Chapter 4
Data Understanding

The main goal of data understanding is to gain general insights about the data that will potentially be helpful for the further steps in the data analysis process, but data understanding should not be driven exclusively by the goals and methods to be applied in later steps. Although these requirements should be kept in mind during data understanding, one should approach the data from a neutral point of view. Never trust any data as long as you have not carried out some simple plausibility checks. Methods for such plausibility checks will be discussed in this chapter. At the end of the data understanding phase, we know much better whether the assumptions we made during the project understanding phase concerning representativeness, informativeness, data quality, and the presence or absence of external factors are justified.

We first take a general look at single attributes in Sect. 4.1 and ask questions like: What kind of attributes do we have, and what do their domains look like? What is the precision of numerical values? Is the domain of an attribute stable over time, or does it change? We also need to assess the data quality. Methods and criteria for this purpose are introduced in Sect. 4.2.

Data understanding requires taking a closer look at the data. However, this does not mean that we must browse through seemingly endless columns of numbers and other values. In this way we would probably overlook most of the important facts. Looking at the data refers to visualization techniques (Sect. 4.3) that can be used to get a quick overview on basic characteristics of the data and enable us to check the plausibility of the data to a certain extent. Visualization techniques are suitable for the analysis of single attributes and of attributes in combination. Apart from the pure visualization, it is also recommended to compute simple statistical measures for correlation between attributes as described in Sect. 4.4.

Outliers, values, or records that are very different from all others should be identified with methods described in Sect. 4.5. They might cause difficulties for some of the methods applied in later steps, or they might be wrong values due to data quality problems. Missing values (see Sect. 4.6) can lead to similar problems as outliers, and by simply ignoring missing values we might obtain wrong data analysis results, so we must be aware of whether we have to deal with missing values and, if we have to, of what kind the missing values are.

Data understanding is also a step that is required for data preparation. For example, data understanding will help us to identify and to characterize outliers and missing values. However, how to treat them—whether to leave them as they are, to exclude them from further analysis steps, or to replace them by more plausible values—is a task for data preparation.

Throughout this chapter we will use the Iris data set [2, 7]: a set of 150 data points describing three different types of iris flowers (Iris setosa, Iris virginica, and Iris versicolor) using four different attributes measuring the length and width of the sepal and the petal leaves. This is a classic data set with a few very simple and obvious properties which lends itself naturally to demonstrate how the several different data analysis methods work. However, readers should not let themselves be fooled into believing that any real-world data sets will ever display just nice and well pronounced features. Quite the opposite: in real-world data the odd and undesired effects often far outweigh the interesting ones.

4.1 Attribute Understanding

In most cases, we assume that the data can be described in terms of a table or data matrix whose rows contain the **instances**, records, or **data objects** and whose columns represent the **attributes**, **features**, or **variables**. The data might not be stored directly in one table but in different tables from which the attributes of interest need to be extracted and joined into a single table. For instance, when we are interested in the correlation between the age of a customer and the number of budget products they buy, we would need a table with two columns representing the attributes *age* and *number of budget products bought*. The age of the customers can be extracted directly from the customer table (computing it from the birth date if necessary), whereas the number of budget products a customer has bought is not stored in any specific table but must be counted in the list of items he has bought.

An attribute or its **domain**—the set of possible values for the attribute—can have different principal characteristics. One of the most basic is the **scale type**: an attribute can be *nominal* (or *categorical*), *ordinal*, or *numeric* (details are provided below). Numeric attributes are further subdivided depending on whether they are *discrete* or *continuous* and whether they have an *interval*, *ratio*, or *absolute scale*.

An attribute is called **nominal** or **categorical** if its domain is a finite set. The possible values for a categorical attribute are often considered as classes or categories. For a purely categorical attribute, there is no additional structure on the domain. This means that two values or categories are either equal or different, but not more or less similar. Examples for categorical attributes are the gender with the values *F* (female) and *M* (male) in the customer data set or the species in the Iris data set.

For categorical attributes, there are sometimes different levels of **granularity**, and it is necessary to make a choice on which level the model for the analysis of the data should be built. The different levels of granularities form a hierarchy of more and more refined domains. A typical example for such refined levels of granularity are product categories. The general category might have the values *drinks, food,* ..., a more refined level for the drinks might distinguish between *water, beer, wine, ...*, and these categories might even be more refined by incorporating the producer. Even the same producer might offer different brands and variants of water, beer, or wine. This is still not the lowest level of refinement. We might even distinguish identical drinks by the size of the container in which they are sold, for instance, 0.5 l, 1.0 l, and 1.5 l bottles. Of course, the most refined level will always provide the most detailed information. Nevertheless, it is often not very useful to carry out an analysis on this level, since it is impossible to extract general structures or rules when we restrict our analysis on the refined level. A general rule like "Customers tend to buy wine and cheese together" might not be discovered on a refined level of granularity where no combination of a specific wine and specific cheese might be frequent. Therefore, it is crucial to choose an appropriate level of granularity for such attributes taking the aim of the analysis and the size of the data set into account. For smaller data sets, little statistical evidence can be found when we look at a very refined level, since the number of instances per value of the attribute decreases with the level of refinement. Especially, if a number of such attributes with different levels of granularity is considered, the number of possible combinations for the values grows extremely fast for refined levels. However, the choice of a suitable granularity is a task within data preparation. Nevertheless, we must make a decision here as well, on which level we want to understand and take a closer look at the data. We might do this even on different levels of granularity.

Another problem that sometimes comes along with categorical attributes is a dynamic domain. For certain categorical attributes, the domain will never change. The range of possible values for the month of birth is *January, ..., December*, and it seems quite improbable that it will be changed in the near future. The situation for products is, however, completely different. Certain products might not be offered by a shop anymore or might vanish completely from the market, whereas other new products enter the market, or already existing products are adopted by the shop.

Such a **dynamic domain** of an attribute can cause problems in the analysis later on. When products are analyzed on a long-term basis of, say, several years, then products that have entered the market just recently will not show significant (accumulated) sales numbers compared to products that have been in the market for decades. Therefore, categorical attributes can lead to undesired or even wrong analysis results when such problems like the level of granularity or dynamic domains are not taken into account. We identify such attributes already at this early stage and make sure that we do not forget to handle them later.

A specific type of categorical attributes are **ordinal** attributes with an additional linear ordering imposed on the domain. An attribute for university degrees with the values none, B.Sc., M.Sc., and Ph.D. represents an ordinal attribute. A Ph.D. is a higher degree than an M.Sc., and an M.Sc. is a higher degree than a B.Sc. However, the ordering does not say that the difference between a Ph.D. and an M.Sc. is the same as the difference between an M.Sc. and a B.Sc.

The domain of a **numerical** attribute are numbers. Numerical attributes can be **discrete**, usually taking integer values, or **continuous**, taking arbitrary real values. Discrete numerical attributes often result from counting processes like the number of children or the number of times a customer has ordered from an on-line shop in the last twelve months.

Sometimes, categorical attributes are coded by numerical values. For instance, the three possible values *food, drinks, nonfood* of the attribute *general product category* might be coded by the numbers 1, 2, and 3. However, this does not turn the attribute *general product category* into a (discrete) numerical attribute. We should bear this fact in mind for later steps of the analysis to avoid that the attribute is suddenly interpreted as a numerical attribute: it does not make sense to carry out numerical operations like computing the mean on such coded categorical attributes. For a discrete attribute, though, especially when it represents some counting process, it is meaningful to calculate the mean value, even though the mean value will usually not be an integer number. It is meaningful to say that on average the customers buy products 2.6 times per year in our on-line shop. But it does not make sense that the average *general product category* we sell is 2.6, which we might obtain when we simply compute the mean value of the products we have sold based on the numerical coding of the general product categories.

