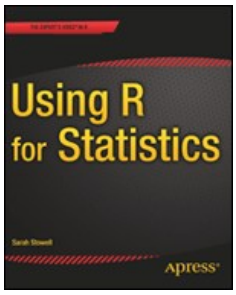

Using R for Statistics

by Sarah Stowell

Chapter 5: Summary Statistics for Continuous Variables

Overview

A good place to begin analyzing your data is with some simple summary statistics. This chapter explains how to calculate summary statistics for continuous variables.

Two main types of summary statistics are univariate summary statistics and measures of association. Univariate statistics are those that are calculated from a single variable. This includes measures of location such as the mean and median, and measures of dispersion such as the variance, standard deviation and range. Measures of association summarise the relationship between two variables, and include the covariance, Pearson's correlation, and Spearman's correlation.

Also covered in this chapter are methods that help you to make inferences about the population from which a sample is drawn. These include the the Shapiro-Wilk and Kolmogorov-Smirnov tests, and confidence and prediction intervals.

You will learn how to:

- calculate univariate statistics
- calculate statistics for different groups of observations
- calculate the covariance, Pearson's correlation, and Spearman's correlation between two variables
- perform a hypothesis test to check whether a correlation is statistically significant
- perform the Shapiro-Wilk and Kolmogorov-Smirnov tests
- calculate confidence and prediction intervals

This chapter uses the trees, `iris`, `warpbreaks`, and `PlantGrowth` datasets, which are included with `R`, and the `bottles` dataset, which is available with the downloads for this book. It is helpful to become familiar with them before beginning the chapter. For the datasets included with R, you can view additional information about them by entering *help(datasetname)*. For more information about the `bottles` dataset, see Appendix C.

Univariate Statistics

To produce a summary of all the variables in a dataset, use the `summary` function. The function summarizes each variable in a manner suitable for its class. For numeric variables, it gives the mean, median, range, and interquartile range. For factor variables, it gives the number in each category. If a variable has any missing values, it will tell you how many.

```
> summary(iris)
```

```
 Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
Min. : 4.300   Min.   :2.000   Min.   :1.000   Min. :0.100   setosa    :50
1st Qu.:5.100  1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
Median :5.800  Median :3.000   Median :4.350   Median :1.300   virginica :50
Mean   :5.843  Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400  3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900  Max.   :4.400   Max.   :6.900   Max.   :2.500
```

To calculate a particular statistic for a single variable, use the relevant function from Table 5-1.

Table 5-1: Functions for Summarizing Continuous Variables; Those Marked with an Asterisk Give a Single Value as Output

Statistic	Function
Mean*	`mean`
Median*	`median`
Standard deviation*	`sd`
Median absolute deviate*	`mad`
Variance*	`var`
Maximum value*	`max`
Minimum value*	`min`
Interquartile range*	`IQR`

Range	`range`
Quantiles	`quantile`
Tukey five-number summary	`fivenum`
Sum*	`sum`
Product*	`prod`
Number of observations*	`length`

For example, to calculate the mean tree height, use the command:

```
> mean(trees$Height)
```

```
[1] 76
```

If the variable has any missing data values, set the `na.rm` argument to `T` as shown below. This tells R to ignore any missing values when calculating the statistic. Otherwise, the result will be either another missing value or an error message, depending on the function:

```
> mean(dataset$variable, na.rm=T)
```

To calculate a particular statistic for each of the variables in a dataset simultaneously, use the `sapply` function with any of the statistics in Table 5-1:

```
> sapply(trees, mean)
```

```
   Girth   Height   Volume

13.24839 76.00000 30.17097
```

Again, if the dataset has any missing values then set the `na.rm` argument to `T`:

```
> sapply(dataset, mean, na.rm=T)
```

If any of the variables in your dataset are nonnumeric, the `sapply` function behaves inconsistently. For example, this command attempts to calculate the maximum value for each of the variables in the `iris` dataset. R returns an error message because the fifth variable in the dataset is a factor variable:

```
> sapply(iris, max)
```

```
Error in Summary.factor(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, :

max not meaningful for factors
```

