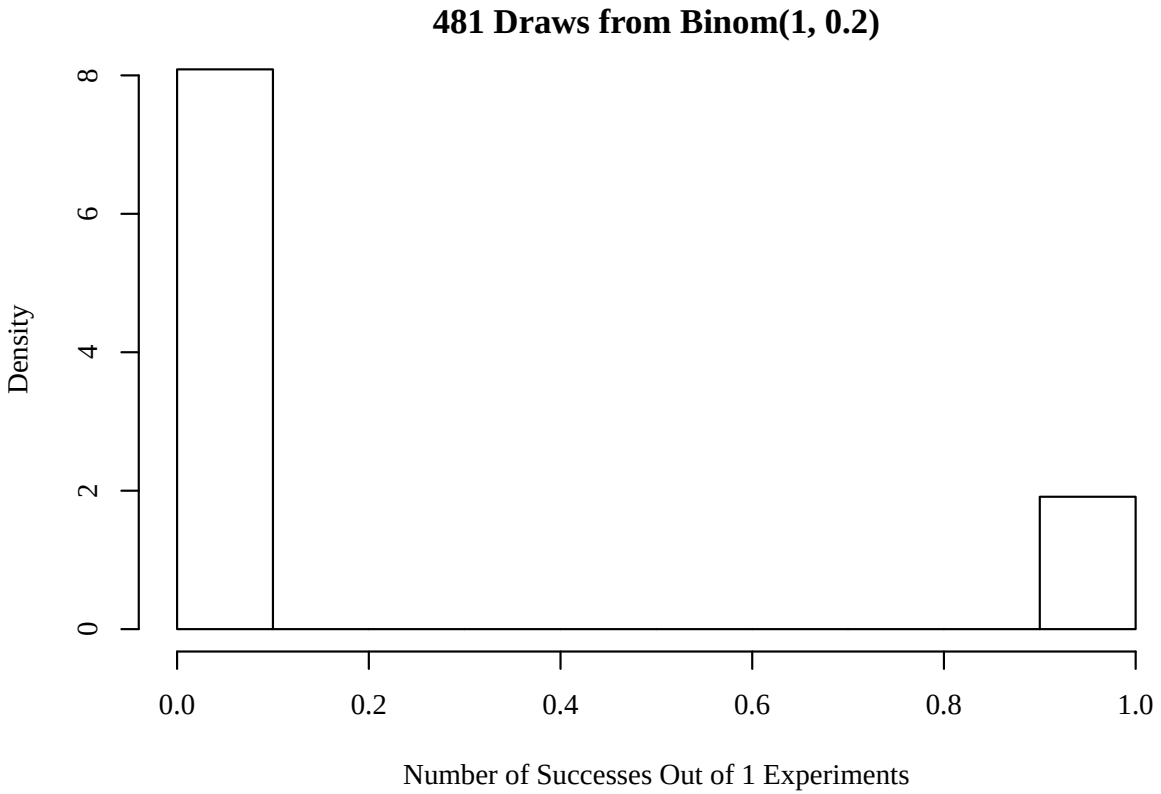
The PMF of Binomial(197, 0.2)

```
plot(xprob ~ xseq, type="h", xlab="Number of Successes",
     ylab="Probability Mass", main="Binomial(197, 0.2)")
points(xseq, xprob)
```



Wonder what would happen if "size" were 1?

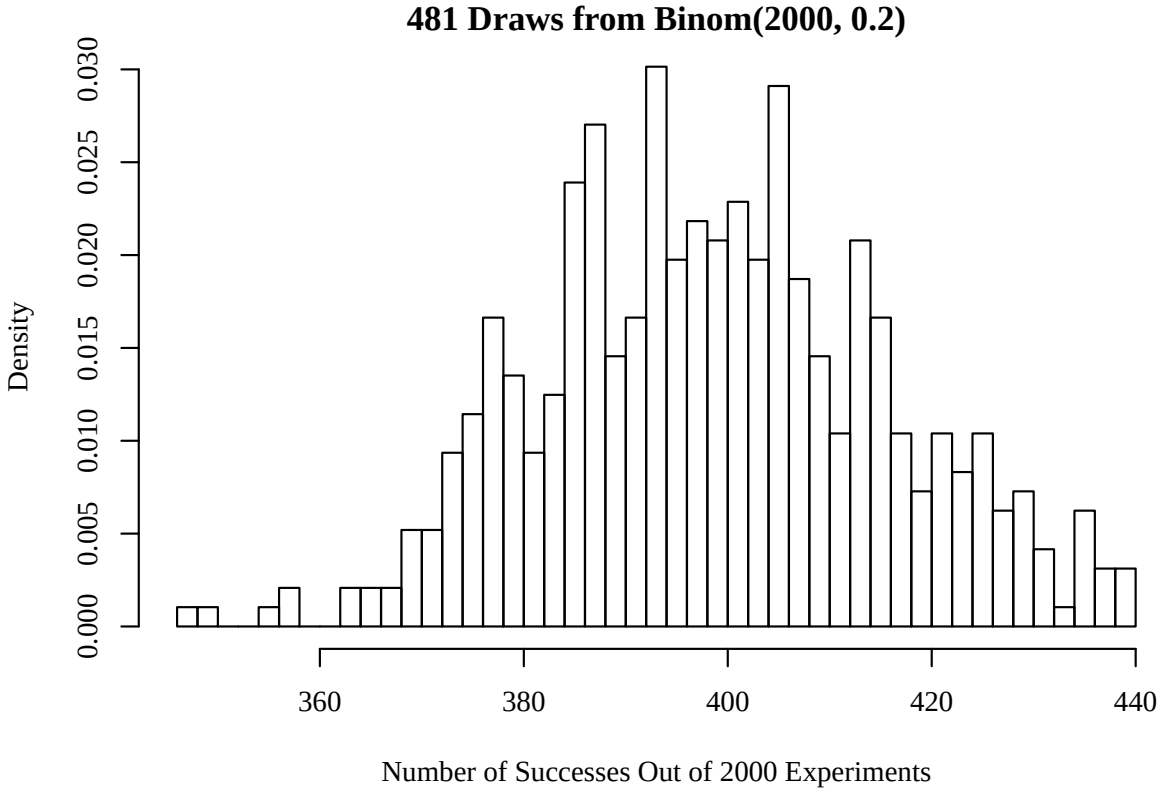
```
x <- rbinom(481, size=1, prob=0.2)
hist(x, prob = TRUE, xlab="Number of Successes Out of 1
    Experiments", main="481 Draws from Binom(1, 0.2)")
```



- This is sometimes called a "Bernoulli" process

Wonder what would happen if "size" were 2000

```
x <- rbinom(481, size=2000, prob=0.2)
hist(x, prob = TRUE, breaks = 50,  xlab = "Number of
    Successes Out of 2000 Experiments", main="481 Draws
    from Binom(2000, 0.2)")
```

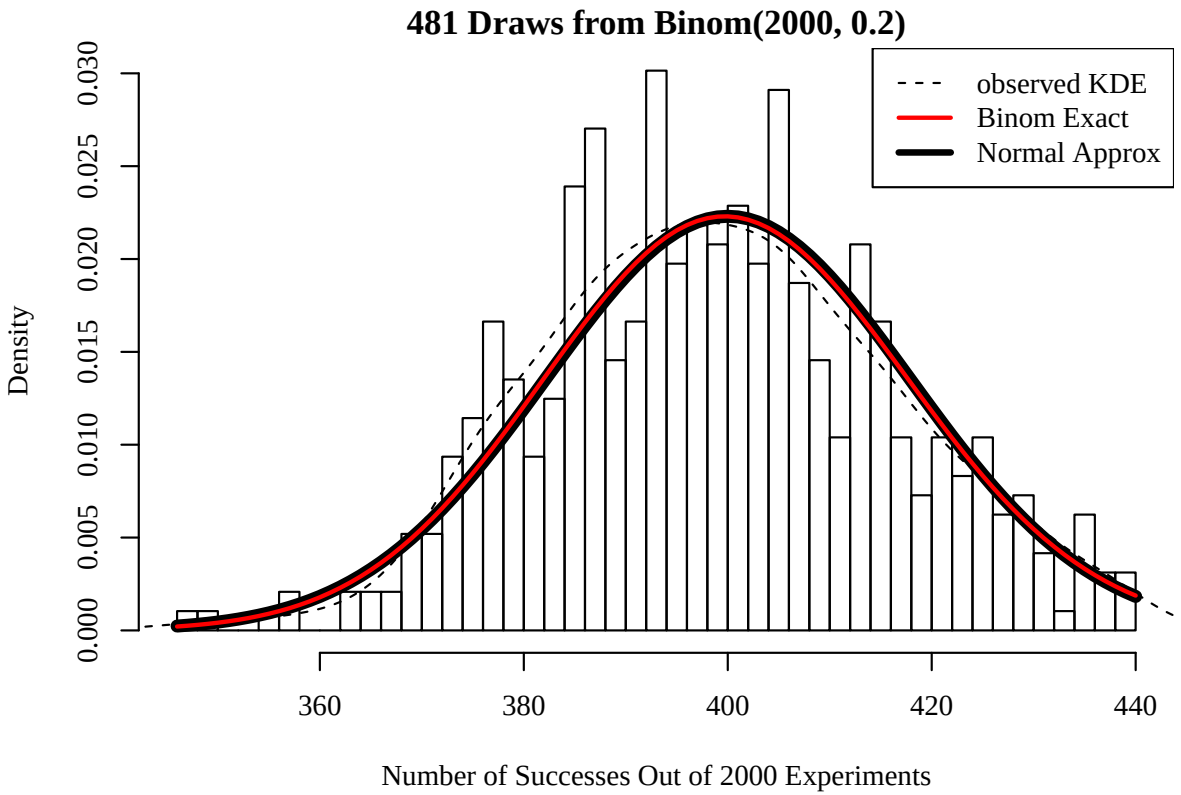


- as "size" growing larger. What does this remind me of?

Wonder what would happen if "size" were 2000

```
hist(x, prob = TRUE, breaks = 50,  xlab="Number of
    Successes Out of 2000 Experiments", main="481 Draws
    from Binom(2000, 0.2)")
lines(density(x), lty = 2)
xseq <- seq( min(x), max(x), by = 1)
xprob <- dnorm(xseq, m = 2000*0.2, sd = sqrt(2000*(0.2)*(0
    .8)))
lines(xseq, xprob, lwd = 6)
xbinprob <- dbinom(xseq, size = 2000, prob = 0.2)
lines(xseq, xbinprob, col = 2, lwd = 2, lty = 1)
legend("topright", legend = c("observed KDE","Binom Exact"
    , "Normal Approx"), lty = c(2,1,1), lwd = c(1,2,3),
    col = c(1,2,1))
```

Wonder what would happen if "size" were 2000



- as "size" growing larger. What does this remind me of?

Let's expore the Tweedie distribution

- I had heard of the Tweedie, but never studied it before.
- Install the package "tweedie".

Run "?tweedie"

Usage:
dtweedie(y, xi=power, mu, phi, power=NULL)
ptweedie(q, xi=power, mu, phi, power=NULL)
rtweedie(n, xi=power, mu, phi, power=NULL)

- 3 parameters, xi, mu, and phi.
- mu, the Greek μ, is the Expected Value
- phi, the Greek ϕ, is "dispersion". Note, that is NOT variance.
- xi, the Greek ξ, is also called **power**.

