Chapter 8: Interactive Notebook for Students

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

Load functions to compute p-values

```
p_rtail = function(sampdist,tstat)
  {
  temp = density(sampdist)
  df = data.frame(temp$x, temp$y)
  formula1 = df$temp.x<tstat
  df1 = df[formula1,]
  plot(df, col = "red", type = "h")
  points(df1, col = "green", type = "h")
  pvalue = length(sampdist[sampdist>tstat])/(length(sampdist))
  return(pvalue)
  }
```

```
p_ltail = function(sampdist,tstat)
  {
  temp = density(sampdist)
  df = data.frame(temp$x, temp$y)
  formula1 = df$temp.x>tstat
  df1 = df[formula1,]
  plot(df, col = "red", type = "h")
  points(df1, col = "green", type = "h")
  pvalue = length(sampdist[sampdist<tstat])/(length(sampdist))
  return(pvalue)
  }

p_2tail = function(sampdist,tstat)
  {
hyp = mean(sampdist)
cutoff1 = hyp - abs(tstat-hyp)
  cutoff2 = hyp + abs(tstat-hyp)
  temp = density(sampdist)
  df = data.frame(temp$x, temp$y)
  formula1 = df$temp.x<cutoff1 | df$temp.x>cutoff2
  df1 = df[formula1,]
  plot(df, col = "green", type = "h")
  points(df1, col = "red", type = "h")
  pvalue = length(sampdist[sampdist<cutoff1 |  sampdist>cutoff2])/(length(sampdist))
  return(pvalue)
  }
```

In this lesson, we will conduct a number of statistical tests based on the principles we discussed in the previous lesson. We will also show the standard R functions to conduct some of these tests. You will notice that the basic procedure to test remains the same regardless of the specific test you perform. After you gain some experience with a few, conducting additional tests should become fairly straightforward.

Tests With Numeric Data

```
admission <- read.csv("../../data/admission.csv")
head(admission)
```

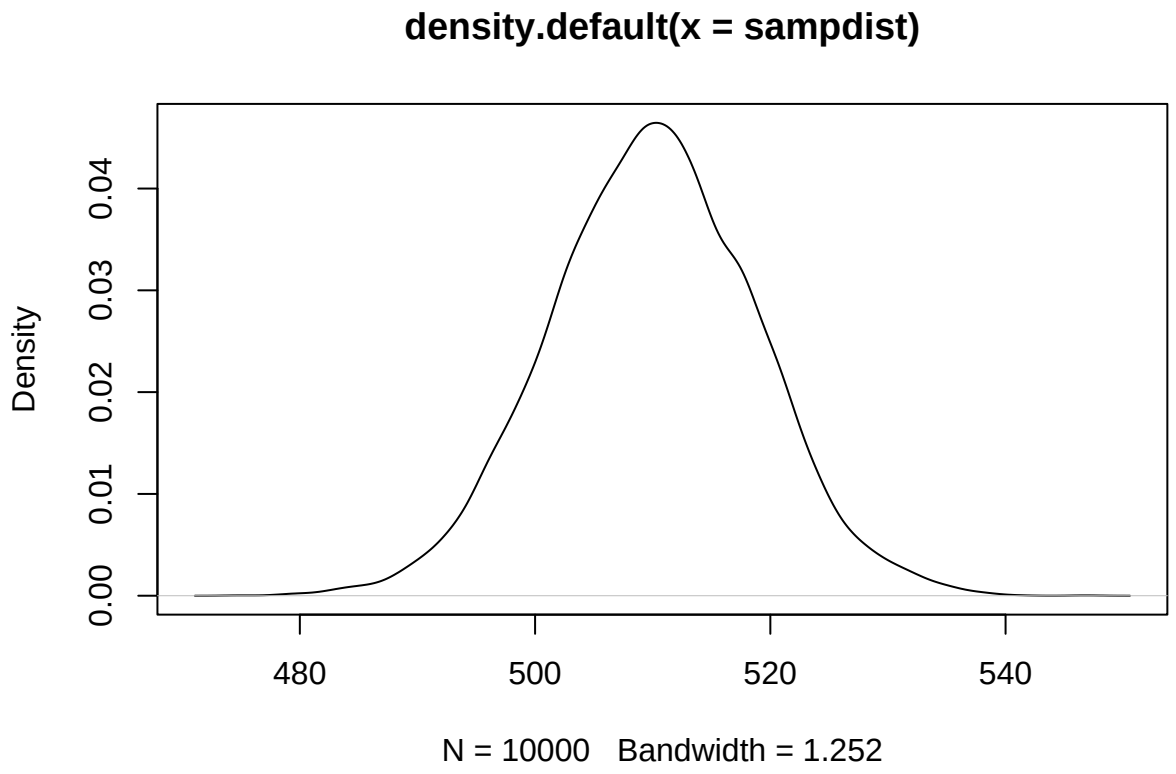
```
##    GPA GMAT    De
## 1 2.96  596 admit
## 2 3.14  473 admit
## 3 3.22  482 admit
## 4 3.29  527 admit
## 5 3.69  505 admit
## 6 3.46  693 admit
```

Mean

Step 1: State the hypothesis.
Population mean is 510.

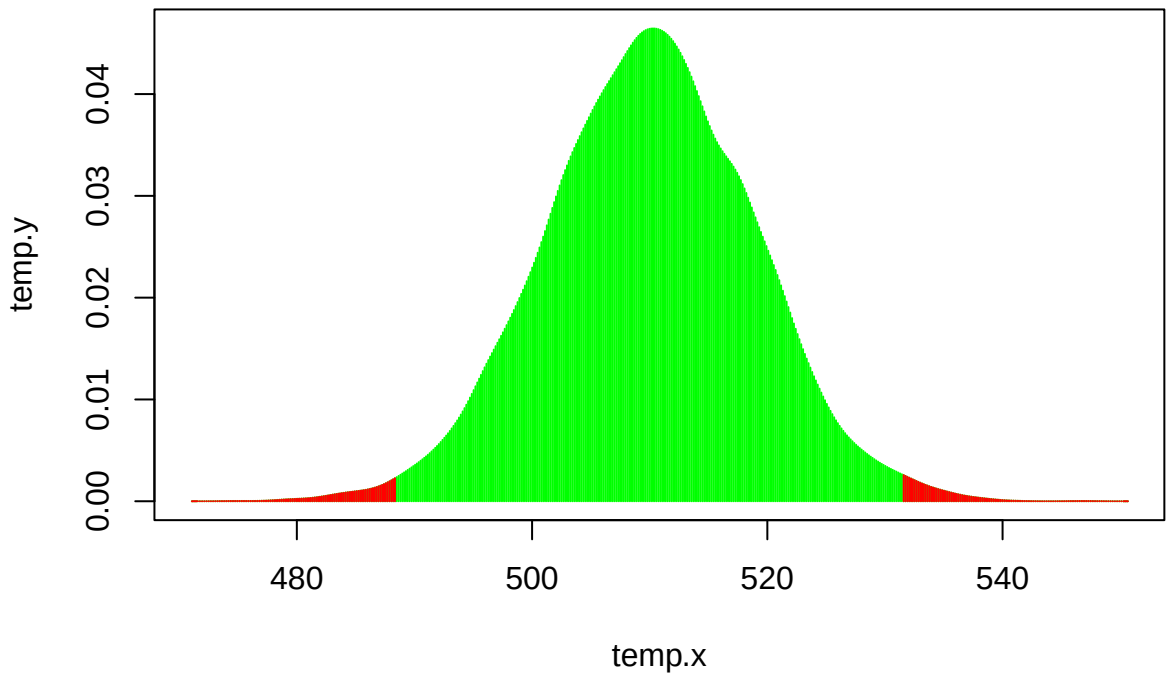
Step 2: Describe the data generation process and the population.
Since GMAT scores are numeric, we will assume it follows a normal distribution, $N(\mu, \sigma)$. Based on the hypothesis, μ is 510. What about σ? Since we do not have explicit information about σ, we will simply use the standard deviation from the sample.
Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  s1 = rnorm(sampsize, mean = 510, sd = sd(admission$GMAT))
  return(mean(s1))
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = mean(admission$GMAT)
p_2tail(sampdist,tstat)
```




```
## [1] 0.0161
```

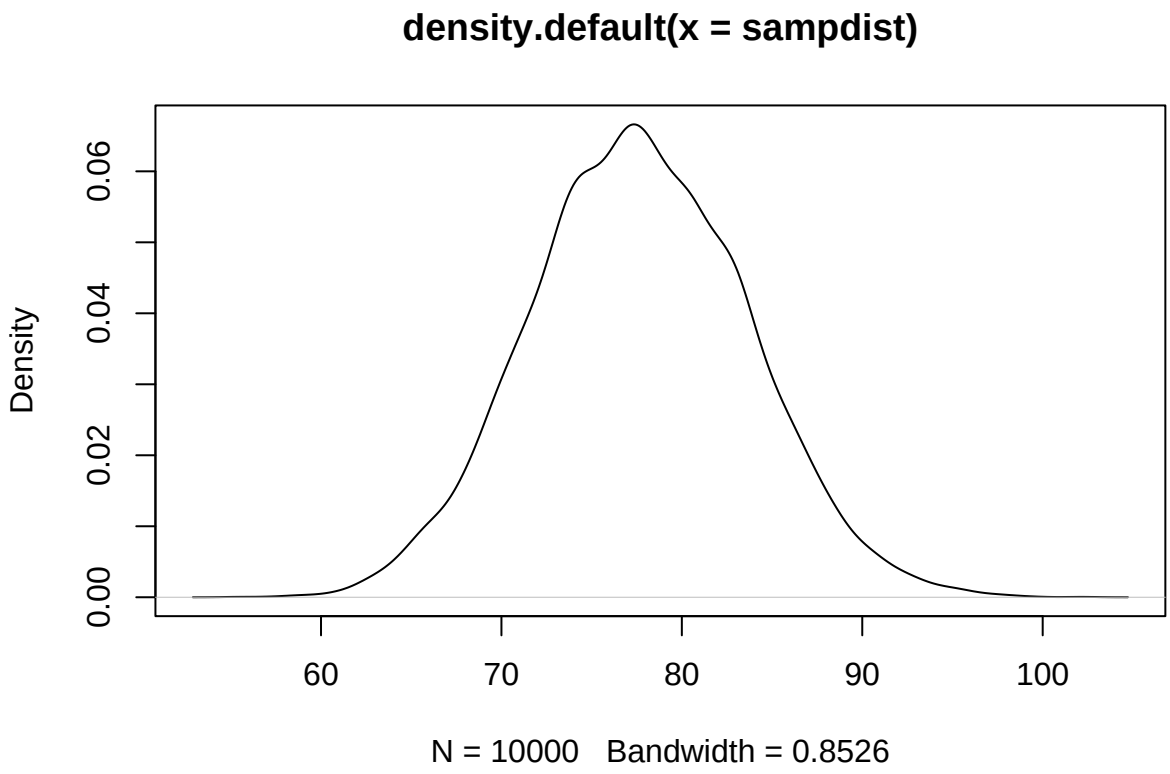
The p value is 0.016, and thus we can reject the null hypothesis that the mean population GMAT score is 510.

Standard Deviation

We want to test the hypothesis that the population standard deviation is 78.

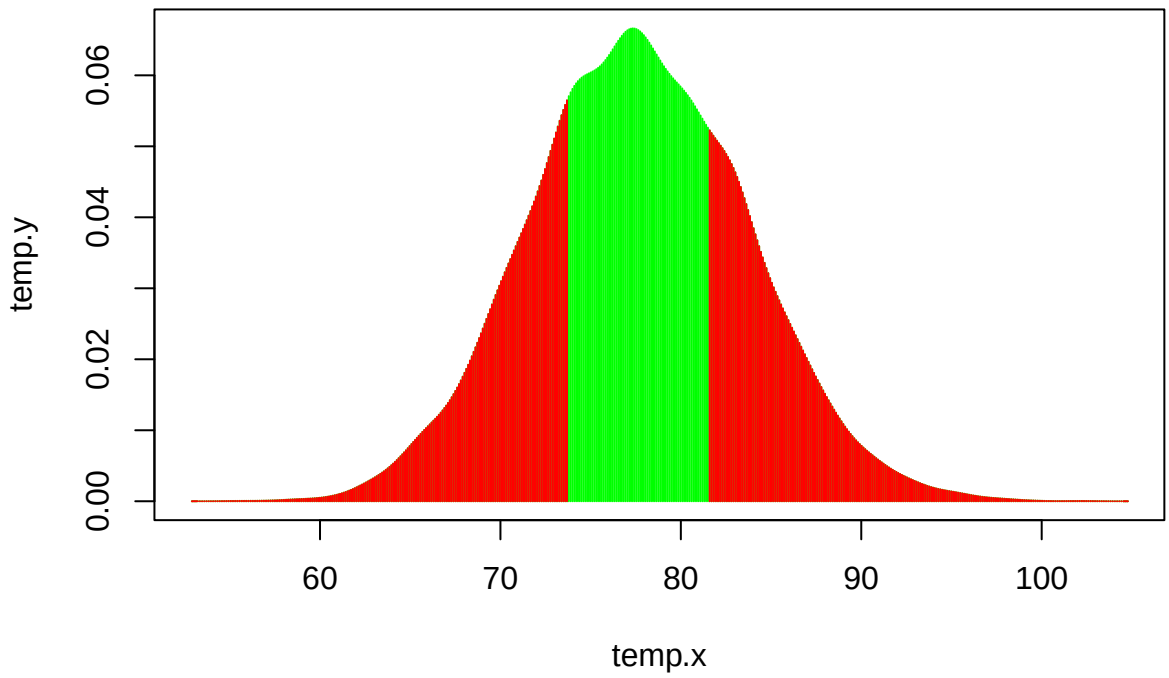
Step 1: State the hypothesis.
Population standard deviation is 78.
Step 2: Describe the data generation process and the population.
Since GMAT scores are numeric, we will assume it follows a normal distribution, $N(\mu, \sigma)$. Based on the hypothesis, σ is 78. μ will be taken from the sample.
Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  s1 = rnorm(sampsize, mean = mean(admission$GMAT), sd = 78)
  return(sd(s1))
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = sd(admission$GMAT)
p_2tail(sampdist,tstat)
```



