Chapter 10
Statistical Inference with R

All models are wrong; some models are useful.
George E.P. Box

10.1 Introduction

Statistical inference is the branch of statistics whereby we arrive at conclusions about a population through a sample of the population. We can make inferences concerning several issues related to the data, for example, the parameters of the probability distribution, the parameters of a given model that explains the relationship between variables, goodness of fit to a probability distribution, and differences between groups (e.g., regarding the mean or the variance).

In Six Sigma projects, improvement is closely linked to the effect that some parameters of the process (input) have on features of the process (output). Statistical inference provides the necessary scientific basis to achieve the goals of the project and validate its results.

In this chapter, some basic statistical inference tools and techniques suitable for Six Sigma are reviewed. In Sect. 10.2, confidence intervals and point estimation are explained. Hypothesis-testing concepts are very important for every inference analysis. You can read about them in Sect. 10.3. Regression and analysis of variance (ANOVA) are the main techniques to study the relationship between variables. Sections 10.4 and 10.5 explain how to use them with R.

10.2 Confidence Intervals

10.2.1 Sampling Distribution and Point Estimation

Through point estimation, one or more parameters of a population's probability distribution can be inferred using a sample. A function over the values of the sample

E.L. Cano et al., *Six Sigma with R: Statistical Engineering for Process Improvement*, Use R! 36, DOI 10.1007/978-1-4614-3652-2_10,
© Springer Science+Business Media New York 2012

is called a *statistic*. For example, the sample mean is a statistic. When we make inferences about the parameters of a population's probability distribution, we use statistics. These statistics have, in turn, a probability distribution. That is, for every sample extracted from a given population, we have a value for the statistic. In this way, we may build a new population of values (of the statistic) that follows some probability distribution.

The probability distribution of a statistic is a *sampling distribution*, and the properties of this distribution allow us to know if the statistic is a good estimator of the parameter under study. Some important properties to study about a statistic to find out if it is a good estimator are unbiasedness, mean square error, consistency, and asymptotic distribution. An estimator is unbiased if its expectation equals the real value of the parameter.

To distinguish the actual value of a parameter from its estimator, a *hat* (ˆ) is placed over the symbol of the estimator (e.g., $\hat{\sigma}^2$ is the estimator for the variance σ^2). We will not explain in detail the properties of the statistics or explain how to study sampling distributions. We will simply introduce some of the most important statistics for estimating proportions, means, and variances.

For binomial distributions, the sample proportion is an unbiased estimator:

$$\hat{p} = \frac{x}{n}.$$

That is, the number of events (x) in n Bernoulli experiments over the number of experiments. The mean and the variance of the statistic are[1]

$$\mu_{\hat{p}} = p,$$
$$\sigma^2_{\hat{p}} = \frac{p(1-p)}{n}.$$

The sample mean is an unbiased estimator of the population mean:

$$\hat{\mu} = \bar{x},$$
$$\mu_{\bar{x}} = \mu,$$
$$\sigma^2_{\bar{x}} = \frac{\sigma^2}{n}.$$

For the variance the unbiased estimator is the sample variance, defined as

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1},$$

whose expectation is the population variance ($E[\hat{s^2}] = \sigma^2$).

[1]For sampling distributions, we set the symbol of the statistic a subscript.

The sample standard deviation is not an unbiased estimator of the population standard deviation. In its place we can use the following unbiased estimator:

$$\hat{\sigma} = \frac{s}{c_4},$$

where c_4 is a constant that depends only on the sample size:

$$c_4 = \left(\frac{2}{n-1}\right)^{\frac{1}{2}} \frac{\Gamma(n/2)}{\Gamma\left[(n-1)/2\right]},$$

where $\Gamma(\cdot)$ can be evaluated using the `gamma` R function. You can compute this constant using the `ss.cc.getc4` function in the `SixSigma` package.

An unbiased estimator for the standard deviation σ of a normal distribution used in many applications of engineering statistics and quality control is

$$\hat{\sigma} = \frac{R}{d_2},$$

where R is the range of the data in the sample (that is, the difference between the maximum and the minimum) and d_2 is the mean of the random variable W (*relative range*) defined as

$$W = \frac{R}{\sigma}.$$

The mean of this random variable is constant and depends only on the sample size n. It can be computed numerically using integration. The `ss.cc.getd2` function[2] in the `SixSigma` package returns the constant d_2 for a given sample size:

```
> ss.cc.getd2(20)
```

```
      d2
3.734949
```

Example 10.1 (Guitar strings). Let us reproduce the example in Chap. 9. The quality manager of a company that produces guitar strings wants to study the resistance to tension of the strings produced one day. Obviously, testing the resistance of the whole population is not possible. Every day, the company produces 1,000 strings of each type (E1, B2, G3, D4, A5, E6), and once the strings are made, they are packaged and numbered.

We can do computations over the sample data to obtain estimators of the mean and variance[3] for tension resistance in the `ss.data.strings` data set:

[2] It in turn uses the `ptukey` function (see `?ptukey`).

[3] R provides the sample variance. If you need the population variance, just multiply it by $\frac{n-1}{n}$.

```
> mean(ss.data.strings$res)
```

```
[1] 6.666667
```

```
> var(ss.data.strings$res)
```

```
[1] 3.45098
```

For the proportion of defective strings, the estimator $\hat{p}$ is

```
> sum(ss.data.strings$res<3)/nrow(ss.data.strings)
```

```
[1] 0.03333333
```

The unbiased estimator for the standard deviation using the sample standard deviation can be computed as follows:

```
> c(Sigma.Est = sd(ss.data.strings$res)/ss.cc.getc4(nrow(ss.
    data.strings)))
```

```
Sigma.Est.c4
    1.861588
```

We can estimate the standard deviation through the range, using the constant d_2:

```
> c(Sigma.Est = diff(range(ss.data.strings$res))/ss.cc.getd2(
    nrow(ss.data.strings)))
```

```
Sigma.Est.d2
    1.749553
```

□

We have obtained an estimation for the parameter of interest for our process. However, any estimation is linked to some uncertainty, and therefore we will have to deal with some *error level*. To quantify this uncertainty, we use interval estimation. Interval estimation consists in giving bounds for our estimation (LL and UL, upper and lower limits). These limits are calculated so that we have confidence in the fact that the real value of the parameter is contained within them. This fact is stated as a *confidence level* and expressed as a percentage. The confidence level reflects the percentage of times that the real value of the parameter is assumed to be in the interval when repeating the sampling. Usually the confidence level is represented by $100 \times (1-\alpha)\%$, with α the confidence coefficient. The confidence coefficient is a measure of the error in our estimation. Common values for the confidence level are 99, 95, or 90%, corresponding, respectively, to $\alpha = 0.01$, $\alpha = 0.05$, and $\alpha = 0.1$.