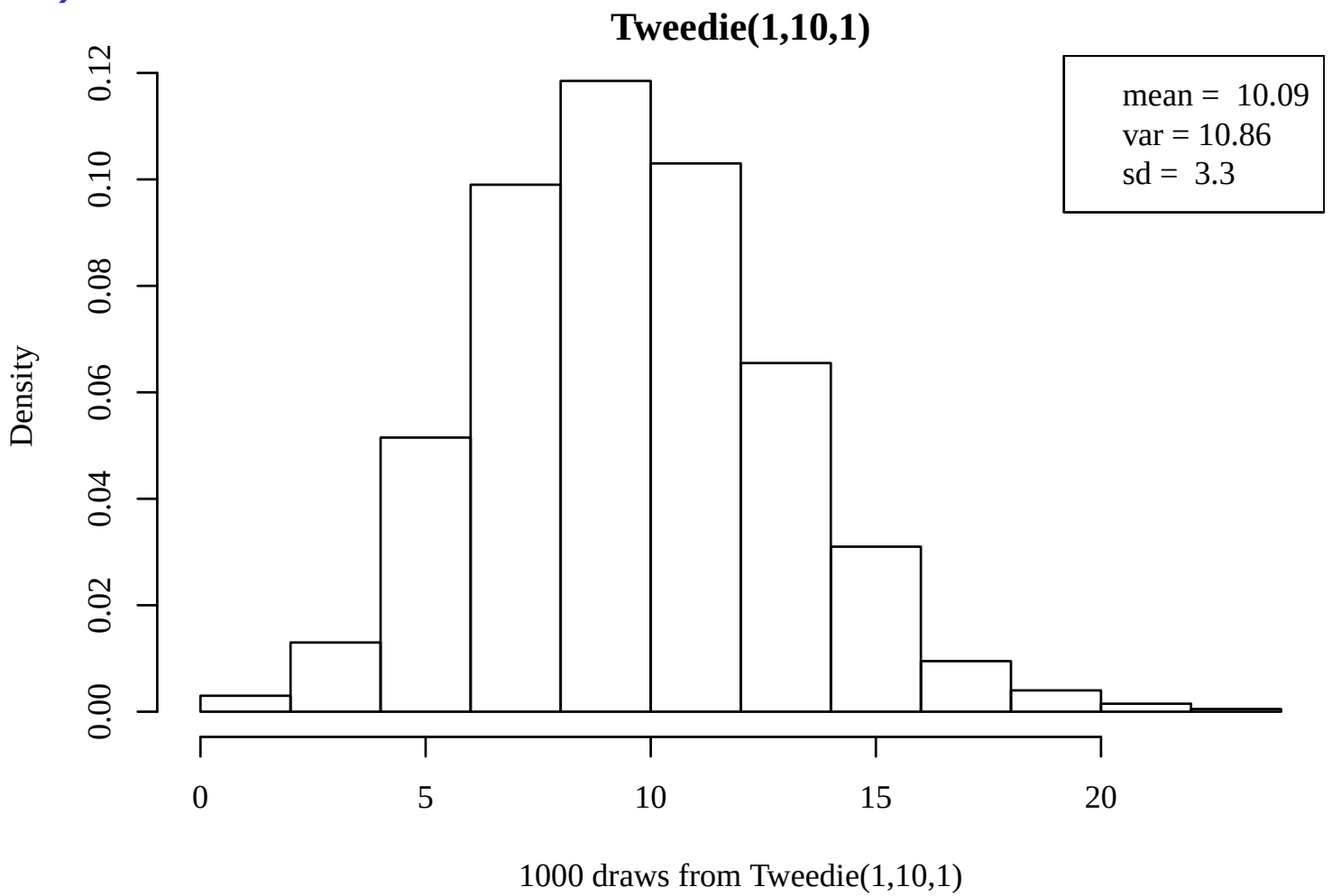
Better run some examples to see what's up.

- The help says ξ is a magic coefficient

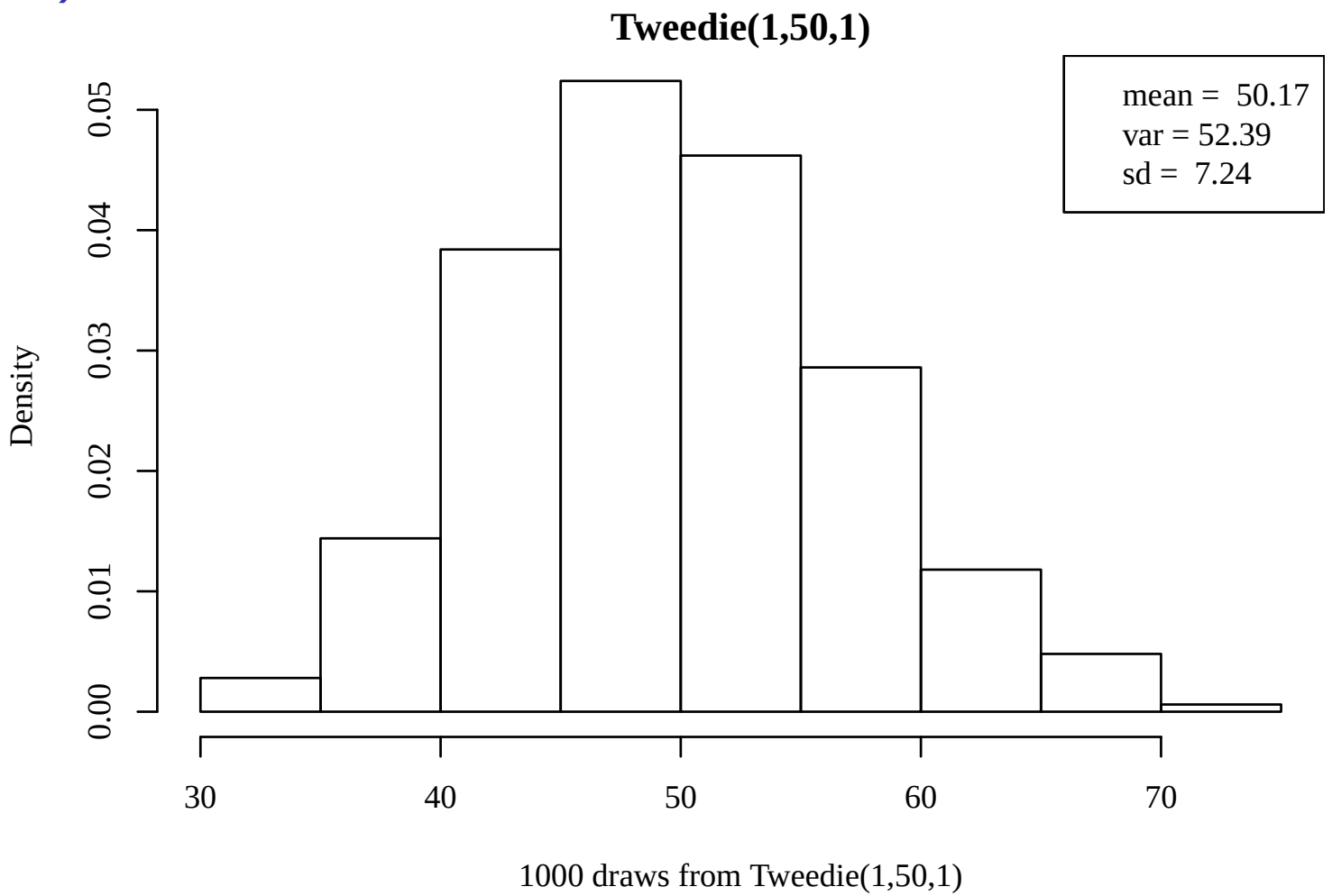
$$Var(x) = \phi * \mu^{\xi}$$

- See why they call xi "power"? The power turns up the variance.
- Calculations very slow for many values of xi.
- "tweedie" package only allows if $\xi >= 1$.

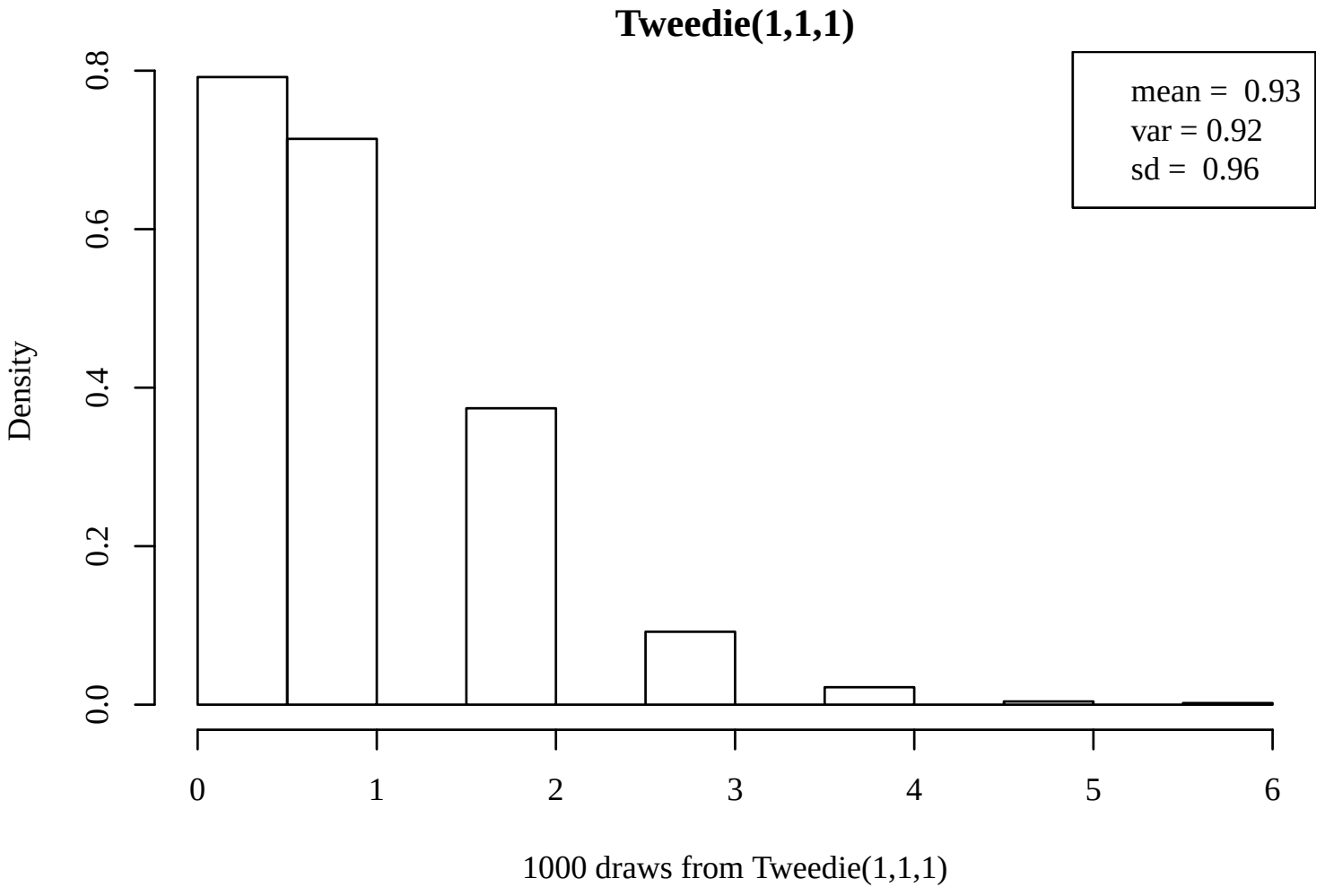
1000 Draws From a Tweedie(xi = 1, mu = 10, phi = 1)



1000 Draws From a Tweedie(xi = 1, mu = 50, phi = 1)



1000 Draws From a Tweedie(xi=1, mu=1, phi=1)



- Weird. It appears to be discrete!

Wow. It is discrete. When xi = 1, at least

```
table(x)
```

```
x
  0   1   2   3   4   5   6
396 357 187  46  11   2   1
```

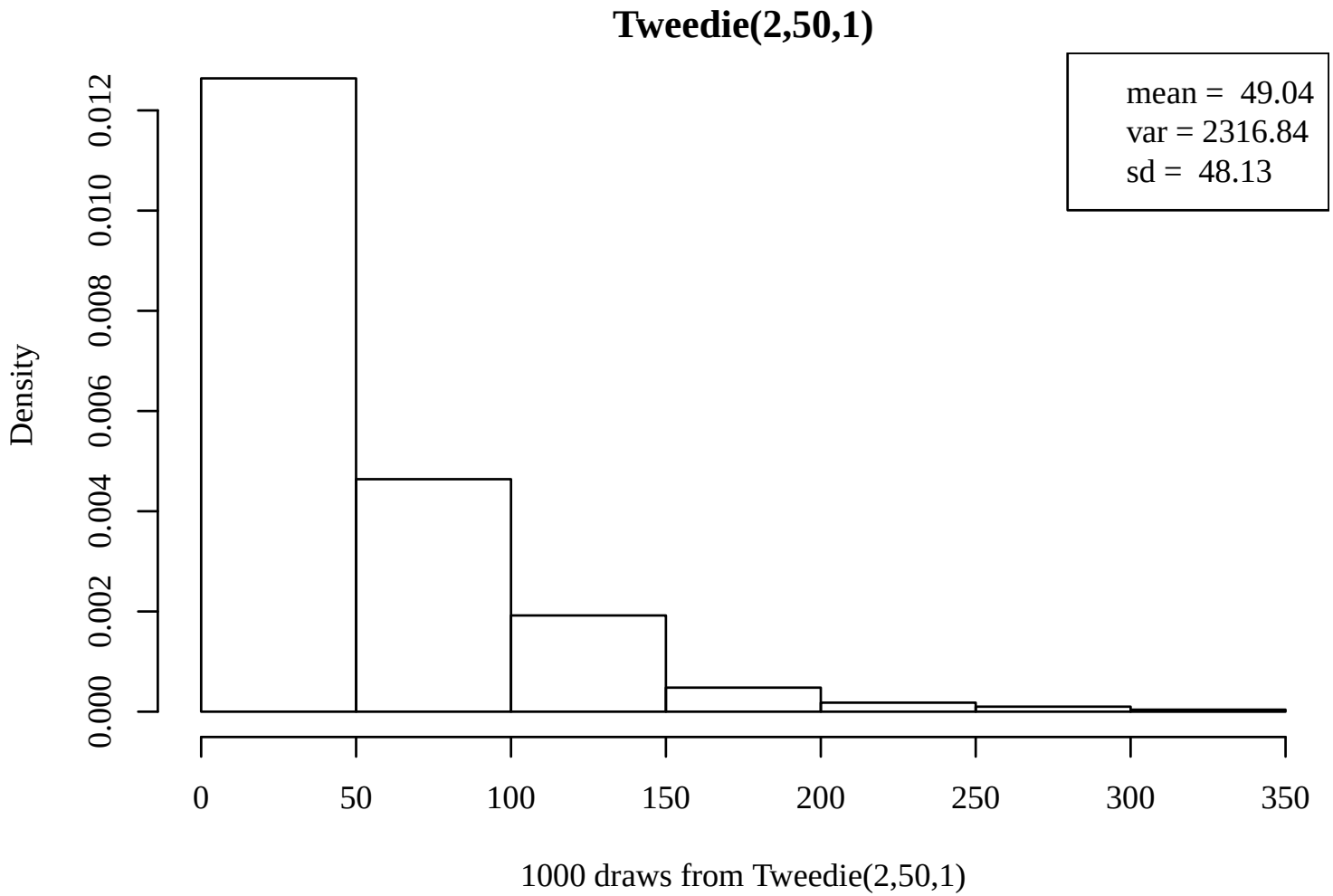
```
x <- rtweedie(1000, xi = 1, mu = 20, phi = 1)
table(x)
```

```
x
  6   8   9  10  11  12  13  14  15  16  17  18  19  20  21
     22  23  24  25  26  27  28  29  30  31  32  33  35
  1   1   3  10  11  18  24  41  58  75  67  86  88 105  83
     68  53  50  42  39  28  21   5   4  11   6   1   1
```

Remembering ?dtweedie

- Now I recall from `?dtweedie` that when `xi = 1`, the Tweedie distribution is a Poisson distribution.
- I also recall that if xi=2, then Tweedie is Gamma distribution.
- But Gamma is a continuous distribution.