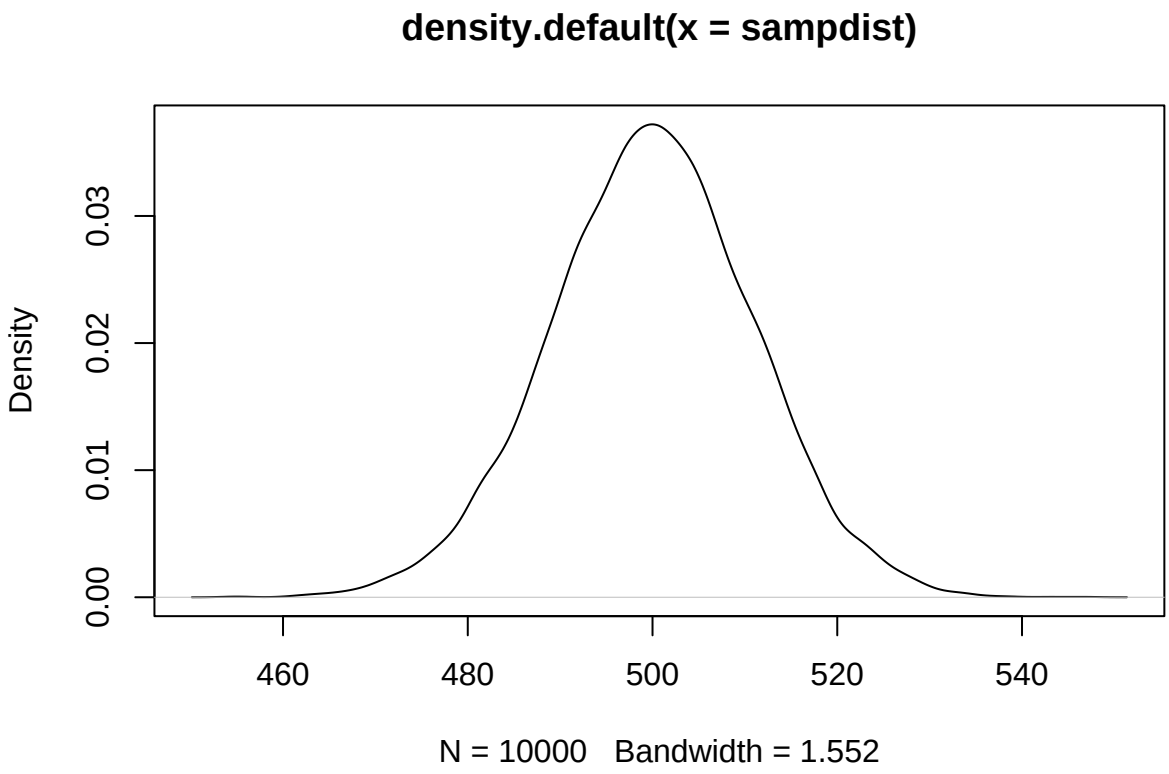

```
## [1] 0.5249
```

Median

We want to test the hypothesis that the population median is 500.

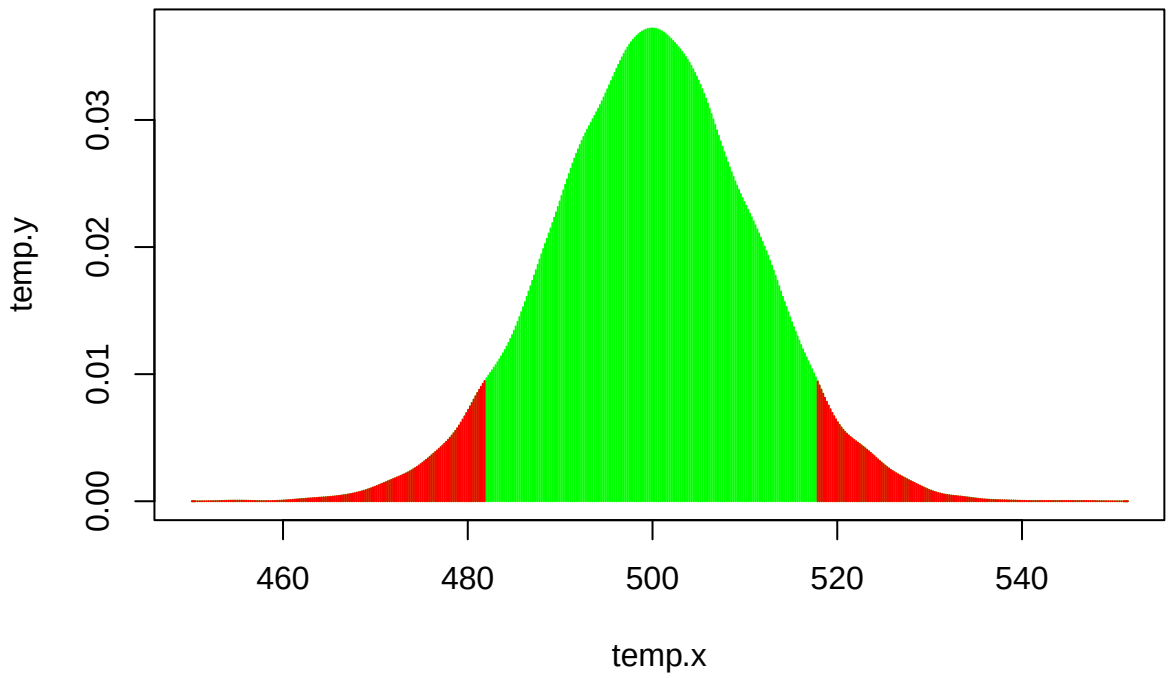
Step 1: State the hypothesis.
Population median is 500.
Step 2: Describe the data generation process and the population.
Since GMAT scores are numeric, we will assume it follows a normal distribution, $N(\mu, \sigma)$. Since a normal distribution is symmetric, the median value is the same as the mean value. So, $\mu = 500$, and σ will be taken from the sample.
Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  s1 = rnorm(sampsize, mean = 500, sd = sd(admission$GMAT))
  return(median(s1))
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = median(admission$GMAT)
p_2tail(sampdist,tstat)
```




```
## [1] 0.1042
```

Percentile

Consider the hypothesis that the 75% percentile value of the GMAT score in the population is 600. We are hypothesizing that 75% of the students in the population will have GMAT scores of 600 or below. Do we reject or not reject this hypothesis based on the data we have?

Step 1: State the hypothesis.
The 75% percentile value of the GMAT score in the population is 600.
Step 2: Describe the data generation process and the population.
Since GMAT scores are numeric, we will assume it follows a normal distribution, $N(\mu, \sigma)$. What should these values be to describe the population where the 75% percentile value is 600? Let us start with the sample mean and standard deviation as the population mean and standard deviation.

```
qnorm(mean = mean(admission$GMAT), sd = sd(admission$GMAT), p = .75)
```

```
## [1] 543.4
```

It is below 600, so we need to change either the mean or the standard deviation. We will experiment with changing mean and/or standard deviation to get the population 75% percentile value to be 600.

```
newm = 600-(qnorm(0.75)*sd(admission$GMAT))
print(newm)
```

```
## [1] 545
```

```
qnorm(p = 0.75,mean = newm,sd = sd(admission$GMAT))
```

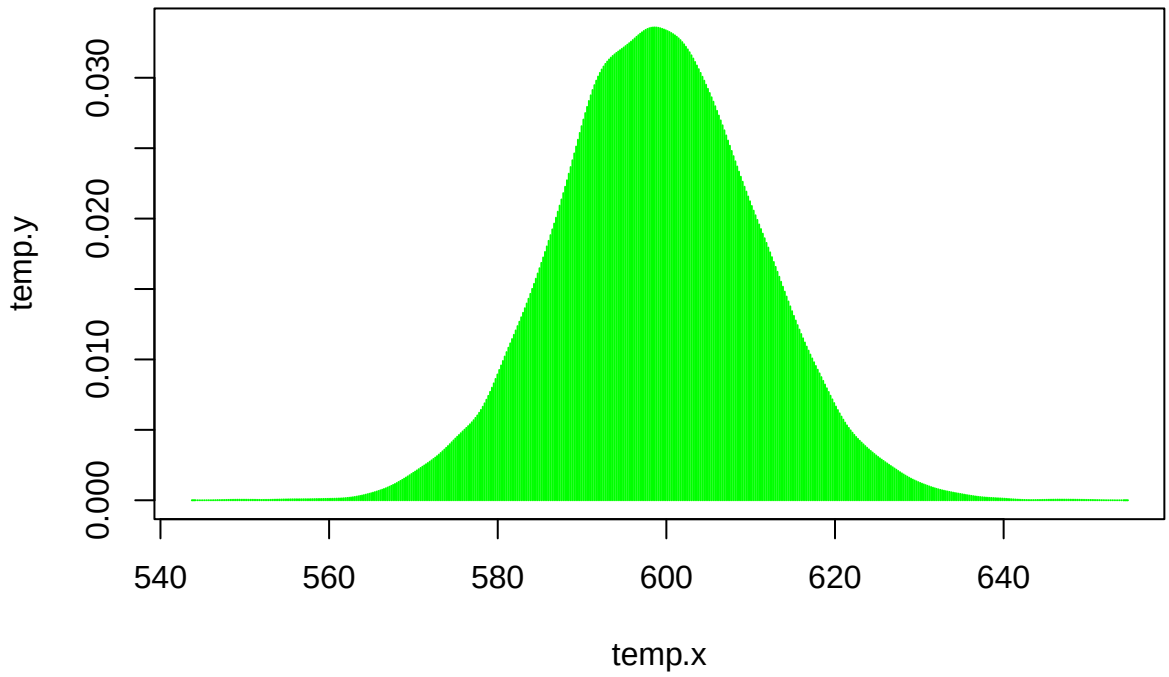
```
## [1] 600
```

Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  s1 = rnorm(sampsize, mean = newm, sd = sd(admission$GMAT))
  return(quantile(s1, probs = .75))
}
sampdist = replicate(10000, f1())
```