A confidence interval is expressed as an inequality.[4] If θ is a parameter, then $[LL, UL]$ means $LL \leq \theta \leq UL$.

[4]In Bayesian statistics, the *credible interval* is the counterpart of the confidence interval, which has a probabilistic meaning.

10.2.2 Proportion Confidence Interval

As we explained in Chap. 9, when we are dealing with proportions, the binomial distribution is the appropriate probability distribution to model a process. The typical application of this model is the calculation of the fraction of nonconforming items in a process.

Due to the central limit theorem (see Sect. 9.3.3 in Chap. 9), we can construct a confidence interval for the proportion using the following formula:

$$\hat{p} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}},$$

where $z_{1-\frac{\alpha}{2}}$ is the quantile of the standard normal distribution that leaves a probability of $\frac{\alpha}{2}$ on the right-hand side. This is the *classical* way to construct a confidence interval for the proportion when the sample size n is large and $\hat{p}$ is not small (under these circumstances, the normal distribution can be used to approximate the binomial distribution). The R function `binom.test` provides an exact confidence interval for the probability of success. However, in [1] it is shown that "approximate results are sometimes more useful than exact results, because of the inherent conservativeness of exact methods." In this regard, the approximation performed by the `prop.test` function may provide more accurate results. Likewise, the `binconf` function in the `Hmisc` package allows us to use different methods and compare the results.

Example 10.2 (Guitar strings (cont.)). To construct a 95% confidence interval for the proportion of defective strings using the normal approximation, we first obtain the value of the standard normal distribution for a probability of $\frac{\alpha}{2} = 0.025$:

```
>   def.z <- qnorm(0.975)
```

The point estimator for the proportion is

```
>   est.p <- sum(ss.data.strings$res<3)/nrow(ss.data.strings)
```

Therefore, our confidence interval is

```
>   est.p + (c(LL=-1,UL=1) * def.z * sqrt((est.p*(1-est.p))/
    nrow(ss.data.strings)))
```

```
          LL          UL
0.001216316 0.065450351
```

The previous normal approximation may not be adequate, given that p is small. A suitable alternative is to use the `prop.test` function, which provides the following interval:

```
> (prop.test(x = sum(ss.data.strings$res<3),
    n = nrow(ss.data.strings)))$conf.int
```

```
[1] 0.01073155 0.08825263
attr(,"conf.level")
[1] 0.95
```

We can compare different methods with the `binconf` function:

```
> require("Hmisc")
> binconf(x = sum(ss.data.strings$res<3),
    n = nrow(ss.data.strings), method = "all")
```

```
             PointEst       Lower      Upper
Exact      0.03333333 0.009155506 0.08314872
Wilson     0.03333333 0.013037551 0.08258034
Asymptotic 0.03333333 0.001216316 0.06545035
```

As a general rule, the result of the `prop.test` is the best option. □

10.2.3 Mean Confidence Interval

Thanks to the central limit theorem, for large[5] sample sizes we can construct confidence intervals for the mean of any distribution using the following formula:

$$\bar{x} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}}.$$

Usually, σ is unknown. In this case, instead of σ and the normal quantile z, we must use the sample standard deviation (s) and Student's t quantile with $n-1$ degrees of freedom ($t_{\alpha/2,n-1}$). A thorough explanation of this important concept is outside the scope of this book. The degrees of freedom can be thought of as the number of data minus the number of constraints used to estimate the parameter under study. Therefore, the confidence interval takes the following form:

$$\bar{x} \pm t_{\alpha/2,n-1} \frac{s}{\sqrt{n}}.$$

Example 10.3 (Guitar strings (cont.)). We want to build a 95% confidence interval for the mean of the tension resistance of the strings in the data set `ss.data.strings`. Assuming that the population variance is unknown, we need the appropriate value of the t distribution:

```
> res.t <- qt(0.975, nrow(ss.data.strings) - 1)
> res.t
```

```
[1] 1.9801
```

[5] A sample size $n \geq 30$ is considered large.

This value will be multiplied by the standard deviation of the mean, that is:

```
> sd.mean <- sd(ss.data.strings$res)/sqrt(nrow(ss.data.
    strings))
> sd.mean
```

```
[1] 0.1695823
```

And finally we can give the confidence interval:

```
> mean(ss.data.strings$res) + c(LL = -1, UL = 1) * (res.t *
    sd.mean)
```

```
      LL       UL
6.330877 7.002457
```

If we know the actual value of the population variance, then we must use a different formula to construct the confidence interval. For example, if we know (through historical data or any other accepted procedure) that the variance of the resistance is 4, then we construct the confidence interval using the following code:

```
> res.z <- qnorm(0.975)
> sd.mean <- 2 / sqrt(nrow(ss.data.strings))
> mean(ss.data.strings$res) + c(LL = -1, UL = 1) * (res.z *
    sd.mean)
```

```
      LL       UL
6.308828 7.024505
```

□

10.2.4 Variance Confidence Interval

Sometimes we need to find out if the variance of a process is within a given range. Confidence intervals are a fast way to verify this issue. There are two important differences between mean and variance confidence intervals:

- Methods for variance are more sensitive to the normality assumption. Thus, a normality test is advisable for validating results.
- The statistic used to construct the confidence interval (χ^2) for the variance is not symmetric, unlike z or t. Therefore, the limits are not symmetric with respect to the point estimator.

The formulas to construct the confidence interval are as follows:

$$\frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}} \leq \sigma \leq \frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}}.$$

Example 10.4 (Guitar strings (cont.)). In the guitar string factory, the length of each string was measured before the resistance was tested. To compute a confidence interval for the variance of the length, we first compute the χ^2 quantiles:

```
> len.chi <- c(qchisq(0.975, nrow(ss.data.strings)),
        qchisq(0.025, nrow(ss.data.strings)))
> len.chi
```

```
[1] 152.21140  91.57264
```

Next, we calculate the bounds of the confidence interval:

```
> len.ci <- ((nrow(ss.data.strings)-1) * var(ss.data.strings$
    len)) / len.chi
> len.ci
```

```
[1] 0.05591480 0.09294119
```

We can obtain a confidence interval for the mean along with a graphical output and a normality test with the function `ss.ci` in the `SixSigma` package (Fig. 10.1):

```
> ss.ci(len, data = ss.data.strings, digits = 3)
```

```
        Mean = 950.016; sd = 0.267
        95% Confidence Interval= 949.967 to 950.064

      LL       UL
949.9674 950.0640
```

□

Confidence intervals can also be obtained using the functions for hypothesis testing available in R (we will talk about hypothesis tests in the next section). The hypothesis testing functions in R return an `htest` object. One of the components of these types of objects is `conf.int`, which contains a confidence interval for the data and the confidence level specified.