To avoid this problem, exclude any nonnumeric variables from the dataset by using bracket notation or the `subset` function, as described in Chapters 1 (under "Data Frames") and 3 (under "Selecting a Subset of the Data"):

```
> sapply(iris[-5], max)
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
         7.9          4.4          6.9          2.5
```

Statistics by Group

You may want to group the values of a numeric variable according to the levels of a factor and calculate a statistic for each group. There are two functions that allow you to do this, called `tapply` and `aggregate`.

You can use the `tapply` function with any of the statistics in Table 5-1. For example, to calculate the mean sepal width for each species for the `iris` dataset:

```
> tapply(iris$Sepal.Width, iris$Species, mean)
```

```
setosa  versicolor  virginica
 3.428       2.770      2.974
```

If the numeric variable has any missing data, set the `na.rm` argument to `T`.

```
> tapply(dataset$variable, dataset$factor1, mean, na.rm=T)
```

You can also group the data by more than one factor, by nesting the factor variables inside the `list` function. R calculates the statistic separately for each combination of factor levels. For example, to display the median number of breaks for each combination of tension and wool type for the `warpbreaks` dataset:

```
> tapply(warpbreaks$breaks, list(warpbreaks$wool, warpbreaks$tension), median)
```

```
   L  M  H
A 51 21 24
B 29 28 17
```

When using more than one grouping variable, you can only use statistical functions that give a single value as output (i.e., those marked with an asterisk in Table 5-1).

Alternatively, you can also use the `aggregate` function to summarize variables by groups. Using the `aggregate` function has the advantage that you can summarize several continuous variables simultaneously. It can also be used with statistical functions that give more than one value as output (such as `range` and `quantile`). However, the results are displayed a little differently, so it is a matter of personal preference whether to use `tapply` or `aggregate`.

To calculate the mean sepal width for each species, use the `aggregate` function as shown:

```
> aggregate(Sepal.Width~Species, iris, mean)
```

```
     Species Sepal.Width
1     setosa       3.428
2 versicolor       2.770
3  virginica       2.974
```

Again, you can also use more than one grouping variable. For example, to calculate the median number of breaks for each combination of wool and tension for the `warpbreaks` dataset:

```
> aggregate(breaks~wool+tension, warpbreaks, median)
```

```
 wool tension breaks
1   A       L     51
2   B       L     29
3   A       M     21
4   B       M     28
5   A       H     24
6   B       H     17
```

To summarize two or more continuous variables simultaneously, nest them inside the `cbind` function as shown:

```
> aggregate(cbind(Sepal.Width, Sepal.Length)~Species, iris, mean)
```

```
     Species Sepal.Width Sepal.Length
1     setosa       3.428        5.006
2 versicolor       2.770        5.936
3  virginica       2.974        6.588
```

You can save the output to a new data frame, as shown here. This allows you to use the results for further analysis:

```
> sepalmeans<-aggregate(cbind(Sepal.Width, Sepal.Length)~Species, iris, mean)
```

Remember to set the `na.rm` argument to `T` if any of the continuous variables have missing values.

Measures of Association

The *association* between two variables is a relationship between them, such that if you know the value of one variable, it tells you something about the value of the other. Positive association means that as the value of one variable increases, the value of the other also tends to increase. Negative association means that as the value of one variable increases, the value of the other tends to decrease. The most commonly used measures of association are:

Covariance: A measure of the linear association between two continuous variables. Covariance is scale dependent, meaning that the value depends on the units of measurements used for the variables. For this reason, it is difficult to directly interpret the covariance value. The higher the absolute covariance between two variables, the greater the association. Positive values indicate positive association and negative values indicate negative association.