In contrast to discrete numerical attributes, a continuous attribute can—at least theoretically—assume any real value. However, such numerical values will always be measured and represented with limited precision. It should be taken into account how precise these values are. Drastic round-off errors or truncations can lead to problems in later steps of the analysis. Suppose, for instance, that a cluster analysis is to be carried out later on and that there is one numerical attribute, say X, that is truncated to only one digit right after the decimal point, while all other numerical attributes were measured and stored with a higher precision. When comparing different records, such truncation for the attribute X influences their perceived similarity and might be a dominating factor for the further analysis only for this reason. Truncation errors and measurements with limited precision should be distinguished from values corrupted with noise. The problem of noise will be tackled in the context of data quality in Sect. 4.2 and will also be discussed in more detail in Chap. 5.

Numerical attributes can have an *interval*, a *ratio*, or an *absolute scale*. For an interval scale, the definition of what zero means is more or less arbitrary. The date is a typical example for an attribute measured on an interval scale. There are calendars with different definitions of the time point zero. For instance, the Unix standard time, counted in milliseconds, has its time point zero in the year 1970 of the Gregorian calendar. The same applies to temperature scales like Fahrenheit and Celsius degrees, where zero refers to different temperatures. Certain operations like quotients are not meaningful for interval scales. For example, it does not make sense to say that a temperature of 21°C is three times as warm as 7°C.[1]

In contrast to this, a **ratio scale** has a canonical zero value and thus allows us to compute meaningful ratios. Examples of ratio scales are height, distance, or duration. Distance can be measured in different units like meters, kilometers, or miles.

[1]Such a statement may make sense, though, for the Kelvin temperature scale, because on this scale the temperature is directly proportional to the average kinetic energy of the particles—and it is meaningful to compute ratios of energies.

But no matter which unit we choose, a distance of zero will always have the same meaning. Especially ratios, which do not make sense for interval scales, are often useful for ratio scales: the quotient of distances is independent of the measurement unit, so that the distance 20 km is always twice as long as the distance 10 km, even if we change the unit kilometers to meters or miles. Whereas for a ratio scale, only the value zero has a canonical meaning and the meaning of other values depends on the choice of the measurement unit, for an **absolute scale**, there is a unique measurement unit. A typical example for an absolute scale is any kind of counting procedure.

4.2 Data Quality

The saying "garbage in, garbage out" applies to data analysis as to any other area. The results of an analysis cannot be better than the quality of the data, so that we should be concerned about the data quality before we carry out any deeper analysis with the data. **Data quality** refers to how well the data fit to their intended use. There are various data quality dimensions.

Accuracy is defined as the closeness between the value in the data and the true value. For numerical attributes, accuracy means how exact the value in the data set is compared to the true value. Noise or limited precision in measurements can lead to reduced accuracy for numerical attributes. Limited precision is often obvious from the data set. For example, in the Iris data set all numerical values are measured with only one digit right after the decimal point. The magnitude of noise can be estimated when measurements for the same value have been taken repeatedly. Accuracy of numerical values can also be affected by wrong or erroneous measurements or simply by errors like transposition of digits when measurements are recorded manually. For categorical attributes, problems with accuracy can result from misspellings like "fmale" for a value of the attribute *gender*, and also from erroneous entries.

We distinguish between *syntactic* and *semantic accuracy*. **Syntactic accuracy** means that a considered value might not be correct, but it belongs at least to the domain of the corresponding attribute. For a categorical attribute like *gender* for which only the values *female* and *male* are admitted, "fmale" violates syntactic accuracy. For numerical attributes, syntactic accuracy does not only mean that the value must be a number and not a string or text. Also certain numerical values can be out of the range of syntactic accuracy. Attributes like *weight* or *duration* will admit only positive values, and therefore negative values would violate syntactic accuracy. Other numerical attributes have an interval as their range like [0, 100] for the percentage of votes for a candidate. Negative values and values larger than 100 should not occur. For integer-valued attributes like the number of items a customer has bought, floating-point values should be excluded.

Problems with **semantic accuracy** mean that a value might be in the domain of the corresponding attribute, but it is not correct. When the attribute *gender* has the value *female* for the customer John Smith, then this is not a question of syntactic accuracy, since *female* is a possible value of the attribute *gender*. But it is obviously a wrong value for a person named "John".[2]

Discovering problems of syntactic accuracy in a data set is a relatively easy task. Once we know the domains of the attributes, we can easily verify, whether the values lie in the corresponding domains or not. A simple measure for syntactic accuracy is the fraction of values that lies in the domains of their corresponding attribute.

The verification of semantic accuracy is much more difficult or often even impossible. Another source for the same data would enable us to check our data, and differences not caused by problems with syntactic accuracy indicate problems with semantic accuracy. Sometimes also certain "business rules" are known for the data. For instance, if we find a record in our data set with the value *male* for the attribute *gender* and *yes* for the attribute *pregnant*, there must be a problem of semantic accuracy based on the known "business rule" that only women can be pregnant.

Whether or in which detail to check syntactic and semantic accuracy depends very much on how the data were generated. Especially, when data were entered manually, there is a higher chance for accuracy problems. In any case, it is recommended to carry out at least some simple tests to see whether there might be problems with accuracy. However, the usual practice is to keep these tests at a minimum and to find out later on that there are problems with accuracy, namely when the data analysis yields implausible results.

Throughout this book we normally assume that the data are already given, for example, as a database table. This is not the best point in time to cope with data quality problems. Chances of avoiding or reducing data quality problems are highest when the data are entered into the database. For instance, instead of letting a user type in the value of categorical attribute with the danger of misspellings, one could provide a fixed selection of values from which the user can choose.

Another dimension of data quality is **completeness** which can be divided into completeness with respect to attribute values and completeness with respect to records. Completeness with respect to attribute values refers to missing values (which will be discussed in Sect. 4.6). When missing values are explicitly marked as such, then a simple measure for this dimension of data quality is the fraction of missing values among all values. But we will see that missing values are not always directly recognizable, so that the fraction of known missing values might only provide a lower bound for the fraction of actually missing values.

Completeness with respect to records means that the data set contains the necessary information that is required for the analysis. Some records might simply be missing for some technical reasons. Data might have been lost because a few years ago the underlying database system was changed and only those data records were transferred to the new database that were considered to be important at that point in time. In a customer database, customers who had not bought anything for a longer time might not have been transferred (in order to eliminate potential

[2]Note, however, that the problem may also reside with the name. Maybe the name of the person was misspelled, and the correct name is "Joan Smith"—then the gender is actually *female*.

zombie customers) to the new database, or older transactions were not stored anymore.

Very often the available dataset itself is biased and not representative. Consider as an example a bank that provides mortgages to private customers. If the aim of the analysis is to predict for future applicants of loans whether they will return the loan, we must take into account that the sample is biased in the sense that we only have information about those customers who have been granted a loan. For those customers who have been denied the loan initially, we have no information whether they would have returned the loan or not. But especially these customers might be the ones for which it is interesting to find a good scheme to predict the risk. For customers with high income and a safe job and no current debt, we need no sophisticated data analysis techniques to predict that there is a good chance that they will return the loan. Of course, it is impossible to obtain a representative sample in the statistical sense in this case. Such a sample would mean that we would have to provide loans to any customer, no matter how bad their financial status is, for a certain period and collect and evaluate these data. Unfortunately, this would be a method entailing almost guaranteed bankruptcy.

The same problems occur in many other areas. For a production plant, we usually have large amounts of data when it is running in a normal mode. For exceptional situations, we will have little or no data. We cannot ask for such data, for instance, by requiring to check what happens if, say, a nuclear plant operates at its limit.

In such cases we might encounter future situations for which we had no corresponding data in our sample. Such possible gaps in the data should be identified. One should be aware that the space of possible values is automatically covered sparsely by the data when we have a larger number of attributes. Consider a set of m numerical attributes, and we want to make sure that we have at least positive and negative values for each attribute in our data set. This does not require a large data set. But if we want to make sure that we have data for all combinations of positive and negative values for the considered attributes, this leads to 2^m possible combinations. If we have $m = 20$ attributes, we have already more than one million possible combinations of positive and negative values. Therefore, if we have a data set with one million records, we have on average one sample for each of these combinations. For a data set with 100,000 records, at least 90% of the combinations will not be covered.

