More on probability theory and Monte Carlo Simulation

jsn-jin @ UCLA

1/24/2020

Contents

Plot Two Density Curves On The Same Graph

Suppose that $X \sim N(0, 4)$ and $Y \sim N(1, 1)$. Note the use of `dnorm` function in the following code chunk.

```
mu.x <- 0
sigma.x <- 2

mu.y <- 1
sigma.y <- 1

vals = seq(from = -5, to = 5, length = 150)
df <- data.frame(vals,
                 dnorm(x = vals, mean = mu.x, sd = sigma.x),
                 dnorm(x = vals, mean = mu.y, sd = sigma.y))

colnames(df)[1] <- "quantile"
colnames(df)[2] <- "X" # rename the second column
colnames(df)[3] <- "Y"

head(df)
```

```
##    quantile           X            Y
## 1 -5.000000 0.008764150 6.075883e-09
## 2 -4.932886 0.009525755 9.068072e-09
## 3 -4.865772 0.010341890 1.347300e-08
## 4 -4.798658 0.011215313 1.992772e-08
## 5 -4.731544 0.012148813 2.934233e-08
## 6 -4.664430 0.013145200 4.301059e-08
```

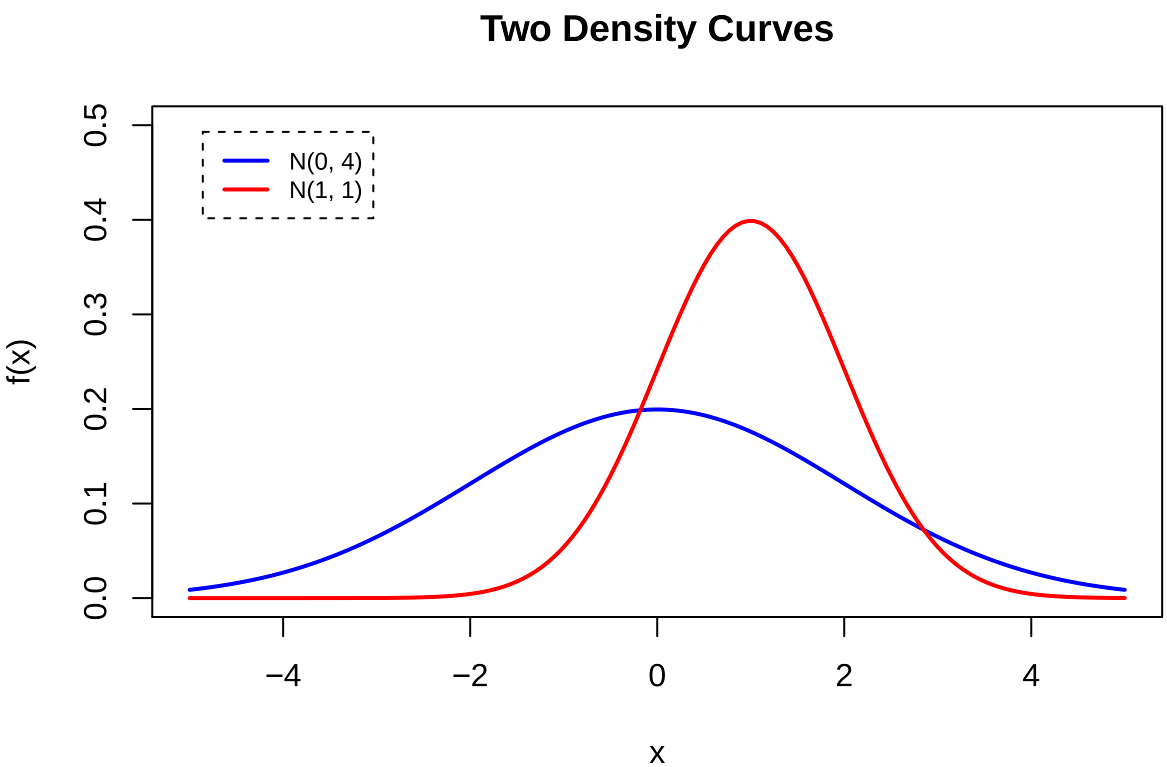
We can generate the plot using the `plot` function:

```
plot(df$quantile,
     df$X,
     type = "l",
     col = "blue",
     lwd = 2,
     ylim = c(0,0.5),
```

```
      xlab = "x",
      ylab = "f(x)",
      main = "Two Density Curves")

lines(df$quantile,
       df$Y,
       type = "l",
       col = "red",
       lwd = 2)

legend(x = "topleft",
        legend = c("N(0, 4)", "N(1, 1)"),
        col = c("blue","red"),
        lwd = 2,
        box.lty = 2,
        cex = 0.75,
        inset = 0.05)
```