1000 Draws From a Tweedie(xi = 2, mu = 50, phi = 1)



With xi=2, Tweedie is like a Gamma

- Recall theoretical $Var(x) = phi * mu^{xi} = phi * mu^2$
- Reminds of $Gamma(\alpha, \beta)$ because the Gamma's
  - $E[x] = \alpha \times \beta$
  - $Var[x] = \alpha\beta^2$
- To make Tweedie variance match, let $\alpha = phi = 1$
- And this draw is continuous

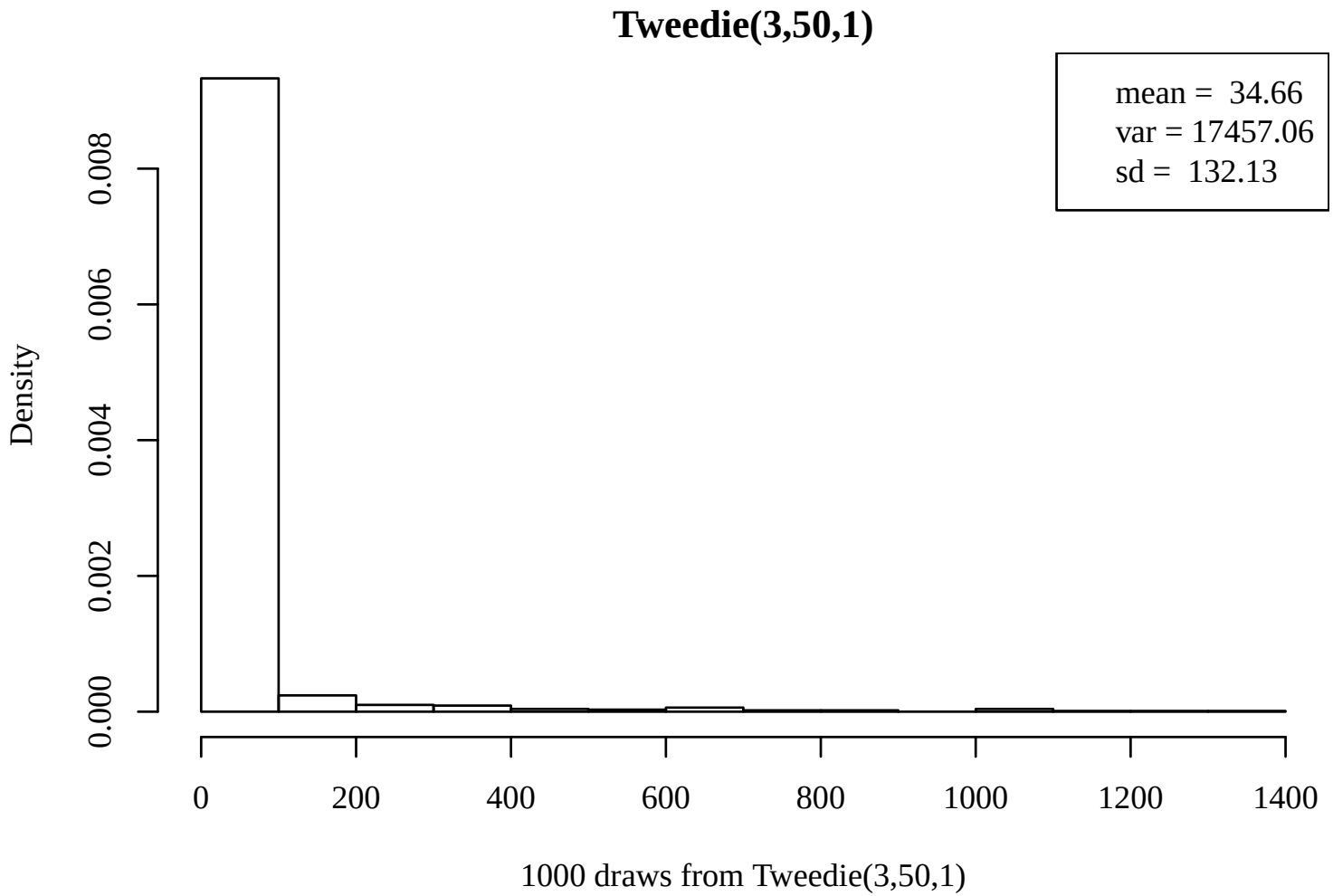
```
head(x)
```

```
[1]  73.426794   9.646750   2.303447 140.960617 102
    .687257  10.829624
```

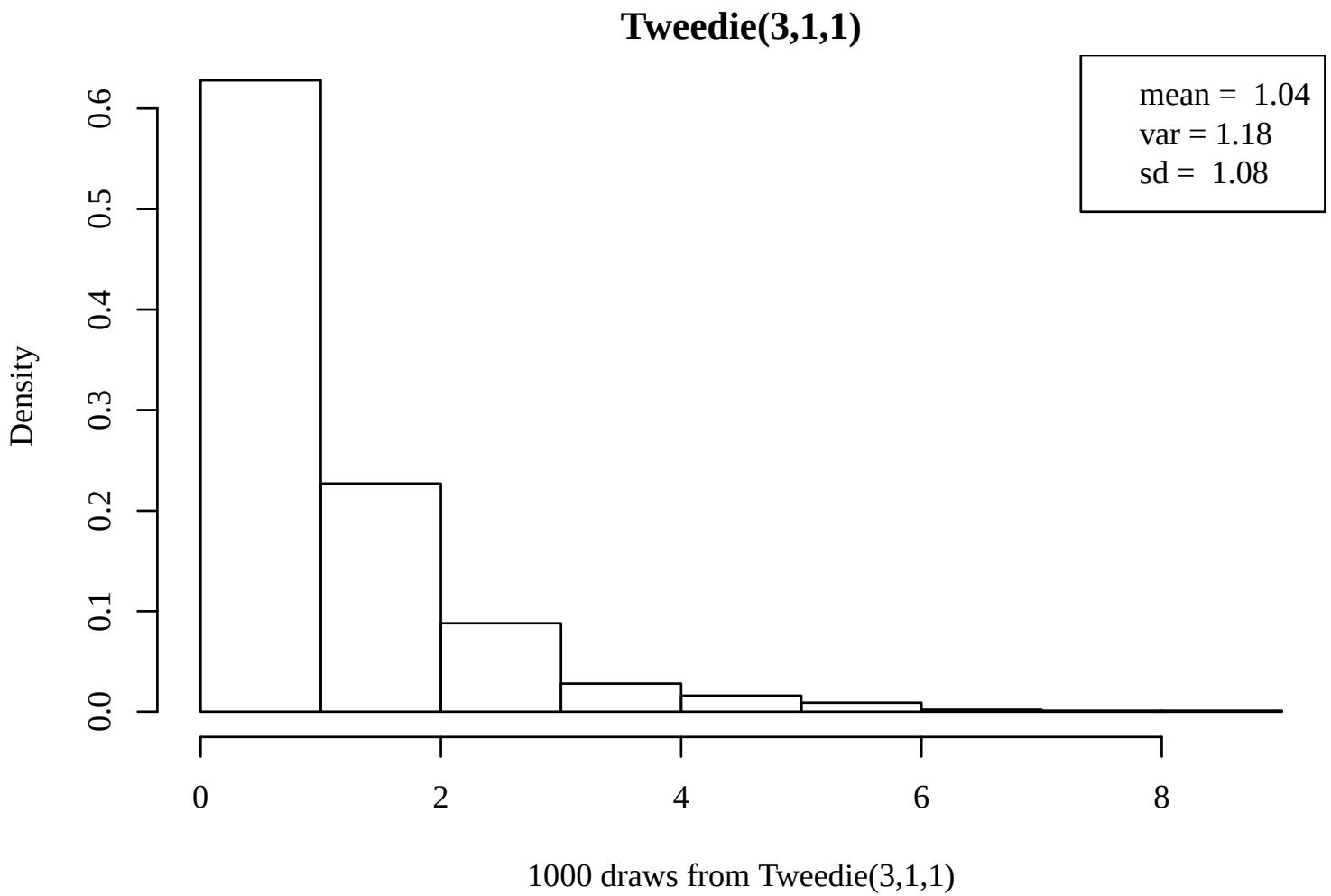
Still trying to figure out advantages.

- Calculations with Tweedie model very difficult, help page has plenty of detail and caution about methods.
- Why would I want that?
  1. I make an abstract model that is Gamma sometimes, or Poisson sometimes, or Normal sometimes?
  2. If I could estimate xi from data, I could let the data "tell me" which distribution to use.

1000 Draws From a Tweedie(xi=3, mu=50, phi=1)



1000 Draws From a Tweedie(xi=3, mu=1, phi=1)



The Basic Idea behind Monte Carlo Analysis

- Draw a lot of samples
  - Collect M samples of size N
  - Calculate an estimate (e.g., mean) for each sample
- What distribution will be observed among all of those estimates?

Refresher on Sampling Distributions

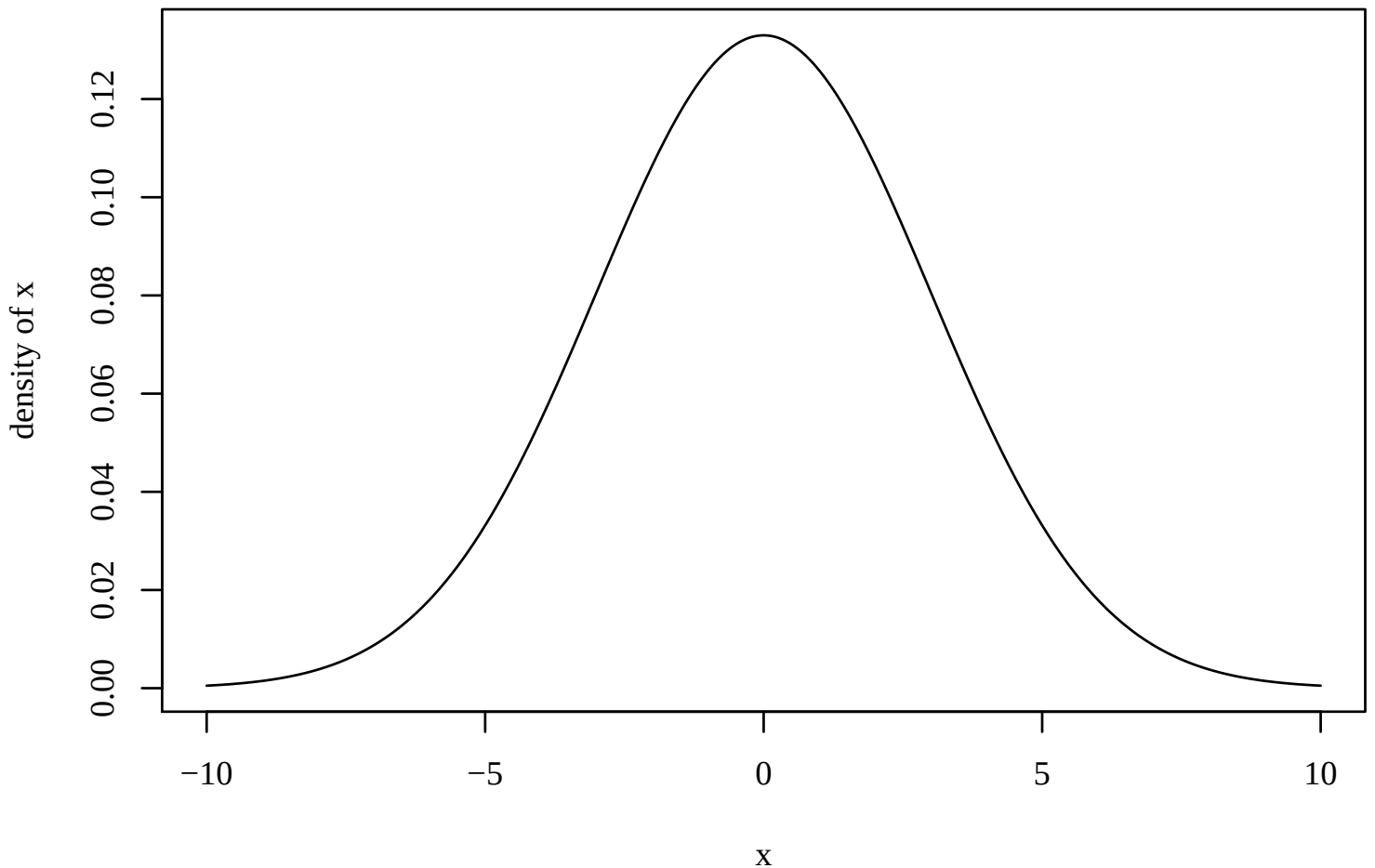
$\bar{x}$ is one sample mean

- If $E[x] = \mu$, then $E[\bar{x}] = \mu$
- If $Var[x] = \sigma^2$, then $Var[\bar{x}] = \frac{Var[x]}{N}$
- Which implies $SD[\bar{x}] = \frac{SD[x]}{\sqrt{(N)}}$
- As sample size $\rightarrow \infty$, the distribution of means tends toward a Normal distribution (The CLT)