Step 4: Get the actual sample and compute the statistic.

```
tstat = quantile(admission$GMAT, probs = .75)
p_2tail(sampdist,tstat)
```



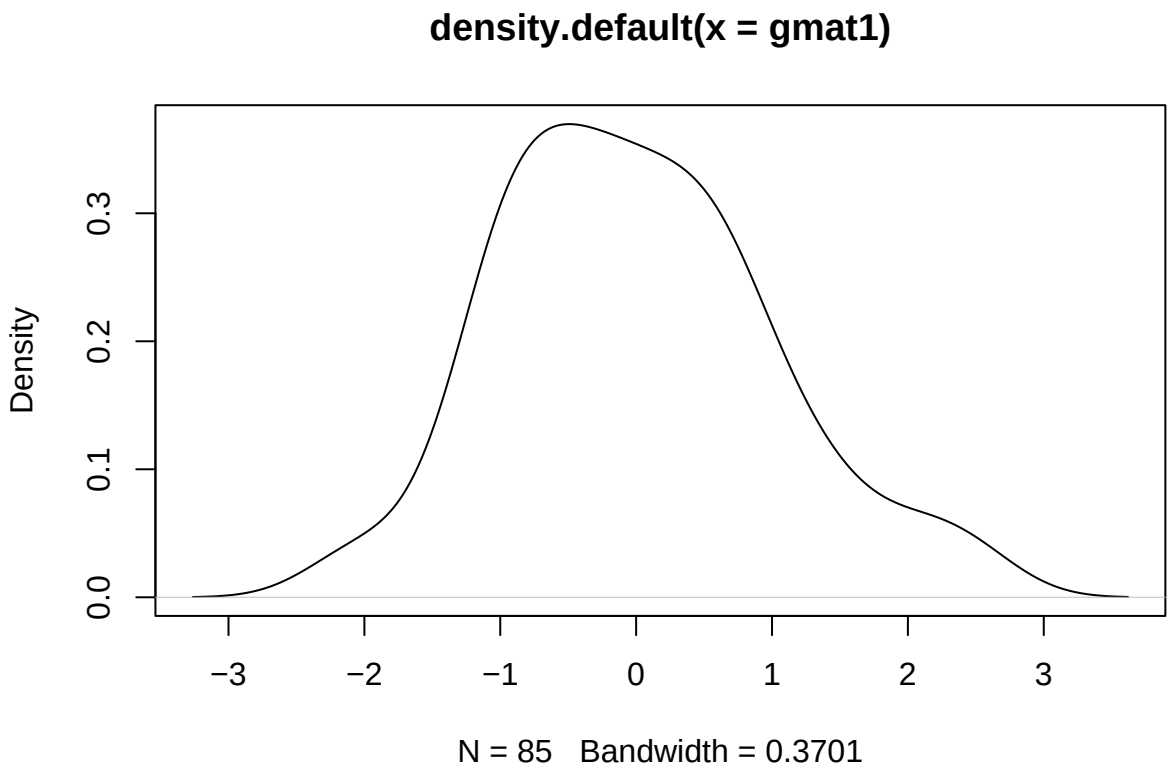

```
## [1] 0
```

The p value is essentially zero and thus we reject the null hypothesis.

Normality Test

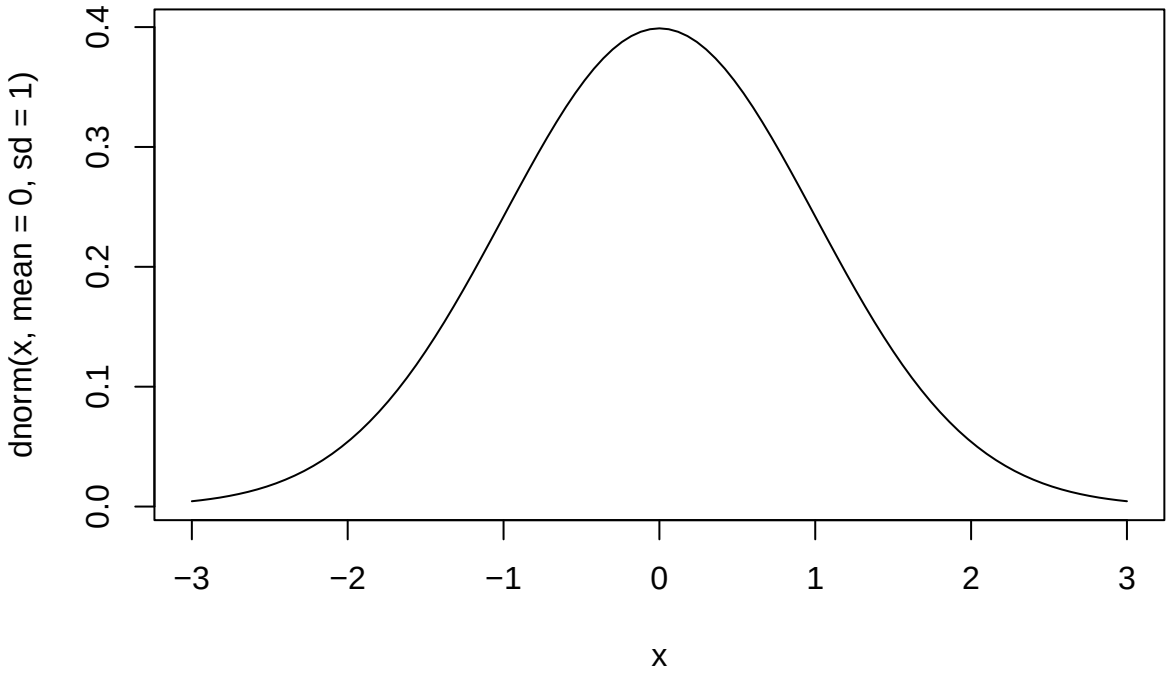
Let us consider the GMAT scores and standardize them.

```
gmat1 = scale(admission$GMAT)
plot(density(gmat1))
```



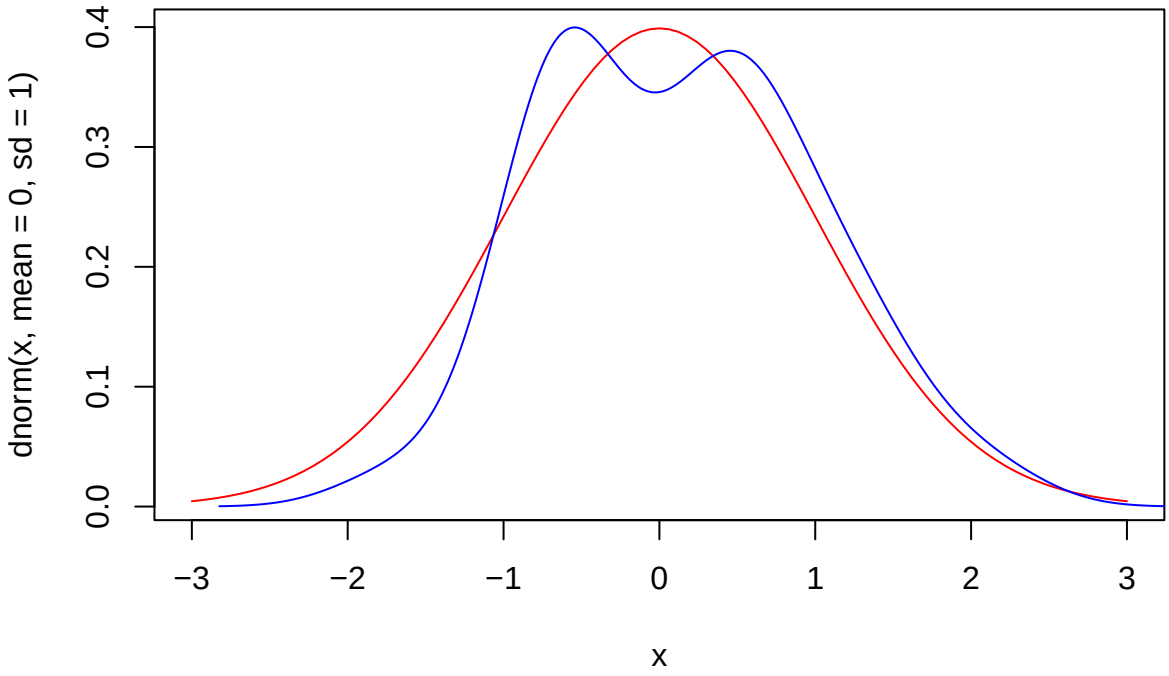
The question we want to ask here is if `gmat1` follows a normal distribution. Let us first plot a normal distribution.

```
curve(dnorm(x, mean = 0, sd = 1), from = -3, to = 3)
```



Let us take a sample of 50 observations and see what the sample looks like.

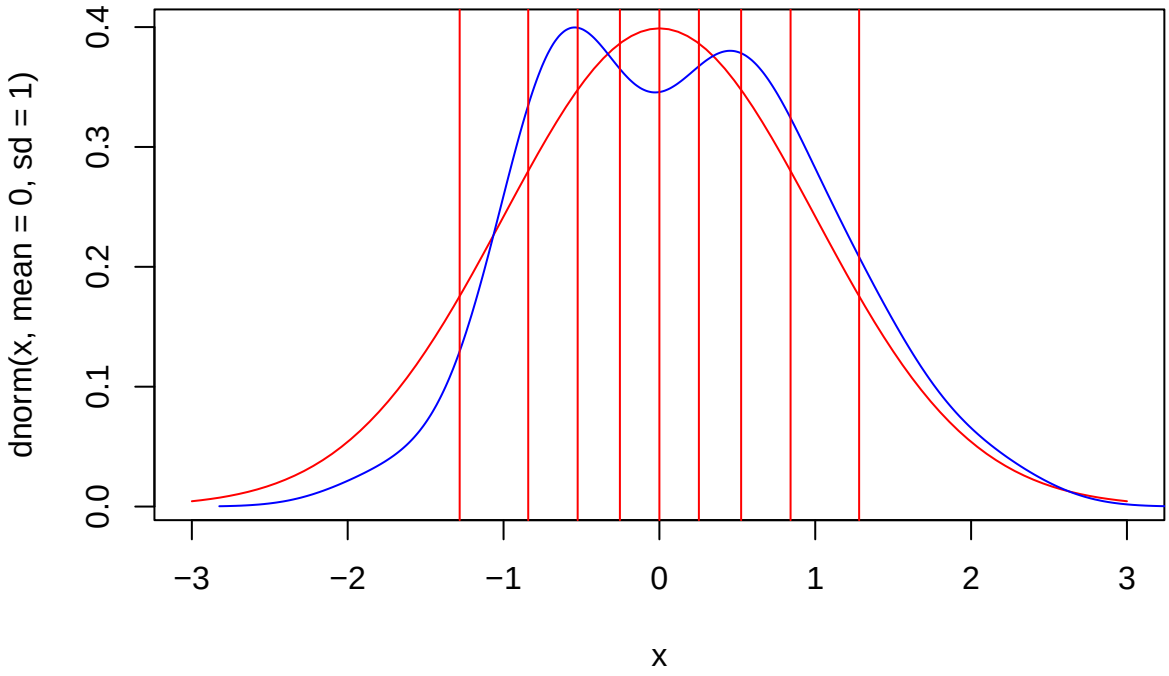
```
set.seed(87654321)
curve(dnorm(x, mean = 0, sd = 1), from = -3, to = 3, col = "red")
s1 = rnorm(n = 50, mean = 0, sd = 1)
lines(density(s1), col = "blue")
```



It is critical to note that when you take a sample from a normal distribution, it will likely not look perfectly normal, but it should be somewhat close. What we need is a measurement of how similar and dissimilar the two curves are.