Example 10.5 (Guitar strings (cont.)). To obtain automatically a confidence interval for the mean of the length of the strings, we first save the object returned by the `t.test` function and then extract the `conf.int` component:

```
> my.test <- t.test (ss.data.strings$len)
> my.test$conf.int
```

```
[1] 949.9674 950.0640
attr(,"conf.level")
[1] 0.95
```

□

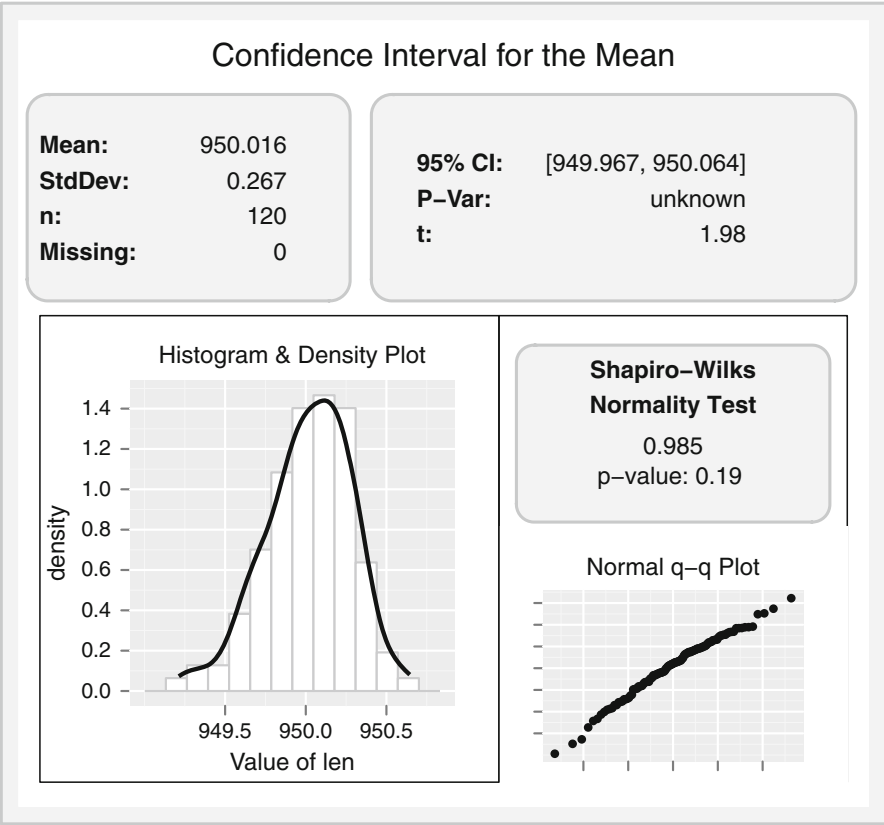

Fig. 10.1 Confidence interval for guitar string example. The output shows the confidence interval and the facts used to construct it. A normality test (including a quantile–quantile plot) and a histogram with a density plot are shown to validate the model assumptions

10.3 Hypothesis Testing

In statistical inference, hypothesis testing is intended to confirm or validate some conjectures about the process we are analyzing. Importantly, these hypotheses are related to the parameters of the probability distribution of the data. For example, if we have data from a process that are normally distributed and we want to verify if the mean of the process has changed with respect to the historical mean, we should make the following hypothesis test:

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

where H_0 denotes the *null hypothesis* and H_1 denotes the *alternative hypothesis*. Thus we are testing H_0 ("the mean has not changed") vs. H_1 ("the mean has changed").

Hypothesis testing can be performed in two ways: one-sided tests and two-sided tests. An example of the latter is when we want to know if the mean of a process has increased:

$$H_0 : \mu = \mu_0,$$
$$H_1 : \mu > \mu_0.$$

Hypothesis testing tries to find evidence about the refutability of the null hypothesis using probability theory.[6] We want to check if a new situation (represented by the alternative hypothesis) is arising. Subsequently, we will reject the null hypothesis if the data do not support it with "enough evidence." The threshold for enough evidence is decided by the analyst, and it is expressed as a significance level α (similarly to the confidence intervals explained in Sect. 10.2). A 5% significance level is a widely accepted value in most cases.

To verify whether the data support the alternative hypothesis, a statistic (related to the underlying probability distribution) is calculated. If the value of the statistic is within the rejection region, then the null hypothesis is rejected. If the statistic is outside the rejection region, then we say that we do not have enough evidence to accept the alternative hypothesis (perhaps it is true, but the data do not support it).

Usually the refutability of the null hypothesis is assessed through the p-value stemmed from the hypothesis test. If the p-value is larger than α, then H_0 should not be rejected, otherwise H_0 must be rejected. The p-value is sometimes interpreted as the probability that the null hypothesis is true. This interpretation is not correct. The p-value is the probability of finding a more extreme sample (in the sense of rejecting H_0) than the one that we are currently using to perform the hypothesis test. So if the p-value is small, the probability of finding a more extreme sample is small, and therefore the null hypothesis should be rejected. Otherwise, if the p-value is large, the null hypothesis should not be rejected. The concept of "large" is determined by the significance level (α). In practice, if $p < \alpha$, then H_0 is rejected. Otherwise, H_0 is not rejected. Thus, for instance, if the confidence level is 95% ($\alpha = 0.05$) and the p-value is smaller than 0.05, we do not accept the null hypothesis taking into account empirical evidence provided by the sample at hand.

There are some functions in R to perform hypothesis tests, for example, `t.test` for means, `prop.test` for proportions, `var.test` and `bartlett.test` for variances, `chisq.test` for contingency table tests and goodness-of-fit tests, `poisson.test` for Poisson distributions, `binom.test` for binomial distributions, `shapiro.test` for normality tests. Usually, these functions also provide a confidence interval for the parameter tested.

[6] We will not explain in detail the foundations of hypothesis testing. Some good references can be found in Sect. 10.6.

Example 10.6 (Guitar strings (cont.)). We can perform a hypothesis test to verify if the length of the strings is different from the target value of 950 mm:

$$H_0 : \mu = 950,$$
$$H_1 : \mu \neq 950.$$

```
> t.test(ss.data.strings$len,
    mu = 950,
    conf.level = 0.95)
```

```
        One Sample t-test

data:  ss.data.strings$len
t = 0.6433, df = 119, p-value = 0.5213
alternative hypothesis: true mean is not equal to 950
95 percent confidence interval:
 949.9674 950.0640
sample estimates:
mean of x
 950.0157
```

As the p-value is (much) greater than 0.05, we accept that the mean is not different from the target.