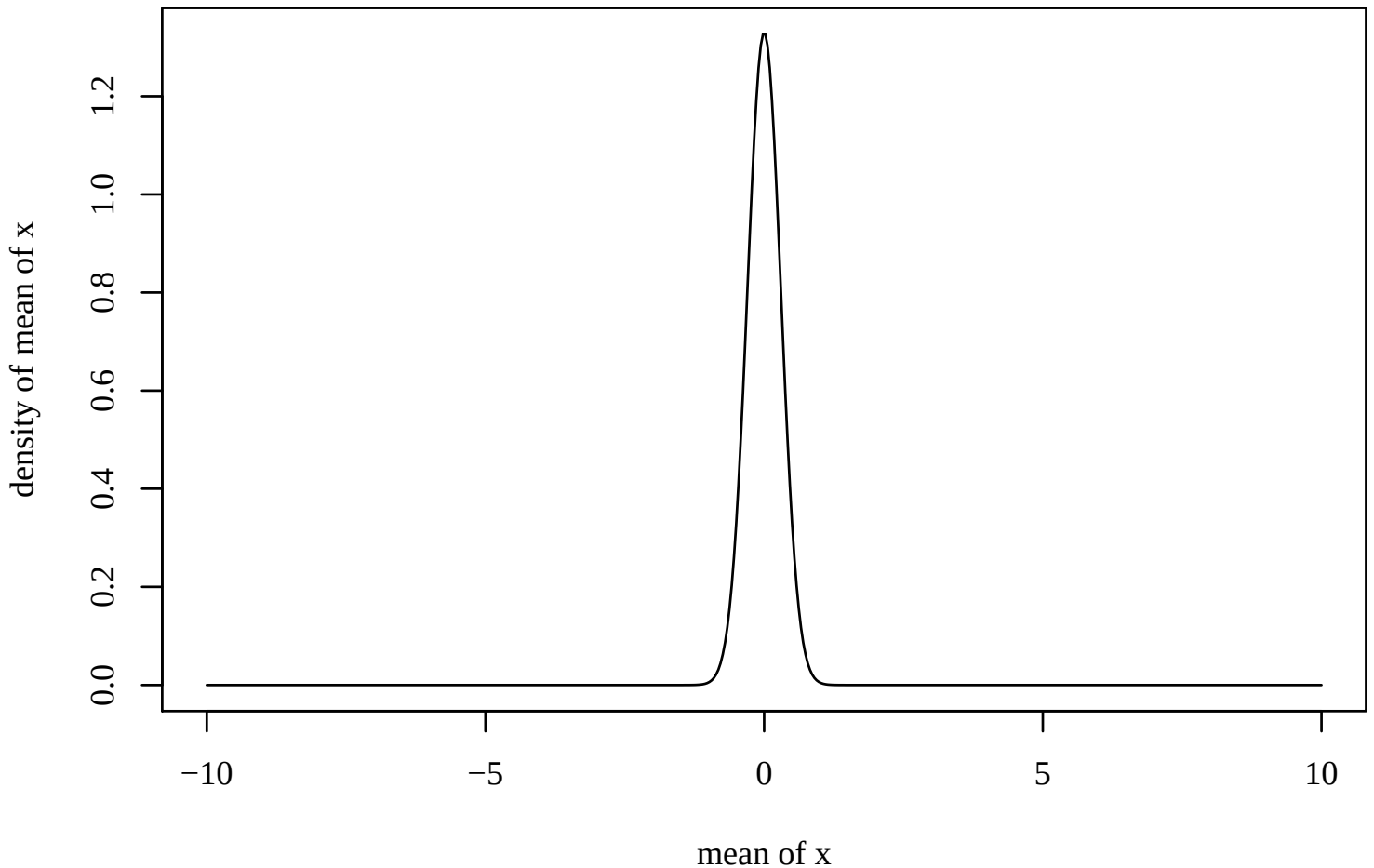
The Distribution of the Mean is "Spike-ish"

Please observe the effect of sample size on the variance of $\bar{x}$ drawn from a Normal distribution.

Distribution of $x \sim Normal(0, 3^2)$



Distribution of Means, Sample=100 ($Normal(0, 3^2/100)$)



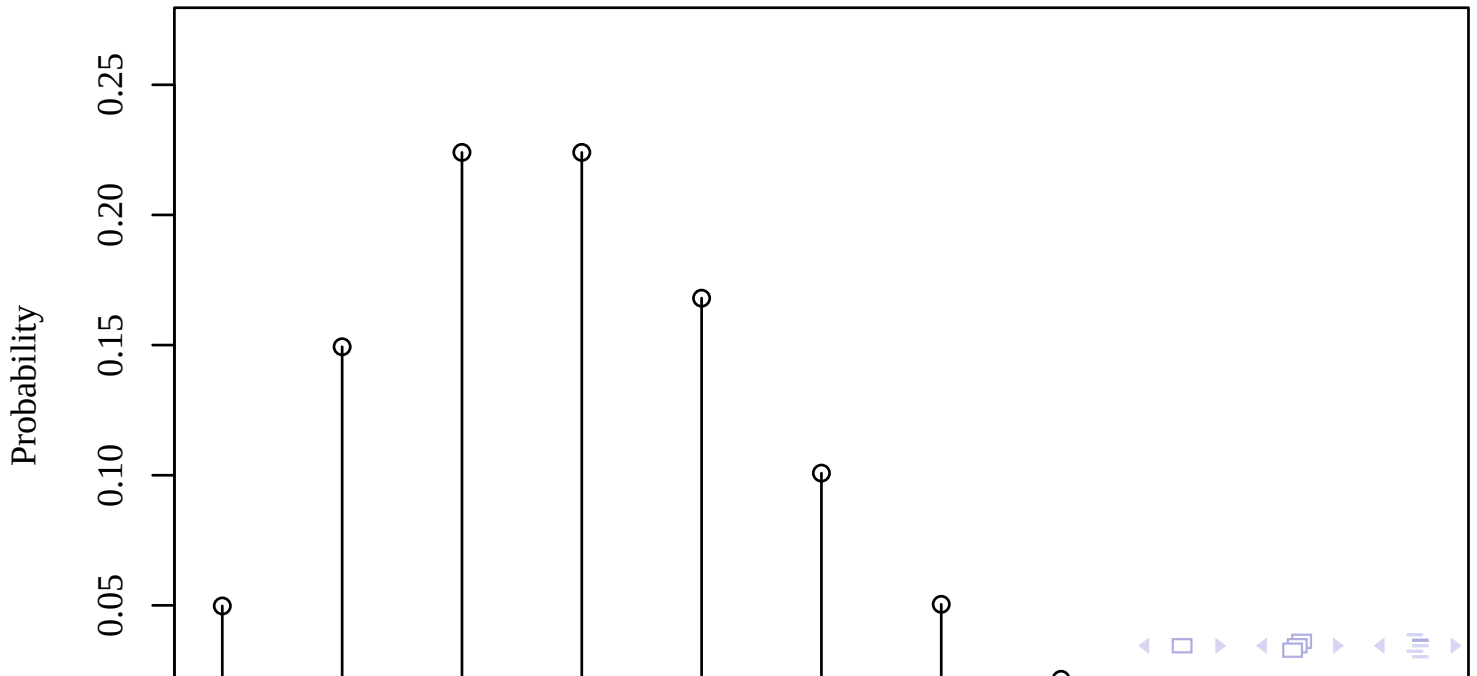
I Have Made a Lot of Examples Like This Over the Years

- Display the distribution of the variable
- Draw samples
- Explore sampling distribution

Example 1: The Poisson Distribution

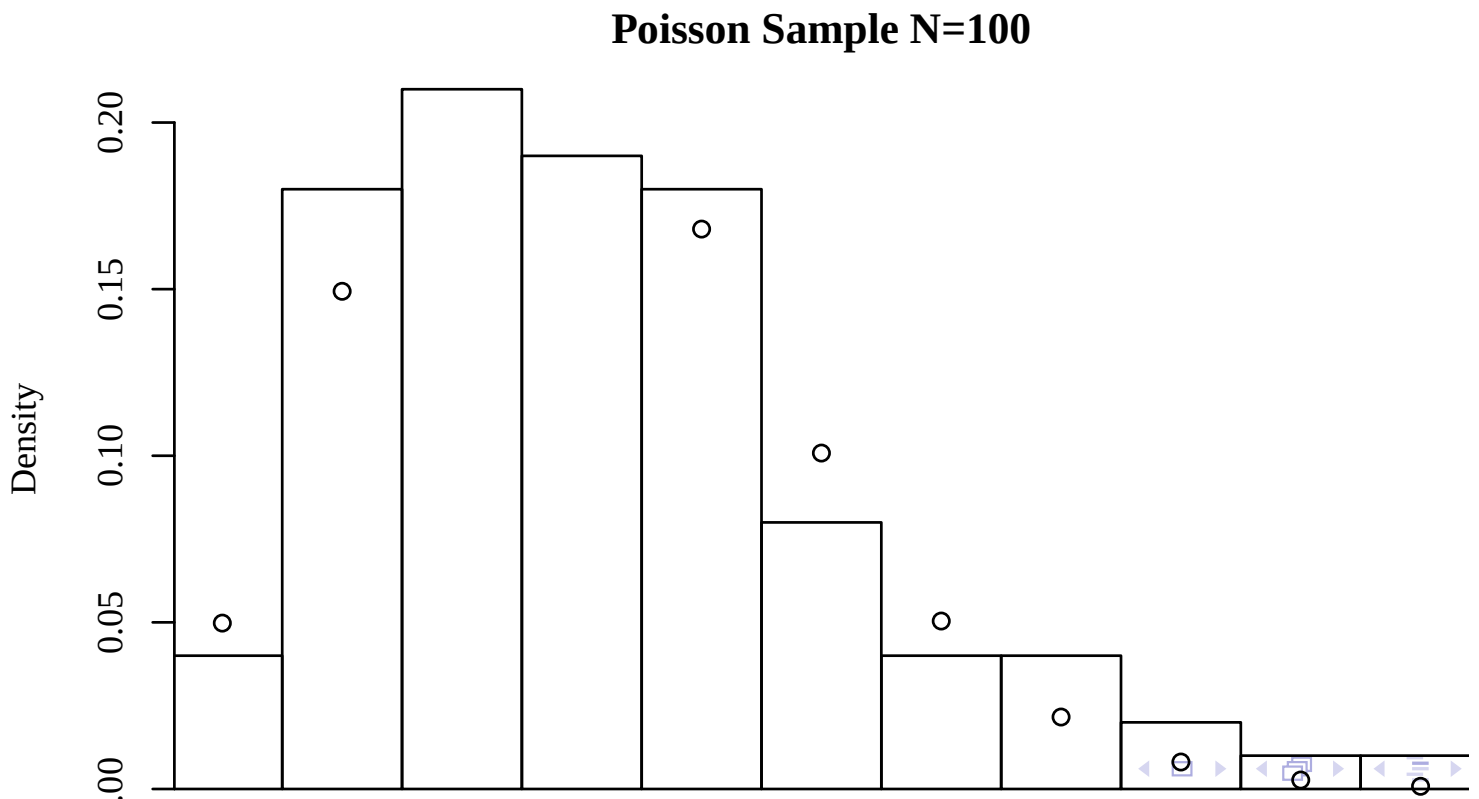
Set the lambda parameter (λ = mean=variance) at 3.

```
xseq <- seq(0,10)
xdpois <- dpois(xseq,lambda=3)
plot(xdpois ~ xseq, type = "h", xlab = "a Poisson variate
    with lambda=3", ylab = "Probability", ylim = c(0, 1.2*
    max(xdpois)) )
points(xseq, xdpois)
```