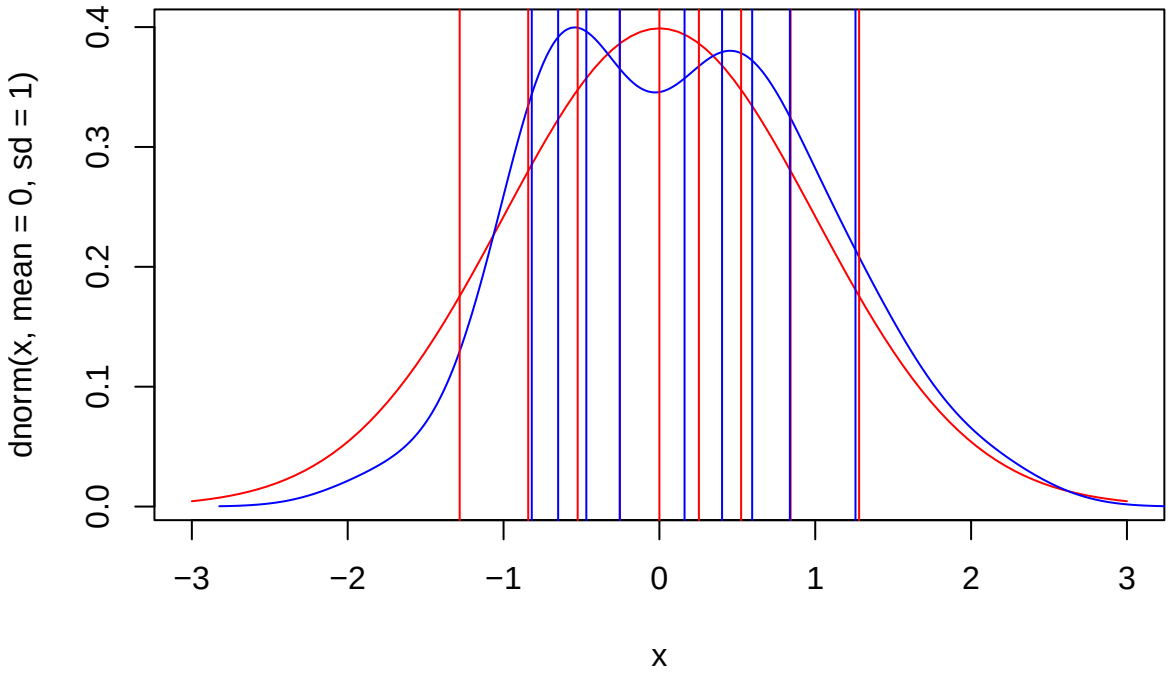
Let us compute the quantile values of the normal distribution at several different quantile levels.

```
quantiles = seq(.1, .9, by = .1)
qvaluespop = qnorm(p = quantiles, mean = 0, sd = 1)
curve(dnorm(x, mean = 0, sd = 1), from = -3, to = 3, col = "red")
lines(density(s1), col = "blue")
abline(v = qvaluespop, col = "red")
```



Now, let us do the same thing for the sample.

```
quantiles = seq(.1, .9, by = .1)
qvaluespop = qnorm(p = quantiles, mean = 0, sd = 1)
curve(dnorm(x, mean = 0, sd = 1), from = -3, to = 3, col = "red")
lines(density(s1), col = "blue")
abline(v = qvaluespop, col = "red")
qvaluessamp = quantile(x = s1, probs = quantiles)
abline(v = qvaluessamp, col = "blue")
```



If the two plots are identical, then the blue and red lines should overlap. We will define a measure that is based on the difference between the two sets of quantile values.

```
print(round(qvaluespop,3))
```

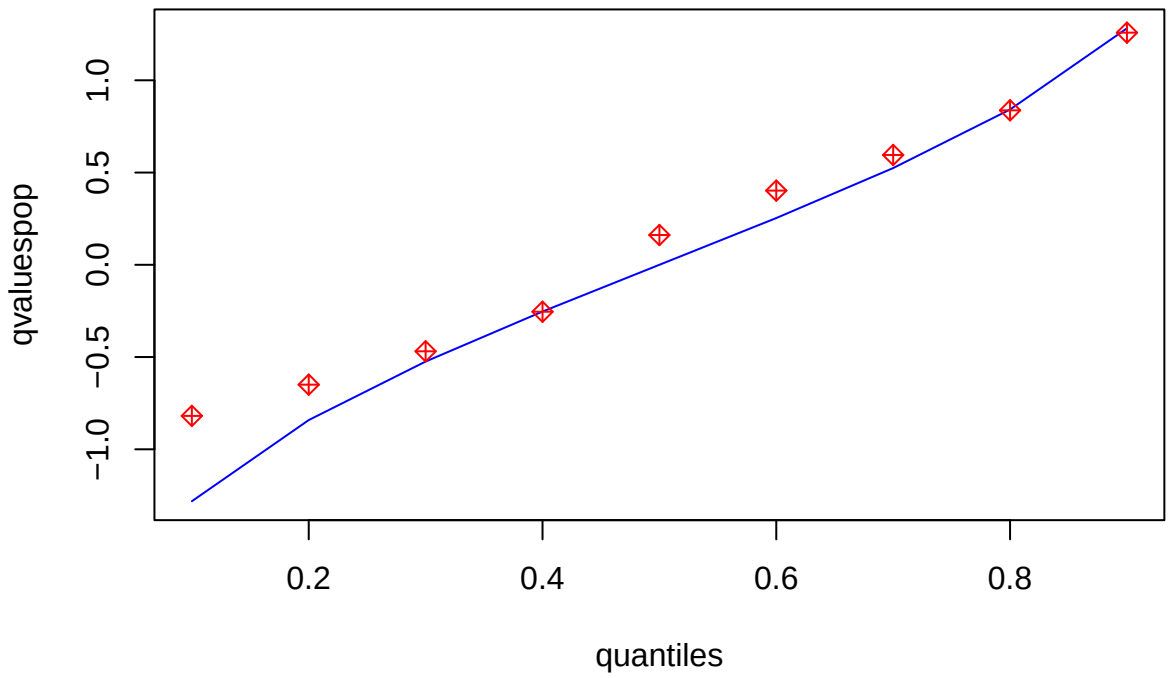
```
## [1] -1.282 -0.842 -0.524 -0.253  0.000  0.253  0.524  0.842  1.282
```

```
print(round(qvaluessamp,3))
```

```
##    10%    20%    30%    40%    50%    60%    70%    80%    90%
## -0.819 -0.650 -0.469 -0.254  0.161  0.402  0.595  0.837  1.258
```

Let us plot these to get a visual on the quantile values of the normal distribution and the sample.

```
plot(x = quantiles, y = qvaluespop, type = "l", col = "blue")
points(x = quantiles, y = qvaluessamp, col = "red", pch = 9 )
```



We will construct a metric to capture the difference between the two quantile values. The metric that we will use is the mean of the absolute differences in quantile values.

```
normmetric = mean(abs(qvaluespop-qvaluessamp))
print(normmetric)
```

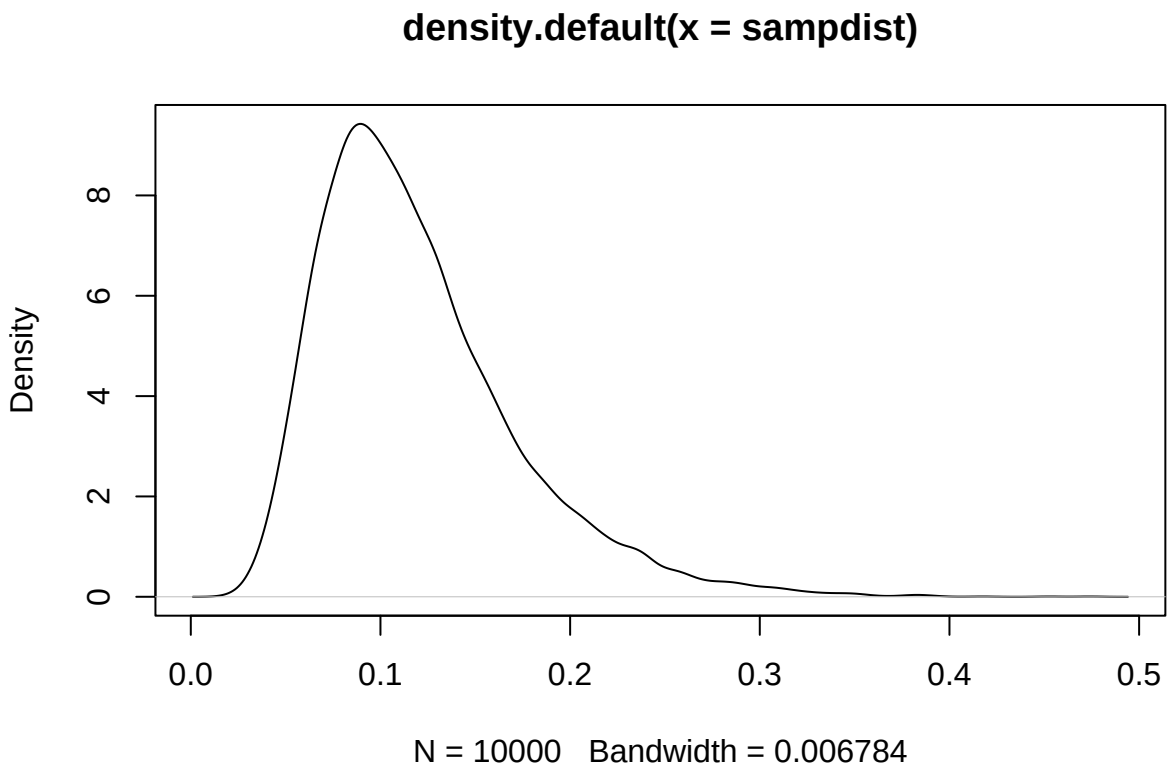
```
## [1] 0.1244
```

The strategy for testing is as follows:

Step 1: State the hypothesis.
Gmat1 follows $N(0, 1)$.
Step 2: Describe the data generation process and the population.
The data comes from a standard normal distribution. Create a sample and compute the `normmetric`.
Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