We can also compare two means, for example, for two types of strings. To compare the length of the string types E6 and E1, we use the following code:

```
>data.E1 <- ss.data.strings$len[ss.data.strings$type == "E1"]
>data.E6 <- ss.data.strings$len[ss.data.strings$type == "E6"]
>t.test(data.E1, data.E6)
```

```
        Welch Two Sample t-test

data:  data.E1 and data.E6
t = -0.3091, df = 36.423, p-value = 0.759
alternative hypothesis: true difference in means is not equal
     to 0
95 percent confidence interval:
 -0.1822016  0.1339911
sample estimates:
mean of x mean of y
 949.9756  949.9997
```

Again, we cannot accept the alternative hypothesis, and therefore we do not reject that there are no differences between the length of the two types of strings (that is, we do not reject the null hypothesis). If we want to compare the variances between the two types of strings, we can use the `var.test` function:

```
> var.test(data.E1, data.E6)
```

```
        F test to compare two variances

data:  data.E1 and data.E6
F = 1.5254, num df = 19, denom df = 19, p-value
= 0.3655
alternative hypothesis: true ratio of variances is not equal
    to 1
95 percent confidence interval:
 0.6037828 3.8539181
sample estimates:
ratio of variances
          1.525428
```

In this case, the statistic used is the ratio between variances, and the null hypothesis is "the ratio of variances is equal to 1," that is, the variances are equal.

We can also compare proportions using the `prop.test` function. Do we have the same defects for every string type? For example, with the following code we can compare types E1 and A5:

```
> defects <- data.frame(type = ss.data.strings$type, res = ss
    .data.strings$res < 3)
> defects <- aggregate(res ~ type, data = defects, sum)
> prop.test(defects$res, rep(20,6))
```

```
        6-sample test for equality of proportions
        without continuity correction

data:  defects$res out of rep(20, 6)
X-squared = 5.1724, df = 5, p-value = 0.3952
alternative hypothesis: two.sided
sample estimates:
prop 1 prop 2 prop 3 prop 4 prop 5 prop 6
  0.05   0.00   0.00   0.10   0.00   0.05
```

The p-value for the hypothesis test of equal proportions is 0.39 (larger than 0.05), so we cannot reject the null hypothesis, and therefore we do not reject that the proportions are equal.

A normality test to check if the data follow a normal distribution can be performed with the `shapiro.test()` function:

```
> shapiro.test(ss.data.strings$len)
```

```
        Shapiro--Wilk normality test

data:  ss.data.strings$len
W = 0.9846, p-value = 0.1902
```

The statistic used to perform this hypothesis test is the *Shapiro–Wilk* statistic. In this test, the hypotheses are as follows:

H_0 : The data are normally distributed.

H_1 : The data are not normally distributed.

The p-value is `0.19`. As it is larger than 0.05, we cannot reject the hypothesis of normality for the data, so we do not have enough evidence to reject normality. Other normality tests can be performed using the `nortest` package. □

Finally, regarding hypothesis testing, the concepts *error type I* and *error type II* should be introduced. A type I error occurs when we reject the null hypothesis and it is true. We commit a type II error when we do not reject the null hypothesis and it is false. The probability of the former is represented as α, and it is the significance level of the hypothesis test ($1-\alpha$ is the confidence level). The probability of the latter is represented as β, and the value $1-\beta$ is the statistical power of the test.

In statistics, committing a type I error is considered more severe than committing a type II error, and that is one reason why α is fixed before the test is performed. The statistical power of the test is usually used to find the appropriate sample size to conduct a hypothesis test. The R functions `power.prop.test` and `power.t.test` can be used to determine the power or the sample size.

Example 10.7 (Guitar strings (cont.)). Using the results of the initial study, the Black Belt plans to perform a new analysis to find out the sample size needed to estimate the mean length of the strings with a maximum error of $\delta = 0.1$ cm. He sets the significance level ($\alpha = 0.05$) and the power ($1-\beta = 0.90$). The sample size can be determined using the following command:

```
> power.t.test(delta = 0.1, power = 0.9, sig.level = 0.05,
        sd = sd (ss.data.strings$len))
```

```
     Two-sample t test power calculation

              n = 151.2648
          delta = 0.1
             sd = 0.2674321
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

 NOTE: n is number in *each* group
```

Therefore, for the new study he must select a sample of 152 strings. □

10.4 Regression

10.4.1 Model Identification

One of the most important inferences to be made in a Six Sigma project concerns the relationship between the critical to quality (CTQ) characteristics of our process (Ys) and the process parameters (Xs), that is, the variables that affect the process. This relationship is represented as a function where the value of the Ys can be inferred

from knowledge of the Xs, plus some error (ε):

$$Y = f(X) + \varepsilon. \tag{10.1}$$

This function is a statistical model, with some parameters that must be estimated. Moreover, the error of the model must be assessed.

Regression is the statistical technique used to estimate the f function in (10.1). The easiest case is the simple linear regression, where f corresponds to a straight line. If we have only one independent variable (y) and one dependent variable (x), the regression model is

$$y = \beta_0 + \beta_1 x + \varepsilon,$$

and the parameters to be estimated are β_0 (the intercept of the straight line) and β_1 (the slope of the straight line). So the estimated model will be

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x.$$

The difference between the estimated values and the actual values in the sample, $\hat{y} - y$, are the *residuals*. The residuals are used to check and validate the model.

Before fitting the regression model, we need to find some evidence regarding the linear relation between the variables. The appropriate statistic to have a first approximation is the correlation coefficient, defined as

$$r = \frac{s_{xy}}{s_x s_y},$$

where $s_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n-1}$ is the covariance between the two samples and s_x and s_y are the sample standard deviations of x and y, respectively. This coefficient ranges from -1 to 1. There is no correlation when it equals 0, perfect positive correlation when it is 1, and perfect negative correlation when it is -1.

The graphical tool suitable for detecting at a glance the relationship between the two variables is the scatterplot (Chap. 8).

Example 10.8 (Guitar strings (cont.)). During the resistance test of the strings, a new measure was taken for those strings with a level of tension larger than 8. This measure is related to the sound volume produced by the string, which is considered a CTQ characteristic.

The Black Belt wonders whether the length of the strings is related to the sound volume. First, a new data frame is created to remove the missing data (those strings that broke down before achieving a level of tension equal to 8). Next, the correlation coefficient is calculated:

```
> data.sound <- na.omit(ss.data.strings)
> cor(data.sound$sound, data.sound$len)
```

```
[1] 0.7904747
```

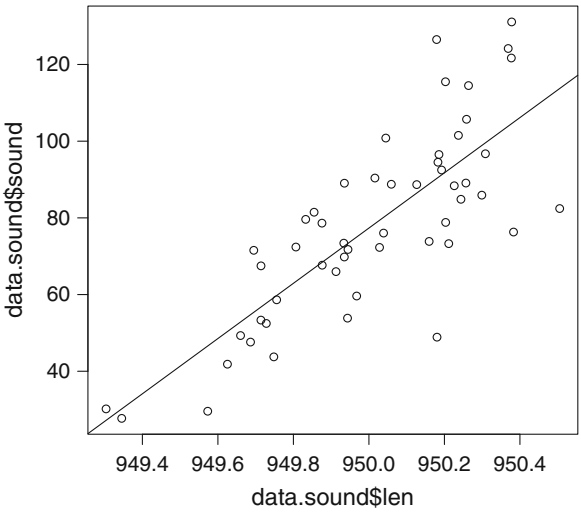

Fig. 10.2 Basic regression scatterplot. The regression line is plotted passing the adjusted model as an argument to the `abline` function

The value of the coefficient (0.79) shows that there is positive correlation between the two variables.