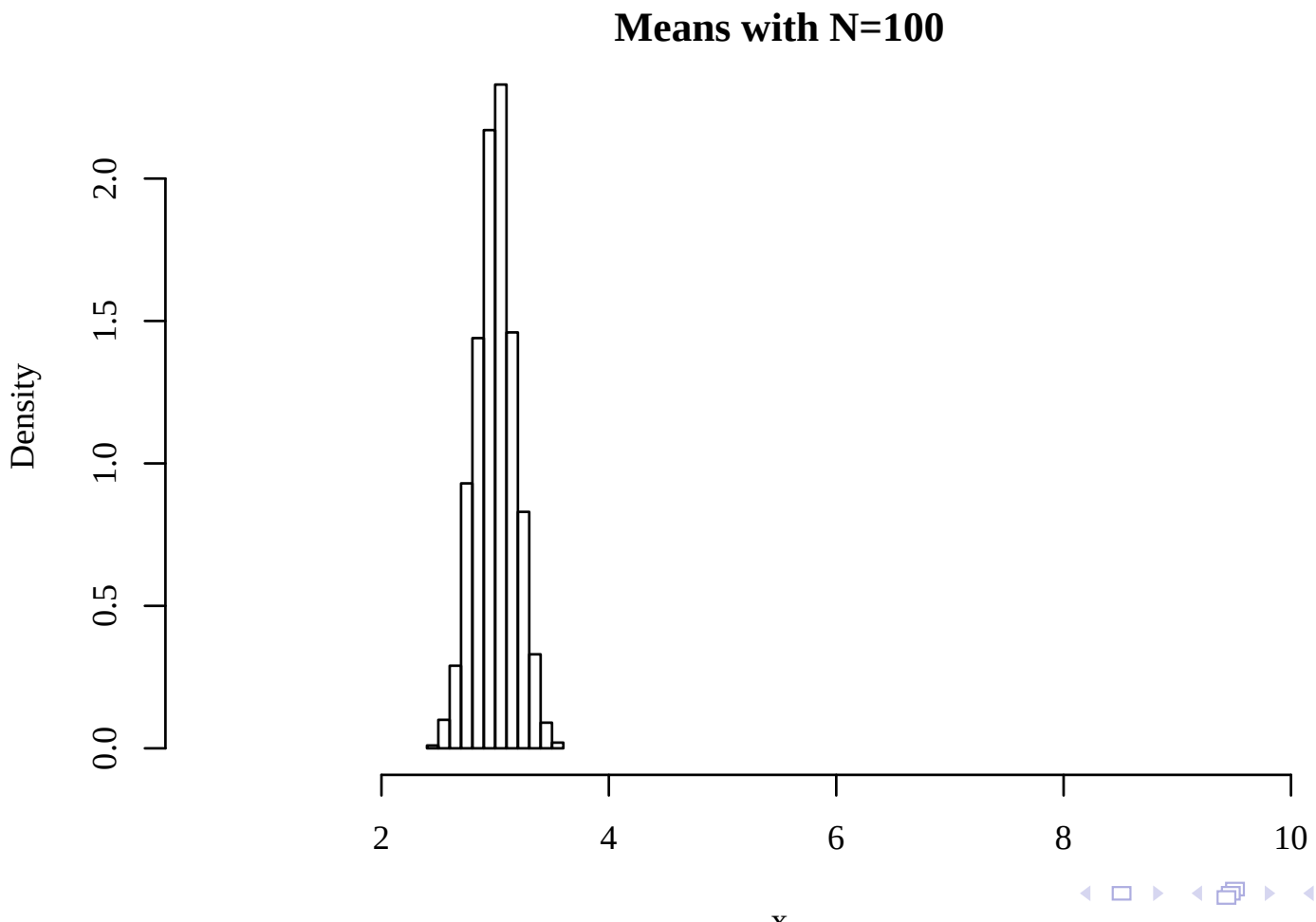
One Example Sample, Poisson(3), SampleSize=100

```
hist(0.5+rpois(100, lambda = 3), xlim = c(0.5, 10.5),
     prob = TRUE, xlab = "x", main = "Poisson Sample N=100
     ")
points(xseq+0.5, xdpois, lty=4)
```



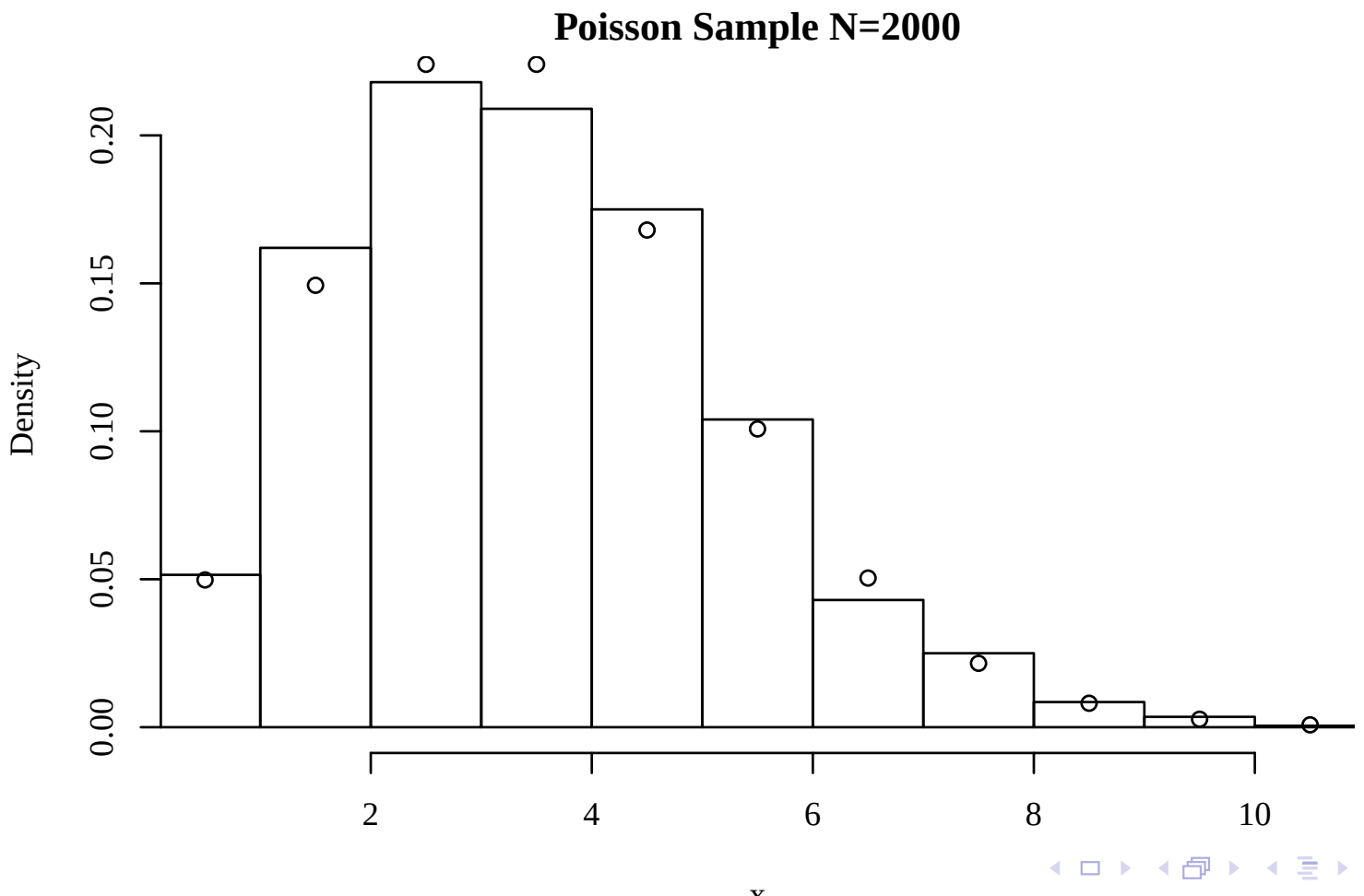
Means from 1000 Poissons, Sample Size=100

```
m100 <- replicate(1000, mean(rpois(100, lambda=3)))
hist(m100, xlim = c(0.5,10.5), prob = TRUE, xlab = "x",
     main = "Means with N=100")
```



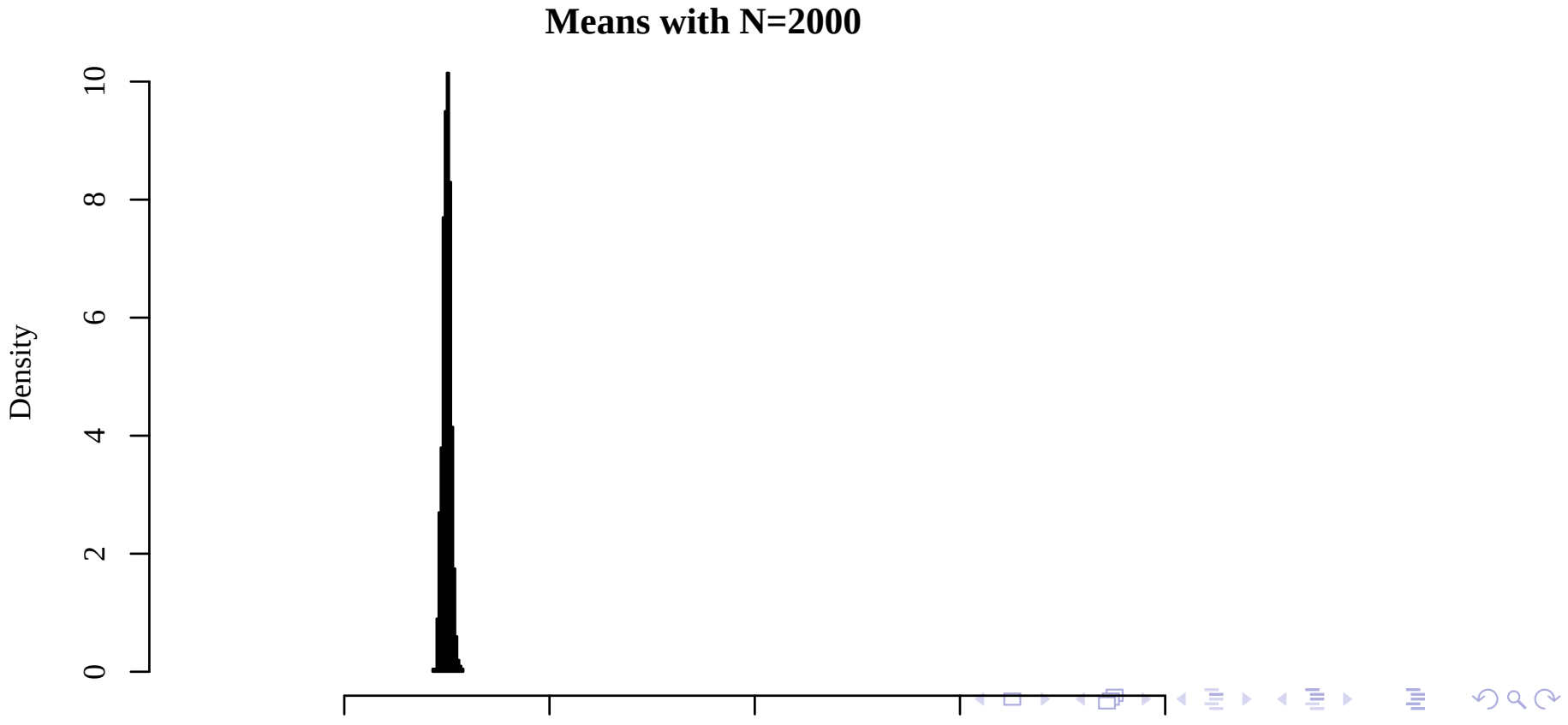
Poisson(3), SampleSize=2000

```
hist(0.5+rpois(2000, lambda=3), xlim = c(0.5,10.5), prob =
    TRUE, xlab = "x", main = "Poisson Sample N=2000")
points(xseq+0.5, xdpois, lty=4)
```

