

GEA1000 QUANTITATIVE REASONING WITH DATA

Tutorial 4

Please work on the problems before coming to class. In class, you will engage in group work.

Through the lectures, we learn that random variables play a very important role in statistical inference. In this tutorial, we will explore how they are integral in the formulation of hypothesis tests, using the one sample t-test as a case study. We will also take a short detour into confidence intervals.

Warming up with a Few Samples

HDB_resale_2019 contains data on HDB resale transactions that occurred in the year 2019. HDB_resale_2019 will be our population data for this section.

- (1) Drawing a sample of 200 rows from HDB_resale_2019

Open the file Tutorial 4 Sample Generation (an Excel workbook) in Excel.

In the worksheet 'Original Data', you will find complete data copied from HDB_resale_2019. In addition, you will also see an additional column labelled 'ID'.

	A	B	C
1	ID	year	month
2	1	2019	
3	2	2019	
4	3	2019	
5	4	2019	
6	5	2019	
7	6	2019	
8	7	2019	
9	8	2019	
10	9	2019	
11	10	2019	
12	11	2019	
13	12	2019	
14	13	2019	
15	14	2019	
16	15	2019	
17	16	2019	
18	17	2019	
19	18	2019	
20	19	2019	
21	20	2019	
22	21	2019	

Original Data

'ID' serves as a unique identifier for the rows, and is combined with a (pseudo-) random number generator to randomly select rows from our population data.

The worksheet 'Sample Generation' consists of formulae written and organised to generate a random sample of 200 rows from our population data. You may hit 'F9' on your keyboard to refresh the formulae, in order to generate a fresh sample.

(On certain keyboards, the key 'F9' is bound to another function. If hitting 'F9' does not refresh the worksheet, you may try 'Fn' + 'F9', given that there is a 'Fn' key on your keyboard. If this too fails, you can try opening [Tutorial 4 Sample Generation](#) in Office 365 on your preferred browser, and repeat the attempts there.)

To prevent the formulae from accidentally refreshing itself as you deal with the sample data, it is recommended that you copy and paste every sample you generate onto a new worksheet.

(2) Generate 3 random samples following the instructions in (1).

Compute the mean `price_psm` of each sample generated. Are the means different?

pnor F9

Things like the sample mean, fall under the broad category of *sample statistics*. In general, a sample statistic is a formula or a computation that can be applied to any sample of numerical data drawn from a population, to return a numerical value.

(3) Construct the 95% confidence interval for mean `price_psm` of each sample drawn in (2). Are the intervals different?

yes

Given any sample of numerical data drawn from a population, we can compute its t-statistic, t , using the following formula:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}, \text{ where}$$

- \bar{x} is the sample mean,
- μ is the population mean,
- s is the sample standard deviation, and
- n is the sample size.

Hence, the t-statistic is a sample statistic.

- (4) What is the mean `price_psm` of the population?
- (5) Compute the t-statistic of `price_psm` for each sample drawn in (2). Are the t-statistics different?
- (6) What property of the sampling process can be used to explain the variation (or lack thereof) in (2), (3) and (5)?

Note that, while the computation of a sample mean requires only sample data, the computation of a t-statistic requires an additional input – the population mean. This happens to fit in well with the framework of hypothesis testing, when we want to conduct a test involving the population mean. We will see why in the last section of this tutorial.

In considering how a sample statistic varies across random samples, we are looking at a random variable. This random variable is called the *sampling distribution* of the sample statistic. It then makes sense to ask about how the density curve of the sampling distribution looks like.

That Certain Sample Statistics Follow Well-known Distributions

There are no questions in this section. Instead, it contains a few hands-on exercises. You are advised to attempt the exercises by following the detailed instructions given. If you run into problems (bugs) in executing the instructions, fret not, because your tutor will demonstrate these exercises in class.

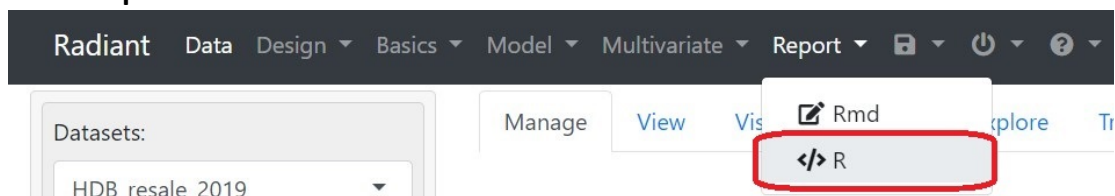
Let's say we want to get a handle on the sampling distributions of some sample statistics. To this end, it is helpful to run simulations and visualise their results. In particular, we want to simulate the drawing of a large number of samples, and the computation of relevant samples statistics for each sample. The point and click interfaces of Radiant and Excel are far too slow for such purposes; it makes more sense, speed-wise, to run some simple code in R.

