Review: normal distribution

Definition

A random variable Y is normally distributed with expectation μ and variance σ^2, denoted $Y \sim \text{Normal}(\mu, \sigma^2)$, if its probability density function is

$$f(y) = (2\pi\sigma^2)^{-1/2} e^{-\frac{(y-\mu)^2}{2\sigma^2}}, \qquad y \in \mathbb{R}$$

Property: Let $a, b \in \mathbb{R}$ be known constants. If $Y \sim \text{Normal}(\mu, \sigma^2)$, then $aY + b \sim \text{Normal}(a\mu + b, a^2\sigma^2)$.

A consequence of this property is that if $Y \sim \text{Normal}(\mu, \sigma^2)$, we can equivalently express it as

$$Y = \mu + \varepsilon, \qquad \varepsilon \sim \text{Normal}(0, \sigma^2).$$

The models we will see in this course will look like the expression above: the outcome Y is equal to a deterministic part, which in this case is μ, and a random "noise" term centered at zero, which in this case is ε.

Inference

Point estimation

Let $Y_1, Y_2, \ldots, Y_n$ be independent Normal(μ, σ^2). Given a sample, how do we estimate μ and σ^2?

The method of moments, maximum likelihood, least squares, and common sense agree that we can estimate the population mean μ with the sample mean

$$\overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i.$$

There are two commonly used estimators for σ^2, the sample variance S^2 and the maximum likelihood estimator $\widehat{\sigma^2}$, defined as

$$S^2 = \frac{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}{n-1}, \qquad \widehat{\sigma^2} = \frac{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}{n}.$$

The estimators S^2 and $\widehat{\sigma^2}$ have the same numerator. The only difference is that the denominator for S^2 is $n-1$, whereas the denominator for $\widehat{\sigma^2}$ is n. In practice, they're similar so it doesn't matter much which one we use.

Confidence intervals

As statisticians, we are interested in quantifying the uncertainty in our estimations. Confidence intervals effectively do that: narrow intervals tell us we are fairly confident in our point estimation, whereas wide intervals tell us we are highly uncertain.

As you know, $100 \cdot (1-\alpha)\%$ confidence intervals have a somewhat complicated interpretation. To make things concrete, let's take $\alpha = 0.05$. Whenever we compute a 95% confidence interval for a population mean μ, it may or it may not contain μ itself (we don't know what μ actually is; that's why we're estimating it). If we go through life making 95% confidence intervals for population means μ (they don't need to be the same population mean), at the end of our career about 95% of the intervals we reported will have contained the population mean μ we were estimating.

Below, there are $100 \cdot (1-\alpha)\%$ confidence intervals for the population mean μ and variance σ^2:

$$\text{IC}_{1-\alpha}(\mu) = \overline{y} \pm \texttt{qt}(1-\alpha/2, n-1)\frac{s}{\sqrt{n}}$$
$$\text{IC}_{1-\alpha}(\sigma^2) = \left[\frac{(n-1)S^2}{\texttt{qchisq}(1-\alpha/2, n-1)}, \frac{(n-1)S^2}{\texttt{qchisq}(\alpha/2, n-1)}\right],$$

where `qt` and `qchisq` are quantile functions of the Student-t and chi-squared (χ^2) distributions; the first argument is the quantile and the second argument is the degree of freedom of the distribution.

Below, there's an example that shows how to use the `t.test` function in `R` to find a confidence interval for μ. You can check that the results we get are the same ones we'd get with the formula.

Example: The vector `x` contains 10 draws from a normal distribution with mean 5 and variance 1.

```
n = 5
x = rnorm(n, mean = 5, sd = 1)
x
```

```
## [1] 5.761782 4.733258 5.088287 7.067214 4.280525
```

The function `t.test` reports, among other things, a confidence interval for the population mean μ, given a vector `x`. We can set the confidence level with the `conf.level` argument. By default, it reports 95% confidence intervals:

```
t.test(x, conf.level = 0.99)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 11.106, df = 4, p-value = 0.000374
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  3.153234 7.619193
## sample estimates:
## mean of x
##  5.386213
```

If we want to see the confidence interval only:

```
t.test(x, conf.level = 0.99)$conf.int
```

```
## [1] 3.153234 7.619193
## attr(,"conf.level")
## [1] 0.99
```