Other problems can be caused by **unbalanced data**. As an example, consider a production line for goods for which an automatic quality control is to be installed. Based on suitable measurements, a classifier is to be constructed that sorts out parts with flaws or faults. The scrap rate in production is usually very small, so that our data might contain far less than 1% examples for parts with flaws or faults.

Timeliness refers to whether the available data are too old to provide up to date information or cannot be considered as representative for predictions of future data. Timeliness is often a problem in dynamically changing domains, where only recently collected data provide relevant information, while older data can be misleading and can indicate trends that have vanished or even reversed.

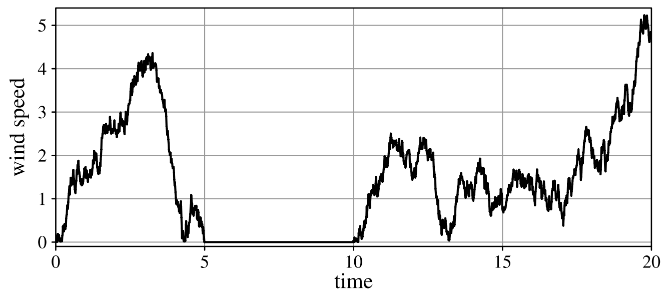


Fig. 4.1 Measured wind speeds with a period of a jammed sensor

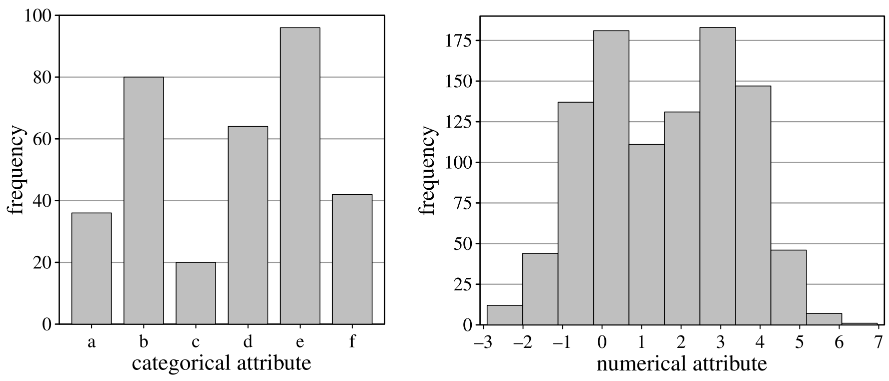


Fig. 4.2 A bar chart (categorical attribute, *left*) and a histogram (numerical attribute, *right*)

4.3 Data Visualization

According to Tukey [24], "there is no excuse for failing to plot and look" when one wants to handle a data analysis problem. The right plots of the data can provide valuable information as the simple time series plot in Fig. 4.1 shows, which enables us to discover zero values that are actually missing values. There are infinitely many ways to plot data, and it is not always easy to find the best ways of plotting a given data set. Nevertheless, there are some very useful standard data visualization techniques that will be discussed in the following.

4.3.1 Methods for One and Two Attributes

A **bar chart** is a simple way to depict the frequencies of the values of a categorical attribute. A simple example for a categorical attribute with six values a, b, c, d, e, and f is shown on the left in Fig. 4.2.

A **histogram** shows the frequency distribution for a numerical attribute. To this end, the range of the numerical attribute is discretized into a fixed number of inter-

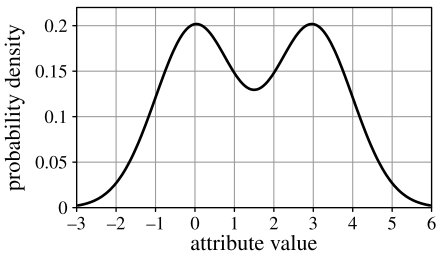


Fig. 4.3 Probability density function of a sample distribution, from which the data for the histogram in Fig. 4.2 was sampled

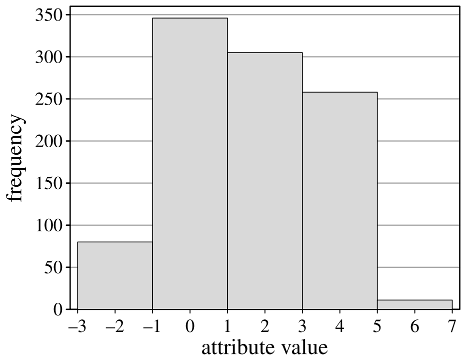


Fig. 4.4 A histogram with too few bins, for the same data set that was depicted in Fig. 4.2 on the right. As a consequence, the distribution looks unimodal but skewed

vals (called *bins*), usually of equal length. For each interval, the (absolute) frequency of values falling into it is indicated by the height of a bar as shown on the right in Fig. 4.2. From this histogram it can be read, for example, that a little bit more than 100 values lie in the vicinity of 1.

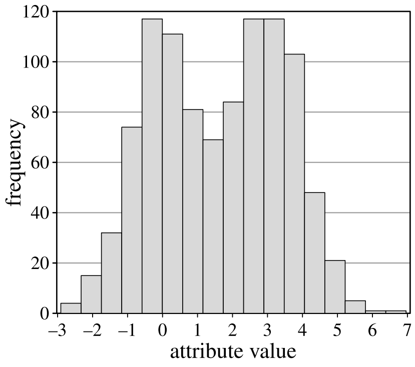
The histogram in Fig. 4.2 resulted from a sample of size 1000 from a mixture of two normal distributions with means 0 and 3, respectively, having both a standard deviation of 1. The density of this distribution is shown in Fig. 4.3.

But how should we choose the number of bins, and how much does this choice influence the result? Figure 4.4 shows a histogram with only five bins for the same data set underlying the histogram shown in Fig. 4.2. With only five bins, the two peaks of the original distribution are no longer visible, and one gets the wrong impression that the distribution is unimodal but skewed.

There is no generally best choice for the number of bins, but there are certain recommendations. **Sturges' rule** [22] proposes to choose the number k of bins according to the following formula:

$$k = \lceil \log_2(n) + 1 \rceil, \tag{4.1}$$

where n is the sample size. Although Sturges' rule is still very often used as a default in various statistics software packages, it is tailored to data from normal distributions and data sets of moderate size [21]. The number of bins of the histogram in Fig. 4.2 has been computed based on Sturges' rule. The size of the data set is $n = 1000$.



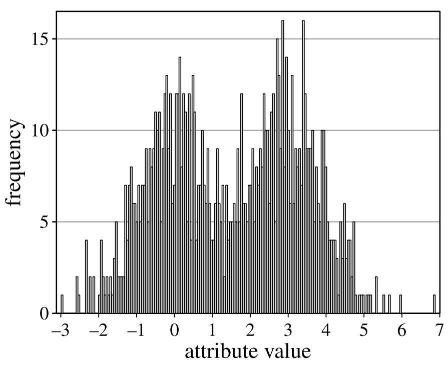


Fig. 4.5 Histograms with a suitable choice for the number of bins (*left*) and too many bins (*right*)

Assuming that, as in Sturges' rule, the bins have equal length, the number of bins can also be determined based on the length h of each bin:

$$k = \left\lceil \frac{\max_i\{x_i\} - \min_i\{x_i\}}{h} \right\rceil, \tag{4.2}$$

where $x_1, \dots, x_n$ is the sample to be displayed. Reasonable values for h are [20]

$$h = \frac{3.5 \cdot s}{n^{\frac{1}{3}}}, \tag{4.3}$$

where s is the sample standard deviation, and [8]

$$h = \frac{2 \cdot \mathrm{IQR}(x)}{n^{\frac{1}{3}}}, \tag{4.4}$$

where $\mathrm{IQR}(x)$ is the interquartile range of the sample, that is, the length of the interval which covers the middle 50% of the data.

For the data set with the histogram displayed in Fig. 4.2, (4.3) yields $k = 16$, and (4.4) leads to $k = 17$. A histogram for the second choice (that is, for $k = 17$) is shown on the left in Fig. 4.5.