Fortunately, Radiant comes with a sufficiently user-friendly editor for R code, which can be accessed via its 'Report' functionality.

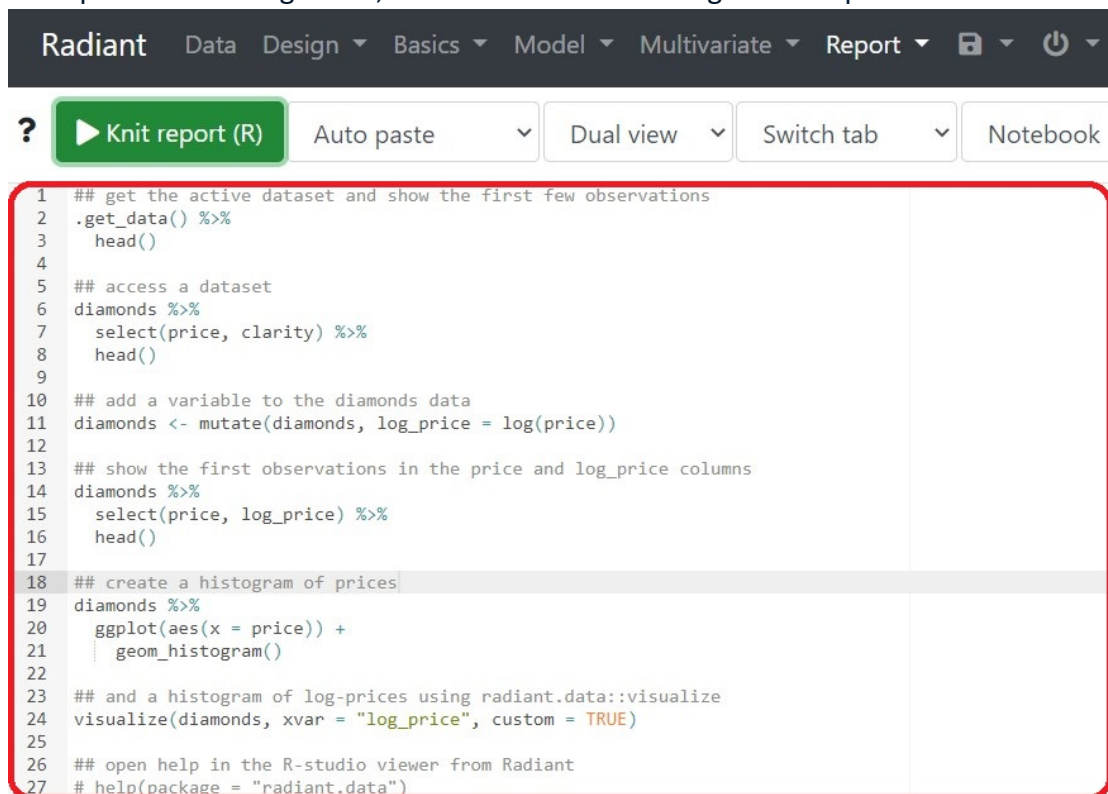
The next four parts require a little navigation and set-up in Radiant. Please follow the instructions below.

A. Ensure that you have `HDB_resale_2019` as your active data set.

B. Go to **Report > R**.



- C. The editor takes up more or less the left half of the window, and may contain examples of code fragments, as shown in the following screencap.