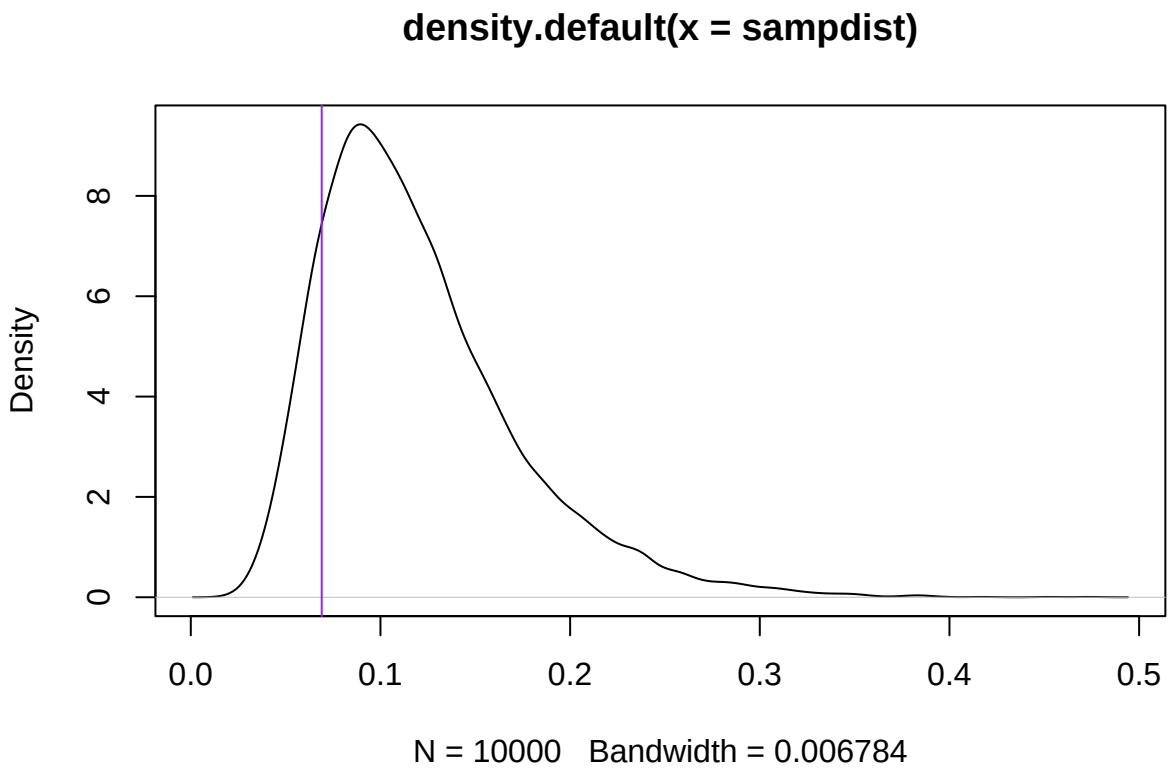
```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  k1 = rnorm(sampsize, mean = 0, sd = 1)
  quantiles = seq(.1, .9, by = .1)
  qvaluespop = qnorm(p = quantiles, mean = 0, sd = 1)
  qvaluessamp = quantile(x = k1, probs = quantiles)
  normmetric = mean(abs(qvaluespop-qvaluessamp))
  return(normmetric)
```

```
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



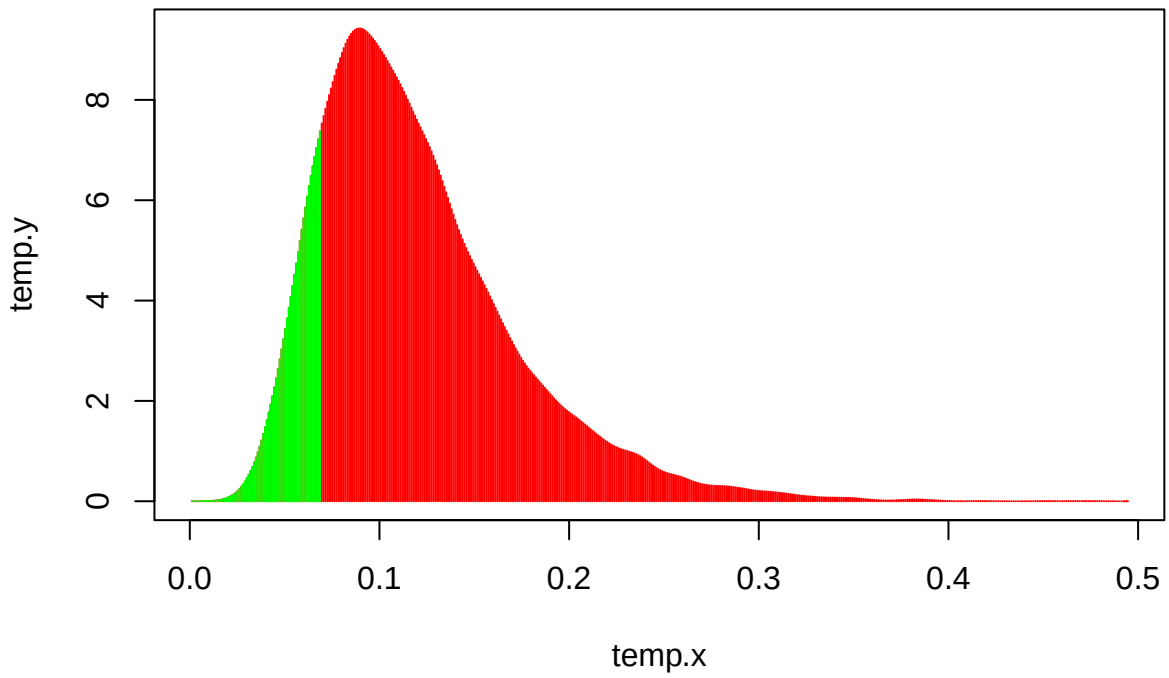
Step 4: Get the actual sample and compute the statistic.

```
plot(density(sampdist))
quantiles = seq(.1, .9, by = .1)
qvaluespop = qnorm(p = quantiles, mean = 0, sd = 1)
qvaluessamp = quantile(x = gmat1, probs = quantiles)
tstat = mean(abs(qvaluespop-qvaluessamp))
abline(v = tstat, col = "purple")
```



Step 5: Plot and compute the p value.

```
tstat = mean(abs(qvaluespop-qvaluessamp))
p_rtail(sampdist,tstat)
```




```
## [1] 0.8632
```

We cannot reject the null hypothesis that the distribution for the GMAT scores is normal.

Inbuilt R Functions

As you would expect, R provides functions to do some of the tests we just conducted. For testing the mean, the function is `t.test()`.

```
t.test(x = admission$GMAT, mu = 510)
```

```
##
##  One Sample t-test
##
## data:  admission$GMAT
## t = -2.4, df = 84, p-value = 0.02
## alternative hypothesis: true mean is not equal to 510
## 95 percent confidence interval:
##  470.9 506.0
## sample estimates:
## mean of x
##     488.4
```

This p value is very close to the p value we got when we wrote out the code.

A function we can use for the test for normality is `shapiro.test()`.

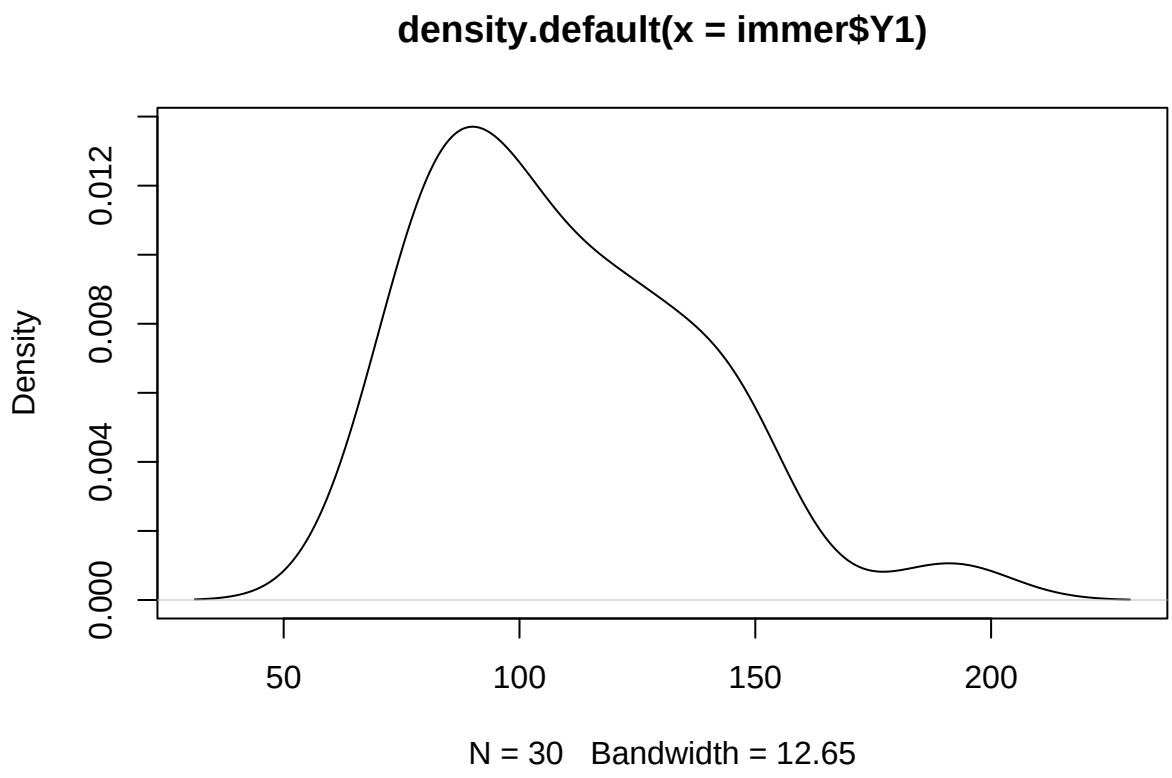
```
shapiro.test(x = admission$GMAT)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  admission$GMAT
## W = 0.98, p-value = 0.2
```

Ab-normal

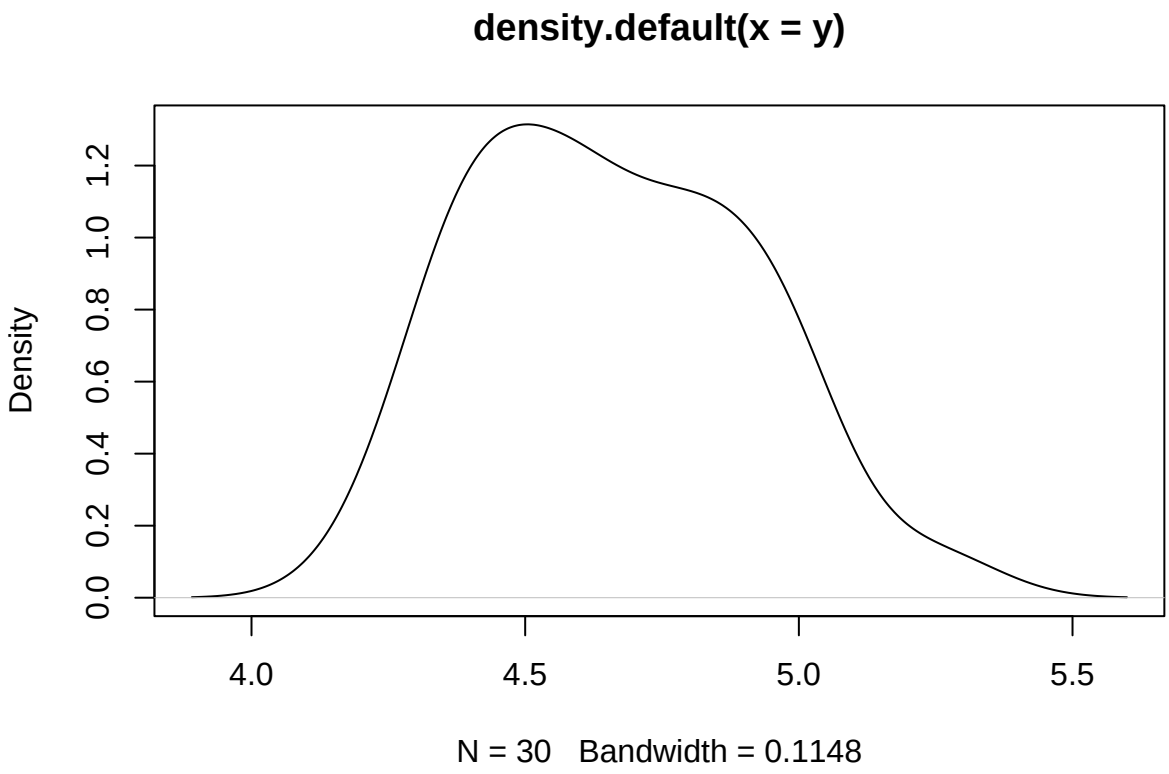
For many tests with numeric variables, we often use the assumption of normality for the population. Sometimes that is not very reasonable.

```
immer = read.csv("../../data/immer.csv")
plot(density(immer$Y1))
```



As you can see, this is not normal. Now let us log transform it and assess normality.