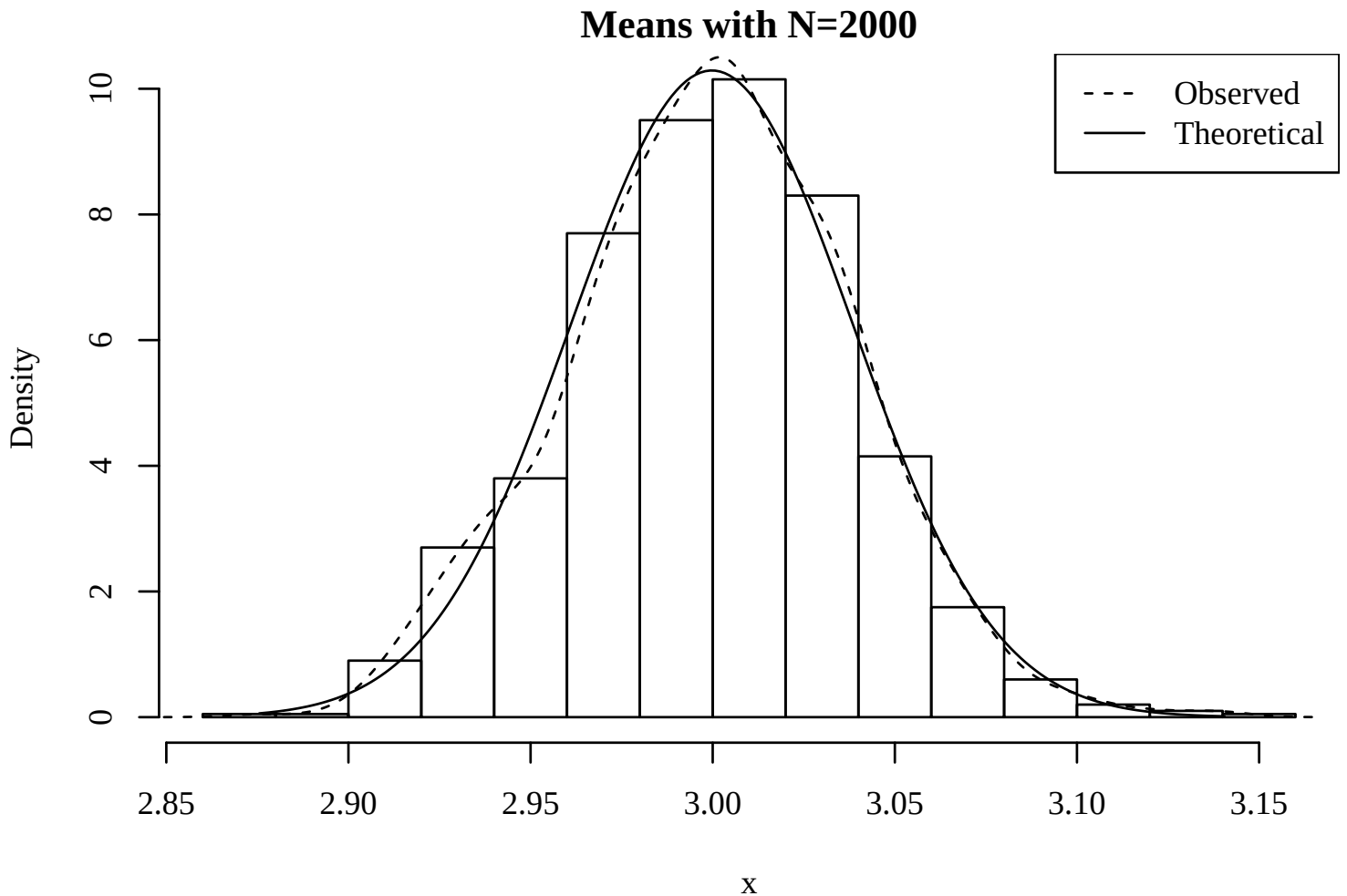


Means from 1000 Poisson samples, Sample Size=2000

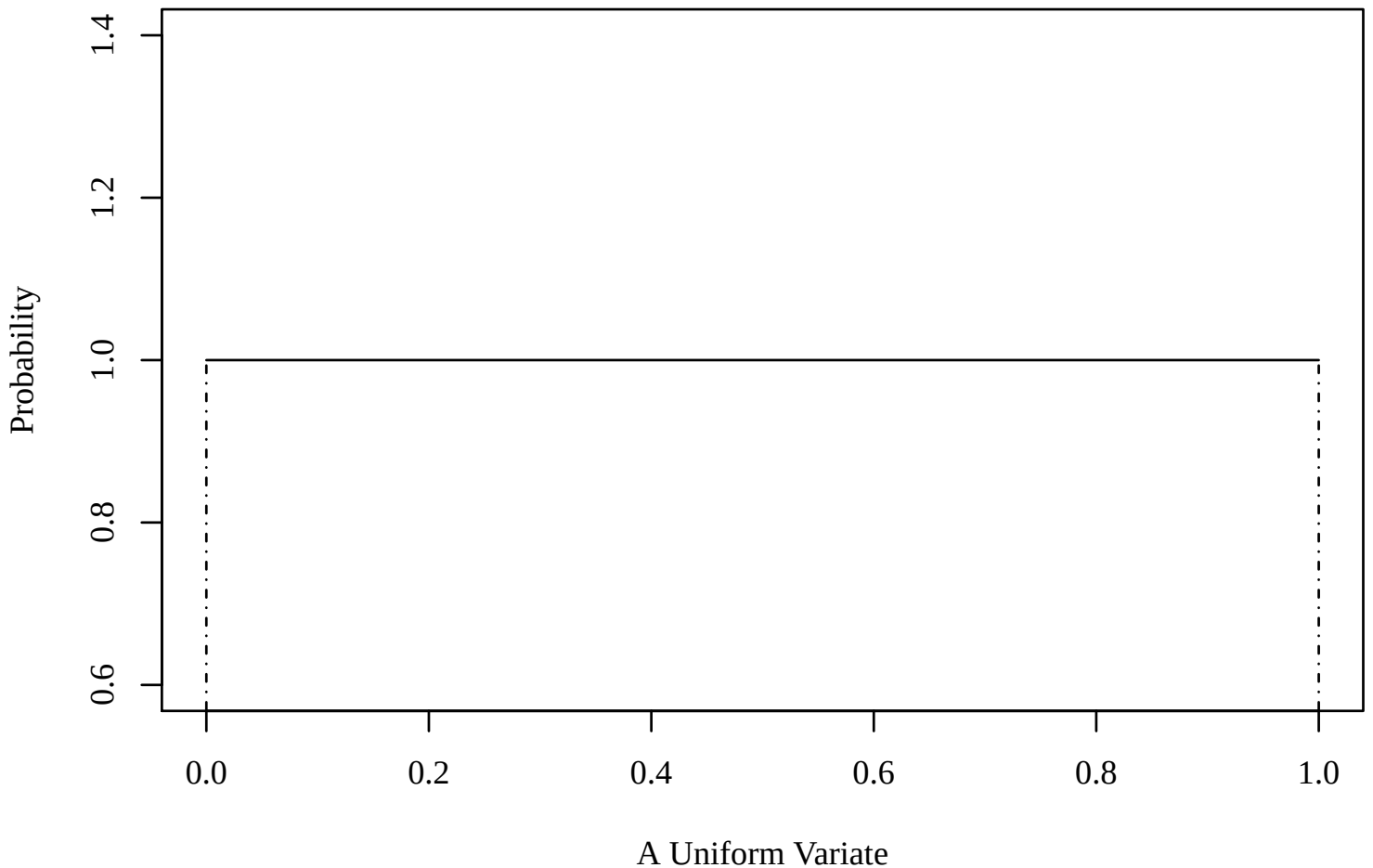
```
m2000 <- replicate(1000, mean(rpois(2000, lambda = 3)))
hist(m2000, xlim = c(0.5, 10.5), prob = TRUE, xlab = "x",
     main = "Means with N=2000")
```



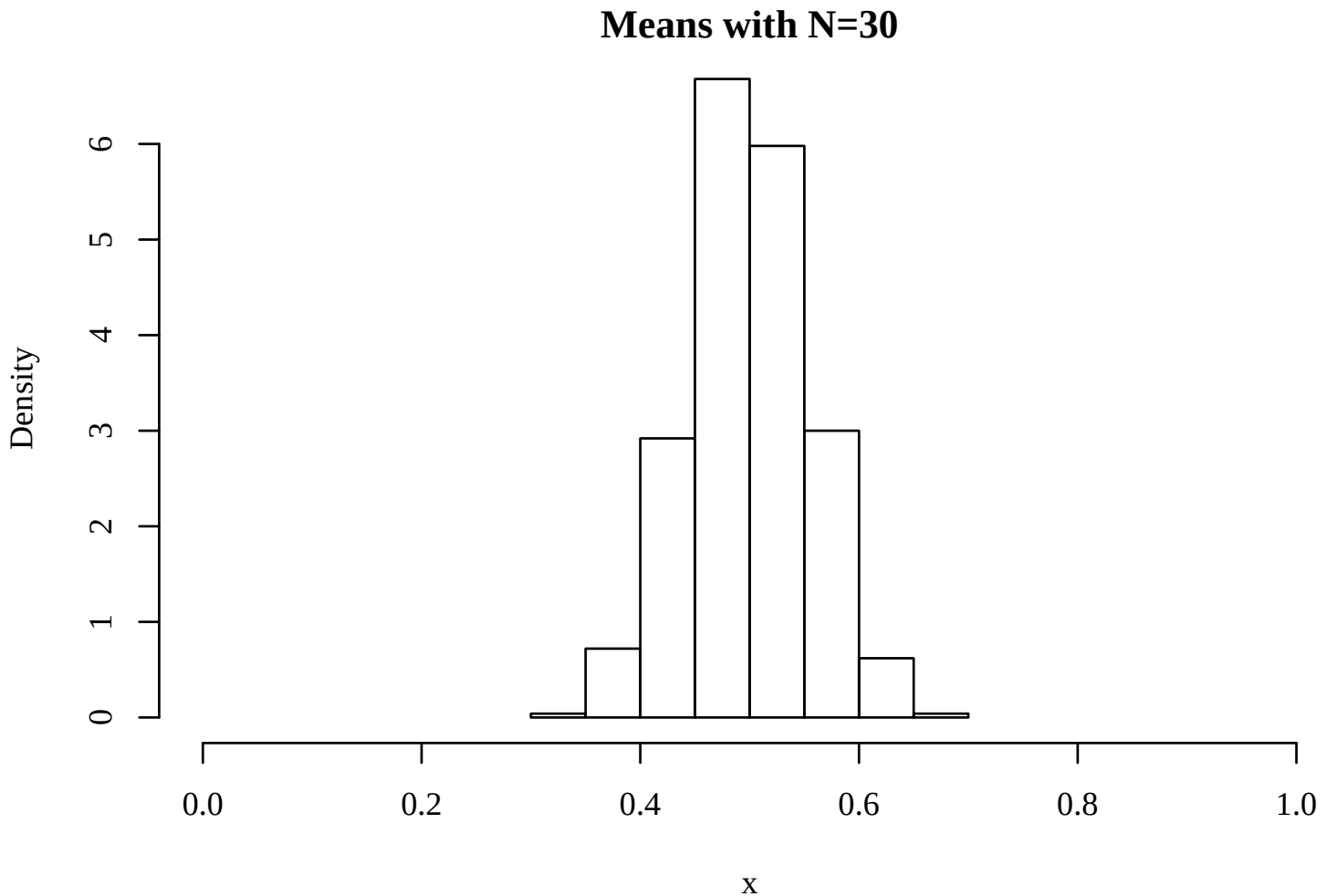
Zoom in on same Figure (N=2000)