The screenshot shows the Radiant IDE interface. At the top is a dark navigation bar with the 'Radiant' logo and several menu items: 'Data', 'Design', 'Basics', 'Model', 'Multivariate', 'Report', and icons for saving and power. Below this is a toolbar with a question mark icon, a green 'Knit report (R)' button, and dropdown menus for 'Auto paste', 'Dual view', 'Switch tab', and 'Notebook'. The main area is a code editor with a red border, containing R code for data manipulation and visualization. The code is as follows:

```
1 ## get the active dataset and show the first few observations
2 .get_data() %>%
3   head()
4
5 ## access a dataset
6 diamonds %>%
7   select(price, clarity) %>%
8   head()
9
10 ## add a variable to the diamonds data
11 diamonds <- mutate(diamonds, log_price = log(price))
12
13 ## show the first observations in the price and log_price columns
14 diamonds %>%
15   select(price, log_price) %>%
16   head()
17
18 ## create a histogram of prices
19 diamonds %>%
20   ggplot(aes(x = price)) +
21     geom_histogram()
22
23 ## and a histogram of log-prices using radiant.data::visualize
24 visualize(diamonds, xvar = "log_price", custom = TRUE)
25
26 ## open help in the R-studio viewer from Radiant
27 # help(package = "radiant.data")
```

For each of the following parts, you may re-run the code a few times, each time clicking on the “Knit Report (R)” button to generate a new report from a fresh set of samples. You should notice that the expected result still holds, even if there are slight variations across the different runs.

(7)

Plotting a histogram of the sample means of 10000 random samples of size 200

Delete all existing code in the editor, then paste the following there.

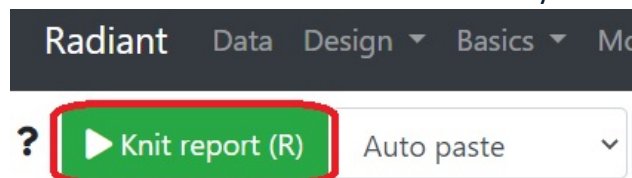
```
resale <- .get_data()

smean = numeric(10000)

for (i in 1:10000){
  s = sample(resale[, "price_psm"], 200)
  smean[i] = mean(s)
}

hist(smean,
     main = "Histogram of Sample Means",
     xlab = "Sample Mean",
     xlim = c(4200, 4800),
     breaks = seq(min(smean), max(smean), l = ((max(smean)-min(smean))/6)+1),
     freq=FALSE)
```

Click the button above the editor that says “Knit Report (R)”.



You should see the knitted report on the right-hand side of the window, and a histogram of the results of our simulation at the bottom of the report.

Let us go through the different parts of the code to see what each of them does.

```
resale <- .get_data()
```

Here, we assign the name `resale` to the active data set.

```
smean = numeric(10000)
```

Here, we define an array `smean` that can store up to 10000 numerical values. We use this array to store the sample means from the 10000 samples.

```
for (i in 1:10000){  
  s = sample(resale[, "price_psm"], 200)  
  smean[i] = mean(s)  
}
```

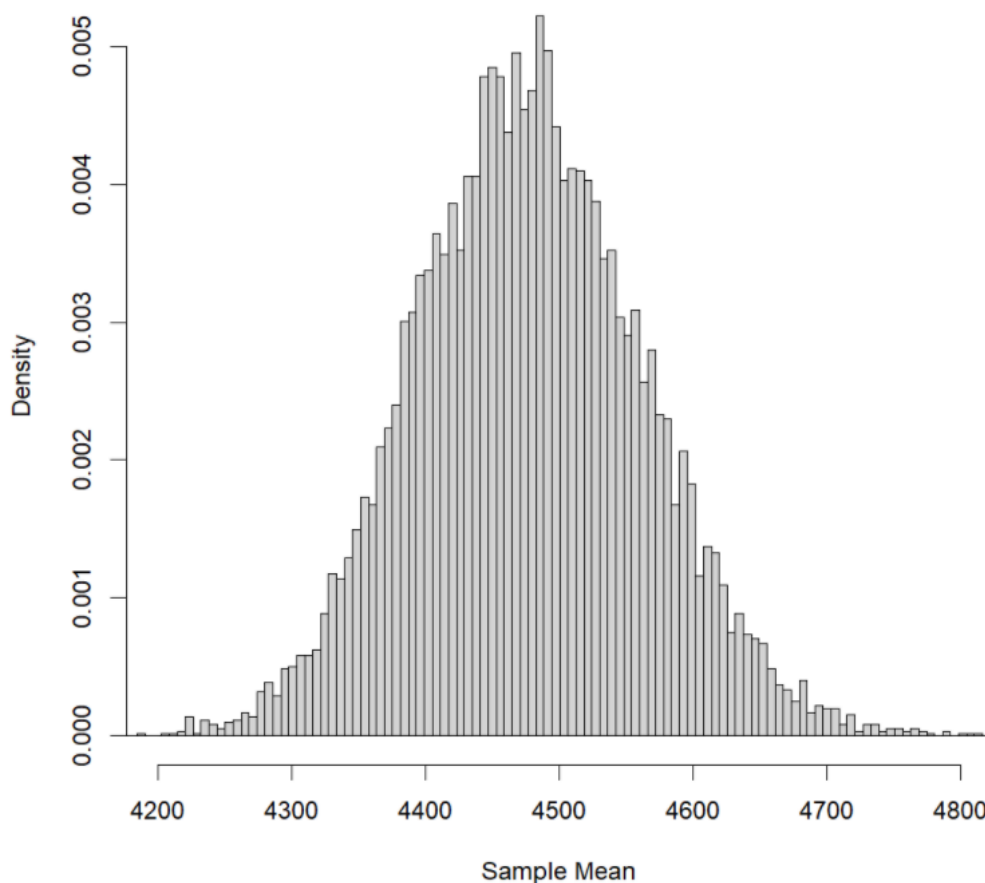
Here, we draw 10000 random samples of size 200. For each sample, we compute its mean `price_psm`, and record the value in `smean`.

```
hist(smean,  
  main = "Histogram of Sample Means",  
  xlab = "Sample Mean",  
  xlim = c(4200, 4800),  
  breaks = seq(min(smean), max(smean), l = ((max(smean)-min(smean))/6)+1),  
  freq=FALSE)
```

