A Demo or Two

March 6, 2017

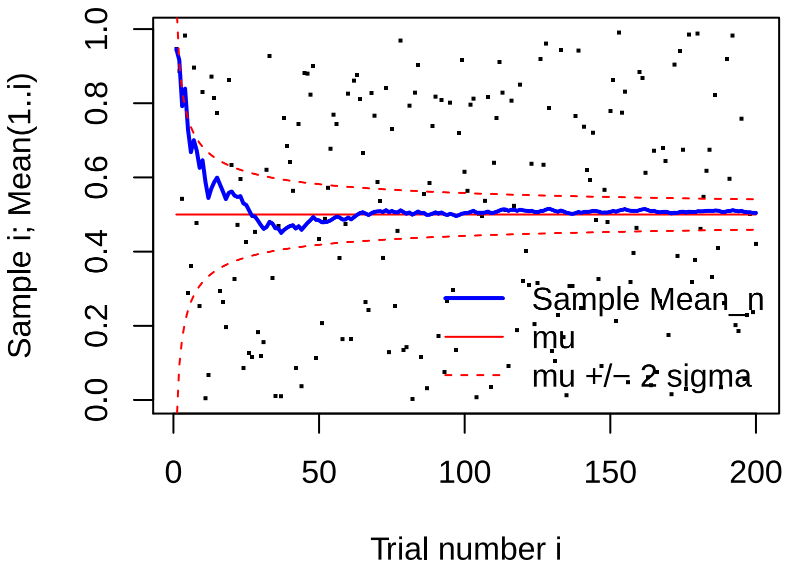
Here are a few simple demonstrations illustrating important concepts from the course. Most use R; see R Quick Start for a quick introduction to R. You should be able to run these demos below by copying the R code shown below and pasting it into an R console window (or putting it in a file and entering “source(’filename’)”).

Laws Of Large Numbers: The Weak and Strong Laws of Large Numbers are important theoretical results, essentially guaranteeing that the average of a large number if independent samples from arbitrary distributions will converge to the expected value of such a variable. As a simple illustration of this, the following looks at averages of i.i.d. Uniform(0,1) random variables:

```
#
# "Regression Towards the Mean" -- by the law of large numbers, the mean of an
#  increasingly large sample of, e.g., uniform RVs, should converge to the mean.
#
# Plot a sample of i.i.d. uniform RVs & successive sample means thereof.
#
# Parameters:
#   n: # samples,
#   ksigma: if > 0, also plot +/- ksigma*sigma envelope around the mean
#   cex: controls point size,
#   others: plot colors
#
# Usage:
#   Copy/paste this into the console window of an R session, or enter
#   "source('filename.R')" to define the function, then enter "rtm()" to run it.
#
rtm <- function(n=200, ksigma=2, cex=2, cmu='red', cavg='blue', csig='red'){
  v <- runif(n)          # n samples from uniform
  mu <- 0.5              # mean &
  sigma <- 1/sqrt(12)    #   variance of each sample
  # plot the samples:
  plot(v,pch='.',xlab='Trial number i',ylab='Sample i; Mean(1..i)',cex=cex)
  # plot a horizontal line at mean:
  lines(c(1,n),c(mu,mu),col=cmu,lwd=1)
  # plot n successive sample means:
  points(1:n,cumsum(v)/(1:n),type='l',col=cavg,lwd=2)
  if(ksigma>0){
    # plot k-sigma envelope around mean:
    points(1:n, mu+ksigma*sigma/sqrt(1:n), type='l',lwd=1,col=csig,lty='dashed')
    points(1:n, mu-ksigma*sigma/sqrt(1:n), type='l',lwd=1,col=csig,lty='dashed')
    # add plot legend:
    legend('bottomright',
        legend=c("Sample Mean_n",    "mu", paste("mu +/-",ksigma,"sigma")),
           col=c(           cavg,     cmu,                              csig),
           lwd=c(              2,       1,                                 1),
           lty=c(        'solid', 'solid',                          'dashed'),
        bty='n')
    }
```