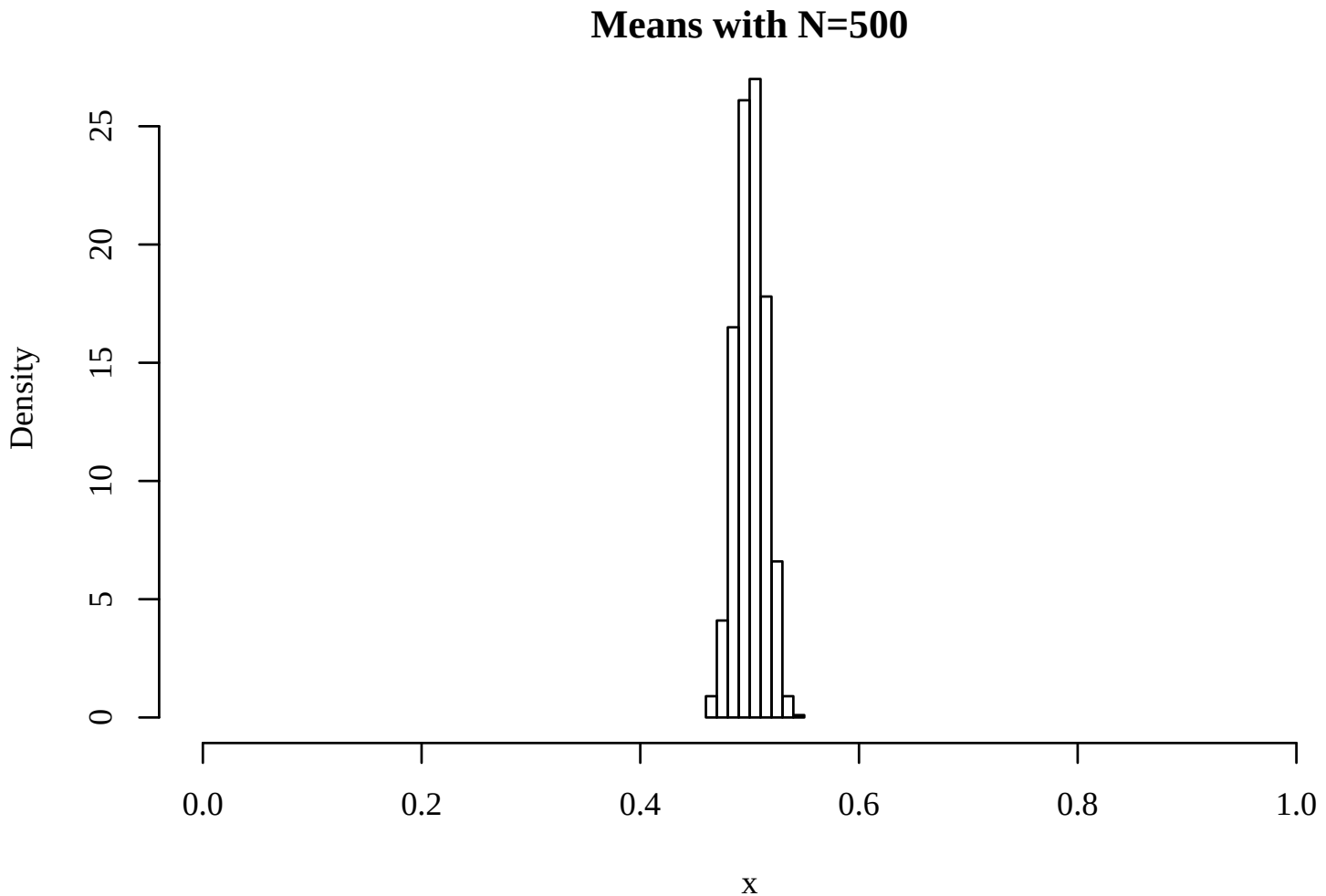
Example 2: The Uniform Distribution



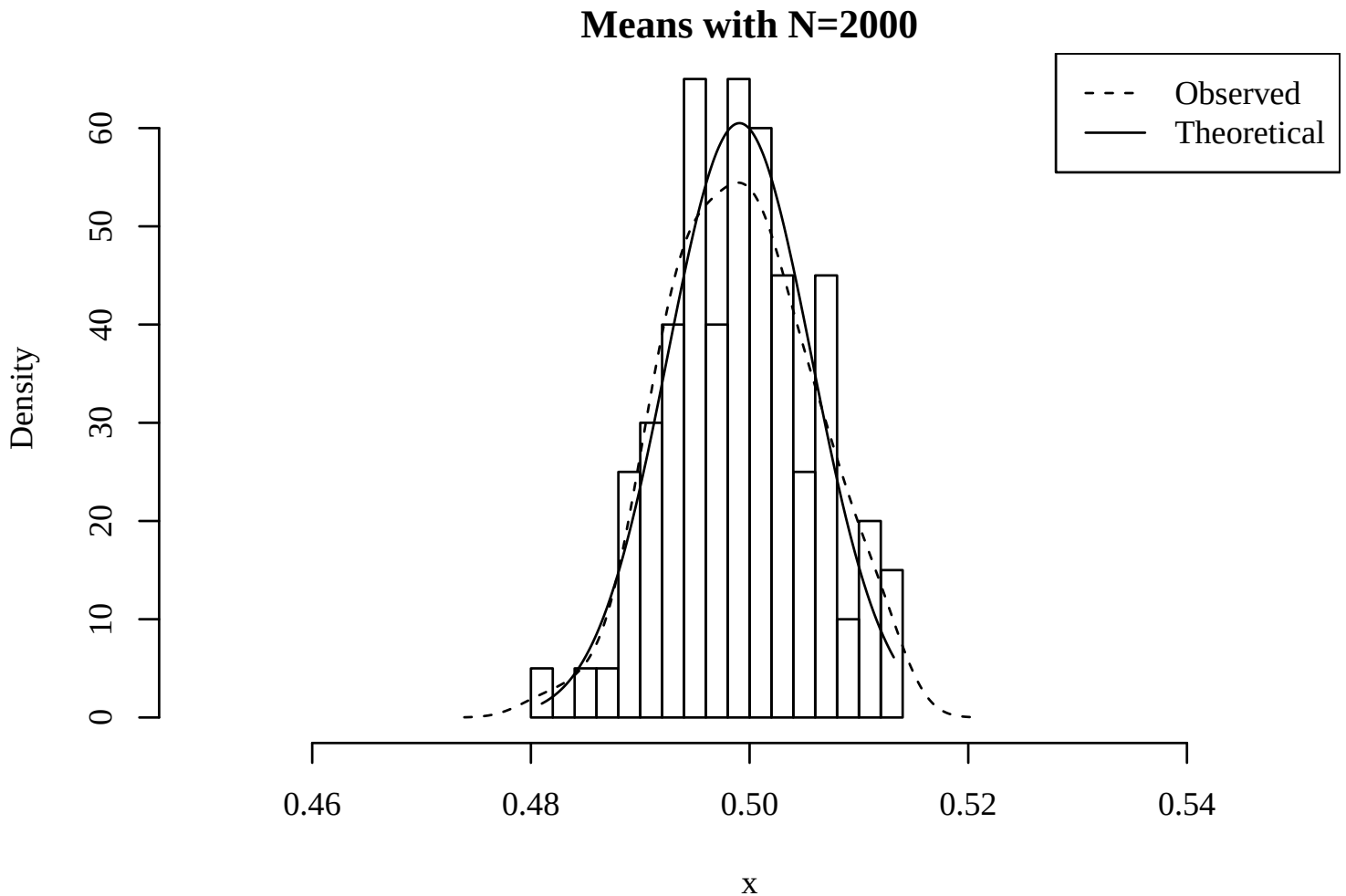
Means from 1000 Uniform samples, Sample Size=30



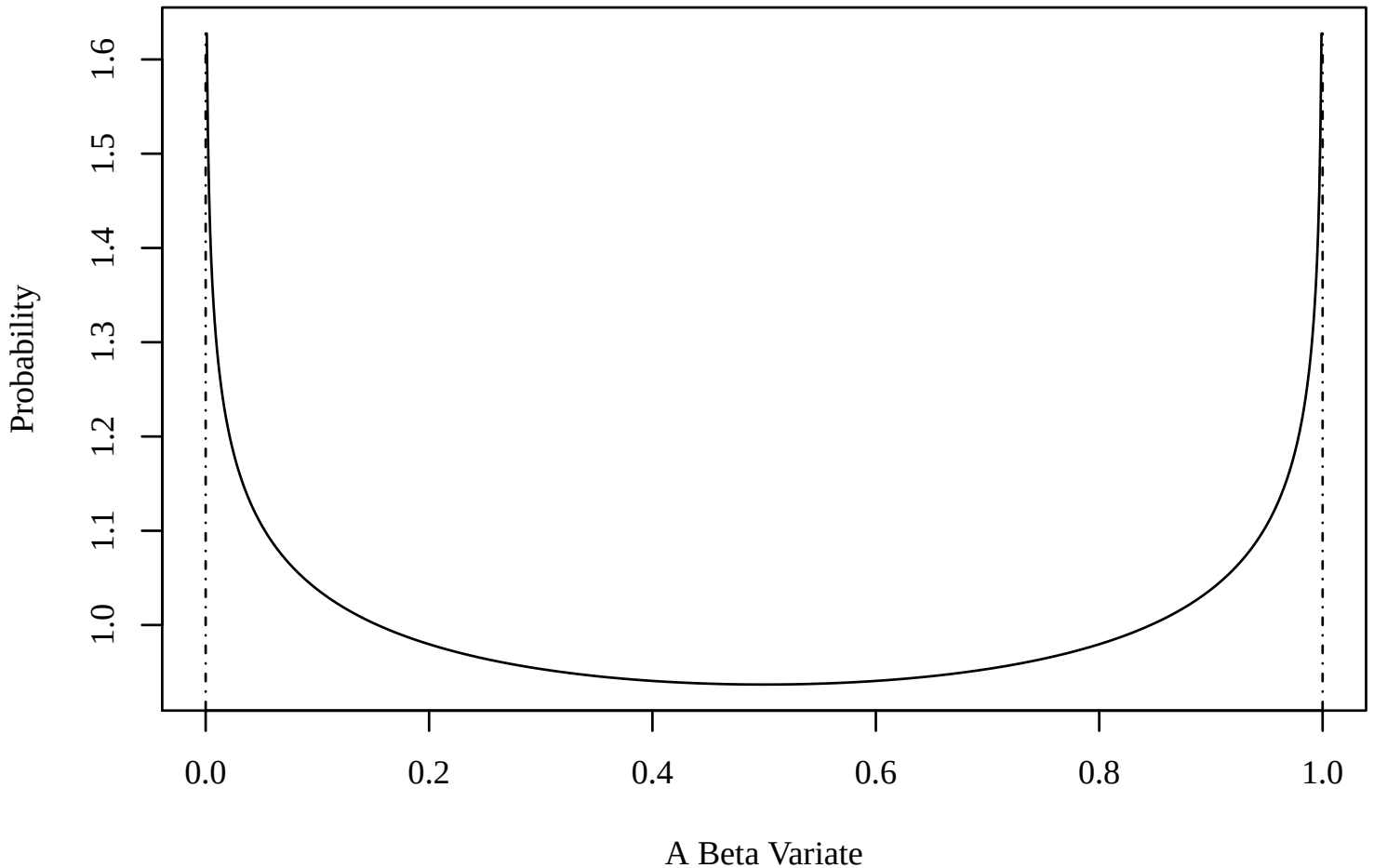
Means from 1000 Uniform samples, Sample Size=500



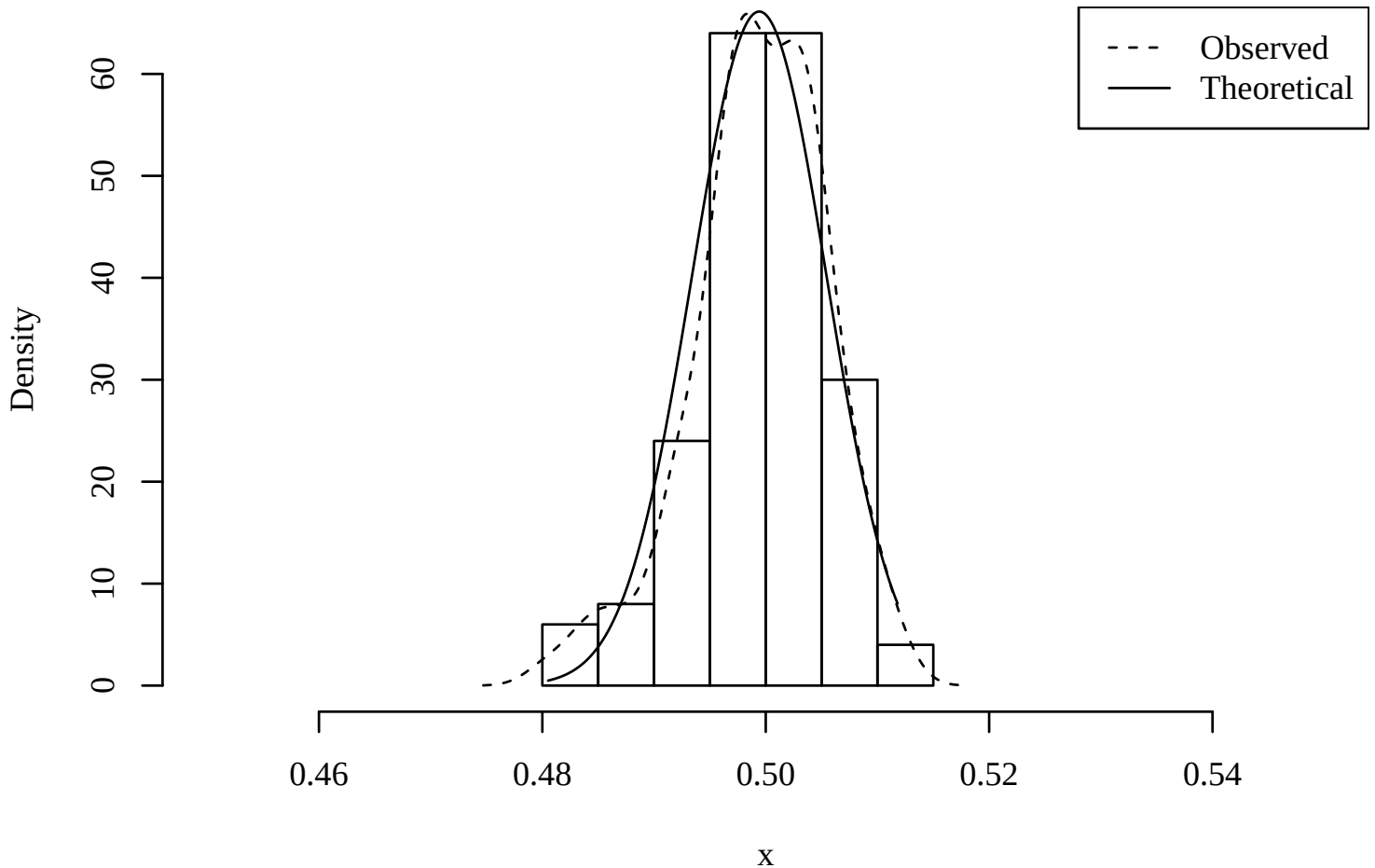
Means from 1000 Uniform samples, Sample Size=2000