Here, we plot a density histogram of all values stored in `smean`. These are simply the mean `price_psm` values of the 10000 samples drawn. In a density histogram, the area of each bar represents the proportion of data points in its corresponding bin, out of the total number of data points. The multiple lines of code here specify a bunch of formatting rules that make the histogram more aesthetically pleasing.

The histogram presented in your report should look somewhat similar in shape to the one shown below.

Histogram of Sample Means



The minor differences between the histogram captured above, and the one in your report, are due to the variability in random sampling. In general, the larger the number of random samples used to plot a histogram of sample statistics, the smaller the variability is between plots.

The shape of this histogram – as well as that of the one in your report – approximates the theoretical density curve of the sampling distribution of the sample mean for `price_psm`, for samples of size 200 drawn from our population.

Central Limit Theorem

If central size is large (assume SRS),
then the distribution of sample mean is

approximately normal with center = population mean

and $\sigma_{\text{Hofr}} = \frac{\sigma}{\sqrt{n}}$ — population std
— sample size

(8)

Constructing the 95% confidence intervals for the sample means of 10000 random samples of size 200

Delete all existing code in the editor, then paste the following there.

```
resale <- .get_data()

pmean = mean(resale[, "price_psm"])

pmean_in_CI = logical(10000)

for (i in 1:10000){
  s = sample(resale[, "price_psm"], 200)
  CI_lb = mean(s) - qt(0.975, df=199)*sd(s)/(sqrt(200))
  CI_ub = mean(s) + qt(0.975, df=199)*sd(s)/(sqrt(200))
  if (pmean >= CI_lb & pmean <= CI_ub) {
    pmean_in_CI[i] = TRUE
  }
}

print(paste("Proportion of intervals containing the population mean =",
  sum(pmean_in_CI)/10000,
  quote = FALSE))
```

Click on “Knit Report (R)”. You should see a statement at the bottom of the knitted report, stating the proportion of intervals containing the population mean.

As in (7), let us analyse the code above, part by part.

```
pmean = mean(resale[, "price_psm"])
```

Here, we are assigning the name `pmean` to the mean `price_psm` of the population. This is done for ease of reference when checking if each confidence interval contains the population mean.

```
pmean_in_CI = logical(10000)
```

Here, we assign the name `pmean_in_CI` to an array that stores up to 10000 ‘TRUE / FALSE’ answers. This array will be used to store answers to whether a confidence interval constructed from one of the 10000 samples, contains the population mean.

```

for (i in 1:10000){
  s = sample(resale[, "price_psm"], 200)
  CI_lb = mean(s) - qt(0.975, df=199)*sd(s)/(sqrt(200))
  CI_ub = mean(s) + qt(0.975, df=199)*sd(s)/(sqrt(200))
  if (pmean >= CI_lb & pmean <= CI_ub) {
    pmean_in_CI[i] = TRUE
  }
}

```

Here, we draw 10000 random samples, and for each sample, construct its confidence interval for the population mean. We then check if the confidence interval constructed contains the population mean, before storing the answer (TRUE if it contains the population mean, FALSE otherwise) in `pmean_in_CI`.

```

print(paste("Proportion of intervals containing the population mean =",
  sum(pmean_in_CI)/10000),
  quote = FALSE)

```

Here, we compute and print the proportion of TRUE out of all the answers stored in `pmean_in_CI`.