Central limit theorem

Let $Y_1, Y_2, \ldots, Y_n$ be independent and identically distributed from a distribution with expectation $\mathbb{E}(Y_i) = \mu$ and finite variance $\text{Var}(Y_i) = \sigma^2$. The distribution can be anything – discrete, continuous, skewed, symmetric, etc. Then, the central limit theorem implies that

$$\overline{Y} = \sum_{i=1}^{n} Y_i/n \approx N(\mu, \sigma^2/n).$$

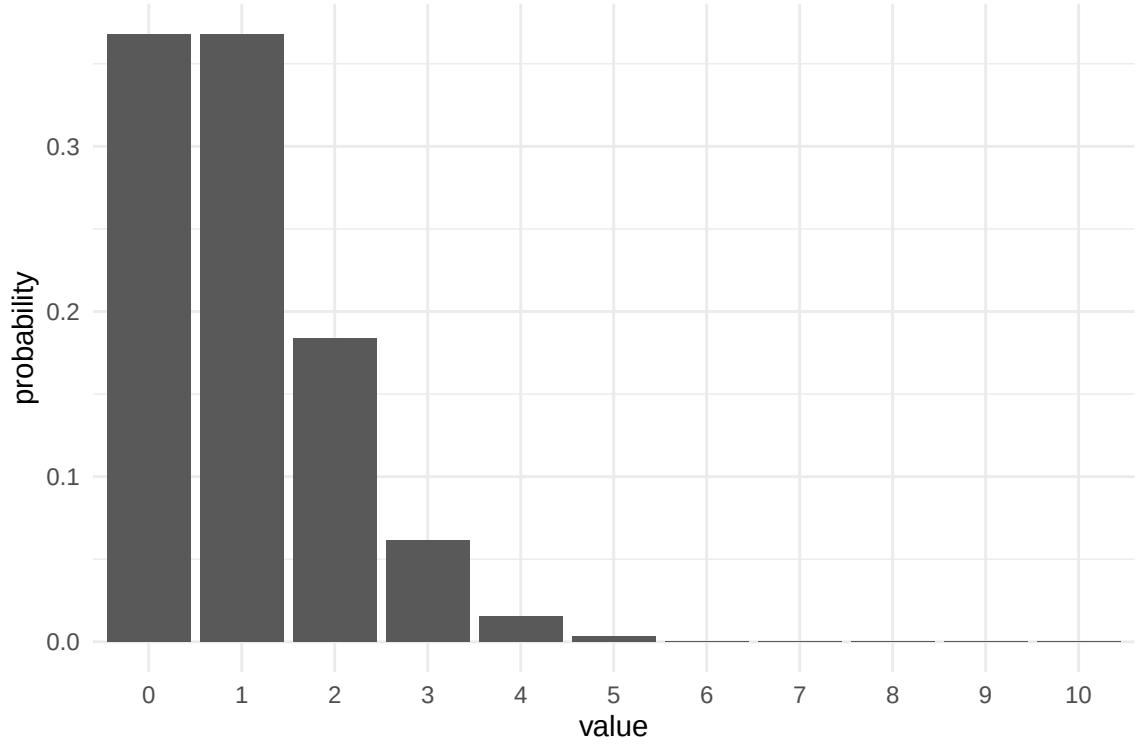
The quality of the approximation gets better as n gets large. If the distribution of the Y_i is close to normal to begin with, the approximation is good for small n. If the distribution is far from normal (for example, if it's heavily skewed) we need a larger n for the approximation to be good.

A consequence of the central limit theorem is that, if n is large enough and the conditions of the theorem hold, we can use the interval for the population mean μ we saw in the previous section even if the data are not normal to begin with.

Example Let's use the confidence interval for μ that assumes normality in an example where the truth is far from being normal. We'll take samples of size 100 from a Poisson(1) distribution, which is discrete and skewed (recall that the normal distribution is continuous and symmetric).