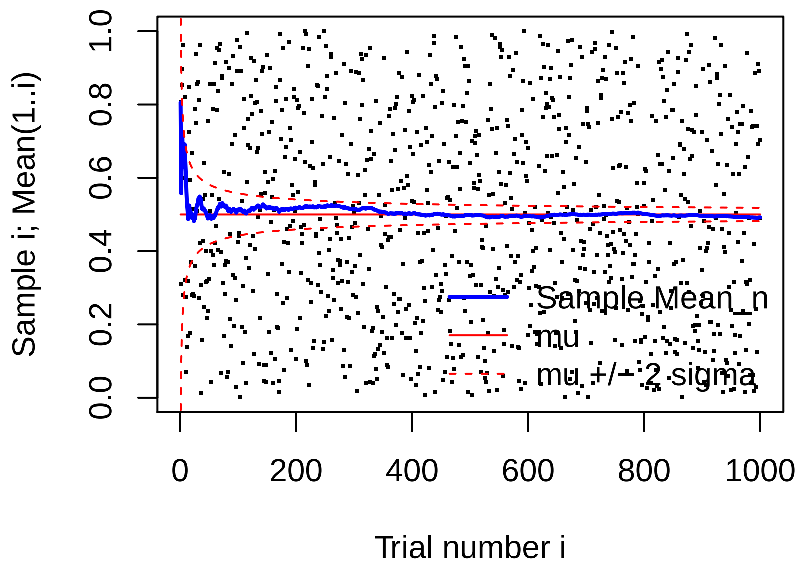
```
}
```

```
rtm() # Call it once, with default parameters.
```



Another plot, with larger n:

```
rtm(n=1000)
```



Exercise: Do something similar for a different distribution (normal, exponential, Poisson,...) in place of uniform.

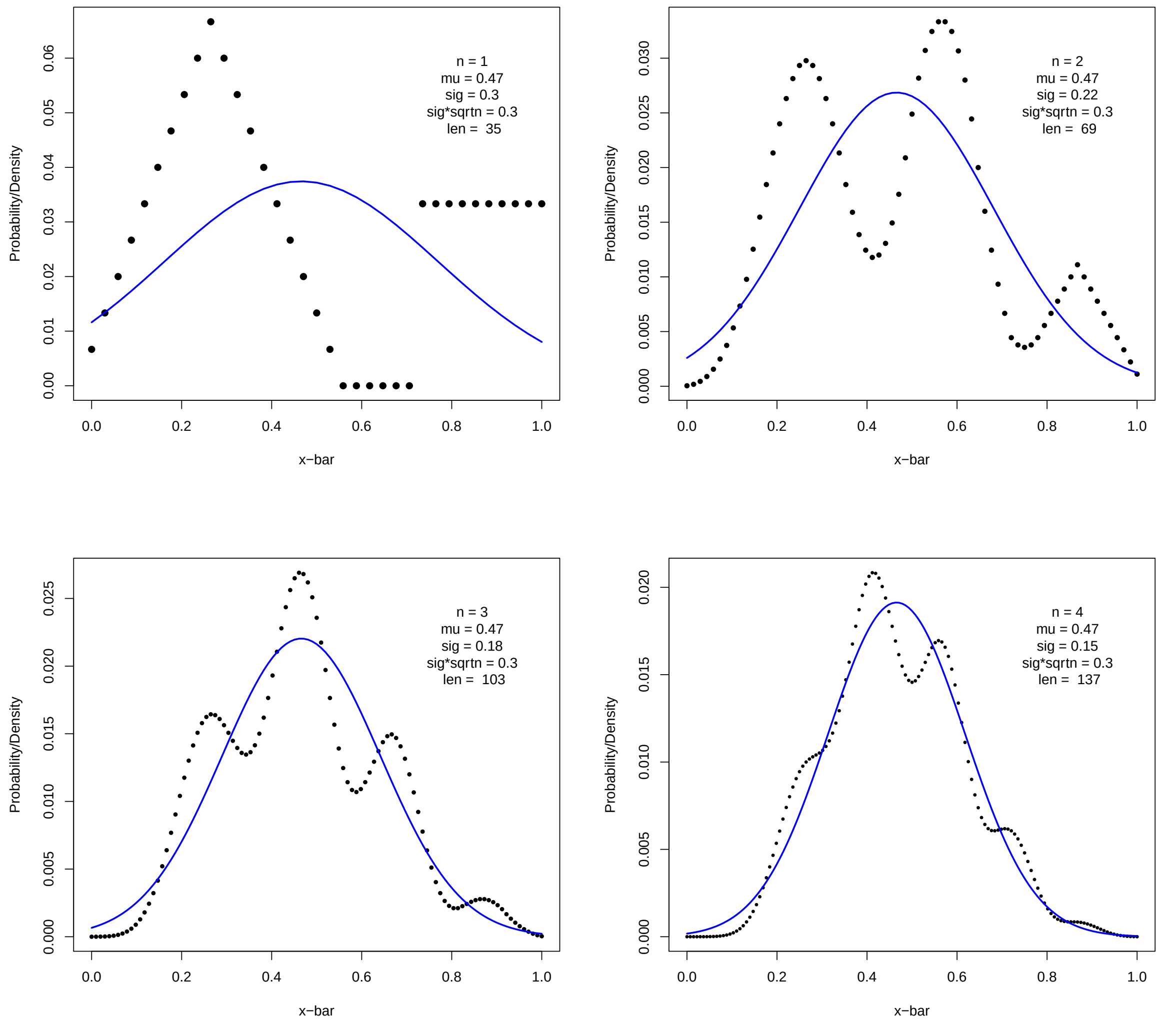
The Central Limit Theorem: Another very important result is the Central Limit Theorem: Not only does the *value* of the average of a large sample of independent random variables from *arbitrary* distributions converge to its expected limit (above), but *the shape of the distribution* of those averages also converges to a well-defined limit—namely, it is approximately normally distributed.

```
# Convergence of any wacky distrib to normal as in CLT
#
# Method: n-fold convolution of initial distribution with itself
#
#   For the n-th convolution, we need both the (n-1)-st and the original
#   distributions, so for convenience these are bundled into a list and
#   returned, making a by-hand iteration simple:
#     bundle1 <- clt(wacky=//*put your wacky distribution here*//)
#     bundle2 <- clt(bundle1) # 2-fold convolution
#     bundle3 <- clt(bundle2) # 3-fold convolution
#     ...
#
# Parameters:
#    bundle  : initially NULL; subsequently, result of previous call
#    plot    = T to see plot
#    verbose = T to annotate plot with mu, sigma, etc.
#    bell    = T to overlay bell curve
#    wacky   = vector of numbers representing relative probabilities of
#              outcomes 1:length(wacky); irrelevant unless bundle == NULL
#    cex     = scale factor for point size
#
clt <- function(bundle=NULL, plot=T, verbose=T, bell=T,
                wacky=c(1:10,9:0,rep(0,5),rep(5,10)), cex=NULL){
  if(is.null(bundle)){
    mywack <- wacky/sum(wacky)                      # normalize
    bundle <- list(n=1,result=mywack,start=mywack) # bundle params/result
```