We should expect the proportion published in the report to be very close to 0.95, as evidenced in the following screencap.

```

resale <- .get_data()

pmean = mean(resale[, "price_psm"])

pmean_in_CI = logical(10000)

for (i in 1:10000){
  s = sample(resale[, "price_psm"], 200)
  CI_lb = mean(s) - qt(0.975, df=199)*sd(s)/(sqrt(200))
  CI_ub = mean(s) + qt(0.975, df=199)*sd(s)/(sqrt(200))
  if (pmean >= CI_lb & pmean <= CI_ub) {
    pmean_in_CI[i] = TRUE
  }
}

print(paste("Proportion of intervals containing the population mean =",
  sum(pmean_in_CI)/10000),
  quote = FALSE)

```

```
[1] Proportion of intervals containing the population mean = 0.9496
```

This is in line with what is taught in the lectures: upon repeated sampling, about 95% of the confidence intervals constructed contains the population mean.

(9) Plotting a histogram of the t-statistics of 10000 random samples of size 200

Delete all existing code in the editor, then paste the following there.

```
resale <- .get_data()

pmean = mean(resale[, "price_psm"])
tsta = numeric(10000)

for (i in 1:10000){
  s = sample(resale[, "price_psm"], 200)
  tsta[i] = (mean(s) - pmean) / (sd(s) / sqrt(200))
}

hist(tsta,
     main = "Histogram of T-Statistics",
     xlab = "T-Statistic",
     xlim = c(-4, 4),
     breaks = seq(min(tsta), max(tsta), l = ((max(tsta) - min(tsta)) / 8 * 100) + 1),
     freq = FALSE)
```

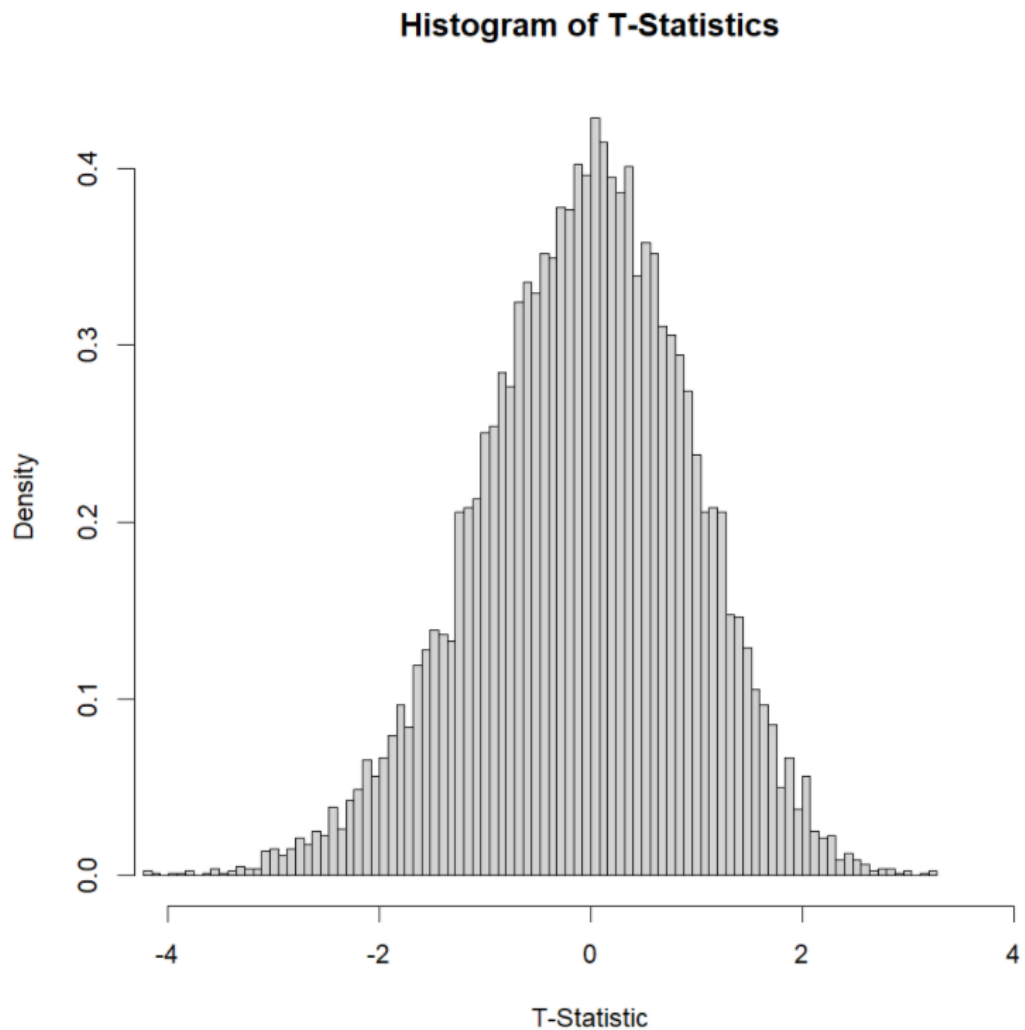
Click on “Knit Report (R)”. As in (7), you should find a histogram at the bottom of the knitted report.