Pearson's correlation coefficient (denoted *r*): A scale independent measure of association, meaning that the value is not affected by the unit of measurement. The correlation can take values between -1 and 1, where -1 indicates perfect negative correlation, 0 indicates no correlation and 1 indicates perfect positive correlation. The correlation coefficient only measures *linear* relationships, so it is important to

check for nonlinear relationships with a scatter plot (see the "Scatter Plots" section in Chapter 8).

Spearman's rank correlation coefficient: A nonparametric alternative to the Pearson's correlation coefficient, which measures nonlinear as well as linear relationships. It also takes values between -1 (perfect negative correlation) and 1 (perfect positive correlation), with a value of 0 indicating no correlation. Spearman's correlation can be calculated for ranked as well as continuous data.

The following subsections explain how to calculate each of these measures of association in R.

Covariance

To calculate the covariance between two variables, use the cov function:

```
> cov(trees$Height, trees$Volume)
```

```
[1] 62.66
```

You can also create a covariance matrix for a whole dataset, which shows the covariance for each pair of variables:

```
> cov(trees)
```

```
           Girth   Height    Volume
Girth   9.847914 10.38333  49.88812
Height 10.383333 40.60000  62.66000
Volume 49.888118 62.66000 270.20280
```

From the output, you can see that the covariance between tree girth and tree height is 10.38. The values along the diagonal of the matrix give the variance of the variables. For example, the tree volumes have a variance of 270.2.

If your dataset has any nonnumeric variables, R will display an error message:

```
> cov(iris)
```

```
Error: is.numeric(x) || is.logical(x) is not TRUE
```

To avoid this problem, exclude the nonnumeric variables using bracket notation or the `subset` function:

```
> cov(iris[-5])
```

```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

If any of the variables have missing values, set the use argument to "`pairwise`" as shown here. Otherwise, the covariance matrix will also have missing values:

```
> cov(dataset, use="pairwise")
```

Another useful option for the use argument is "`complete`". This option completely excludes observations that have missing values for any of the variables, while "`pairwise`" excludes only those observations that have missing values for either of the variables in a given pair. Enter `help(cov)` to view more details about these options.

Pearson's Correlation Coefficient

To calculate the Pearson's correlation coefficient between two variables, use the `cor` function:

```
> cor(trees$Girth, trees$Volume)
```

```
[1] 0.9671194
```

The value is very close to 1, which indicates a very strong positive correlation between tree girth and tree volume. This means that trees with a larger girth tend to have a larger volume.

You can also create a correlation matrix for a whole dataset:

```
> cor(trees)
```

```
           Girth    Height    Volume
```

```
Girth   1.0000000 0.5192801 0.9671194
Height  0.5192801 1.0000000 0.5982497
Volume  0.9671194 0.5982497 1.0000000
```

Notice that the values along the diagonal of the matrix are all equal to 1, because a variable always correlates perfectly with itself.

Remember to exclude any nonnumeric variables using bracket notation or the `subset` function, as shown here for the `iris` dataset:

```
> cor(iris[-5])
```

Again if any of the variables have missing values, set the `use` argument to "pairwise":

```
> cor(dataset, use="pairwise")
```

Spearman's Rank Correlation Coefficient

The `cor` function can also calculate the Spearman's rank correlation coefficient between two variables. Set the `method` argument to `"spearman"`:

```
> cor(trees$Girth, trees$Volume, method="spearman")
```

```
[1] 0.9547151
```

You can also create a correlation matrix for a whole dataset. Remember to exclude any nonnumeric variables using bracket notation:

```
> cor(trees, method="spearman")
```

```
           Girth    Height    Volume
Girth  1.0000000 0.4408387 0.9547151
Height 0.4408387 1.0000000 0.5787101
Volume 0.9547151 0.5787101 1.0000000
```

If any of the variables have missing values, set the `use` argument to `"pairwise"`:

```
> cor(dataset$var1, dataset$var2, method="spearman", use="pairwise")
```

Hypothesis Test of Correlation

A hypothesis test of correlation determines whether a correlation is statistically significant. The null hypothesis for the test is that the population correlation is equal to zero, meaning that there is no correlation between the variables. The alternative hypothesis is that the population correlation is not equal to zero, meaning that there is some correlation between the variables. You can also perform a one-sided test, where the alternative hypothesis is either that the population correlation is greater than zero (the variables are positively correlated) or that the population correlation is less than zero (the variables are negatively correlated).

> **Note** See Chapter 10 for more details about hypothesis testing.

You can perform a test of the correlation between two variables with the `cor.test` function:

```
> cor.test(dataset$var1, dataset$var2)
```

By default, R performs a test of the Pearson's correlation. If you would prefer to test the Spearman's correlation, set the `method` argument to `"spearman"`:

```
> cor.test(dataset$var1, dataset$var2, method="spearman")
```

By default, R performs a two-sided test, but you can adjust this by setting the `alternative` argument to `"less"` or `"greater"` as required:

```
> cor.test(dataset$var1, dataset$var2, alternative="greater")
```

The output includes a 95% confidence interval for the correlation estimate. To adjust the size of this interval, use the `conf.level` argument:

```
> cor.test(dataset$var1, dataset$var2, conf.level=0.99)
```

Example 5-1: Hypothesis Test of Correlation Using the Trees Dataset

Suppose that you want to perform a hypothesis test to help determine whether the correlation between tree girth and tree volume is statistically significant.

To perform a two-sided test of the Pearson's product moment correlation between tree girth and volume at the 5% significance level, use the command:

```
> cor.test(trees$Girth, trees$Volume)
```

The output is shown here:

```
        Pearson's product-moment correlation

data:  trees$Girth and trees$Volume
t = 20.4783, df = 29, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
  0.9322519 0.9841887
sample estimates:
        cor
  0.9671194
```

The correlation is estimated at 0.967, with a 95% confidence interval of 0.932 to 0.984. This means that as tree girth increases, tree volume tends to increase also.

Because the p-value of 2.2e-16 is much less than the significance level of 0.05, we can reject the null hypothesis that there is no correlation between girth and volume, in favor of the alternative hypothesis that the two are correlated.

Scientific Notation

Scientific notation is a way of expressing very large or very small numbers more compactly.

For example, the number 720000 is equal to 7.2 x 100000 or 7.2 x 10^5. R and many other programming languages use the letter e to express "times ten to the power of", so that 7.2 x 10^5 is displayed as 7.2e5.

Scientific notation works similarly for very small numbers, for example the number 0.000072 is equal to 7.2 x 0.00001 or 7.2 x 10^{-5}. Using the R notation, this would be displayed as 7.2e-5.

Comparing a Sample with a Specified Distribution

Sometimes you may wish to determine whether your sample is consistent with having been drawn from a particular type of distribution such as the normal distribution. This is useful because many statistical techniques (such as analysis of variance) are only suitable for normally distributed data.

Two methods that allow you to do this are the Shapiro-Wilk and Kolmogorov-Smirnov tests. To visually assess how well your data fits the normal distribution, use a histogram or normal probability plot (covered in Chapter 8).

Shapiro-Wilk Test

The Shapiro-Wilk test is a hypothesis test that can help to determine whether a sample has been drawn from a normal distribution. The null hypothesis for the test is that the sample is drawn from a normal distribution and the alternative hypothesis is that it is not.

You can perform a Shapiro-Wilk test with the `shapiro.test` function.

