Chapter 6
Basic Inference Methods

6.1 Introduction

Example 6.1 (Sleeping patterns of college students).

To illustrate some basic inferential methods, suppose a college instructor is interested in the sleeping patterns of students in a particular mathematics class. He has read that the recommended hours of sleep for a teenager is nine hours each night. That raises several questions:

- Is the median sleeping time for students in this course nine hours?
- If the answer to the first question is no, what proportion of students do get at least nine hours of sleep in a particular night?
- What is a reasonable estimate of the average number of hours these math students get per night?

The instructor decides to collect some data from one representative class to answer these questions. Each student is asked what time he or she got to bed the previous night, and what time he or she woke up the next morning. Based on the answers to these questions, the instructor computes the number of hours of sleep for each of 24 students in his class. The sleeping times are placed in the vector `sleep`.

```
> sleep = c(7.75, 8.5, 8, 6, 8, 6.33, 8.17, 7.75,
+ 7, 6.5, 8.75, 8, 7.5, 3, 6.25, 8.5, 9, 6.5,
+ 9, 9.5, 9, 8, 8, 9.5)
```

In the next sections, this data is analyzed to investigate the sleeping patterns of students.

J. Albert and M. Rizzo, *R by Example*, Use R, DOI 10.1007/978-1-4614-1365-3_6,

6.2 Learning About a Proportion

6.2.1 Testing and estimation problems

Let M denote the median hours of sleep for the population of students who take this math course. We are interested in testing the hypothesis H that $M = 9$ hours. This testing problem can be restated as a test of a population proportion. Let p denote the proportion of students who get at least nine hours of sleep on a particular night. If the population median is $M = 9$ hours, then the proportion $p = 0.5$. So we are interested in testing the hypothesis

$$H : p = 0.5.$$

In the event that H is rejected, one typically is interested in learning about the location of the proportion, and one constructs an interval estimate that contains p with a given confidence.

6.2.2 Creating group variables by the `ifelse` function

The relevant data for this hypothesis test is the sample size and the number of students in the sample who get at least nine hours of sleep. Using the `ifelse` function, we define a new variable `nine.hours` that records for each observation if the student got at least nine hours of sleep ("yes") or didn't ("no"). Then we tabulate this "yes, no" data by the `table` function.

```
> nine.hours = ifelse(sleep >= 9, "yes", "no")
> table(nine.hours)
nine.hours
 no yes
 19   5
```

Only five out of 24 students indicated that they had at least nine hours of sleep. If H is true, the number of yes's has a binomial$(n = 24, p = 0.5)$ distribution with mean np and variance $np(1-p)$. In addition, if n is large, this variable is approximately normally distributed.

6.2.3 Large-sample test and estimation methods

The traditional test for a proportion is based on the assumption that, when the population proportion is $p = 0.5$, the number of yes's y in a sample of n is approximately normally distributed with mean $n/2$ and standard deviation $\sqrt{n/4}$. The Z statistic

$$Z = \frac{y - np}{\sqrt{np(1-p)}},$$

is approximately standard normal. One computes the statistic z_{obs} from the sample and one decides whether to accept or reject H by computing the lower tail probability $P(Z \le z_{obs})$. If the alternative hypothesis is that $p < 0.5$, the p-value is equal to the lower tail probability; if the alternative is two-sided where $p \neq 0.5$, the p-value is double the lower-tail probability.

This traditional Z test is implemented using the `prop.test` function. We first define `y` to be the number of yes's and `n` to be the sample size. In the `prop.test` function, we indicate by the `p=0.5` argument that we are testing the hypothesis that the proportion is equal to 0.5, and `correct=FALSE` indicates that no continuity correction is used in the calculation of the Z statistic. The summary of this test is displayed by printing the variable `Test`.

```
> y = 5; n = 24
> Test = prop.test(y, n, p=0.5, alternative="two.sided",
+   correct=FALSE)
> Test
        1-sample proportions test without continuity correction

data:  y out of n, null probability 0.5
X-squared = 8.1667, df = 1, p-value = 0.004267
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.09244825 0.40470453
sample estimates:
        p
0.2083333
```

The variable `Test` contains all of the calculations of the test and we request components of `Test` to obtain specific quantities of interest. A vector of the names of components is obtained using the `names` function.

```
> names(Test)
[1] "statistic"   "parameter"   "p.value"     "estimate"    "null.value"
[6] "conf.int"    "alternative" "method"      "data.name"
```

The estimate of the proportion p is the sample proportion of students y/n obtained by asking for the component `estimate`.

```
> Test$estimate
        p
0.2083333
```

The component `statistic` gives the value of the chi-square statistic z_{obs}^2 (the square of the observed Z statistic) and `p.value` gives the associated p-value. It is a two-sided p-value since we indicated the alternative was two-sided.

```
> Test$statistic
X-squared
 8.166667
> Test$p.value
[1] 0.004266725
```

Since the p-value is close to zero, we have strong evidence to say that the proportion of "nine hours or greater" sleepers among the students is not 0.5.