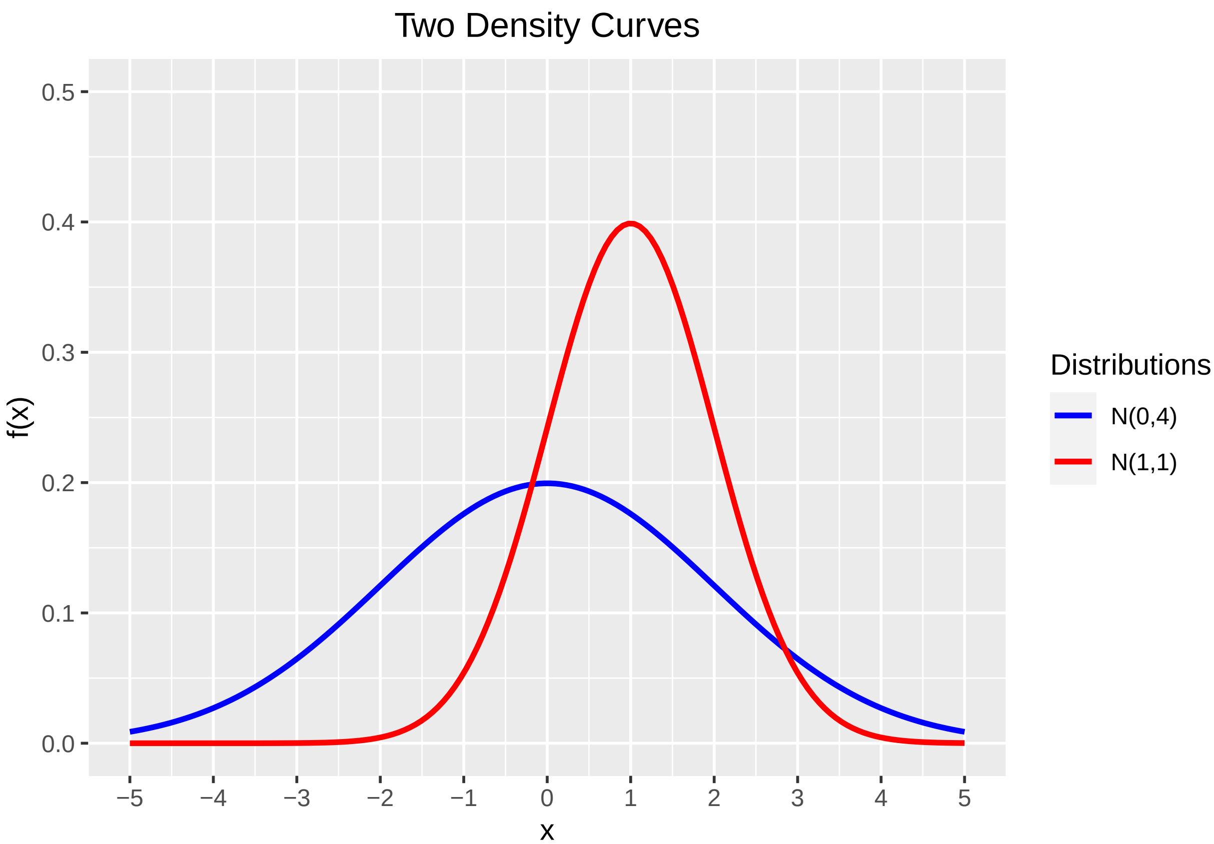
Or, we can apply the `ggplot` function:

```
library(ggplot2)
g <- ggplot(data = df) +
  geom_line(mapping = aes(x = quantile, y = X, color = "N(0,4)"), size = 1) +
  geom_line(mapping = aes(x = quantile, y = Y, color = "N(1,1)"), size = 1) +
  scale_color_manual(values = c('N(0,4)' = 'blue', 'N(1,1)' = 'red')) +
  labs(color = 'Distributions') +
  ylab("f(x)") +
  xlab("x") +
  xlim(-5, 5) +
  ylim(0, 0.5) +
```

```r
  ggtitle("Two Density Curves") +
  scale_x_continuous(breaks = seq(-5, 5, by = 1)) +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

```r
print(g)
```



Shade Areas Under The Density Curve

Suppose we are interested in $Pr(1 \leq X < 2)$ and $Pr(X \leq -2)$

```r
g <- g + geom_area(data = subset(df, quantile >= 1 & quantile < 2),
                   aes(x = quantile, y = X), fill = "green", alpha = 0.5)

g <- g + geom_area(data = subset(df, quantile < -2),
                   aes(x = quantile, y = X),  fill = "darkgrey", alpha = 0.5)
print(g)
```

Computing Probabilities (integrals) - A Numerical Way

Please refer to this handout for more information.

Univariate Case

To compute $Pr(1 \leq X < 2)$, we can directly apply the `pnorm` function:

```
prob_univ = pnorm(2, mean = mu.x, sd = sigma.x) - pnorm(1, mean = mu.x, sd = sigma.x)
prob_univ
```

```
## [1] 0.1498823
```

Or, we can solve for $Pr(1 \leq X < 2)$ numerically, where $X \sim N(0, 4)$.