```
y = log(immer$Y1)
plot(density(y))
```




```
shapiro.test(y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.97, p-value = 0.4
```

The p value is now 0.42, and thus the assumption of normality is reasonable. Now you can do the standard tests on this transformed data.

Tests With Factor Data

Suppose we have a claim that 40% of the students are admitted. Let us design a statistical test to verify this claim.

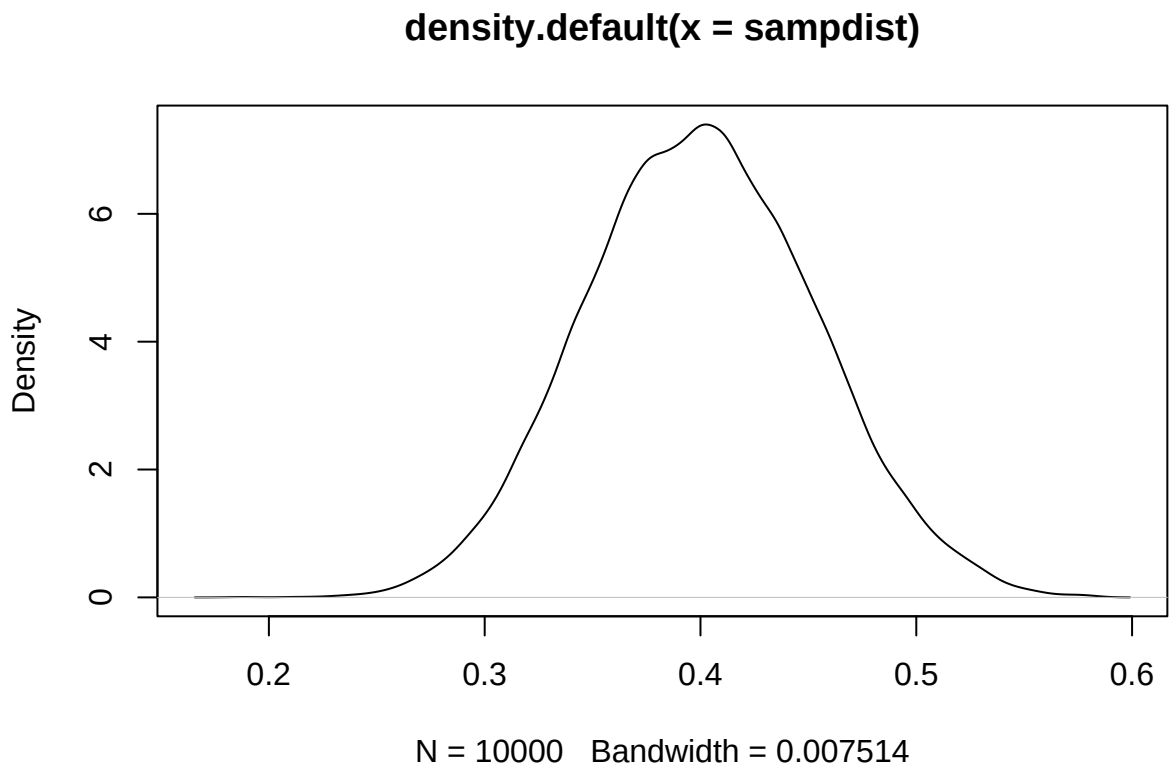
Step 1: State the hypothesis.
40% of the students are admitted.
Step 2: Describe the data generation process and the population.
This is a discrete distribution with two possible outcomes: getting admitted and other. We can describe this as follows:

```
admitdomain = c("admit", "other")
admitprob = c(.4, .6)
```

Step 3: Create a sampling distribution.
The sampling distribution is created with the following code.

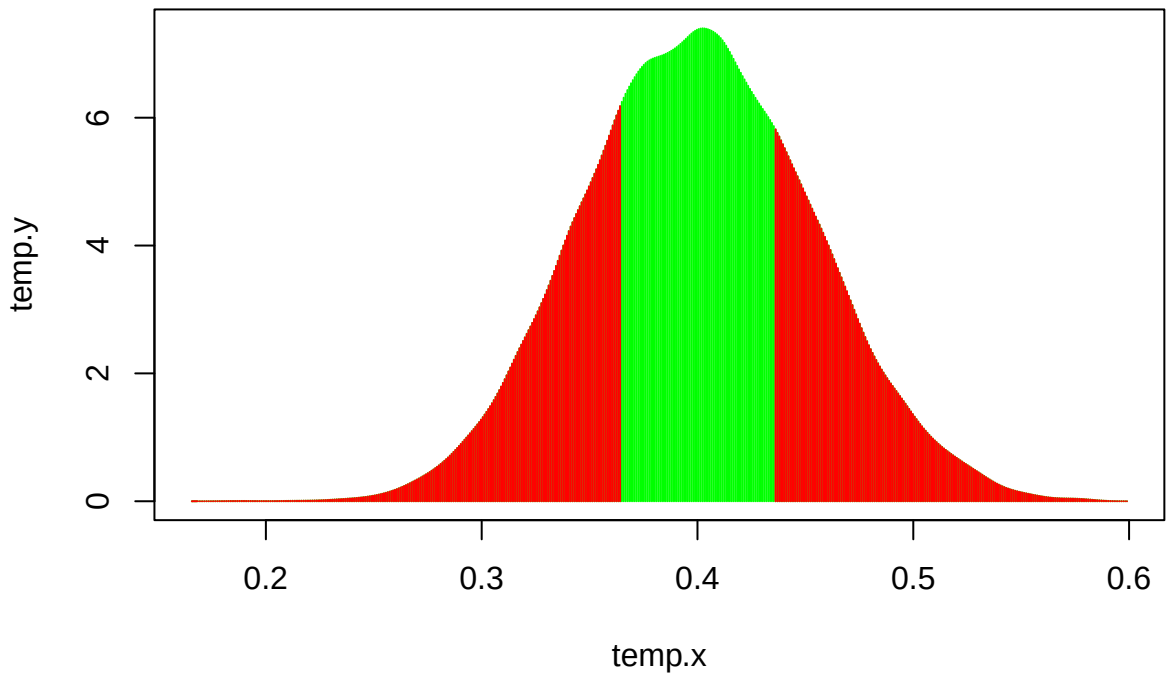
```
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  k1 = sample(x = admitdomain, size = sampsize, replace = T, prob = admitprob)
  return(prop.table(table(k1))[1])

}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = prop.table(table(admission$De))[1]
p_2tail(sampdist,tstat)
```




```
## [1] 0.4414
```

Since the p value is 0.44, we cannot reject the null hypothesis.

```
table(admission$De)
```

```
##
##    admit   border notadmit
##       31       26       28
```

```
nrow(admission)
```

```
## [1] 85
```

The function for the proportions test is `prop.test()`.

```
prop.test(x = 31, n = 85, p = .4)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  31 out of 85, null probability 0.4
## X-squared = 0.31, df = 1, p-value = 0.6
```

```
## alternative hypothesis: true p is not equal to 0.4
## 95 percent confidence interval:
##  0.2650 0.4768
## sample estimates:
##      p
## 0.3647
```

Tests With Two Numeric Variables

Let us consider the two variables GMAT and GPA. Recall our earlier discussion about univariate and multivariate distributions. If two variables are independent, we can describe them as separate univariate distributions. However, if they are not independent, we have to describe them together as a multivariate distribution. For numeric variables, if they are independent, their correlation should be 0. In this example, we want to test if GPA and GMAT are independent.

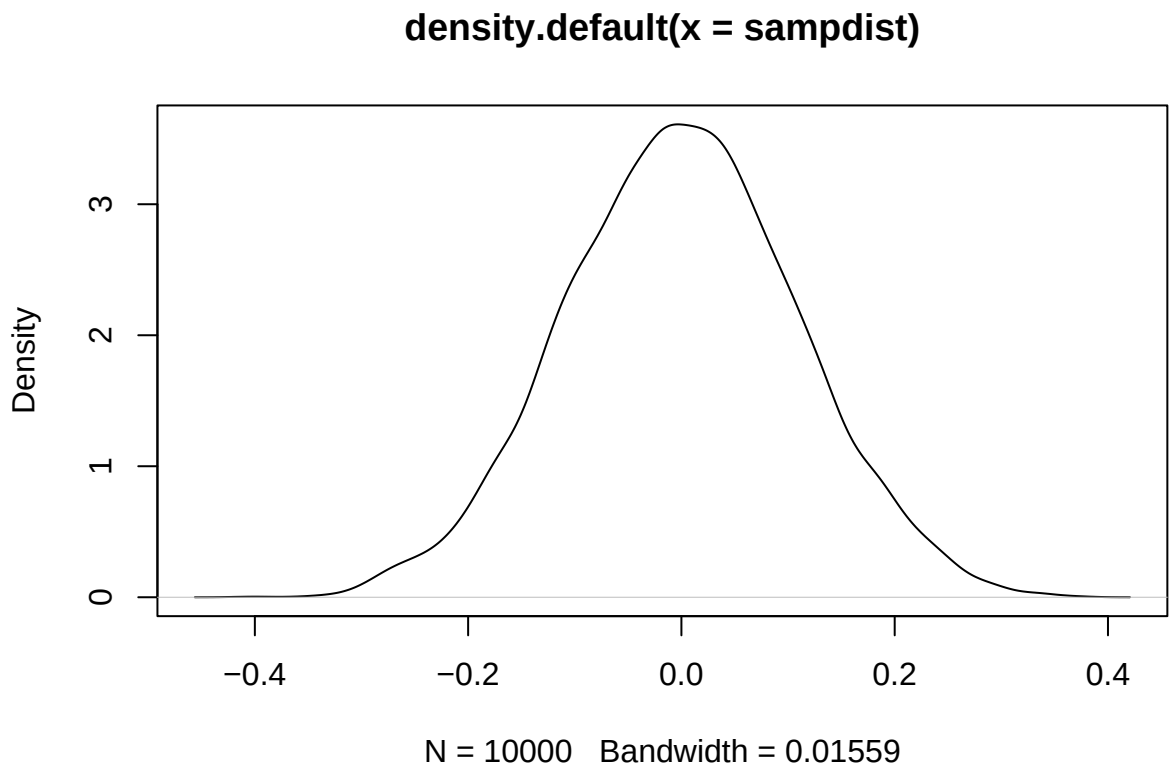
To simplify, let us first standardize the two variables. There is a function in R called `scale()` that can be used to standardize numeric variables.

```
gmat1 = scale(admission$GMAT)
gpa1 = scale(admission$GPA)
```

The test is conducted as follows.

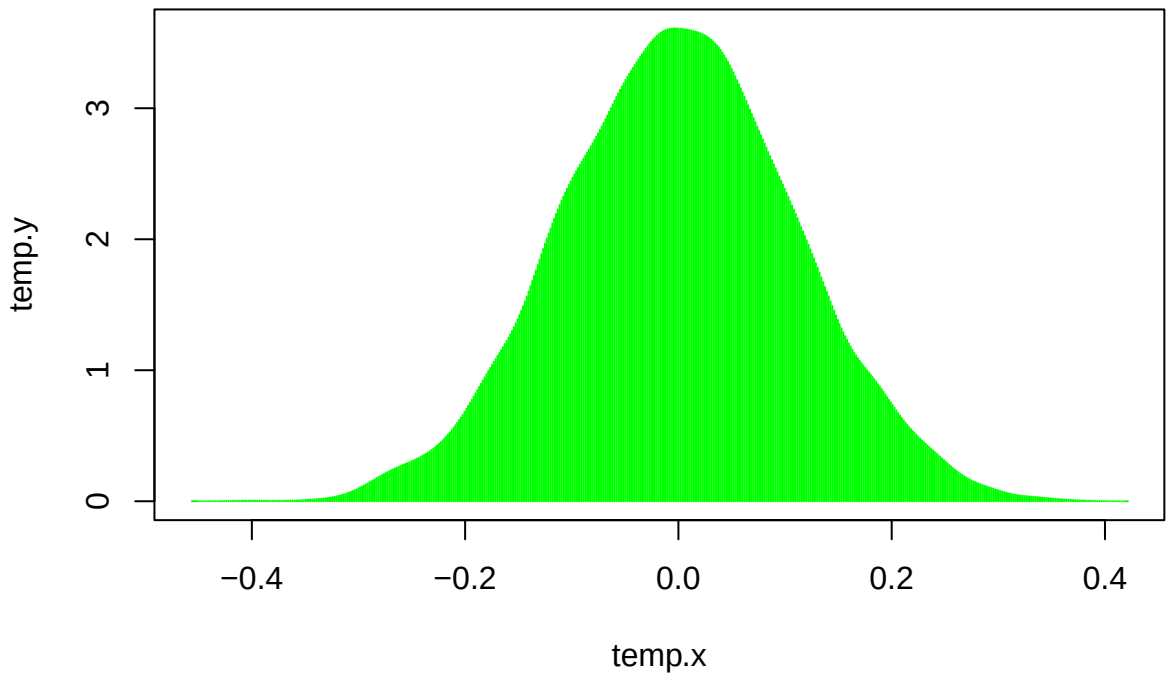
Step 1: State the hypothesis.
`gmat1` and `gpa1` are independent.
Step 2: Describe the data generation process and the population.
If they are independent, gmat1 $\sim N(0,1)$ and gpa1 $\sim N(0,1)$.
Step 3: Create a sampling distribution.
Draw samples from `gpa1` and `gmat1` and compute the correlation. Repeat to create the sampling distribution.