In the case where the hypothesis $p = 0.5$ is rejected, a next step is to estimate the proportion by a confidence interval. The component `conf.int` displays a 95% confidence interval. This particular interval, the Wilson score interval, is found by inverting the score test for the proportion.

```
> Test$conf.int
[1] 0.09244825 0.40470453
attr(,"conf.level")
[1] 0.95
```

We are 95% confident that the interval (0.092, 0.405) contains the proportion of heavy sleepers.

6.2.4 Small sample methods

One problem with the traditional inference method is that the Z statistic is assumed to be normally distributed, and the accuracy of this normal approximation can be poor for small samples. So there are several alternative inferential methods for a proportion that have better sampling properties when the sample size n is small.

One "small-sample" method is to adjust the Z for the fact that y is a discrete variable. In our example, the "continuity-adjusted" Z statistic for testing the hypothesis $H : p = 0.5$ is based on the statistic

$$Z_{adj} = \frac{y + 0.5 - np}{\sqrt{np(1-p)}}.$$

This test is implemented using the `prop.test` function with the `correct=TRUE` argument.

```
> y = 5; n = 24
> Test.adj = prop.test(y, n, p=0.5, alternative="two.sided",
+   correct=TRUE)
> c(Test.adj$stat, p.value=Test.adj$p.value)
 X-squared    p.value
7.04166667 0.00796349
```

Note that we obtain slightly different values of the chi-square test statistic Z^2 and associated p-value. The result of 5 successes in 24 trials is slightly less significant using this test.

A second alternative testing method is based on the underlying exact binomial distribution. Under the hypothesis $H : p = 0.5$, the number of successes y has a binomial distribution with parameters $n = 24$ and $p = 0.5$ and the exact (two-sided) p-value is given by

$$2 \times P(y \leq 5 | p = 0.5).$$

This procedure is implemented using the function `binom.test`. The inputs are the number of successes, the sample size, and the value of the proportion under the null hypothesis.

```
> Test.exact = binom.test(y, n, p=0.5)
> c(Test.exact$stat, p.value=Test.exact$p.value)
number of successes             p.value
        5.000000000         0.006610751
```

One can check the computation of the p-value using the binomial cumulative distribution function `pbinom`. The probability that y is at most 5 is given by `pbinom(5, size=24, prob=0.5)` and so the exact p-value is given by

```
> 2 * pbinom(5, size=24, prob=0.5)
[1] 0.006610751
```

which agrees with the output of `binom.test`. One can also obtain the "exact" 95% Clopper-Pearson confidence interval by displaying the component `conf.int`.

```
> Test.exact$conf.int
[1] 0.07131862 0.42151284
attr(,"conf.level")
[1] 0.95
```

This particular confidence interval is guaranteed to have 95% coverage, but it can be a bit longer than other computed "95% intervals."

A third popular "small-sample" confidence interval was developed by Agresti and Coull [2]. A 95% interval is constructed by simply adding two successes and two failures to the dataset and then using the simple formula

$$\tilde{p} - 1.96se, \tilde{p} + 1.96se,$$

where $\tilde{p} = (y+2)/(n+4)$ and se is the usual standard error based on the modified data $se = \sqrt{\tilde{p}(1-\tilde{p})/(n+4)}$. There is no R function in the base package that computes the Agresti-Coull interval, but it is straightforward to write a function to compute this interval. In the user-defined function `agresti.interval` below, the inputs are the number of successes `y`, the sample size `n`, and the confidence level.

```
agresti.interval = function(y, n, conf=0.95){
  n1 = n + 4
  y1 = y + 2
  phat = y1 / n1
  me = qnorm(1 - (1 - conf) / 2) * sqrt(phat * (1 - phat) / n1)
  c(phat - me, phat + me)
}
```

After this function has been read into R, one computes the interval for our data by typing

```
> agresti.interval(y, n)
[1] 0.0896128 0.4103872
```

(The function `add4ci` in the `PropCIs` package will also compute the Agresti-Coull interval.)