The code above is very similar in structure to the code used in (7); the only difference being that all computations and mentions of sample means have been replaced with those of t-statistics.

Again, for the sake of convenience in later computations, we first assign the name `pmean` to the mean `price_psm` of the population through the following line of code.

```
pmean = mean(resale[, "price_psm"])
```

The histogram presented in your report should look somewhat similar in shape to the one shown below.



The shape of this histogram – as well as that of the one in your report – approximates the theoretical density curve of the sampling distribution of the t-statistic for `price_psm`, for samples of size 200 drawn from our population.

(10) Overlaying a certain density curve on the histogram of (9)

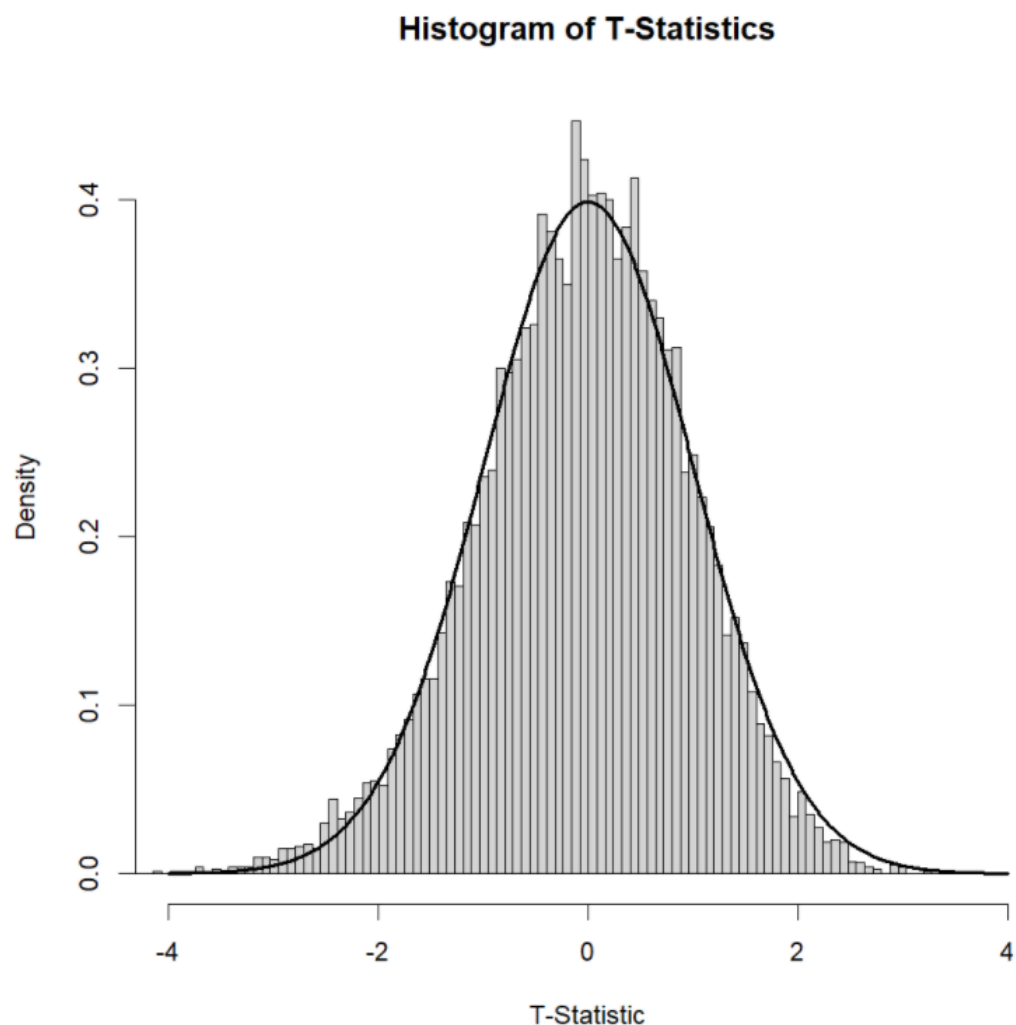
Append the following code to the code in (9).

```
x <- seq(-4, 4, by = 0.01)
td <- dt(x, df = 199)
lines(x, td, lwd = 2)
```