```
# define the integrand
f <- function(x) {1/sqrt(2*pi*4)*exp(-x^2/(2*4))}
integrate(f, lower = 1, upper = 2)
```

```
## 0.1498823 with absolute error < 1.7e-15
```

Likewise, we can obtain $P(X \leq -2)$ in two ways:

```
pnorm(-2, mean = mu.x, sd = sigma.x)
```

```
## [1] 0.1586553
```

```
integrate(f, lower = Inf, upper = -2) # based on previously defined integrand f
```

```
## 0.1586553 with absolute error < 9.6e-08
```

Multivariate Case

Consider the random vector

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 4 \end{pmatrix} \right).$$

We are interested in $P(-1 \leq X_1 \leq 1, -2 \leq X_2 \leq 3)$.

Method 1:

```
library(mvtnorm)
cov <- matrix(c(1, 0.5, 0.5, 4), ncol =2, byrow = TRUE)
prob_mtv = pmvnorm(lower = c(-1, -2), upper = c(1, 3), mean = c(0, 0), sigma = cov)
prob_mtv
```

```
## [1] 0.5354662
## attr(,"error")
## [1] 1e-15
## attr(,"msg")
## [1] "Normal Completion"
```

Method 2:

```
library(cubature)
f <- function(x, mu = c(0, 0), var = matrix(c(1, 0.5, 0.5, 4), ncol = 2, byrow = TRUE)){
  k = length(x)
  det_cov = det(var)
  inv_cov = solve(var)
  # the density function of a bivariate normal distribution
  pdf = (2 * pi)^(-k / 2) * det_cov^(-1 / 2) * exp(-1 / 2 * t(x - mu) %*% inv_cov %*% (x - mu))
  return(pdf)
}
adaptIntegrate(f, lowerLimit = c(-1, -2), upperLimit = c(1, 3))
```

```
## $integral
## [1] 0.5354662
##
## $error
## [1] 4.876613e-06
##
## $functionEvaluations
## [1] 289
##
## $returnCode
## [1] 0
```

Comment: **If you do not have the cumulative distribution function of a random variable/vector, at least you can define the pdf and solve for the probability of interest numerically.**

Monte Carlo Simulation

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.

We will apply the monte carlo simulation to compute the probabilities of interest

Example: $Pr(1 \leq X < 2)$

First, generate 1000 random draws from the target distribution.

```r
set.seed(123) # for reporducible randomness
# runs <- 5000
runs <- 50000
mu.x <- 0
sigma.x <- 2

X <- rnorm(runs, mu.x, sigma.x)
head(X)
```

```
## [1] -1.1209513 -0.4603550  3.1174166  0.1410168  0.2585755  3.4301300
```

Now we compute the fraction of realizations falling between 1 and 2. This fraction is "approximately" the probability of interest when the number of runs is very large. (Weak law of large numbers)

```r
head(X >= 1 & X < 2)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
sum(X >= 1 & X < 2)/runs
```

```
## [1] 0.14968
```

```r
pnorm(2, 0 ,2) - pnorm(1, 0, 2)
```

```
## [1] 0.1498823
```

Note that the fraction is "close to" the previously computed 0.1498823. At least, this is easier to implement than integrating the pdf over $[1, 2)$.

Example: $Pr(-1 \leq X_1 < 1, -2 \leq X_2 < 3)$

Likewise, we can apply the monte carlo simulation to computing joint probabilities.

```r
# use the mvrnorm() function from the MASS package
library(MASS)
set.seed(2020)
runs <- 5000
X <- mvrnorm(runs, mu = c(0, 0), Sigma =  matrix(c(1, 0.5, 0.5, 4), ncol =2, byrow = TRUE ))
class(X)
```

```
## [1] "matrix"
```

```r
head(X)
```

```
##            [,1]       [,2]
## [1,]  1.2273667  0.5723406
## [2,] -0.5275094  0.7027548
## [3,] -1.6426291 -1.9806630
## [4,] -0.2681952 -2.2699777
## [5,] -0.8510619 -5.5853062
## [6,]  0.5369915  1.3875911
```

```r
res = sum(-1 <= X[, 1] & X[, 1] < 1 & -2 <= X[, 2] & X[, 2] < 3)/runs
res
```

```
## [1] 0.5356
```

The 0.5356 is "close to" 0.5354662.

Example: $Pr(-3 \leq 2X_1 + 3X_2 < 3)$

Now we consider a new random variable $Y \equiv 2X_1 + 3X_2$. What is the probability $Pr(-3 \leq Z < 3)$

```
Y <- 2 * X[, 1] + 3 * X[, 2]
res_Y <- sum( -3 <= Y & Y < 3 )/runs
res_Y
```

```
## [1] 0.3476
```

Note that Y is also normally distributed with mean 0 and variance 47. So we can verify the computed probability as follows.

```
pnorm(3, mean = 0, sd = sqrt(47)) - pnorm(-3, mean = 0, sd = sqrt(47))
```

```
## [1] 0.3383201
```

As we can see, the result from the monte carlo simulation is "close to" the true probability. However, **you do not need to derive the marginal distribution of the new random variable Y in the MC simulation**.