We have illustrated three methods for constructing a 95% interval estimate for a proportion. In the following code, we create a data frame in R that gives the method, the function for implementing the method, and the lower and upper bounds for the interval.

```
> cnames = c("Wilson Score Interval", "Clopper-Pearson",
+ "Agresti-Coull")
> cfunctions = c("prop.test", "binom.test", "agresti.interval")
> intervals = rbind(Test$conf.int, Test.exact$conf.int,
+   agresti.interval(y, n))
> data.frame(Name=cnames, Function=cfunctions,
+   LO=intervals[ , 1], HI=intervals[ , 2])
                  Name         Function         LO        HI
1 Wilson Score Interval        prop.test 0.09244825 0.4047045
2       Clopper-Pearson       binom.test 0.07131862 0.4215128
3         Agresti-Coull agresti.interval 0.08961280 0.4103872
```

In terms of interval length, the shortest interval is the Wilson score interval, followed in order by the Agresti-Coull interval and the Clopper-Pearson interval. Although it is desirable to have short intervals, one wants the interval to have the stated 95% coverage probability. In Chapter 13, we will explore the coverage probability of these three procedures.

6.3 Learning About a Mean

6.3.1 Introduction

In the previous section, we focused on the proportion of students who got at least nine hours of sleep and learned that this proportion was quite small. Next it is reasonable to return to the original collection of observed sleeping times and learn about the mean μ of the population of sleeping times for all college students.

6.3.2 One-sample t statistic methods

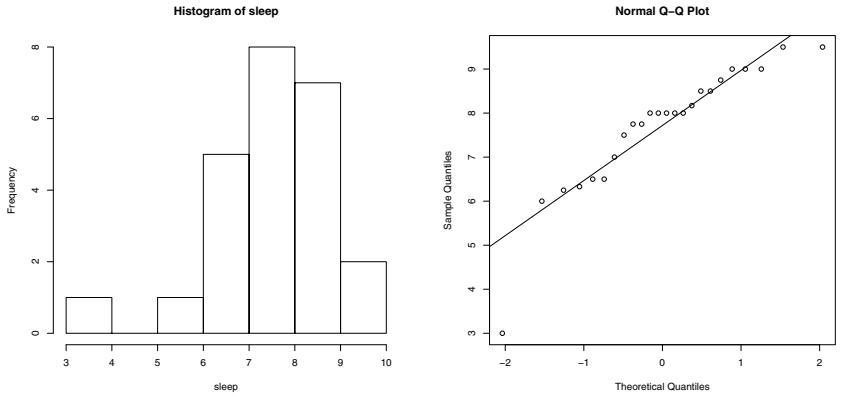
The R function `t.test` performs the calculations for the traditional inference procedures for a population mean. If the observations represent a random sample from a normal population with unknown mean μ, the statistic

$$T = \frac{\sqrt{n}(\bar{y} - \mu)}{s}$$

has a t distribution with $n - 1$ degrees of freedom, where $\bar{y}$, s, and n are respectively the sample mean, sample standard deviation, and sample size.

To illustrate this function, suppose we wish to test the hypothesis that the mean sleeping time is 8 hours and also construct a 90% interval estimate for the population mean. Before we use `t.test`, we should check if it is reasonable to assume the sleeping times are normally distributed. A histogram of the times is constructed by the `hist` function. The `qqnorm` function produces a normal probability plot of the times and the `qqline` function overlays a line on the plot passing through the first and third quartiles. The resulting graphs are displayed in Figure 6.1.

```
> hist(sleep)
> qqnorm(sleep)
> qqline(sleep)
```



(a) (b)

Fig. 6.1 Histogram and normal probability plot of the sleeping times of college students.

Looking at the graphs in Figure 6.1, it is clear that there is one unusually small sleeping time (about 3 hours) that is not consistent with the normal distribution assumption. There are several ways one could handle this problem. If there was some mistake in the recording of this particular time, one could remove the outlier and apply the t methods to the modified data. Alternatively, one could apply a different inferential procedure that rests on a more general assumption about the distribution of the population. Here we will illustrate both approaches.

We identify the position of the outlier by constructing an index plot of the sleeping times by the `plot` function. From the display in Figure 6.2, it is clear that the 14th observation is the outlier, and in further examination, we find that this observation was incorrectly coded. A new data vector `sleep.new` is defined that is the original dataset with the 14th observation deleted.

```
> plot(sleep)
> sleep.new = sleep[-14]
```

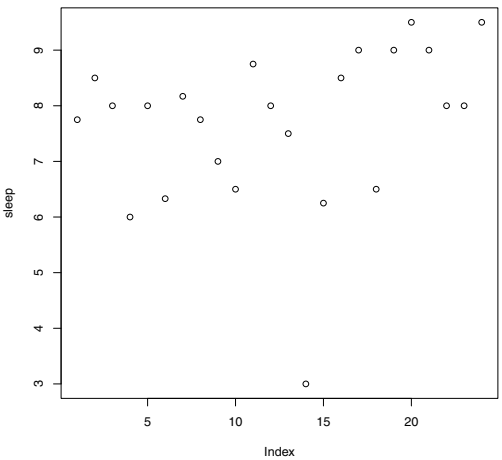


Fig. 6.2 Index plot of the sleeping times. This helps identify the outlier as the 14th observation.