Click on “Knit Report (R)”. You see a curve superimposed on a histogram at the bottom of the knitted report.

The three lines of code we appended are used to plot the density curve of a particular distribution – the t-distribution with 199 degrees of freedom – over a histogram of t-statistics simulated and computed according to the code in (9).

You should find in your report, a curve overlayed onto a histogram on the same pair of axes, similar to what is shown below.



The shape of this histogram – as well as that of the one in your report – should fit rather well with the curve plotted.

Mathematically, it can be proven that the sampling distribution of the t-statistic for samples of size n (drawn from a large enough population) more or less equals the t-distribution with $n - 1$ degrees of freedom. The etymology of “degrees of freedom” is beyond the scope of this course; we need only think of it as a parameter used to uniquely identify a t-distribution, the way mean and variance are used to uniquely identify a normal distribution.

One Sample T-Test

HDB_resale_2018_sample is a sample of 200 rows randomly selected from a data set of all HDB resale transactions that occurred in year 2018. The latter will be our population data for this section.

Give all numerical answers correct to 3 decimal places.

(11) What is the mean `price_psm` of this sample?

4490.842 ✓

Now, we know that the sample gives some evidence that the mean `price_psm` of HDB resale flats sold in 2018 (i.e., the population mean `price_psm`), is less than 4500.

But is the evidence *sufficiently strong*?

In the remaining part of this section, we will run through the steps to answer this question. Specifically, we will do a one sample t-test on the `price_psm` data in HDB_resale_2018_sample.

(12) State the null and alternative hypotheses of the test.

Q72

Null = 4500

Alt < 4500

(13) If the null hypothesis in (12) is true, what can we infer about the sampling distribution of the sample statistic $\frac{\bar{x} - 4500}{s/\sqrt{200}}$, for samples of size 200 drawn from our population, where

- \bar{x} is the sample mean, and
- s is the sample standard deviation?

(Hint: refer to the paragraph that follows (10).)

t-statistic has distribution = t-distribution with deg. of freedom = 199.

Recall that p-value is “the probability of obtaining a test result at least as extreme as the result observed [in the sample], assuming the null hypothesis is true”. Here,

- a “test result” is usually a sample statistic, and
- “extreme” can be interpreted as “favourable to the alternative hypothesis”.

(14) What is the relevant test result observed in our sample?

$$t\text{-statistic: } \frac{4490.8 - 4500}{123.69/\sqrt{200}} = -0.105$$

Just as a sample mean `price_psm` less than 4500 gives evidence of the population mean `price_psm` being less than 4500, a value of $\frac{\bar{x}-4500}{s/\sqrt{200}}$ less than 0 does too. In fact, the further the value of $\frac{\bar{x}-4500}{s/\sqrt{200}}$ is below 0, the greater evidence we have that the population mean `price_psm` is less than 4500.

(15) State the **range of possible test results** “at least as extreme as the result observed”. Briefly justify your answer.

$$p \leq -0.105$$

We know that the probability that a random variable takes on a certain range of values, is equal to the area under its density curve across the said range. This means we can visualise a p-value as the area under a suitable density curve, across the range of possible test results “at least as extreme as the result observed”.

(16) Give a visual representation of the p-value in our test, as area under some density curve.

(Hints:

- Refer to (13) and (15).
- For visualisations of common density curves in Radiant, go to **Basics > Probability Tool: Probability calculator.**)

(17) Evaluate the p-value. 0.458