Example 3: Beta(0.9,0.9)



Means from 1000 Beta Samples, Sample Size=2000



More Ambitious Experiments are Possible

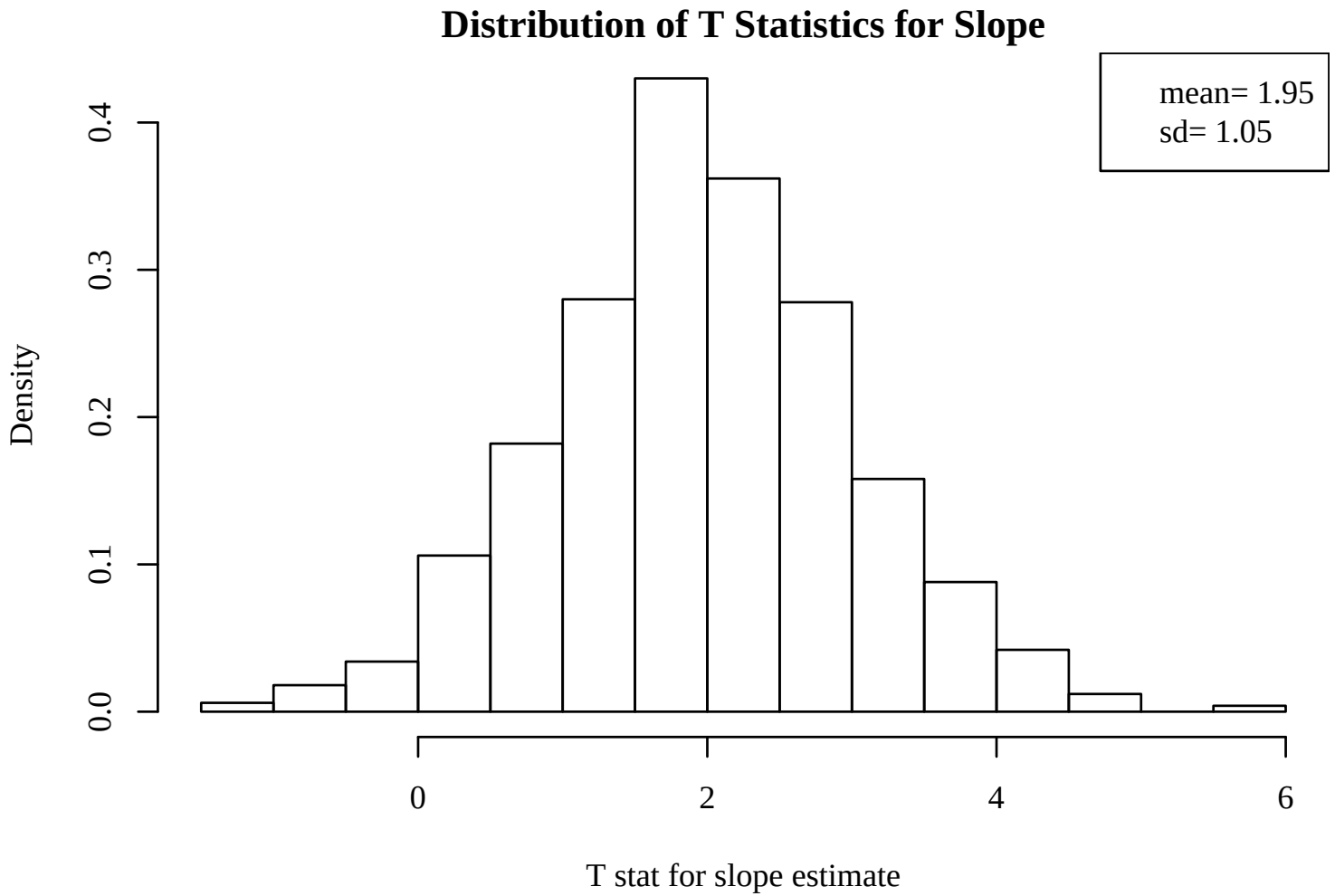
- The focus on "means" is just for simplicity and familiarity
- We can create "true sampling distributions" for any kind of estimator
- More detail in my lectures on functions and programming with R

replicate

- R makes it particularly easy to conduct experiments with the "replicate" function

```
grabTofB <- function(){
    x <- rpois(1000, lambda=17)
    y <- 2 + .223 * x + 15 * rnorm(1000)
    seB <- coef(summary(lm(y~x)))[2,3]
}
rep1000<- replicate(1000,grabTofB())
m1000 <- round(mean(rep1000),2)
sd1000 <- round(sd(rep1000),2)
h1000 <- hist(rep1000, prob= TRUE, main="Distribution
     of T Statistics for Slope", xlab="T stat for
    slope estimate")
legend("topright",legend=c(paste("mean=",m1000),paste
    ("sd=",sd1000)))
```

The Sampling Distribution From Regression Estimates



Detour: Random Generators

- In my advanced class on R programming, I noticed a little blip. I'd like to ask you what you think.
- Here is my example that generates 3 sets of 3 numbers.

```
set.seed(1234)
x1 <- rnorm(3)
x2 <- rpois(3, lambda = 7)
x3 <- rnorm(3)
X <- cbind(x1, x2, x3)
X
```

```
           x1 x2         x3
[1,] -1.2070657  2 0.03572991
[2,]  0.2774292  5 0.11297506
[3,]  1.0844412  8 1.42855203
```

Lets vote: Yes or No

- Do you believe that the result for column x3 will be the same if we draw 3 gamma variables in the middle stanza?

```
set.seed(1234)
x1 <- rnorm(3)
x2 <- rgamma(3, 1, 1)
x3 <- rnorm(3)
Y <- cbind(x1, x2, x3)
```

Check the Result

- Recall the original set was

```
              x1 x2          x3
[1,] -1.2070657  2 0.03572991
[2,]  0.2774292  5 0.11297506
[3,]  1.0844412  8 1.42855203
```

- And the new results are:

```
set.seed(1234)
x1 <- rnorm(3)
x2 <- rgamma(3, 1, 1)
x3 <- rnorm(3)
Y <- cbind(x1, x2, x3)
```

- You can guess why this might be a very serious problem. Right? Slight changes in the middle of a program have a destructive effect for all following calculations.

How To Understand: Generator versus Distribution

Generator Actually, *Pseudo Random Number Generator*. It creates a very long stream of (unpredictable) integers. (The R default is MT19937, the Mersenne-Twister. There are others, the most famous at the current time is Pierre L'Ecuyer's MRG32k3a.)

- R allows one "Random Generator" to exist at a time.
- Any R function that needs random numbers will draw from that one stream. Thus, rnorm, rpois, rgamma, etc, draw on the one random generator.
- The creation of a number that approximates a "normal" or "gamma" may require several draws from the generator (accept-reject sampling algorithm, for example).
- The key point is that, unless we are very careful, apparently "innocent" changes in a program can cause havoc.

MVN Havoc (and how to avoid it)

- Multivariate Normal distribution common in simulations.
- Consider mvrnorm (from the MASS) package.

```
set.seed(12345)
X0 <- MASS::mvrnorm(n=10, mu = c(0,0,0), Sigma =
    diag(3))
## create a smaller data set, starting at same
    position
set.seed(12345)
X1 <- MASS::mvrnorm(n=5, mu = c(0,0,0), Sigma =
    diag(3))
## Create a larger data set
set.seed(12345)
X2 <- MASS::mvrnorm(n=15, mu = c(0,0,0), Sigma =
    diag(3))
```

- My *guess* was that the first 5 rows of sets X0, X1, and X2 should all be the same.

Disappointing result. Check 5 rows

- The top 5 rows

```
X0[1:5, ]
```

```
           [,1]       [,2]      [,3]
[1,]  0.7796219 -0.1162478 0.5855288
[2,]  1.4557851  1.8173120 0.7094660
[3,] -0.6443284  0.3706279 -0.1093033
[4,] -1.5531374  0.5202165 -0.4534972
[5,] -1.5977095 -0.7505320 0.6058875
```

```
X1[1:5, ]
```

```
           [,1]       [,2]      [,3]
[1,] -0.1162478 -1.8179560 0.5855288
[2,]  1.8173120  0.6300986 0.7094660
[3,]  0.3706279 -0.2761841 -0.1093033
[4,]  0.5202165 -0.2841597 -0.4534972
[5,] -0.7505320 -0.9193220 0.6058875
```

Disappointing result. Check 5 rows ...

```
X2[1:5, ]
```

```
           [,1]       [,2]       [,3]
[1,] 0.8118732  0.8168998  0.5855288
[2,] 2.1968335 -0.8863575  0.7094660
[3,] 2.0491903 -0.3315776 -0.1093033
[4,] 1.6324456  1.1207127 -0.4534972
[5,] 0.2542712  0.2987237  0.6058875
```

The Cautionary Tale

- Simulations based on MASS mvrnorm must be very cautious.
- Expanding the sample size from 100 to 200 causes 2 very bad things to happen.
  1. The first 100 rows are not replicated exactly
  2. But they are repeated partially.

Disappointing result. Check 5 rows

- The top 5 rows

```
X0[1:5, ]
```

```
           [,1]       [,2]      [,3]
[1,]  0.7796219 -0.1162478 0.5855288
[2,]  1.4557851  1.8173120 0.7094660
[3,] -0.6443284  0.3706279 -0.1093033
[4,] -1.5531374  0.5202165 -0.4534972
[5,] -1.5977095 -0.7505320 0.6058875
```

```
X1[1:5, ]
```

```
           [,1]       [,2]      [,3]
[1,] -0.1162478 -1.8179560 0.5855288
[2,]  1.8173120  0.6300986 0.7094660
[3,]  0.3706279 -0.2761841 -0.1093033
[4,]  0.5202165 -0.2841597 -0.4534972
[5,] -0.7505320 -0.9193220 0.6058875
```

Disappointing result. Check 5 rows ...

```
X2[1:5, ]
```

```
           [,1]       [,2]       [,3]
[1,] 0.8118732  0.8168998  0.5855288
[2,] 2.1968335 -0.8863575  0.7094660
[3,] 2.0491903 -0.3315776 -0.1093033
[4,] 1.6324456  1.1207127 -0.4534972
[5,] 0.2542712  0.2987237  0.6058875
```

The Cautionary Tale

- Simulations based on MASS mvrnorm must be very cautious.
- Expanding the sample size from 100 to 200 causes 2 very bad things to happen.
  1. The first 100 rows are not replicated exactly
  2. But they are repeated partially.

Package mvtnorm has been fixed

- Correcting the "first N" observations problem requires a relative small fix, I could show you if you really want to know.
- Because MASS is distributed with R, for the sake of historical continuity, its authors decided not to change it..
- However, the authors of mvtnorm responded favorably, as you see now the evidence:

Package mvtnorm has been fixed ...

```
library(mvtnorm)
set.seed(12345)
X0 <- mvtnorm::rmvnorm(n=10, mean = c(0,0,0), sigma
    = diag(3))
## create a smaller data set, starting at same
    position
set.seed(12345)
X1 <- mvtnorm::rmvnorm(n=5,  mean = c(0,0,0), sigma
    = diag(3))
X0
```

```
            [,1]       [,2]       [,3]
[1,]  0.5855288  0.7094660 -0.1093033
[2,] -0.4534972  0.6058875 -1.8179560
[3,]  0.6300986 -0.2761841 -0.2841597
[4,] -0.9193220 -0.1162478  1.8173120
[5,]  0.3706279  0.5202165 -0.7505320
[6,]  0.8168998 -0.8863575 -0.3315776
[7,]  1.1207127  0.2987237  0.7796219
[8,]  1.4557851 -0.6443284 -1.5531374
[9,] -1.5977095  1.8050975 -0.4816474
```

Package mvtnorm has been fixed ...

```
[10,]  0.6203798  0.6121235 -0.1623110
```

```
     X1
```

```
             [,1]       [,2]       [,3]
[1,]  0.5855288  0.7094660 -0.1093033
[2,] -0.4534972  0.6058875 -1.8179560
[3,]  0.6300986 -0.2761841 -0.2841597
[4,] -0.9193220 -0.1162478  1.8173120
[5,]  0.3706279  0.5202165 -0.7505320
```

- In case you want the old, bad way from mvtnorm, they added an argument "pre0.9_9994" so you can restore the bad old days.