With the outlier removed, the assumption that the data comes from a normal population seems reasonable, and we can apply the procedures based on the t distribution. The general form of the `t.test` function is given by

```
t.test(x, y=NULL,
       alternative=c("two.sided", "less", "greater"),
       mu=0, paired=FALSE, var.equal=FALSE,
       conf.level=0.95, ...)
```

For single sample inference, the data is contained in the vector `x`. We indicate by the `mu` argument what value of the population mean is to be tested and the `alternative` argument indicates if the alternative hypothesis is two-sided, less than the population mean value, or greater than the mean value. Also the `conf.level` argument indicates the confidence interval for our interval estimate. In our example, we are interested in testing the hypothesis $\mu =$ 8 hours, the alternative is two-sided (the default value), and we wish to construct a 90% interval. The form of the `t.test` function is as follows.

```
> t.test(sleep.new, mu=8, conf.level=0.90)
        One Sample t-test

data:  sleep.new
t = -0.4986, df = 22, p-value = 0.623
alternative hypothesis: true mean is not equal to 8
90 percent confidence interval:
 7.516975 8.265633
sample estimates:
mean of x
 7.891304
```

The output gives the value of the t-test statistic and the two-sided p-value for testing the hypothesis that $\mu = 8$. Since the p-value is large, there is insufficient evidence from the data to conclude the mean sleeping time of students is not equal to 8 hours. From a frequentist perspective, we are 90% confident that the interval (7.52, 8.27) contains the mean μ.

6.3.3 Nonparametric methods

If we wish to use the complete sleeping dataset, it would be inappropriate to use the t procedures due to the single outlier. But there are alternative inferential procedures we can use based on less restrictive assumptions about the population of sleeping times. The Wilcoxon signed rank procedure makes the general assumption that the population is symmetric about a median M. In this setting, if one wishes to test the hypothesis that the median sleeping time $M = 8$ hours, then the test statistic is obtained by ranking the absolute values of the differences of each sample value from 8, and then computing the sum of the ranks of the differences that are positive. If the actual median is different from 8, then the sum of ranks corresponding to the positive differences will be unusually small or large, so one rejects the hypothesis in the tail region of the null distribution of the Wilcoxon statistic.

The Wilcoxon signed rank method is implemented as `wilcox.test` with the following general syntax.

```
wilcox.test(x, y=NULL,
            alternative=c("two.sided", "less", "greater"),
            mu=0, paired=FALSE, exact=NULL, correct=TRUE,
            conf.int=FALSE, conf.level=0.95, ...)
```

The arguments are similar to those in `t.test`. The vector `x` is the sample of observations, the constant `mu` is the value to be tested, and the `alternative` argument indicates the direction of the alternative hypothesis. By indicating `conf.int = TRUE`, one can have the function compute the Wilcoxon signed-rank interval estimate for the median and `conf.level` indicates the desired probability of coverage.

We can test the hypothesis $M = 8$ with a two-sided alternative and obtain a 90% interval estimate for the median of sleeping times (with the original dataset) using the command

```
> W = wilcox.test(sleep, mu=8, conf.int=TRUE, conf.level=0.90)
Warning messages:
1: In wilcox.test.default(sleep, mu = 8, conf.int = TRUE, conf.level = 0.9) :
  cannot compute exact $p$-value with ties

> W
        Wilcoxon signed rank test with continuity correction

data:  sleep
V = 73.5, p-value = 0.3969
alternative hypothesis: true location is not equal to 8
90 percent confidence interval:
 7.124979 8.374997
sample estimates:
(pseudo)median
      7.749961
```

We see that a warning message is produced when we execute this function. For small samples (such as this one), the `wilcox.test` will compute exact p-values and interval estimates, but these exact methods cannot be used when there are ties in the dataset. When there are ties, as in this sample of sleeping times, the function will give p-values and interval estimates based on a normal approximation to the signed-rank statistic.