To confirm this relationship, we construct a scatterplot. Using the following code we can plot it using R standard graphics, including the regression line (Fig. 10.2):

```
> plot(data.sound$len, data.sound$sound)
> abline(lm(sound ~ len, data = ss.data.strings))
```

It is apparent that the longer a string is, the higher the sound it produces. We can plot a more sophisticated regression chart including a smooth line, confidence bands, and more information about the data (coloring the points according to string type) using the `ggplot2` package (Fig. 10.3) by typing the following code:

```
> qplot(len, sound, data = data.sound,
        geom = c("point", "smooth"),
        xlab = "String length",
        ylab = "String sound")  +
    geom_point(aes(col=type)) +
    opts(title = "Scatterplot for regression")
```

□

10.4.2 Model Fitting

Once we have identified the model, we estimate it, that is, we calculate $\hat{\beta}_0$ and $\hat{\beta}_1$. The function `lm` fits linear models (see Sect. 10.4.4 for additional models). The R functions that fit models accept certain specific arguments for the model to be fitted. There are two essential common arguments: `data` and `formula`. The former is the data frame where the data are stored, containing the variables we want to

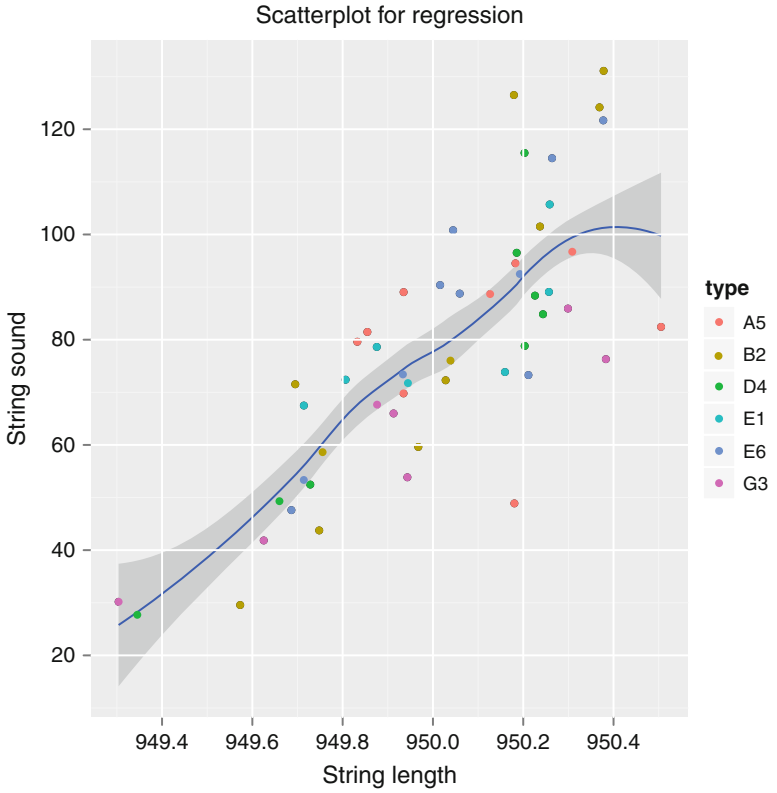

Fig. 10.3 Sophisticated scatterplot using `ggplot2` package. The line and bounds are computed using the `loess` function, which fits a polynomial regression. The smoothness can be adjusted by changing the `span` argument in the `qplot` function. It is also possible to include just the regression line. The points are colored according to string type to identify possible patterns

study. The latter is a special R expression to represent functions in symbolic form. It consists of two terms connected by the special *tilde* operator $\sim$. The term on the left is for the *response* or dependent variable. The term on the right is for the independent or explanatory variables. For example, for the simple linear regression model $y = \beta_0 + \beta_1 x$, the expression to pass as formula is $y \sim x$ (assuming that we have two vectors named x and y in the R workspace).

The term on the right-hand side can consist of a single variable or a series of terms combined with some operators indicating the influence in the model (`+`, `-`, `:`, `*`, `^`, `%in%`). The following special symbols can be used to specify terms: `0` (avoid intercept), "`.`" (all but the response), and `1` (empty model). Type `?formula` in the R console to learn more about model formulae in R.

Example 10.9 (Guitar strings (cont.)). The linear model to estimate the relationship between the sound (response) and length of strings is

```
> lm(sound ~ len, data = data.sound)
```

```
Call:
lm(formula = sound ~ len, data = data.sound)

Coefficients:
(Intercept)          len
  -68346.72        72.03
```

By simply executing the `lm` function, we obtain the coefficients. These coefficients provide the formula for our model:

$$\text{sound} = -68346.72 + 72.03 \times \text{length}.$$

The usual way to proceed is to save the model in an object and access the elements in it. The elements stored in a `lm` model are listed below. The typical output for regression can be obtained with the `summary` generic function over a model object:

```
> my.model <- lm(sound ~ len, data = data.sound)
> names(my.model)
```

```
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

```
> summary(my.model)
```

```
Call:
lm(formula = sound ~ len, data = data.sound)

Residuals:
    Min      1Q  Median      3Q     Max
-41.438  -7.483  -1.345  10.369  36.252

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -68346.719   7497.948  -9.115 3.30e-12
len             72.025      7.892   9.126 3.18e-12

Residual standard error: 15.32 on 50 degrees of freedom
Multiple R-squared: 0.6249,     Adjusted R-squared: 0.6173
F-statistic: 83.28 on 1 and 50 DF,  p-value: 3.181e-12
```

In this output, we obtain, after the call of the function, some statistics about the residuals distribution. Next, the coefficients and the result of their hypothesis tests are shown. The null hypothesis in these tests is "the parameter equals 0." Therefore, as the p-values are very small, we accept that the parameters are not zero. In the last part of the output, the R-squared statistic shows the proportion of the variability of

the response that is explained by the linear model. The hypothesis test performed next is for testing the overall goodness of fit of the regression model. For simple linear models, the p-value is the same as that for the slope of the regression line.

The values returned for the parameters correspond to an estimation. We can compute a confidence interval for these estimators using the `confint` function over the model object:

```
> confint(my.model)
```

```
                  2.5 %       97.5 %
(Intercept) -83406.79017 -53286.64744
len             56.17277     87.87787
```

Other functions that can be run with a model as argument include `plot`, `residuals`, `predict`, and `anova`, among others. □

10.4.3 Model Validation

Once we have estimated the model, we need to validate the assumptions we made before fitting the model. For linear models, the main assumption is the normality and independence of the residuals. To verify the assumptions, we can use analytical tools such as a normality test for the residuals using the `shapiro.test` function or graphical tools such as those provided by the generic `plot` function over the model object.

Example 10.10 (Guitar strings (cont.)). Once we have fitted the model, we can perform a normality test over the residuals:

```
> shapiro.test(residuals(my.model))
```

```
        Shapiro-Wilk normality test

data:  residuals(my.model)
W = 0.9903, p-value = 0.9455
```

We cannot reject the null hypothesis (residuals are normal), so we accept normality for the residuals of the regression model.

The graphical output provided by the generic `plot` function is shown in Fig. 10.4. In the following code, the first line allows us to print in a single window the four graphics produced:

```
> par(mfrow=c(2,2))
> plot(my.model)
```

□

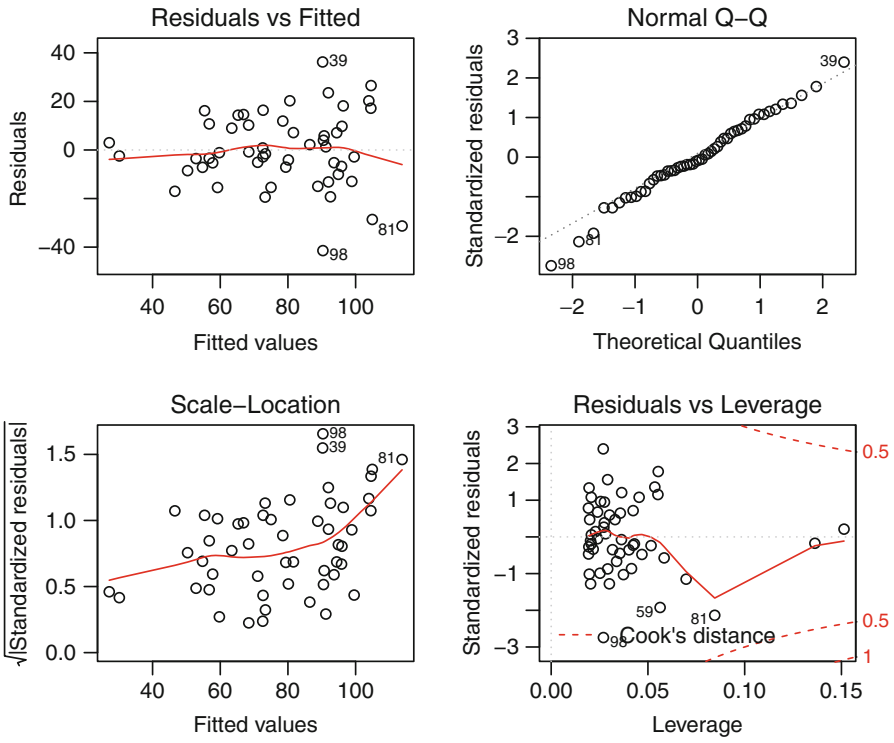

Fig. 10.4 Graphic output for linear regression model. The plot in the *top left* is a scatterplot of the residuals vs. the fitted values (the lack of patterns confirms the normality); the plot in the *top right* is a quantile–quantile plot for normal distributions (the points are approximately over a straight line, again confirming normality). The plot in the *bottom left* is a scatterplot for the $\sqrt{|\text{standardized residuals}|}$ vs. the fitted values. It is similar to the first one, and a triangular shape would be indicative of a lack of homoscedasticity (constant variance, which is one of the model assumptions). The last plot (*bottom right*) shows Cook's distance for all the points. Those points outside of the 0.5 line affect the parameter estimation and might be outliers

10.4.4 Other Models

So far, we have built step by step a simple linear regression model. In practice, many other models may describe our process. The first extension of the simple regression model is the multiple regression model, where we can include more than one independent variable, just adding up more terms on the right side of the formula. R can also fit models with different types of variables, including factors. When more than one variable is present, a variable selection technique should be considered. We can automatically select the significant variables with the function `step` over a model object. If we prefer to do it by ourselves, the function `update.formula` allows us to modify the model without rewriting the whole formula.

For multiple regression models, a new assumption is very important: the independent variables must be uncorrelated among each other. When this issue cannot be avoided, ridge regression is an alternative. The functions `lm.ridge` (`MASS` package), `ridge` (`survival` package), and `lpridge` (`lpridge` package) can fit regression models by ridge regression.

Other extension of the linear models are the *generalized linear model* (GLM), which allows us to deal with nonnormal data and adjust models of different *families* (binomial, Poisson, etc.). The R function `glm` is the appropriate one to fit these models. The *generalized additive model* (GAM) and *mixed models* are alternatives when we do not have linear relationships. The `gam` and `mgcv` packages contain functions for fitting these kinds of models.

State-of-the-art techniques to predict and estimate relationships between variables include neural networks (NN), partial least-squares (PLS) regression, and support vector machine (SVM). Packages for these techniques are available at CRAN: `AMORE`, `monmlp`, `nnet`, and `neuralnet` for NN; `pls`, `plsdof`, `plspm`, `plsRglm`, `plsgenomics`, `plsRbeta`, and `plsRcox` for PLS; and `e1071`, `kernlab`, and `RWeka` for SVM.

Example 10.11 (Guitar strings (cont.)). We can try to fit a multiple regression model by adding the variable `res` to our model to find out if tension also contributes to the sound. We can update the model with the following code:

```
> new.model <- update(my.model, . ~ . + res )
```

or just rewrite the formula:

```
> new.model <- lm(sound ~ len + res, data = data.sound)
```

Now we can see the model summary:

```
> summary(new.model)
```

```
Call:
lm(formula = sound ~ len + res, data = data.sound)

Residuals:
    Min      1Q  Median      3Q     Max
-41.194  -7.949  -1.641  10.702  36.496

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -68604.720   7604.325  -9.022 5.47e-12
len             72.285      8.001   9.035 5.24e-12
res              1.361      4.054   0.336    0.739

Residual standard error: 15.46 on 49 degrees of freedom
Multiple R-squared: 0.6257,     Adjusted R-squared: 0.6104
F-statistic: 40.96 on 2 and 49 DF,  p-value: 3.496e-11
```

The new estimate parameter for resistance (1.361) is not significantly different from zero (p-value = 0.739). Therefore, we do not need it in the model. We can

automatically select the significant variables with the `step` function over the full model using the `backward` method[7]:

```
> step(new.model, direction = "backward")
```

```
Start:  AIC=287.67
sound ~ len + res

       Df Sum of Sq   RSS    AIC
- res   1      26.9 11735 285.79
<none>              11708 287.67
- len   1   19502.4 31210 336.66