```
> shapiro.test(dataset$variable)
```

Example 5-2: Shapiro-Wilk Test Using the Trees Dataset

Suppose that you want to perform a Shapiro-Wilk test to help determine whether the tree heights follow a normal distribution. You will use a 5% significance level. To perform the test, use the command:

```
> shapiro.test(trees$Height)
```

```
        Shapiro-Wilk normality test

data:  trees$Height
W = 0.9655, p-value = 0.4034
```

From the output we can see that the p-value for the test is 0.4034. Because this is not less than our significance level of 0.05, we cannot reject the null hypothesis. This means there is no evidence that the tree heights do not follow a normal distribution.

Kolmogorov-Smirnov Test

A one-sample Kolmogorov-Smirnov test helps to determine whether a sample is drawn from a particular theoretical distribution. It has the null hypothesis that the sample is drawn from the distribution and the alternative hypothesis that it is not.

A two-sample Kolmogorov-Smirnov test helps to determine whether two samples are drawn from the same distribution. It has the null hypothesis that they are drawn from the same distribution and the alternative hypothesis that they are not.

Note that both the one and two-sample Kolmogor-Smirnov tests require continuous data. The test cannot be performed if your data contains ties (i.e some of the values are exactly equal). This is may be an issue if your data is not recorded to a sufficient number of decimal places.

You can perform a Kolmogorov-Smirnov test with the `ks.test` function. To perform a one-sample test with the null hypothesis that the sample is drawn from a normal distribution with a mean of 100 and a standard deviation of 10, use the command:

```
> ks.test(dataset$variable, "pnorm", 100, 10)
```

To test a sample against another theoretical distribution, replace "`pnorm`" with the relevant cumulative distribution function. A list of functions for standard probability distributions is given in Table 7.1 in Chapter 7. You must also substitute the mean and standard deviation with any parameters relevant to the distribution. Use the `help` function to check the parameters for the distribution of interest.

To perform a two-sample test to determine whether two samples are drawn from the same distribution, use the command:

```
> ks.test(dataset$sample1, dataset$sample2)
```

If your data is in stacked form (with the values for both samples in one variable), you must first unstack the dataset as explained in Chapter 4 (under "Unstacking Data").

Example 5-3: One-Sample Kolmogorov-Smirnov Test Using Bottles Data

Consider the `bottles` dataset, which is available with the downloads for this book. The dataset gives data for a sample of 20 bottles of soft drink taken from a filling line. The dataset contains one variable named `Volume`, which gives the volume of liquid in millilitres for each of the bottles.

The bottle filling volume is believed to follow a normal distribution with a mean of 500 milliliters and a standard deviation of 25 milliliters. Suppose that you wish to use a one-sample Kolmogorov-Smirnov test to determine whether the data is consistent with this theory. The test has the null hypothesis that the bottles volumes are drawn from the described distribution, and the alternative hypothesis that they are not. A significance level of 0.05 will be used for the test.

To perform the test, use the command:

```
> ks.test(bottles$Volume, "pnorm", 500, 25)
```

This gives the following output:

```
        One-sample Kolmogorov-Smirnov test

data: bottles$Volume
D = 0.2288, p-value = 0.2108
alternative hypothesis: two-sided
```

From the output we can see that the p-value for the test is 0.2108. As this is not less than the significance level of 0.05, we cannot reject the null hypothesis. This means that there is no evidence that the bottle volumes are not drawn from the described normal distribution.

Example 5-4: Two-Sample Kolmogorov-Smirnov Test Using the Plantgrowth Data

Consider the `PlantGrowth` dataset (included with R), which gives the dried weight of thirty batches of plants, each of which received one of three different treatments. The `weight` variable gives the weight of the batch and the `groups` variable gives the treatment received (`ctrl`, `trt1` or `trt2`).

Suppose that you want to use a two-sample Kolmogorov-Smirnov test to determine whether batch weight has the same distribution for the treatment groups `trt1` and `trt2`.

First you need to unstack the data with the `unstack` function.