Using the `names` function, one can see the names of the components of the object producted by the `wilcox.test` function.

```
> names(W)
[1] "statistic"   "parameter"   "p.value"     "null.value"  "alternative"
[6] "method"      "data.name"   "conf.int"    "estimate"
```

The `statistic` component gives the value of the Wilcoxon test statistic, the `p.value` component gives the (two-sided, in this case) p-value and the `conf.int` component contains the interval estimate.

```
> W$statistic
   V
73.5
> W$p.value
[1] 0.3968656
> W$conf.int
[1] 7.124979 8.374997
attr(,"conf.level")
[1] 0.9
```

Comparing with the output from `t.test`, both the t and Wilcoxon methods indicate insufficient evidence that the "average" sleeping time from the population is not 8 hours. The 90% Wilcoxon interval estimate for the population median is wider than the t interval estimate for the population mean.

6.4 Two Sample Inference

6.4.1 Introduction

Example 6.2 (The twins dataset (continued)).

A basic inferential problem is to compare the locations of two continuous-valued populations. To illustrate different "two-sample" methods, we consider data from Ashenfelter and Krueger [3] who were interested in relating people's education and income. It can be difficult to learn about the effect of education on income since there are many variables associated with income, such as a person's natural ability, his family background, and his innate intelligence. To control for possible confounding variables, the authors collected information on education, income, and background from a group of twins. Since twins have similar family backgrounds, they provide a useful control for confounding variables in this problem. (This particular dataset was previously used in Chapter 3 to illustrate statistical methods for categorical data.)

The datafile `twins.txt` contains information about 183 pairs of twins for sixteen variables. We read the data into R by the `read.table` function and store the data frame in the variable `twins`.

```
> twins = read.table("twins.txt", header=TRUE)
```

Each pair of twins was randomly assigned the labels "twin 1" and "twin 2." The variable `HRWAGEH` gives the hourly wage for twin 2. If one graphs the wages, one finds that they are strongly right-skewed and one can remove the skewness by a log transformation; the variable `log.wages` contains the log wages.

```
> log.wages = log(twins$HRWAGEH)
```

6.4.2 Two sample t-test

The variable `EDUCH` contains the self-reported education of twin 2 in years. Suppose one is interested in comparing the log wages of the "high school" twins with 12 or fewer years of education with the log wages of the "college" twins with more than 12 years of education. We define a new categorical variable `college` using the `ifelse` function that is "yes" or "no" depending on the years of education.

```
> college = ifelse(twins$EDUCH > 12, "yes", "no")
```

A first step in comparing the log wages of the two groups is to construct a suitable graph and Figure 6.3 displays parallel boxplots of the log wages using the `boxplot` function. Both groups of log wages look approximately symmetric with similar spreads and the median log wage of the college twins

appears approximately 0.5 larger than the median log wage of the high school twins.

```
> boxplot(log.wages ~ college, horizontal=TRUE,
+   names=c("High School", "Some College"), xlab="log Wage")
```

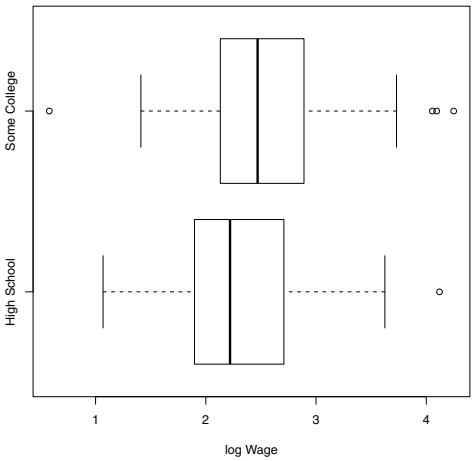


Fig. 6.3 Parallel boxplots of the log wages for the high school and college twins.

Let μ_H and μ_C denote respectively the mean log wage of the population of high school twins and college twins. The standard t-test of the hypothesis H that $\mu_H = \mu_C$ is implemented by the function `t.test`. The argument has the form `log.wages` $\sim$ `college` where the `log.wages` is the continuous response variable and `college` categorizes the response into two groups.

```
> t.test(log.wages ~ college)
        Welch Two Sample t-test

data:  hs.log.wages and college.log.wages
t = -2.4545, df = 131.24, p-value = 0.01542
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.42999633 -0.04620214
sample estimates:
 mean in group no mean in group yes
         2.282119          2.520218
```

From the output, we see that:

- The value of the t-test statistic is -2.4545.

- Using the Welch procedure for unequal population variances, the t statistic is approximately distributed as t with 131.24 degrees of freedom under the assumption of equal population means.
- The two-sided p-value is 0.01542, so there is significant evidence that the means are different.
- The 95% confidence interval for the difference in means $\mu_H - \mu_C$ is $(-0.430, -0.046)$.

The traditional t-test for the difference of means assumes the population variances are equal. One can implement this traditional test using the `var.equal=TRUE` argument.

```
> t.test(log.wages ~ college, var.equal=TRUE)$p.value
[1] 0.01907047
```

In this example, the Welch test and the traditional t-test give approximately the same p-value.