```
gmat1 = scale(admission$GMAT)
gpa1 = scale(admission$GPA)
set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  k1 = rnorm(sampsize, mean = 0, sd = 1)
  k2 = rnorm(sampsize, mean = 0, sd = 1)
  return(cor(x = k1, y = k2))
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = cor(gmat1, gpa1)[1,1]
p_2tail(sampdist,tstat)
```




```
## [1] 0
```

The results clearly indicate that we must reject the null hypothesis that `gmat1` and `gpa1` are independent.

The inbuilt function to do a correlation test is `cor.test()`.

```
cor.test(gmat1, gpa1)
```

```
##
##  Pearson's product-moment correlation
##
## data:  gmat1 and gpa1
## t = 4.7, df = 83, p-value = 0.000009
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2745 0.6135
## sample estimates:
##    cor
## 0.4606
```

Now suppose our null hypothesis is that the correlation between the two is 0.5. How do you test this hypothesis? Since the null hypothesis states that these two variables are not independent, we have to define the hypothesized population as a bivariate normal distribution. We discussed this in an earlier lesson.

Step 1: State the hypothesis.
The correlation between `gmat1` and `gpa1` in the population is .5.

Step 2: Describe the data generation process and the population.
The bivariate normal distribution is defined as:

$$N\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & .5\\.5 & 1\end{bmatrix}\right)$$

Note that the covariance is:

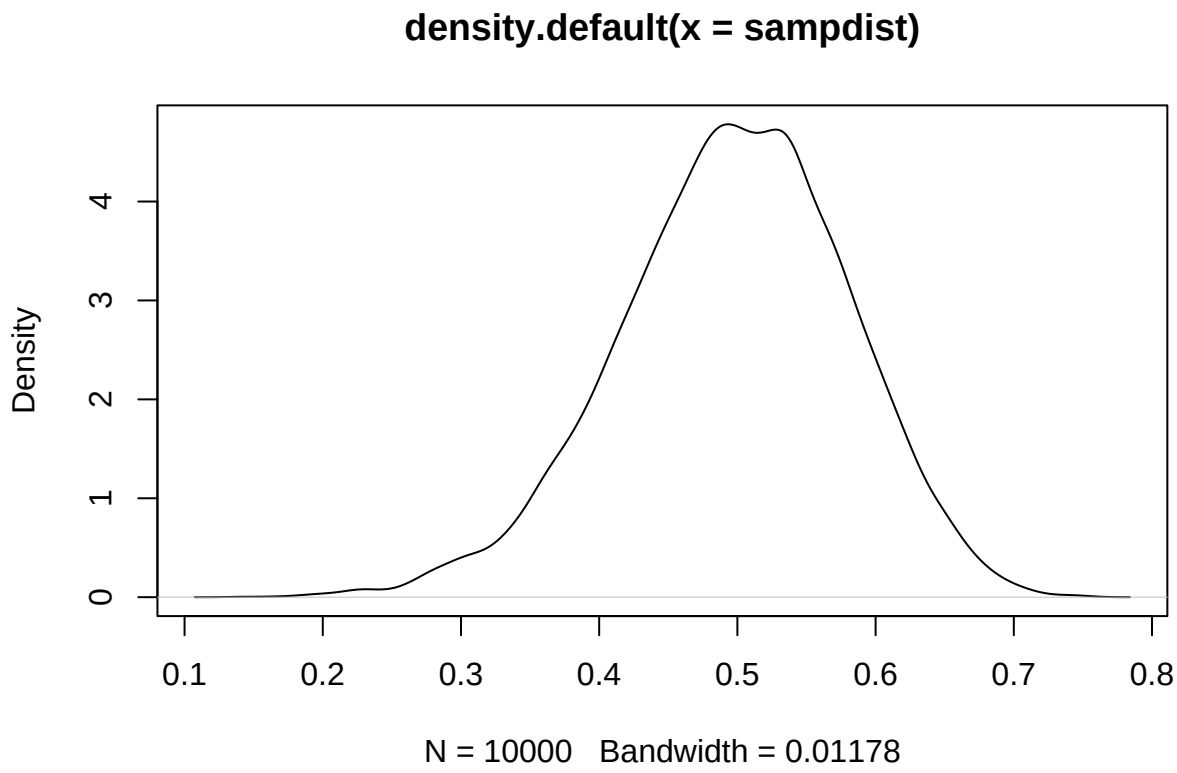
$$\rho_{1,2} \times \sigma_1\sigma_2$$

.

Step 3: Create a sampling distribution.
Draw samples from the bivariate normal distribution and compute the correlation. Repeat to create the sampling distribution.

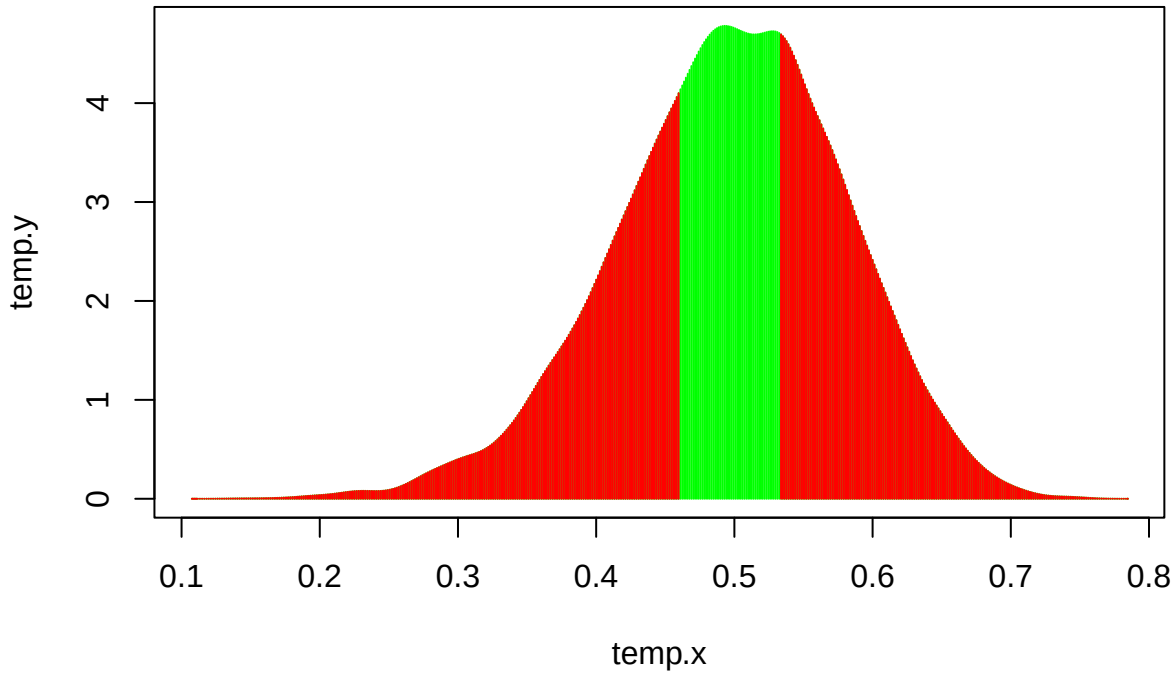
```
library(mvtnorm)
M = c(0,0)
S = matrix(c(1,.5,.5,1), nrow = 2, ncol = 2)

set.seed(87654321)
sampsize = nrow(admission)
f1 = function(){
  x = rmvnorm(nrow(admission),mean = M,sigma = S)
  return(cor(x[,1],x[,2]))
}
sampdist = replicate(10000, f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
tstat = cor(gmat1, gpa1)[1,1]
p_2tail(sampdist,tstat)
```




```
## [1] 0.6641
```

We do not reject the null hypothesis. There is no inbuilt R function to do the test for a hypothesized level of correlation, but with some simple code, we are able to quickly design and execute this test.

Tests With Two Factor Variables

```
survey = read.csv("../../data/survey.csv")
```

We want to test the hypothesis related to two variables `Exer` and `Smoke`. Note that these are both factor variables. Our null hypothesis is that these two variables are independent. Let us take a look at the empirical distribution of each of them.

```
distexer = prop.table(table(survey$Exer))
distsmoke = prop.table(table(survey$Smoke))
distexer
```

```
##
##   Freq   None   Some
## 0.4852 0.1013 0.4135
```

```
distsmoke
```

```
##
##   Heavy   Never   Occas   Regul
## 0.04661 0.80085 0.08051 0.07203
```

If the two variables are independent, then we can describe them separately. The joint probability if they are independent is $P(Smoke \text{ and } Exer) = P(Smoke) \times P(Exer)$. Therefore, the probability that someone exercises frequently and is a heavy smoker should be 0.4852321 * 0.04661017. Assuming independence, we can compute the joint probability as follows:

```
jointdist = distexer %*% t(distsmoke)
round(jointdist,3)
```

```
##
##        Heavy Never Occas Regul
##   Freq 0.023 0.389 0.039 0.035
##   None 0.005 0.081 0.008 0.007
##   Some 0.019 0.331 0.033 0.030
```

The expected number of people in each combination of categories can be calculated as:

```
n = nrow(survey)
E = n * jointdist
round(E,1)
```

```
##
##        Heavy Never Occas Regul
##   Freq   5.4  92.1   9.3   8.3
##   None   1.1  19.2   1.9   1.7
##   Some   4.6  78.5   7.9   7.1
```

Now let us generate a sample of data from the two distributions.

```
o1 = sample(x = c("Freq", "None","Some"), size = n, replace = T, prob = distexer)
o2 = sample(x = c("Heavy", "Never", "Occas","Regul"), size = n, replace = T, prob = distsmoke)
O = table(o1,o2)
O
```

```
##       o2
## o1     Heavy Never Occas Regul
##   Freq     3    98     6     8
##   None     0    20     2     3
##   Some     7    65    10    15
```

How close are the observed and expected values? A commonly used metric called chi-square statistic is defined as:

$$\sum \frac{(O-E)^2}{E}$$

Ideally, the chi-square value will be 0. We can easily calculate this.