```
> PlantGrowth2<-unstack(PlantGrowth)
```

Once the data is in unstacked form, perform the test as shown:

```
> ks.test(PlantGrowth2$trt1, PlantGrowth2$trt2)
```

This gives the following output:

```
       Two-sample Kolmogorov-Smirnov test
```

```
data: PlantGrowth2$trt1 and PlantGrowth2$trt2
D = 0.8, p-value = 0.002057
alternative hypothesis: two-sided
```

The p-value of 0.002057 is less than the significance level of 0.05. This means that there is evidence to reject the null hypothesis that the batch weight distribution is the same for both treatment groups, in favor of the alternative hypothesis that the batch weight distribution is different for the two treatment groups.

Confidence Intervals and Prediction Intervals

A confidence interval for the population mean gives an indication of how accurately the sample mean estimates the population mean. A 95% confidence interval is defined as an interval calculated in such a way that if a large number of samples were drawn from a population and the interval calculated for each of these samples, 95% of the intervals will contain the true population mean value.

A prediction interval gives an indication of how accurately the sample mean predicts the value of a further observation drawn from the population.

The simplest way to obtain a confidence interval for a sample mean is with the `t.test` function, which provides one with the output. The "Student's T-Tests" section in Chapter 10 discusses the function in more detail. If you are only interested in obtaining a confidence interval, use the command:

```
> t.test(trees$Height)
```

```
        One Sample t-test

data: trees$Height
t = 66.4097, df = 30, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  73.6628 78.3372
sample estimates:
mean of x
       76
```

From the results, you can see that the mean tree height is 76 feet, with a 95% confidence interval of 73.7 to 78.3 feet. By default, R calculates a 95% interval. For a different size confidence interval such as 99%, adjust the `conf.level` argument as shown:

```
> t.test(trees$Height, conf.level=0.99)
```

You can also calculate one-sided confidence intervals. For an upper confidence interval, set the `alternative` argument to "`less`". For a lower confidence interval, set it to "`greater`".

```
> t.test(dataset$variable, alternative="greater")
```

To calculate a prediction interval for the tree heights, use the command:

```
> predict(lm(trees$Height~1), interval="prediction")[1,]
```

```
    fit     lwr     upr
76.0000 62.7788 89.2212
Warning message:
In predict.lm(lm(trees$Height ~ 1), interval = "prediction") :
  predictions on current data refer to _future_ responses
```

From the output you can see the the prediction interval for the tree heights is 62.8 to 89.2 feet.

Again, you can adjust the confidence level with the `level` argument as shown:

```
> predict(lm(dataset$variable~1), interval="prediction", level=0.99)[1,]
```

The commands for creating prediction intervals use the `lm` and `predict` functions. You will learn more about these functions in Chapter 11, which will help you to understand what these complex commands are doing. For now, you can just use them by substituting the appropriate dataset and variable names.

Summary

You should now be able to use R to summarize the continuous variables in your dataset and examine the relationship between them. You should also be able to make some inferences about the population from which a sample is drawn, such as whether it has a normal distribution.

This table summarizes the most important commands covered in this chapter.

Task	Command
Statistic for each variable	`sapply(dataset, statistic)`
Statistic by group	`tapply(dataset$var1, dataset$factor1, statistic)`
Statistic by group	`aggregate(variable~factor, dataset, statistic)`
Covariance	`cov(dataset$var1, dataset$var2)`
Covariance matrix	`cov(dataset)`
Pearson's correlation coefficient	`cor(dataset$var1, dataset$var2)`
Correlation matrix	`cor(dataset)`
Spearman's rank correlation coefficient	`cor(dataset$var1, dataset$var2, method="spearman")`
Spearman's correlation matrix	`cor(dataset, method="spearman")`
Hypothesis test of correlation	`cor.test(dataset$var1, dataset$var2)`
Shapiro-Wilk test	`shapiro.test(dataset$variable)`
One-sample Kolmogorov-Smirnov test	`ks.test(dataset$sample1, "pnorm", mean, sd)`
Two-sample Kolmogorov-Smirnov test	`ks.test(dataset$sample1, dataset$sample2)`
Confidence interval	`t.test(dataset$variable)`
Prediction interval	`predict(lm(dataset$variable~1), interval="prediction")[1,]`

Now that you have learned how to summarize continuous variables, we can move on to the next chapter, which will look at categorical variables.