6.4.3 Two sample Mann-Whitney-Wilcoxon test

A nonparametric alternative to the two-sample test is the Mann-Whitney-Wilcoxon test. One tests the general hypothesis that two independent samples come from the same continuous population. If one denotes the two samples as x and y, the test statistic is equal to

$$W = \text{the number of pairs } (\mathrm{x_i}, \mathrm{y_j}) \text{ where } \mathrm{x_i} > \mathrm{y_j}.$$

This method is implemented using the `wilcox.test` function with the same argument format (response by grouping variable) as the `t.test` function.

```
> wilcox.test(log.wages ~ college, conf.int=TRUE)
        Wilcoxon rank sum test with continuity correction

data:  hs.log.wages and college.log.wages
W = 2264, p-value = 0.01093
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -0.44266384 -0.06455011
sample estimates:
difference in location
            -0.2575775
```

The value of the W statistic for our dataset is 2264. The two-sided probability of getting a value as extreme as 2264 if the two samples come from the same continuous population is 0.01093. This p-value is very close to the value obtained using the two-sample t-test.

By specifying the argument option `conf.int = TRUE`, `wilcox.test` will also display a 95% confidence interval for the difference of location parameters of the two populations. Comparing the output of `wilcox.test` with the output of `t.test`, we see this confidence interval is similar to the interval for the difference of population means.

6.4.4 Permutation test

Another testing procedure for comparing two independent samples is a permutation test. As before, we define a variable `log.wages` that contains the log wages for the first twin and a vector `college` that indicates if the first twin had some college education or not.

```
> log.wages = log(twins$HRWAGEH)
> college = ifelse(twins$EDUCH > 12, "yes", "no")
```

Using the `table` function, we see that there are 112 college and 71 no-college twins in our sample.

```
> table(college)
college
 no yes
 71 112
```

Consider the hypothesis that college education has no impact on the wages of the twins. Under this hypothesis, the labeling of the twins into the college and no-college categories is not helpful in understanding the variability of the log wages. In this case, the distribution of any test statistic, say the t-test, will be unchanged if we arbitrarily change the labels of the twins in the college/no-college classification. One obtains an empirical distribution of the test statistic under the null hypothesis by

- Randomly allocating the 71 college and 112 no-college labels among the 183 twins.
- Computing the value of the test statistic for the randomly permuted data.
- Repeating this process a large number of iterations.

The collection of test statistics provides an estimate of the sampling distribution of the statistic when the null hypothesis is true. (Figure 6.4 is a picture of this sampling distribution.) The observed value of the test statistic (for our original sample) is then compared to the distribution of the permutation replicates. To make this comparison, one computes the probability that the test statistic under the randomization distribution is at least as extreme as the observed statistic. If this p-value is sufficiently small, this gives evidence against the assumption that the labeling of the twins into the two groups is not informative.

This testing procedure is straightforward to program by writing a short function. The function `resample` is written which randomly permutes the college labels (using the `sample` function) and returns the value of the t-test statistic from the `t.test` function.

```
> resample = function()
+   t.test(log.wages ~ sample(college))$statistic
```

One repeats `resample` using the `replicate` function. There are two arguments to `replicate`, the number of iterations and the name of the function to be replicated. The values of the t statistic from the 1000 iterations is stored in the vector `many.T`.

```
> many.T = replicate(1000, resample())
```

The value of the t statistic from the observed data is obtained by the `t.test` function using the labels from the vector `college`.

```
> T.obs = t.test(log.wages ~ college)$statistic
> T.obs
        t
-2.454488
```

To see if the observed test statistic of -2.45 is extreme, we use the `hist` function to construct a histogram of the t statistics from the randomized distribution in Figure 6.4. We use the `abline` function to add a vertical line at the observed t statistic.

```
> hist(many.T)
> abline(v=T.obs)
```

The (two-sided) p-value is twice the probability of obtaining a t statistic smaller than `T.obs`.

```
> 2 * mean(many.T < T.obs)
[1] 0.024
```

This computed p-value from the permutation test is similar to the values computed using the t-test and Wilcoxon procedures.