```
sum(((O-E)^2)/E)
```

```
## [1] 17.77
```

This is the chi-square value.

Now let us do the test.

Step 1: State the hypothesis.
`Smoke` and `Exer` are independent.
Step 2: Describe the data generation process and the population.
Since they are independent, we can describe them separately as:

Exer

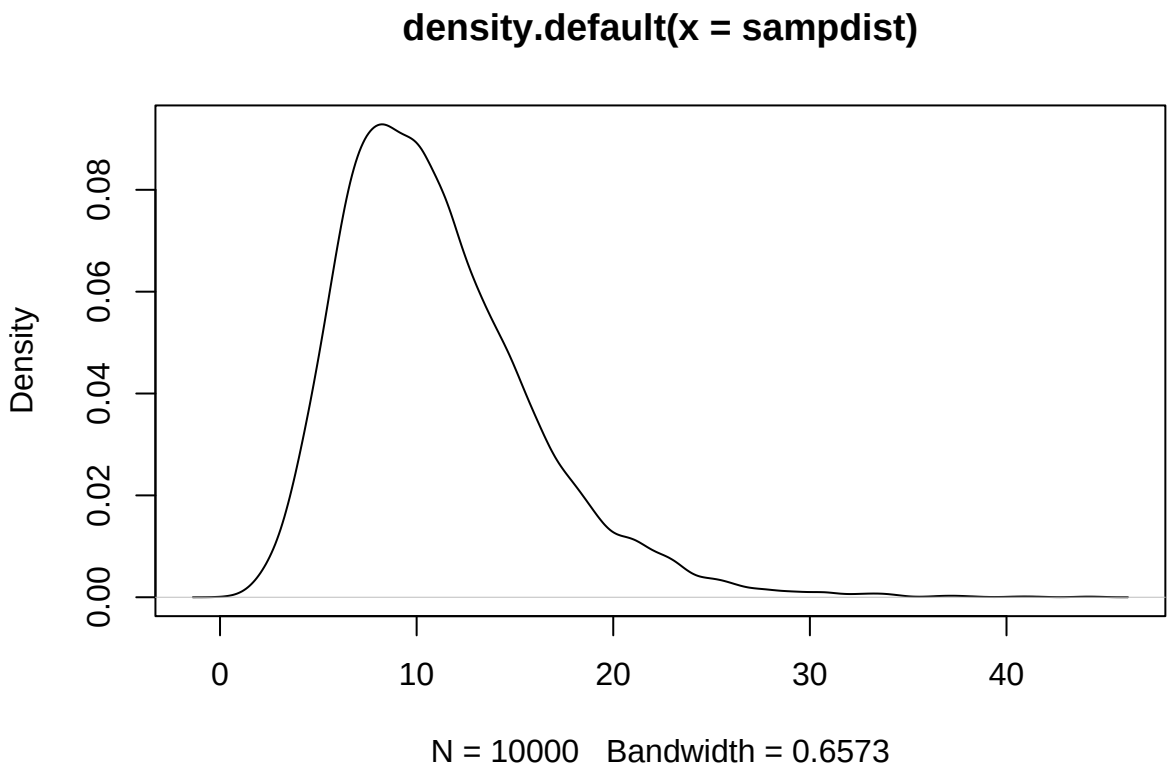
Freq	None	Some
0.4852321	0.1012658	0.4135021

Smoke

Heavy	Never	Occas	Regul
0.04661017	0.80084746	0.08050847	0.07203390

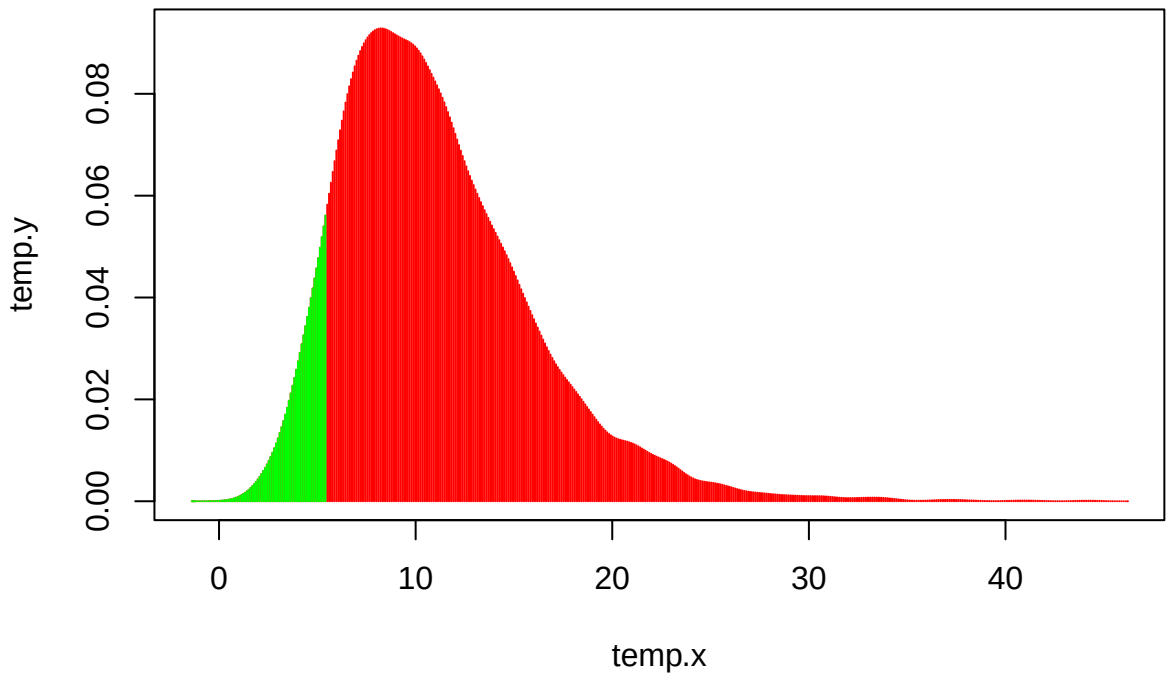
Step 3: Create a sampling distribution.

```
sampsize = nrow(survey)
f1 = function(){
  o1 = sample(x = c("Freq", "None","Some"), size = sampsize, replace = T, prob = distexer)
  o2 = sample(x = c("Heavy", "Never", "Occas","Regul"), size = sampsize, replace = T, prob = distsmoke)
  O = table(o1,o2)
  chi = sum(((O-E)^2)/E)
  return(chi)
}
sampdist = replicate(10000,f1())
plot(density(sampdist))
```



Step 4: Get the actual sample and compute the statistic.

```
O = table(survey$Exer,survey$Smoke)
tstat = sum(((O-E)^2)/E)
p_rtail(sampdist,tstat)
```




```
## [1] 0.9124
```

Thus, we cannot reject the null hypothesis that `Smoke` and `Exer` are independent. The test we just did is called the chi-square test. The inbuilt R function is `chisq.test()`. Let us execute this.

```
chisq.test(x = survey$Exer, y = survey$Smoke)
```

```
##
##  Pearson's Chi-squared test
##
## data:  survey$Exer and survey$Smoke
## X-squared = 5.5, df = 6, p-value = 0.5
```

The p values are slightly different because the function assumes that the sampling distribution is a chi-square distribution. We made no such assumption and created the sampling distribution from scratch.

One advantage of designing your own testing strategy is that you can customize it. Suppose we are only interested in people who smoke heavily and exercise very frequently. Are these two events independent? If you recall from the previous discussion, we use a concept called `lift` to evaluate independence. Let us use that concept to design a new test.

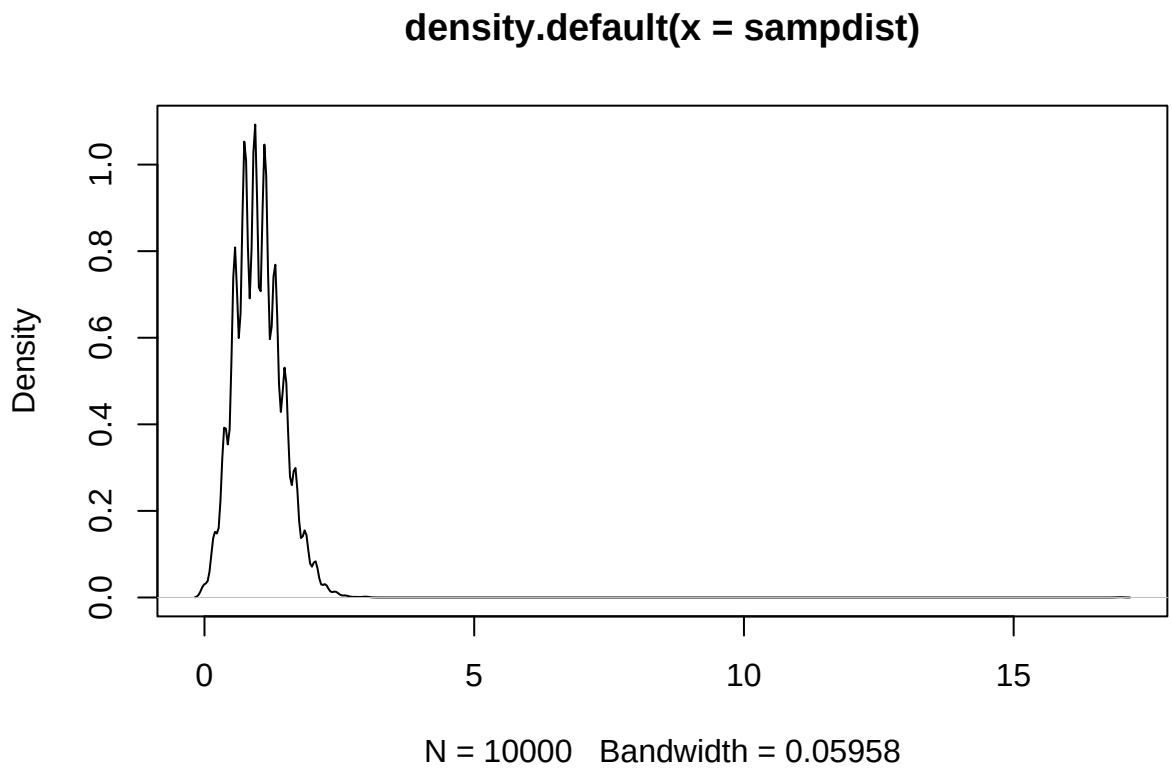
Step 1: State the hypothesis.
Heavy smoking and frequent exercising are independent.
Step 2: Describe the data generation process and the population.
Since they are independent, we can describe them separately as:

$P(Exer\ Freq) = 0.4852321$
$P(Smoke\ Heavy) = 0.04661017$

The metric we will use here is lift, which is the ratio of observed and expected values.

Step 3: Create a sampling distribution.

```
set.seed(87654321)
sampsize = nrow(survey)
f1 = function(){
  o1 = sample(x = c("Freq", "None","Some"), size = sampsize, replace = T, prob = distexer)
  o2 = sample(x = c("Heavy", "Never", "Occas","Regul"), size = sampsize, replace = T, prob = distsmoke)
  O = table(o1,o2)
  lift = O[1,1]/E[1,1]
  return(lift)
}
sampdist = replicate(10000,f1())
plot(density(sampdist))
```

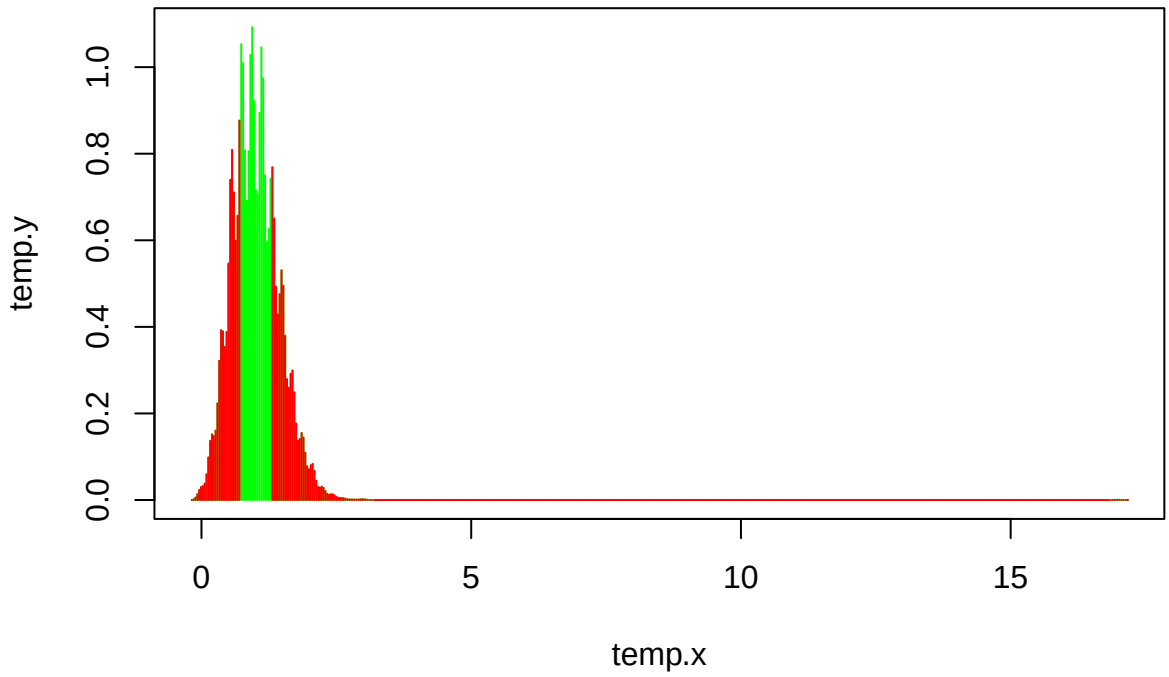


Step 4: Get the actual sample and compute the statistic.

```
O = table(survey$Exer,survey$Smoke)
tstat = O[1,1]/E[1,1]
tstat
```

```
## [1] 1.306
```

```
p_2tail(sampdist,tstat)
```




```
## [1] 0.3865
```

As the p value indicates, we cannot reject the hypothesis, and thus someone smoking heavily is unrelated to someone exercising frequently.