As we have seen in Fig. 4.4, the histogram can be misleading when the number of bins is chosen too small. Choosing the number of bins too high usually leads to a very scattered histogram in which it is difficult to distinguish true peaks from random peaks. An example is shown on the right in Fig. 4.5, where $k = 200$ was chosen for the number of bins for the same data underlying Fig. 4.2.

All of these methods (that is, (4.2), (4.3), and (4.4) for determining the number of bins or the length of the bins) are highly sensitive to outliers, since they divide the range between the smallest and the largest value of the sample into bins of equal size. A single outlier can make this range extremely large, so that for a smaller number of bins, the bins themselves become very large, and for a larger number of bins, most of the bins can be empty. To avoid this problem, one can either leave out extreme values from the sample (for instance, the 3% smallest and the 3% largest values) for calculating and displaying the histogram, or one can deviate from the principle of bins of equal length.

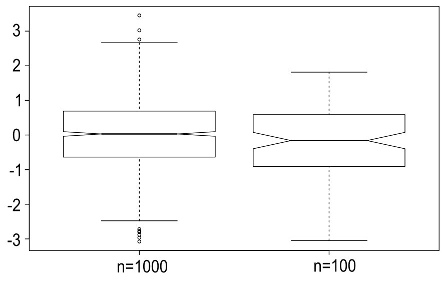


Fig. 4.6 Two boxplots for a sample from a standard normal distribution

Boxplots are a very compact way to visualize and summarize main characteristics of a sample from a numerical attribute. Figure 4.6 shows two boxplots from samples from a standard normal distribution with mean 0 and variance 1. The left boxplot is based on sample of size $n = 1000$, whereas a sample of size $n = 100$ was used for the right boxplot.

The line in the middle of a boxplot indicates the sample median. The notch in the box is not always shown. It indicates a 95% confidence interval for the median. The box itself corresponds to the interquartile range covering the middle 50% of the data. The whiskers are drawn in the following way. The maximum length of each whisker is 1.5 times the length of the interquartile range. But if there is no data point at the maximum length of a whisker, the corresponding whisker is shortened until it reaches the next data point. Data points lying outside the whiskers are considered as outliers and are indicated in the form of small circles.

Comparing the two boxplots in Fig. 4.6, we can observe the following:

- Although both boxplots come from samples from the same normal distribution, they look different, since they are based on different samples.
- The notch of the left boxplot, representing a 95% confidence interval for the median, is much smaller than the notch of the right boxplot because of the larger sample size for the left boxplot.
- Theoretically, the whiskers for a sample from a symmetric distribution like the normal distribution should have roughly the same length. For the boxplot based on the smaller sample size, we can see that whiskers differ significantly in length, since—by chance—the largest value among the sample of 100 values was not greater than 2, whereas the smallest value was smaller than -3.
- In contrast to the boxplot on the left-hand side, the right boxplot does not contain any outliers. This is again due to the smaller sample size. The theoretical length of the interquartile range for a standard normal distribution is 1.349. Therefore, the probability of a point lying outside the (theoretical) range $[-2.698, 2.698]$ of the whiskers is almost 0.7%. Therefore, for a sample from a normal distribution of size $n = 1000$, we can expect roughly 7 outliers on average in a boxplot and less than one for a sample of size $n = 100$.

The boxplots of asymmetric distributions look completely different. If we sample from an exponential distribution, whose probability density function is shown in Fig. 4.7, we obtain boxplots as they are shown in Fig. 4.8. The boxplots on the left and right represent samples of sizes $n = 1000$ and $n = 100$, respectively.

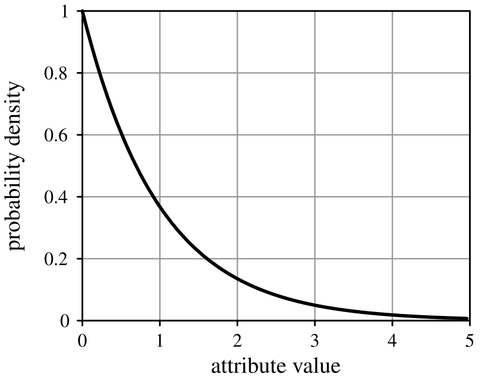


Fig. 4.7 The probability density function of the exponential distribution with $\lambda = 1$

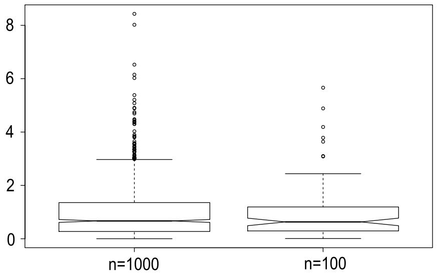


Fig. 4.8 Two boxplots for a sample from an exponential distribution

Barcharts, histograms, and boxplots are visualizations for single attributes. In most cases, we have to deal with a number of attributes, and we are not only interested in the characteristics of single attributes but also in the relations and dependencies between the attributes. However, the display for visualizing the data is two-dimensional, and even if we use 3D-techniques from computer graphics, we cannot directly display more than two or three variables at the same time in a simple coordinate system, unless we use additional features such as symbols, color and size. **Scatter plots** refer to displays where two attributes are plotted against each other. The two axes of the coordinate system represent the two considered attributes, and each instance in the data set is represented by a point, a circle, or any other symbol.

Simple scatter plots are not suited for larger data sets. For a data set with one million objects and a window size of 500 $\times$ 500 pixels, we would have on average four data objects per pixel. For larger data sets, many points or symbols in the scatter plot will be plotted at the same position, and we cannot distinguish whether a point in the scatter plot represents one or 100 objects. In the worst case, a scatter plot for a larger data set might simply look like the one in Fig. 4.9, providing only information about the range of the data, but no hint concerning the distribution of the data.

This can be amended by using **density plots** or plots based on **binning**. Using semitransparent points for plotting the data is one way to generate a density plot. Each plotted point is semitransparent, and the more points are plotted at the same place, the less transparent the image will become in this place. Binning was already used to generate histograms, and the principle is used for the scatter plots. The two-

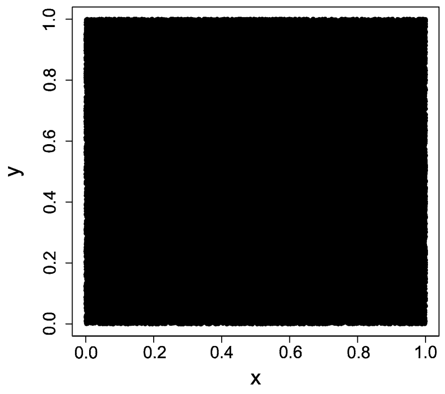


Fig. 4.9 A scatter plot of a data set with $n = 100{,}000$ instances

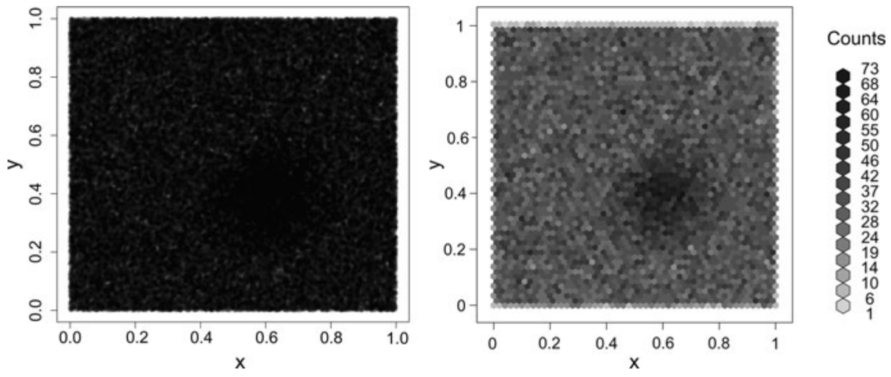


Fig. 4.10 Density plot (*left*) and a plot based on hexagonal binning (*right*) for the same data set as shown in Fig. 4.9

dimensional domain of the data for the scatter plot is partitioned into bins of the same size. Possible forms for the bins are rectangles or hexagons. The intensity of the color for the bin is chosen proportional to the number of data objects falling into the bin. Figure 4.10 shows a density plot on the left and a plot based on a hexagonal binning on the right for the same data set displayed in Fig. 4.9. Both plots indicate a higher density of the data around the point $(0.6, 0.4)$, which cannot be seen in the simple scatter plot in Fig. 4.9.