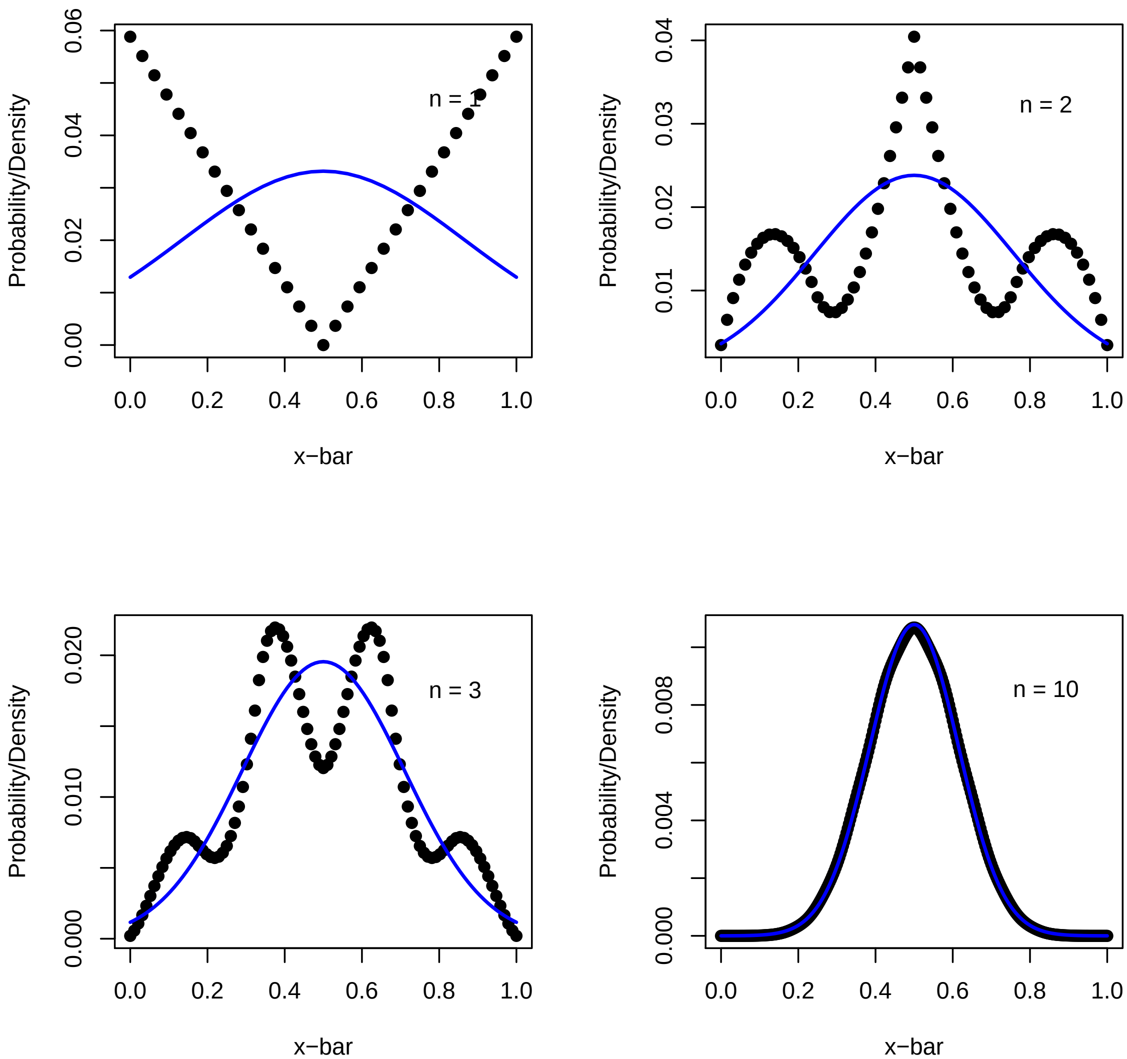
```
  }
  if(plot){
    len <- length(bundle$result)
    x <- (0:(len-1))/(len-1)
    y <- bundle$result
    plot(x,y,xlab='x-bar',ylab='Probability/Density',cex=cex,pch=19)
    mu   <- sum(x*y)
    sig2 <- sum((x-mu)^2*y)
    sig  <- sqrt(sig2)
    chatter <- ifelse(!verbose,'', paste(
      '\nmu =', round(mu,2),
      '\nsig =',round(sig,2),
      '\nsig*sqrtn =', round(sig*sqrt(bundle$n),2),
      '\nlen = ', length(x)));
    text(.85,.8*max(bundle$result), paste('n =',bundle$n,chatter))
    if(bell){points(x,dnorm(x,mu,sig)/length(x),type='l',lwd=2,col='blue')}
  }
  return(
    list(
      n      = bundle$n+1,
      result = convolve(bundle$result,rev(bundle$start),type='o'),
      start  = bundle$start))
}
```

```
# Make a "movie" of above; if "file" is NULL, display to screen, else write a
# multi-page .pdf file. The "..." formal and actual parameters have a special
# meaning in R: accept extra named arguments to this function and pass them to
# inner calls.
clt.movie <- function(filename ="central.limit.thm.movie.pdf", frames=50, ...){
  opar<-par(no.readonly=T);on.exit(par(opar))
  if(!is.null(filename)){
    # noninteractive version: open .pdf graphics "device"
    pdf(filename,onefile=T,width=9,height=7)
  } else {
    # interactive version: pause after each plot & ask to continue
    devAskNewPage(TRUE)
  }
  bundle <- clt(...)
  for(i in 2:frames){
    # tweak cex to make dots smaller when there are more of them
    bundle <- clt(bundle, cex=(1-i/frames)*.6+.4, ...)
  }
  if(!is.null(filename)){dev.off()}  # close .pdf
}
```

```
clt.movie(NULL,4)
```




```r
# Another CLT example, just showing 4 of 10 frames
# Default is a vee-shaped distribution
clt.vee <- function(dist=abs(-16:16), ...){
  opar <- par(mfrow=c(2,2),no.readonly=T) # graph params: 4 plots in 2x2 grid
  on.exit(par(opar))
  bundle <- clt(wacky=dist, plot=TRUE, ...)
  for(i in 2:10) {
    # show plots only for 1-, 2-, 3-, and 10-fold convolution
    bundle <- clt(bundle, plot = (i %in% c(2,3,10)), ...)
  }
}
clt.vee(verbose=F)
```



Exercises: The above should work for any disctrete distribution defined on a finite number of points. Try it on some other ones. Try to find ones that make the convergence to the normal as slow as possible, say.