6.5 Paired Sample Inference Using a t Statistic

In the above analysis, we were comparing the wages of two groups of people with different educational levels. It is difficult to get precise estimates of the effect of education on wage since there are many other confounding variables such as family background, natural ability, and innate intelligence that may also explain the differences between the two groups. This particular study took a sample of twins. By considering the wages of twins who differ in educational level but are similar with respect to other important variables such

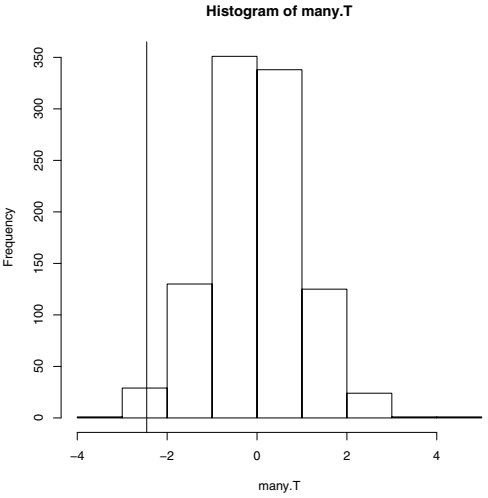


Fig. 6.4 Simulated randomization distribution of the two sample t statistic under the null hypothesis in the permutation test. The vertical line gives the location of the observed t statistic.

as intelligence and family background, one might more accurately estimate the effect of education. Here we illustrate methods of comparing two means from paired data.

In the dataframe `twins`, the variables `EDUCL` and `EDUCH` give the educational levels for twin 1 and twin 2. We first create a new data frame consisting only of the twins with different educational levels. This new data frame `twins.diff` is created using the `subset` function.

```
> twins.diff = subset(twins, EDUCL != EDUCH)
```

(The `!=` symbol means "not equal.") Since there are twins with missing values of educational level, we use the `complete.cases` function to remove any rows from the dataframe where any of the variables contain a `NA` code.

```
> twins.diff = twins.diff[complete.cases(twins.diff), ]
```

For these twins with different educational levels, we let `log.wages.low` be the log hourly wage for the twin with the lower educational level and `log.wages.high` the log hourly wage for the twin with the higher educational level. We compute these new variables using of the `ifelse` function. For example, if the condition `EDUCL` $<$ `EDUCH` is true, then `log.wages.low` will be equal to `log(HRWAGEL)`; otherwise `log.wages.low` will be equal to `log(HRWAGEH)`. A similar conditional expression is used to compute the variable `log.wages.high`.

```
> log.wages.low = with(twins.diff,
+   ifelse(EDUCL < EDUCH, log(HRWAGEL), log(HRWAGEH)))
```

```
> log.wages.high = with(twins.diff,
 +  ifelse(EDUCL < EDUCH, log(HRWAGEH), log(HRWAGEL)))
```

When we are done with this data cleaning, we have data on log wages for 75 pairs of twins. We combine the twins data by the `cbind` function and display the log wages for the first six pairs of twins using of the `head` function.

```
> head(cbind(log.wages.low, log.wages.high))
  log.wages.low log.wages.high
1      2.169054       2.890372
2      3.555348       2.032088
3      2.484907       2.708050
4      2.847812       2.796061
5      2.748872       3.218876
6      2.079442       2.708050
```

Let μ_L and μ_H denote respectively the mean log wages for the twins with the lower and higher educational levels. Due to the paired design, one can perform a test for the difference in means $d = \mu_L - \mu_H$ by working with the single sample of paired differences `log.wages.low - log.wages.high`. We use the `hist` function to construct a histogram of the paired differences in Figure 6.5. Since the paired differences in log wages look approximately

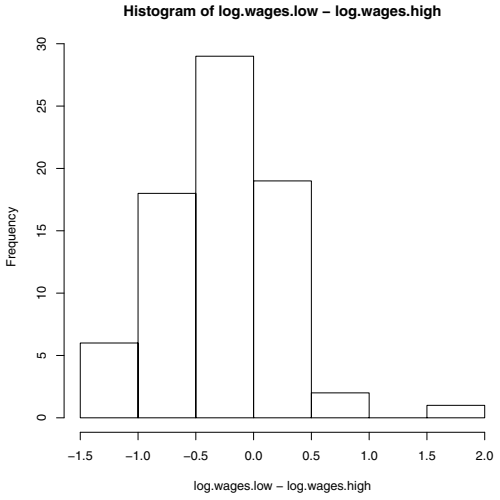


Fig. 6.5 Histogram of the paired differences to see the effect of education on log wages.

normal, it is reasonable to apply a t-test on the differences by the `t.test` function. To construct a test based on the paired differences, the argument option `paired = TRUE` is used.

```
> t.test(log.wages.low, log.wages.high, paired=TRUE)
        Paired t-test

data:  log.wages.low and log.wages.high
t = -4.5516, df = 74, p-value = 2.047e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3930587 -0.1537032
sample estimates:
mean of the differences
             -0.2733810
```

Here the observed difference in means is clearly statistically significant and a 95% confidence interval for the difference is $(-0.393, -0.154)$. In an exercise, one will be asked to rerun the `t.test` function on this dataset without using the `paired = TRUE` option and comment on the difference in the confidence intervals for the difference in means using the two options.