Step:  AIC=285.79
sound ~ len

       Df Sum of Sq   RSS    AIC
<none>              11735 285.79
- len   1     19545 31280 334.77

Call:
lm(formula = sound ~ len, data = data.sound)

Coefficients:
(Intercept)          len
  -68346.72        72.03
```

As expected, the model obtained contains only the explanatory variable `len`. □

10.5 Analysis of Variance

Analysis of variance (ANOVA) is the appropriate statistical technique to analyze the relationship between variables when the explanatory variables are qualitative (also called factors). The possible values of the factors are called *levels*. When we have only one factor with two levels, we can perform a t-test to test the difference between means, as explained in Sect. 10.3. Otherwise, we use one-way ANOVA. One-way means the principal effects to be evaluated, that is, the effect of belonging to one of the groups determined by the levels. Two-way ANOVA is performed when we have more than one factor and we are interested in measuring the effect of second-order interactions (that is, if the response in one level of a factor depends on the level of another factor). Similarly, we can perform multiway ANOVA. However, it is very unlikely to improve our analysis with more than third-order interactions.

[7] We can choose one of the following methods in the `step` function: backward, forward, or both.

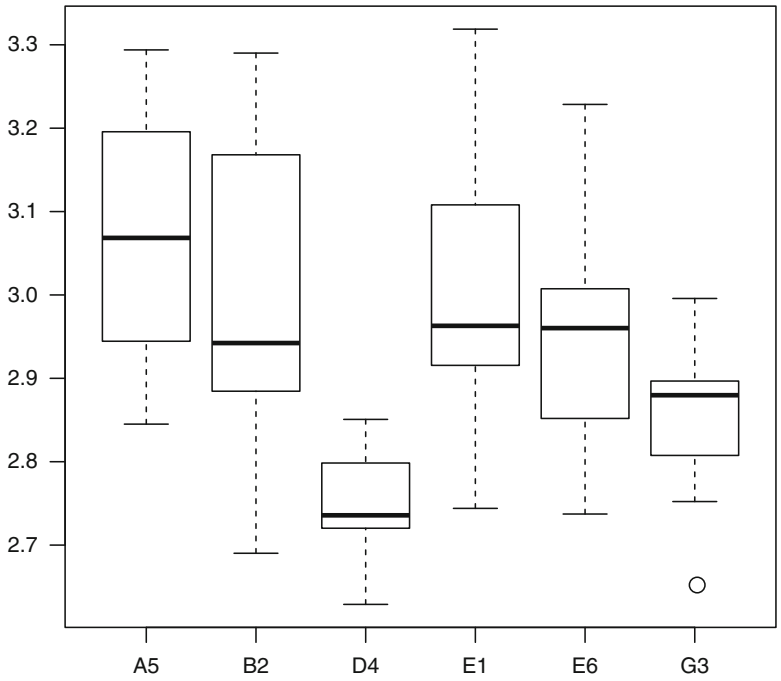

Fig. 10.5 Group box plots for guitar string example. The power needed for strings D4 and G3 seems to be different from the rest. We need to verify if this difference is significant

10.5.1 Model Identification

The counterpart of the scatterplot to reveal a possible relationship between the response and the factors is the box plot by groups. It consists in plotting in the same graphic box plots for each group defined by the factor levels.

Example 10.12 (Guitar strings (cont.)). A measurement of the power needed to pluck a string at a tension level of 8 was taken during the tension test. The Black Belt wants to know if there are differences between the various types of strings for this important characteristic. First, we should plot the box plots for the groups (Fig. 10.5):

```
> boxplot(power ~ type, data = data.sound)
```

We can see that strings D4 and G3 look different from the rest. □

10.5.2 Model Fitting and Validation

We can use the functions `aov` and `lm` to perform the ANOVA. When using `lm`, the ANOVA table is printed calling the function `anova` over the object. The effects of each factor level are the parameters to estimate in the model:

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij},$$

where α_i is the effect of level i. In practice, μ is replaced by the sample mean of the first level (reference level), and the effects are related to the reference level.

Example 10.13 (Guitar strings (cont.)). We fit the model with the `lm` function[8] and save the result in an object:

```
> model.power <- lm(power ~ type, data = data.sound)
> names(model.power)
```

```
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "contrasts"
[10] "xlevels"       "call"          "terms"
[13] "model"
```

This object contains 13 components. By calling the generic `anova` function over the model, we get the following ANOVA table:

```
> anova(model.power)
```

```
Analysis of Variance Table

Response: power
          Df  Sum Sq  Mean Sq F value   Pr(>F)
type       5 0.57869 0.115739  4.8131 0.001273
Residuals 46 1.10615 0.024047
```

The ANOVA table shows a hypothesis test where the null hypothesis is "all the effects are equal to 0." As the p-value is lower than 0.05, we can reject the null hypothesis and accept that there are differences among the means of the groups.

The parameters are stored in the component `coefficients` of the model object, and they can be printed using the `summary` function. The number of parameters is equal to the number of levels in the factor. The intercept parameter is the mean of the response for the first level, and the remaining parameters represent the effect of this level on the mean of the first level. We can also obtain a confidence interval for each parameter:

```
> summary(model.power)
```

[8] The `aov` function can also be used. The difference is basically the presentation of the results.

```
Call:
lm(formula = power ~ type, data = data.sound)

Residuals:
      Min        1Q    Median        3Q       Max
-0.304940 -0.104136 -0.002825  0.076641  0.308359

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.07123    0.05169  59.416  < 2e-16
typeB2      -0.07609    0.06970  -1.092  0.28065
typeD4      -0.32267    0.07535  -4.282 9.32e-05
typeE1      -0.06081    0.07815  -0.778  0.44048
typeE6      -0.10966    0.07125  -1.539  0.13065
typeG3      -0.22355    0.07815  -2.861  0.00634

Residual standard error: 0.1551 on 46 degrees of freedom
Multiple R-squared: 0.3435,     Adjusted R-squared: 0.2721
F-statistic: 4.813 on 5 and 46 DF,  p-value: 0.001273
```

```
> confint(model.power)
```

```
                2.5 %      97.5 %
(Intercept)  2.9671822  3.17527524
typeB2      -0.2163871  0.06420548
typeD4      -0.4743378 -0.17099267
typeE1      -0.2181127  0.09649442
typeE6      -0.2530739  0.03376260
typeG3      -0.3808542 -0.06624708
```

The intervals corresponding to types D4 and G3 do not contain a zero value and therefore differ from the intercept. However, types B2, E1, and D6 do not differ with respect to A5 (intercept).

We can make pairwise comparisons between all the groups using the function `pairwise.t.test`:

```
> pairwise.t.test(power, type,
    p.adj = "bonferroni",
    data = ss.data.strings)
```