Scatter plots can be enriched with further information in order to involve more attributes. Different plot symbols or colors can be used for plotting the points in order to include information about a categorical attribute. Color intensity and the size of the symbols are possible means to indicate the value of additional numerical attributes.

Figure 4.11 shows two scatter plots of the Iris data set—one displaying the sepal length versus the sepal width and the other one the petal length versus the petal width—in which different species are displayed by different colors. Both plots show that the red circles, representing the species Iris setosa, can be well distinguished from the other two species Iris versicolor and Iris virginica displayed as triangles

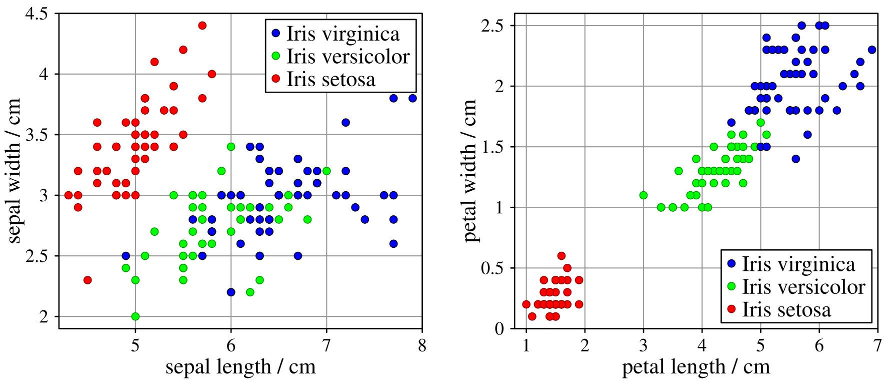


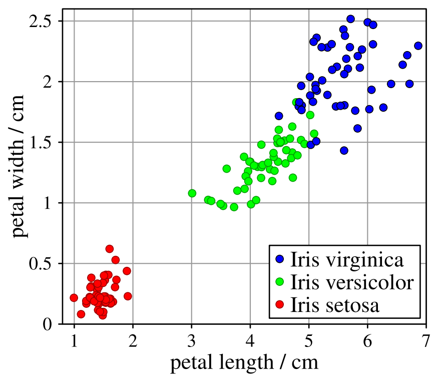
Fig. 4.11 Scatter plots of the iris data set for sepal length vs. sepal width (*left*) and for petal length vs. petal width (*right*). All quantities are measured in centimeters (cm)

and crosses, respectively. However, the left chart in Fig. 4.11 gives the impression that Iris virginica and Iris versicolor are very difficult to distinguish, at least when we only take the sepal length and the sepal width into account. But when we consider the petal length and the petal width (right chart in Fig. 4.11), we can still see the overlap of the corresponding symbols for the species, but there is a clear tendency that Iris virginica tends to larger values than Iris versicolor for the petal length and width.

Comparing the number of red circles in Figs. 4.11 (left and right), there seem to be less red circles on the right. But how can some of the objects suddenly vanish in the scatter plot? When we count the number of red circles, we see that in both scatter plots there are less than 50, although the data set contains 50 instances of Iris setosa that should be displayed by red circles. The circles are not missing in the scatter plots. Some circles are simply plotted at exactly the same position, since their measured sepal length and width or their measured petal length and width coincide. Recall that these values were only measured with a precision of just one digit right after the decimal point. To avoid this impression of seeing less objects than there actually are, one can add **jitter** to the scatter plot. Instead of plotting the symbols exactly at the coordinates specified by the values in the data set, we add a small random value to each original value in the data table. The left chart in Fig. 4.12 shows the resulting scatter plot with jitter where we have added random values from a uniform distribution on the interval $[-0.04, 0.04]$ to the original values. This ensures that a point originally lying left or below another point will always remain left or below the other point, even when the jitter is added.

Jitter is essential when categorical attributes are used for the coordinate axes of a scatter plots, since categorical attributes have only a limited number of possible values, so that plotting of objects at exactly the same position occurs very often when no jitter is added.

From a scatter plot we can already extract important information. Consider again the scatter plot displayed in Fig. 4.12. We can see that the petal length and width are correlated. Objects with larger values for the petal length also tend to have larger



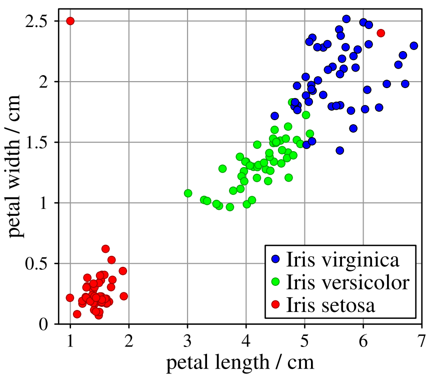


Fig. 4.12 The same scatter plot as in Fig. 4.11 on the right, but with jitter (*left*) and with jitter and two outliers (*right*; the outliers are the *red points* in the top left and top right corners)

values for the petal width. The scatter plot also shows that Iris setosa—the red circles in the scatter plot—can be easily distinguished from the other two species just on the basis of the petal length or width. The scatter plot does not indicate that the other two species cannot be separated clearly. It only shows that, solely based on the petal length and width, it is not possible to distinguish the two species perfectly. Outliers can also be discovered in scatter plots. The left chart in Fig. 4.12 does not have any outliers. In the right chart, however, we have added two artificial outliers. The data point in the upper left corner is a clear outlier with respect to the whole data set. Note that the values for the attributes petal length and width are both in the general range of the corresponding attributes in the data set. But there is no other object in the data set with a similar combination of these attribute values. The second outlier in the right chart of Fig. 4.12—the circle in the upper right corner—is not an outlier with respect to the values for the petal length and width or their combination. However, it is an outlier for the class Iris setosa displayed by red circles. Whenever such outliers are discovered, one should check the data or the data generating process again to ensure that the outliers are not due to erroneous data.

It should be noted that the scatter plots—like all other visualization techniques—are very useful tools to discover simple structures and patterns or peculiar deviations like outliers in a data set. But there is no guarantee that a scatter plot or any visualization technique will automatically show all or even any interesting or deviating pattern in the data set. A scatter plot with no outliers does not mean that there are no outliers in the data set. It only means that there are no outliers with respect to the combination of the attributes displayed in the scatter plot. In this sense, visualization techniques are like test cases for computer programs. Test cases can discover errors in a program. But if the test cases have not indicated any errors, this does not imply that the program does not have any bugs. In the same way, a visualization technique might give hints to certain interesting patterns in the data set. But if one cannot see any interesting patterns in a visualization, it does not mean that there are no patterns in the data set.

4.3.2 Methods for Higher-Dimensional Data

In this section, we restrict our considerations to numerical attributes. Scatter plots are projections of the data set to a two-dimensional plane where the two dimensions of the projection plane correspond to two selected attributes. The data set can be considered as a subset of a space that has as many dimensions as the data set has attributes. How can such a higher-dimensional data set be represented in two or three dimensions? The idea is to preserve as much of the "structure" of the higher-dimensional data in the lower-dimensional representation. But what does it mean to preserve the structure? There is no unique answer to this question, and various approaches have been proposed to solve this problem. A very efficient method is principal component analysis that will be introduced in the next section.