Exercises

6.1 (Gender of marathoners). In 2000, the proportion of females who competed in marathons in the United States was 0.375. One wonders if the proportion of female marathoners has changed in the ten-year period from 2000 to 2010. One collects the genders of 276 people who competed in the 2010 New York City Marathon – in this sample, 120 were women.

a. If p denotes the proportion of 2010 marathoners who are female, use the `prop.test` function to test the hypothesis that $p = 0.375$. Store the calculations of the test in the variable `Test`.
b. From the components of `Test`, construct a 95% interval estimate for p.
c. Using the function `binom.test`, construct an exact-test of the hypothesis. Compare this test with the large-sample test used in part (a).

6.2 (Ages of marathoners). The datafile "nyc.marathon.txt" contains the gender, age, and completion time (in minutes) for 276 people who completed the 2010 New York City Marathon. It was reported that the mean ages of men and women marathoners in 2005 were respectively 40.5 and 36.1.

a. Create a new dataframe "women.marathon" that contains the ages and completion times for the women marathoners.
b. Use the `t.test` function to construct a test of the hypothesis that the mean age of women marathoners is equal to 36.1.
c. As an alternative method, use the `wilcox.test` function to test the hypothesis that the median age of women marathoners is equal to 36.1. Compare this test with the t-test used in part (b).
d. Construct a 90% interval estimate for the mean age of women marathoners.

6.3 (Ages of marathoners, continued). From the information in the 2005 report, one may believe that men marathoners tend to be older than women marathons.

a. Use the `t.test` function to construct a test of the hypothesis that the mean ages of women and men marathoners are equal against the alternative hypothesis that the mean age of men is larger.
b. Construct a 90% interval estimate for the difference in mean ages of men and women marathoners.
c. Use the alternative Mann-Whitney-Wilcoxon test (function `wilcox.test`) to test the hypothesis that the ages of the men and ages of the women come from populations with the same location parameter against the alternative that the population of ages of the men have a larger location parameter. Compare the result of this test with the t-test performed in part (a).

6.4 (Measuring the length of a string). An experiment was performed in an introductory statistics class to illustrate the concept of measurement bias. The instructor held up a string in front of the class and each student guessed at the string's length. The following are the measurements from the 24 students (in inches).

```
22 18 27 23 24 15 26 22 24 25 24 18
18 26 20 24 27 16 30 22 17 18 22 26
```

a. Use the `scan` function to enter these measurements into R.
b. The true length of the string was 26 inches. Assuming that this sample of measurements represents a random sample from a population of student measurements, use the `t.test` function to test the hypothesis that the mean measurement μ is different from 26 inches.
c. Use the `t.test` function to find a 90% confidence interval for the population mean μ.
d. The t-test procedure assumes the sample is from a population that is normally distributed. Construct a normal probability plot of the measurements and decide if the assumption of normality is reasonable.

6.5 (Comparing snowfall of Buffalo and Cleveland). The datafile "buffalo.cleveland.snowfall.txt" contains the total snowfall in inches for the cities Buffalo and Cleveland for the seasons 1968-69 through 2008-09.

a. Compute the differences between the Buffalo snowfall and the Cleveland snowfall for all seasons.
b. Using the `t.test` function with the difference data, test the hypothesis that Buffalo and Cleveland get, on average, the same total snowfall in a season.
c. Use the `t.test` function to construct a 95% confidence interval of the mean difference in seasonal snowfall.

6.6 (Comparing Etruscan and modern Italian skulls). Researchers were interested if ancient Etruscans were native to Italy. The dataset "Etruscan-Italian.txt" contains the skull measurements from a group of Etruscans and modern Italians. There are two relevant variables in the dataset: `x` is the skull measurement and `group` is the type of skull.

a. Assuming that the data represent independent samples from normal distributions, use the `t.test` function to test the hypothesis that the mean Etruscan skull measurement μ_E is equal to the mean Italian skull measurement μ_I.
b. Use the `t.test` function to construct a 95% interval estimate for the difference in means $\mu_E - \mu_I$.
c. Use the two-sample Wilcoxon procedure implemented in the function `wilcox.test` to find an alternative 95% interval estimate for the difference $\mu_E - \mu_I$.

6.7 (President's heights). In Example 1.2, the height of the election winner and loser were collected for the U.S. Presidential elections of 1948 through 2008. Suppose you are interested in testing the hypothesis that the mean height of the election winner is equal to the mean height of the election loser. Assuming that this data represent paired data from a hypothetical population of elections, use the `t.test` function to test this hypothesis. Interpret the results of this test.