This is how the probability mass function of a Poisson(1) looks like. It's supported on the non-negative integers, but the probability that exceeds 10 is low:

```
library(tidyverse)
qplot(x = factor(0:10),
      y = dpois(0:10, 1),
      geom = "col") +
      xlab("value") +
      ylab("probability") +
      theme_minimal()
```



For each sample, we'll compute a 95% confidence interval for μ. We'll repeat this process 1000 times (that is, we will find 1000 intervals, each of them from a Poisson(1) sample of size 100). In this example, we know that the expectation of a Poisson(1) distribution is 1. If the interval is good, the intervals should contain 1 about 95% of the time.

```
n_inter = 1000
n = 100
# getting draws from Poisson
draws = matrix(rpois(n_inter*n, 1), nrow = n_inter, ncol = n)
# checks whether intervals contain 1
contains = apply(draws, 1, function(x) {
                  int = t.test(x)$conf.int
```

```
                  int[1] <= 1 & int[2] >= 1
                  })
# percentage of intervals that contain the truth
100*mean(contains)
```

```
## [1] 95.4
```

The percentage is pretty close to 95%. If n is bigger, the approximation will get better. Also, if we increase the parameter of the Poisson from 1 to something bigger, the approximation will get bigger because as the parameter increases, the distribution looks increasingly symmetric.

Hypothesis testing

In hypothesis testing, we usually have two hypotheses: H_0, which is called the null hypothesis, and H_1, which is called the alternative hypothesis.

When we perform a hypothesis test, we can make two types of error: rejecting H_0 when it is true (false positive, also known as type I error) and not rejecting H_0 when H_1 is true (false negative, also known as type II error). Most textbooks include the following table to illustrate this:

	Don't Reject H_0	Reject H_0
H_0 is true	OK	Type I error (false positive)
H_1 is true	Type II error (false negative)	OK

Most hypothesis tests have a fixed type I error rate, often denoted α.

The basic algorithm to perform all the tests we will see in the course is the following:

1. Fix a type I error rate α, also known as the significance level.
2. Compute the observed value t_{obs} of a test statistic T that has a known distribution when H_0 is true.
3. Compute the p-value, which can usually be interpreted as the probability that T is as "extreme" or more "extreme" than the observed value t_{obs} assuming that H_0 is true.
4. We reject H_0 if the probability that we computed in step 3 is below α. If it isn't, we don't reject H_0.

In previous courses, you have seen the t-test for testing a population mean μ. Now, we will review the two-sided t-test. The null and alternative hypotheses are

$$H_0 : \mu = \mu_0 \qquad H_1 : \mu \neq \mu_0,$$

where μ_0 is a known value. One-sided t-tests have one-sided alternative hypothesis like $H_1 : \mu \leq \mu_0$ or $H_1 : \mu \geq \mu_0$. I won't cover them here but I'll assume you know how to do them.

Given a random sample $Y_1, Y_2, \ldots, Y_n$, the test statistic is

$$T = \frac{\overline{Y} - \mu_0}{s/\sqrt{n}} \overset{H_0}{\sim} T_{n-1},$$

where $T \overset{H_0}{\sim} T_{n-1}$ means that the statistic T is distributed as Student-t with $n-1$ degrees of freedom under H_0.

Once we get data $y_1, y_2, \ldots, y_n$, we can compute the observed value of T, which we denote t_{obs}. Then, the p-value is

$$2P(T_{n-1} \geq |t_{\text{obs}}|),$$

where T_{n-1} is a Student-t with $n-1$ degrees of freedom and $|t_{\text{obs}}|$ is the absolute value of the observed statistic. If the p-value is smaller than α, we reject H_0. If it is not, we do not reject H_0.

Let's see how to do t-tests with `R`.

Example A coffee roaster sells packages whose expected weight should be $\mu = 200$g (there is a requirement on how small the population variance should be; we will ignore it for now). They sampled 10 packages at random and they want to run the hypothesis test

$$H_0 : \mu = 200\text{g} \qquad H_1 : \mu \neq 200\text{g}.$$

The significance level is $\alpha = 0.05$. If they reject H_0, they will adjust the process.

The data are stored in the vector `x` below:

```
x = c(210, 198, 201, 196, 205,
      195, 191, 204, 198, 200)
```

The function `t.test` will do the test for us:

```
t.test(x, mu = 200)
```

```
##
##  One Sample t-test
##
## data:  x
## t = -0.11513, df = 9, p-value = 0.9109
## alternative hypothesis: true mean is not equal to 200
## 95 percent confidence interval:
##  195.8702 203.7298
## sample estimates:
## mean of x
##     199.8
```

The p-value is greater than 0.05, so we do not reject H_0. If we want to run a one-sided test instead, we can change the `alternative` argument in `t.test` to `less` or `greater`.