4.3.2.1 Principal Component Analysis

Principal component analysis (**PCA**), which is also briefly described in the appendix on statistics, is a method from statistics to find a projection to a plane—or more generally, to a linear subspace—which preserves as much as possible of the original variance in the data. In order to restrict the search for the best projection plane to planes through the origin of the coordinate system, the data are first centered around the origin by subtracting the mean value for each attribute from the attribute values. In this way, the projection to the plane can be represented by a matrix M mapping the data points $\mathbf{x} \in \mathbb{R}^m$ to the plane by

$$\mathbf{y} = M \cdot (\mathbf{x} - \bar{\mathbf{x}}), \tag{4.5}$$

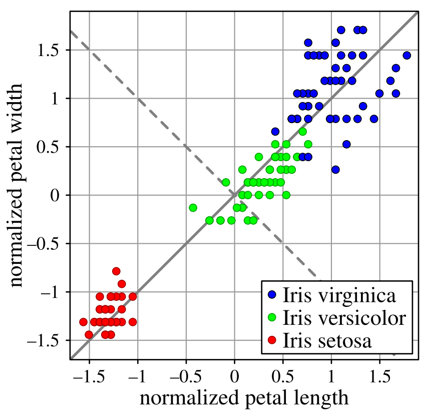
where $\bar{\mathbf{x}}$ denotes the (empirical) mean value (or vector of (empirical) mean values)

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i .$$

M is a $2 \times m$ matrix when PCA is used for visualizing the data on a plane. As we will see later on, PCA can be understood as a more general dimension reduction technique. Therefore, we do not restrict our considerations to projections to a plane of dimension 2 but to any linear subspace of dimension $q \le m$. In this case, M is a $q \times m$ matrix. It should be noted that arbitrary matrices are not admitted for the projection, since only projections, but no scalings, are allowed. Scalings would enable the overall variance of the projection to be changed, which is not desired.

The solution of the optimization problem of finding the projection plane such that the variance of the projected data is maximized leads to an eigenvalue problem. The projection matrix M for PCA is given by

$$M = (\mathbf{v_1}, \ldots, \mathbf{v_q}),$$



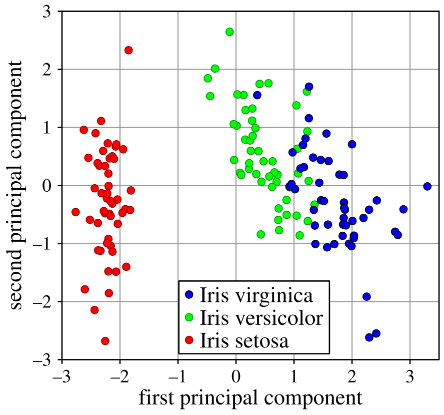


Fig. 4.13 *Left*: the first (*solid line*) and the second principal component (*dashed line*) of an example data set (Iris data). *Right*: the example data set projected to the space that is spanned by the first and second principal components (resulting from a PCA involving all four attributes)

where $\mathbf{v_1}, \ldots, \mathbf{v_q}$ are the normalized eigenvectors[3] of the **covariance matrix** of the data

$$C = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})^\top$$

for the q largest eigenvalues $\lambda_1 \geq \cdots \geq \lambda_q$.

The vectors $\mathbf{v_1}, \ldots, \mathbf{v_q}$ are called **principal components**. Figure 4.13 illustrates the concept of principal components on a two-dimensional example. Of course, for such an example, a dimension reduction is not necessary. There are two principal components. The vector corresponding to the largest eigenvalue is the first component, and its direction is indicated by the solid line. The dashed line represents the direction of the second component, corresponding to the second largest eigenvalue, which in this two-dimensional case is the smallest eigenvalue. It is obvious that the projection of the data set to the first principal component preserves a large fraction of the original variance, whereas a projection to the second principal component only would lead to a significant loss of the original variance.

The left chart in Fig. 4.13 is a plot of the petal length and width of the Iris data set. However, apart from the necessary centering of the data around the origin by subtracting the mean, the data have been z-score standardized by the transformation

$$x \mapsto \frac{x - \hat{\mu}_X}{\hat{\sigma}_X}, \tag{4.6}$$

where $\hat{\mu}_X$ and $\hat{\sigma}_X$ are the mean value and empirical standard deviation of attribute X (see also Sect. 6.3.2). Without standardization, the result of PCA would depend on

[3] λ is called an eigenvalue of a matrix A if there is a nonzero vector $\mathbf{v}$ such that $A\mathbf{v} = \lambda\mathbf{v}$. The vector $\mathbf{v}$ is called an eigenvector to the eigenvalue λ.

Table 4.1 Preservation of the variance of the Iris data set depending on the number of principal components

	Principal component			
	PC1	PC2	PC3	PC4
Proportion of variance	0.73	0.229	0.0367	0.00518
Cumulative proportion	0.73	0.958	0.9948	1.00000

the scaling of the attributes. If no standardization is carried out, the attribute with the largest variance can easily dominate the first principal component. For the example in Fig. 4.13 with z-score standardization, the first principal component is the vector $(\sqrt{2}/2, \sqrt{2}/2)$. If the petal length is measured in meters instead of centimeters, but the petal width is still measured in centimeters, the first principal component without z-score standardization becomes the vector $(0.0223, 0.9998)$, since the variance of the petal length has been decreased drastically by the scaling factor 0.01 resulting from the change from centimeters to meters, so that more or less only the petal width contributes to the variance in the data.

PCA can be used for visualization purposes by restricting to the first two principal components. More generally, PCA can carry out a dimension reduction to any lower-dimensional space; even more, PCA also provides information about over how many dimensions the data set actually spreads. This information can be extracted from the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_m$ of the covariance matrix. When we project the data to the first q principal components $v_1, \ldots, v_q$ corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_q$, this projection will preserve a fraction of

$$\frac{\lambda_1 + \cdots + \lambda_q}{\lambda_1 + \cdots + \lambda_m} \tag{4.7}$$

of the variance of the original data. Table 4.1 shows the corresponding result of PCA applied to the Iris data set without the categorical attribute for the species. A projection of this four-dimensional data set to the first principal component, i.e., to only one dimension, covers already 73% of the variance of the original data set. A projection to a plane defined by the first two principal components covers already 95.8% of the variance. This means that the four numerical attributes of the Iris data set are not located on a two-dimensional plane in the four-dimensional space but do not deviate too much from the plane defined by the first two principal components. The right chart in Fig. 4.13 shows the projection of the Iris data set to the first two principal components where PCA was carried out after the z-score standardization had been applied.

The importance of the scaling effect carried out by the z-score standardization can also be observed by revisiting the example where we had considered only the petal length and width of the Iris data set. We had applied PCA to the original data and the data where we had changed the measurement of the petal length from centimeters to meters without scaling, resulting in the vector $(0.7071, 0.7071)$ as the first principal component for the original data and the vector $(0.0223, 0.9998)$ for the modified data. The variance preserved by the projection to the first principal component is 98.1% in the first case and 99.996% in the second case. In the latter case, the first principal component corresponds more or less to the petal width,

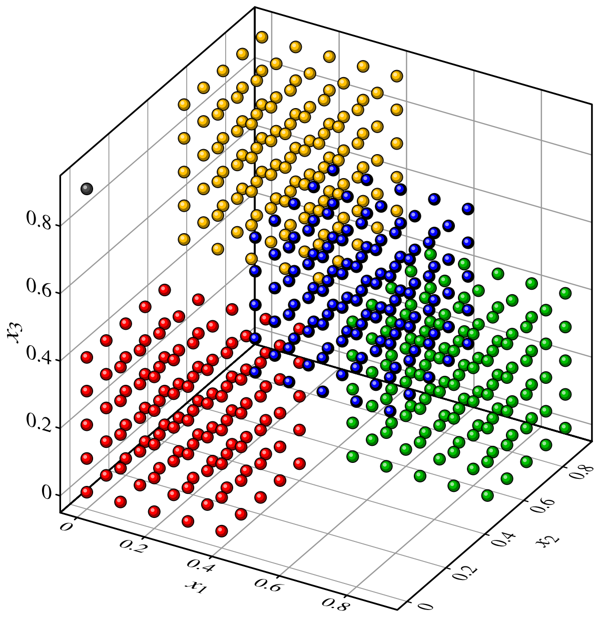


Fig. 4.14 A data set distributed over a cube in a chessboard-like pattern. The colors are only meant to make the different cubes more easily discernible. They do *not* indicate classes. Note the outlier in the upper left corner

since the variance of the petal length measured in meters is almost negligible, and therefore the projection can preserve close to 100% of the original variance.