```
        Pairwise comparisons using t tests with pooled SD

data:  power and type

   A5      B2      D4      E1      E6
B2 1.00000 -       -       -       -
D4 2.9e-05 0.00018 -       -       -
E1 1.00000 1.00000 0.00023 -       -
E6 1.00000 1.00000 0.00304 1.00000 -
G3 0.00409 0.01852 1.00000 0.01882 0.16540

P value adjustment method: bonferroni
```

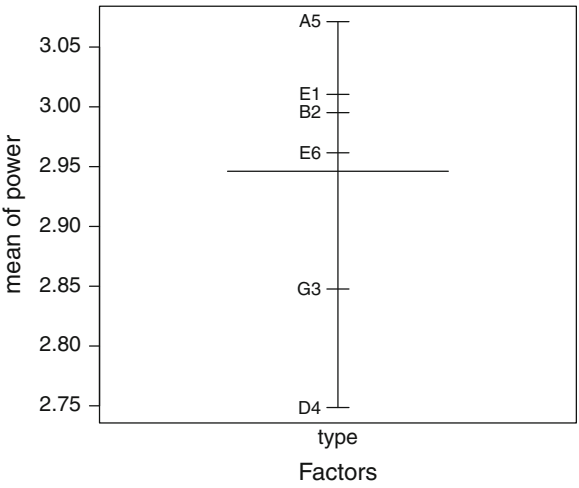

Fig. 10.6 Plot of effects in ANOVA for guitar string example. The mean of the power is lower for types G3 and D4

The output is a matrix with p-values corresponding to the individual hypothesis tests comparing the means. We can see that G3 and D4 are different from the rest of the string types, but there is no difference between them.

To visualize the effects, we can plot a simple chart using the `plot.design` function (Fig. 10.6) or a dot plot with the sample means linked by lines using the `ggplot2` package (Fig. 10.7).

```
> plot.design(power ~ type, data = data.sound)
```

```
> qplot(type, power, data = ss.data.strings) +
    stat_summary(fun.y = mean, geom = "line",
        aes(group = 1), col = "orangered") +
    stat_summary(fun.y = mean, geom = "point",
        shape = 17, size = 3, col = "red") +
    opts(title = "Effects of factor Type of string")
```

□

10.5.3 Additional Models and Related Tools

ANOVA is the suitable analytical tool to use after an experiment has been done. The correct design of that experiment is crucial for the results to be acceptable. Design of experiments (DoE) will be explained in some detail in Chap. 11.

When we are jointly analyzing factors and continuous variables as explanatory variables, analysis of covariance (ANCOVA) allows us to study interactions between both types of variables.

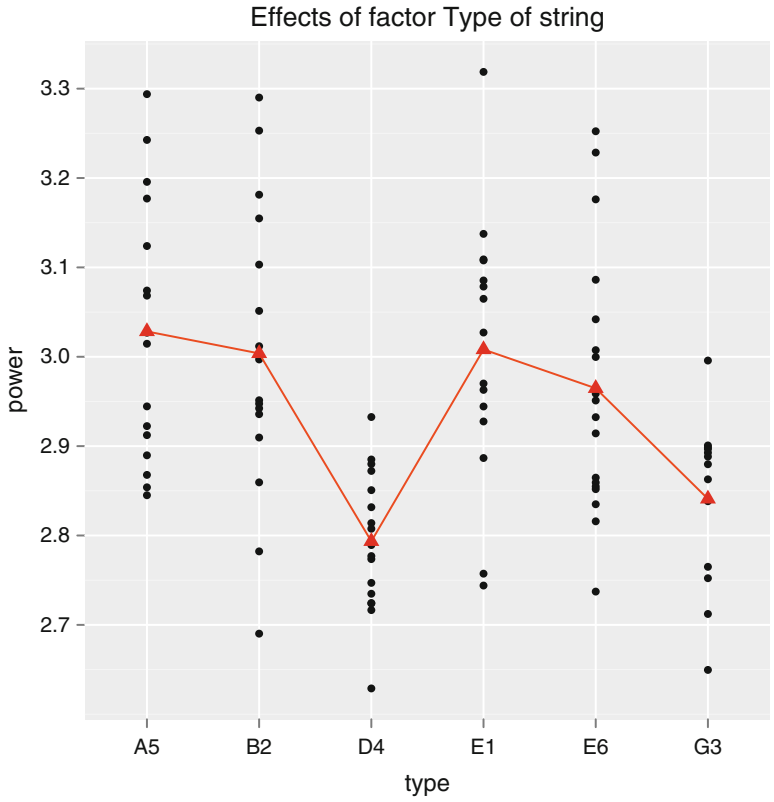

Fig. 10.7 Effects of string type. The means look quite different for types D4 and G3, with D4 having the lower mean value

Multivariate analysis of variance (MANOVA) is an extension that is used when more than one response variable may be influenced by some dependent variables. The R function `manova` accepts the same arguments as the `aov` function, expecting a matrix of continuous variables as the left-side component of the formula.

10.6 Summary and Further Reading

In this chapter we reviewed the basics of statistical inference. Point and interval estimation are the techniques used to make inferences about the parameters and probability distribution of a population. Hypothesis testing can be used throughout any inference analysis, with the interpretation results relying on the important concepts of *p*-value and null/alternative hypotheses.

Inference about the relationship of variables starts with linear models, such as regression or ANOVA. The main R functions and their outputs were explained. However, one of the most common statistical mistakes made in inference is to infer cause-and-effect correspondence from that relationship. It is not enough by itself, as there may be many other unstudied variables that are the real cause of some effect. DoE, described in the next chapter, will help to confirm those cause-and-effect relationships.

Extensions of the linear models can be read in [24]. The free resources [45] and [25] contain a number of R examples explaining the inference techniques reviewed in this chapter. The book [38], freely available on the book's Web site, is a nice reference on state-of-the-art prediction and estimation techniques. Regarding SVM, a complete review with applications is the work in [68]. References [18] and [20] discuss the application of statistics with R. Moreover, almost all books on Six Sigma contain entire chapters devoted to statistical inference.

Case Study

Estimate the parameters of the probability distribution of the flight time (mean and variance) and calculate a confidence interval. Test if the flight time follows a normal distribution. Obtain a confidence interval for the proportion of defects using any valid criterion for what constitutes a defect. Fit a regression model using as response variable the flight time and as independent variable any other measurable characteristic of the prototypes or the environment (e.g., temperature, wind speed). Determine if the estimated parameters are significant. Make an ANOVA using the flight time as the response and any categorical variable as the independent variable (e.g., a design characteristic, operator). Fit a model and determine if there are differences between the groups.

Practice

10.1. Test the normality hypothesis for the resistance of the strings.

10.2. Fit a linear model using the resistance of the strings as the response variable and the length of the strings as the independent variable. Is the linear model a good model to explain this relationship?

10.3. Determine whether the length of the strings is related to the type of string using ANOVA.