In order to illustrate the advantages and limitations of PCA for visualization purposes, we consider an artificial three-dimensional data set illustrated in Fig. 4.14. The data fill the unit cube in chessboard-like manner. When the unit cube is divided into eight subcubes, these subcubes are alternatingly empty and filled with data. There is also one outlier close to the upper left corner of the surface in the front of the cube. The scatter plots resulting from projections to two axes of the coordinate system are shown in Fig. 4.15. All these scatter plots give the wrong impression that the data are uniformly distributed over a grid in the data space. The scatter plots provide neither a hint to the chessboard pattern in the three-dimensional data space nor to the single outlier.

Figure 4.16 shows the projection to the first two principal components of the data set after a z-score standardization has been carried out. The outlier can now be identified easily. From Fig. 4.16 it is also obvious that the data cannot be distributed uniformly over the original three-dimensional space but that there must be some inherent pattern. Of course, it is impossible to recover the three-dimensional chessboard pattern in a two-dimensional projection completely.

4.3.2.2 Projection Pursuit

PCA has the advantage that the best projection with respect to the given criterion—the preservation of the variance—can be computed directly based on the eigenvectors of the covariance matrix. **Projection pursuit** [10] takes a different approach.

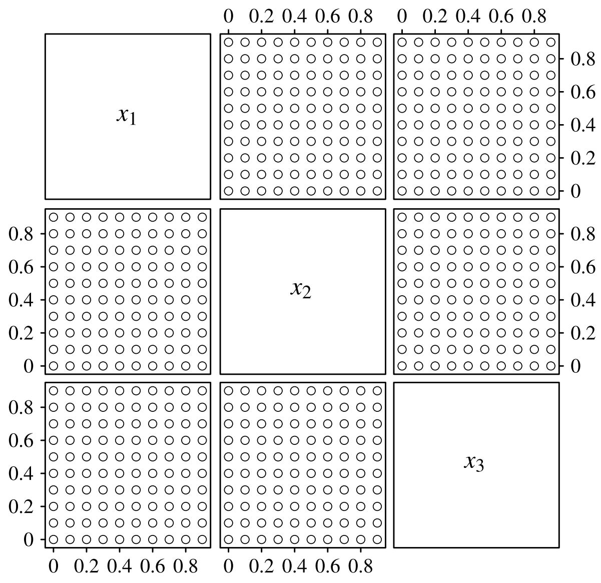


Fig. 4.15 The scatter plots for the data set shown in Fig. 4.14

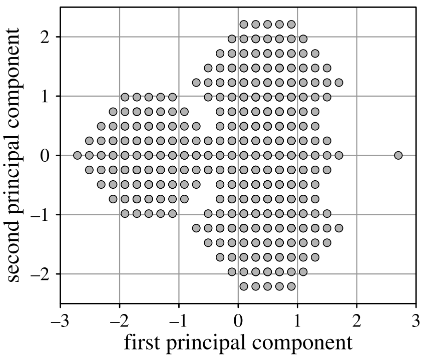


Fig. 4.16 The projection to the first two principal components of the data set shown in Fig. 4.14

The projection of the data should show interesting aspects of the data. But what does interesting mean? Normally, for projection pursuit, interestingness of a projection is defined as the deviation from a normal distribution, according to the observation that most of the projections of high-dimensional data will resemble a normal distribution [6]. The more the projected data deviate from a normal distribution, the more interesting is the projection. Various criteria [5, 9–12] can be defined to measure how much a projection deviates from a normal distribution.

In contrast to PCA, there is no way to find the most interesting projections with respect to the deviation from a normal distribution. Projection pursuit simply generates random projections and chooses the ones yielding the best values for the measures of interestingness.

4.3.2.3 Multidimensional Scaling

PCA is a method for dimension reduction that is based on two assumptions:

- The representation of the data in a lower-dimensional space should be obtained by a projection to a linear subspace.
- The criterion to evaluate the representation of the data in the lower-dimensional space is the preservation of the variance.

Projection pursuit shares the first assumption of PCA but replaces the second one by the deviation of the projection from the normal distribution. **Multidimensional scaling** (**MDS**) is also a dimension-reduction technique but differs in both assumptions from PCA and projection pursuit.

- MDS is not restricted to mappings in the form of simple projections. In contrast to PCA, MDS does not even construct an explicit mapping from the high-dimensional space to the low-dimensional space. It only positions the data points in the low-dimensional space.
- The representation of the data in the low-dimensional space constructed by MDS aims at preserving the distances between the data points and not the variance in the data set, for which PCA was designed.

For a data set with n data objects, MDS requires a distance matrix $[d_{ij}^{(X)}]_{1 \le i,j \le n}$, where $d_{ij}^{(X)}$ is the distance between data object i and data object j. MDS assumes that the distance is symmetric, i.e., $d_{ij}^{(X)} = d_{ji}^{(X)}$ for all $i, j \in \{1, \ldots, n\}$. The distances $d_{ii}^{(X)}$ must be zero, so that each data object has no distance to itself. MDS does not require the data, but only the distance matrix. Usually, the distance matrix is derived from the data by computing the Euclidean distances between the data objects (see Sect. 7.2 for more details on distance measures).

It is important to note that it is recommended to carry out a normalization of the data first, for instance, by applying a z-score standardization. Without such a normalization, MDS has to cope with the same problems as PCA. The variance and the difference between values of an attribute highly depend on the scaling or the measurement unit of the attribute. In the same way as a single attribute or few attributes might dominate the variance, just because the corresponding measurement units tend to larger values, these attributes will contribute more to the Euclidean distance than attributes with very small values and ranges.

Since MDS is mainly used for visualization purposes, the original high-dimensional data should be represented by points in two or sometimes also in three dimensions. Each data object should be represented by a point in the low-dimensional space, usually in $\mathbb{R}^2$ or $\mathbb{R}^3$. Let us restrict our considerations for the moment to a two-dimensional visualization. Then MDS must define a point $\mathbf{p}_i = (p_x^{(i)}, p_y^{(i)})$ to be plotted in the plane for each data object x_i. The plot of these points should preserve the original distances $d_{ij}^{(X)}$ between the data points in the original high-dimensional space. Of course, it is usually impossible to preserve the distances exactly in the two-dimensional representation. Therefore, we require that the distances $d_{ij}^{(Y)} = \|\mathbf{p_i} - \mathbf{p_j}\|$ between the points in the two-dimensional representation should deviate as little as possible from the original distances $d_{ij}^{(X)}$. One way to measure this deviation is the sum of the squared errors between the original distances and the distances in the two-dimensional representation.

$$E_0 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \big(d_{ij}^{(Y)} - d_{ij}^{(X)}\big)^2. \tag{4.8}$$

This equation does not refer explicitly to the two-dimensional representation. The distances $d_{ij}^{(Y)}$ in the lower-dimensional space can be derived from points in $\mathbb{R}^2$, but the distances could also be computed for points in $\mathbb{R}^q$ for any $q \in \mathbb{N}$.

The sum of squared errors depends on the number of data objects and on the values for the original distances. For more data points and for larger distances $d_{ij}^{(X)}$, E_0 will tend to become larger as well. In order to obtain an error measure independent of these effects, E_0 is often normalized to

$$E_1 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (d_{ij}^{(X)})^2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \big(d_{ij}^{(Y)} - d_{ij}^{(X)}\big)^2. \tag{4.9}$$

The factor $\frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (d_{ij}^{(X)})^2}$ is independent of the distances $d_{ij}^{(Y)}$ and therefore independent of the lower-dimensional representation of the data. It does not affect the result of the minimization of E_0 or E_1.

The relative error

$$E_2 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_{ij}^{(Y)} - d_{ij}^{(X)}}{d_{ij}^{(X)}}\right)^2 \tag{4.10}$$

is an alternative to the absolute error in (4.9). Very often, neither the absolute nor the relative error is considered for MDS, but a compromise between them given by

$$E_3 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}^{(X)}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(d_{ij}^{(Y)} - d_{ij}^{(X)})^2}{d_{ij}^{(X)}}. \tag{4.11}$$

When MDS is carried out based on the error measure E_3, it is also called **Sammon mapping**. The error E_3 for a concrete representation of the data in the lower-dimensional space is called **stress**.

So far, we have only proposed error measures for MDS, but we still need to find a way to minimize these error measures. The minimization requires finding n suitable points in $\mathbb{R}^m$ and therefore involves $m \times n$ variables, where m is the dimension for the MDS representation of the data, and n is the number of data objects. This means that even a two-dimensional MDS representation of a small data set like the Iris data leads to a minimization problem with $2 \times 150 = 300$ parameters to be optimized. Unfortunately, there is no known analytical solution for any of the error measures in (4.8)–(4.11). Therefore, a heuristic optimization strategy is needed. Typically, a **gradient method** is applied. Gradient methods are discussed in more

Table 4.2 Multidimensional scaling

```
Algorithm MDS($\mathcal{D}$)

input:     data set $\mathcal{D} \subset \mathbb{R}^m$ with $|\mathcal{D}| = n$ or distance matrix $[d_{i,j}^{(X)}]_{1 \le i,j \le n}$
parameter: dimension $q$ for the representation, stepwidth $\alpha > 0$, stop criterion SC
output:    set $Y$ of $n$ points in $\mathbb{R}^q$

1   Initialize $Y = \{\mathbf{y_1}, \dots, \mathbf{y_n}\} \subset \mathbb{R}^q$ randomly or better with a PCA projection
2   If the input is a data set, compute the distances $d_{ij}^{(X)}$ between the data objects
3   do
4     Compute $d_{i,j}^{(Y)} = \| \mathbf{y}_i - \mathbf{y}_j \|$ (for all $i, j = 1, \dots, n$)
5     Compute $\partial E_1 / \partial \mathbf{y_k}$ according to (4.12) (for all $k = 1, \dots, n$)
6     update $\mathbf{y_k}^{\text{new}} = \mathbf{y_k}^{\text{old}} - \alpha \cdot \partial E_1 / \partial \mathbf{y_k}$ (for all $k = 1, \dots, n$)
7   while SC is not satisfied
```

detail in Sect. 5.3.2. In order to minimize a differentiable function—here any of the error measures (4.8)–(4.11)—one can compute the vector of partial derivatives with respect to the parameters of the object function and then follow the opposite direction of the gradient, since the gradient points in the direction of steepest ascend and our aim is not to maximize but to minimize the error function. The gradient method starts with a random or a suitable initial solution, computes the gradient at the given solution, goes in the opposite direction of the gradient, and takes the resulting solution as the next solution. For this new solution, the gradient is computed again, and the process is repeated until no or little improvements can be achieved.

As an example, we compute the gradient for error measure E_1 in (4.9) with respect to one data point $\mathbf{y}_k$ in the lower-dimensional space:

$$\frac{\partial E_1}{\partial \mathbf{y_k}} = \frac{2}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (d_{ij}^{(X)})^2} \sum_{j \neq k} \left(d_{kj}^{(Y)} - d_{kj}^{(X)}\right) \frac{\mathbf{y_k} - \mathbf{y_j}}{d_{kj}^{(Y)}}, \tag{4.12}$$

where we have used

$$\frac{\partial d_{ij}^{(Y)}}{\partial \mathbf{y_k}} = \frac{\partial}{\partial \mathbf{y_k}} \|\mathbf{y_i} - \mathbf{y_j}\| = \begin{cases} \frac{\mathbf{y_k} - \mathbf{y_j}}{d_{kj}^{(Y)}} & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

The basic structure of the MDS algorithm is described in Table 4.2.

It is recommended to start with a good initialization which can be obtained by first carrying out a principal component analysis and then projecting the data to the first q principal components. The stepwidth α determines how far to go in the opposite direction of the gradient. If α is chosen too large, the algorithm might never find a (local) minimum of the error function. Small values for α lead to slow convergence of the algorithm. It is also possible to use an adaptive stepwidth α that determines a new value for α in each step of the algorithm. The stop criterion can be a fixed number of iterations or a lower bound for the change of the error function, i.e., when E_1 changes less than a given threshold in one step of the algorithm, the iteration scheme is terminated.

The algorithm is identical for the error measures (4.10) and (4.11) except for lines 5 and 6 where the gradient of E_1 has to be replaced by the gradients of the corresponding error measures.

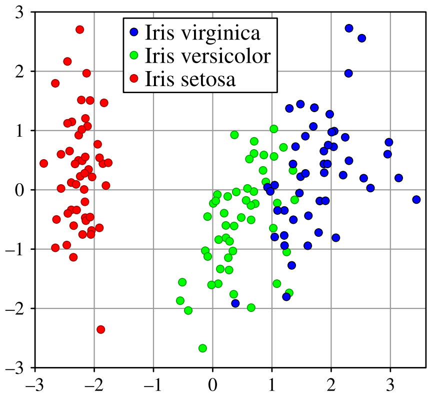


Fig. 4.17 The Sammon mapping of the Iris data, scatter plots of which are shown in Fig. 4.11 on p. 46

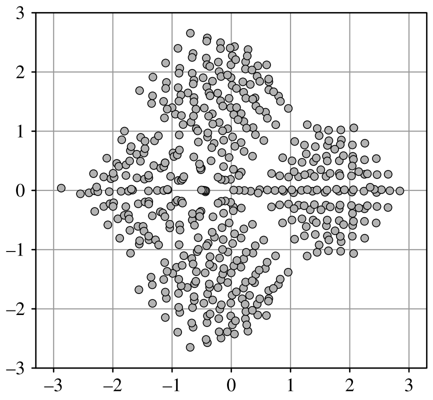


Fig. 4.18 The Sammon mapping of the data set that is shown in Fig. 4.14 on p. 51

Figure 4.17 shows the result of MDS in the form of the Sammon mapping applied to the Iris data set. The categorical attribute for the species has not been used for MDS. But the corresponding points resulting from the Sammon mapping are marked by colors corresponding to the Iris flower species. Before the distances $d_{ij}^{(X)}$ in the original four-dimensional space defined by the sepal and petal length and width are computed, a z-score standardization is applied to each of the four numerical attributes.

Before the MDS algorithm can be applied, we have to make sure that no distance between two different data objects is zero because these distances occur in the denominator of the gradient, so that a zero distance would lead to a division by zero error. There are two Iris flowers—number 102 and number 143—with exactly the same values for all attributes, leading to a distance of zero between them. Therefore, we removed one of them from the data set before carrying out the computations for MDS.

Figure 4.18 shows the result of the Sammon mapping applied to the "3D chessboard pattern" data set in Fig. 4.14, where we have again carried out z-score standardization in advance. The four cubes filled with data can even be recognized in this two-dimensional representation. However, the outlier—the left-most point—is not as well separated from the other data points as in the case of PCA in Fig. 4.16.

4.3.2.4 Variations of PCA and MDS

MDS differs from PCA in various aspects. MDS is based on the idea of preserving the distances among the original data objects, whereas PCA focuses on the variance.

PCA provides an explicit mapping from the space in which the data objects are located to the lower-dimensional space, whereas MDS provides only an explicit representation of the data objects in the lower-dimensional space. This means that when a new or hypothetical data object is considered, it can be represented immediately in case of PCA simply by projecting the data object to the corresponding principal components. This is not possible for MDS.

The computational complexity of PCA is lower than the complexity of MDS. The computation of the covariance matrix can be carried out in linear time with respect to the size of the data set. Once the covariance matrix has been calculated, the computation of the eigenvalues and eigenvectors depends only on the number of attributes, but not on the number of data anymore. The number of attributes is usually much smaller than the number of data objects. For MDS, it is necessary to consider the pairwise distances between data objects leading to a quadratic complexity in the number of data objects. Although a quadratic complexity is often considered as feasible in computer science, it is unacceptable for larger data sets. Consider a data set with one million data objects. The distance matrix contains 10^{12} entries in this case. Since the distance matrix is symmetric and the entries in the diagonal are all zero, we only need to know $(10^{12} - 10^{6})/2 = 4999995 \cdot 10^{5}$ entries. If we want to store the distances in 4-byte floating-point values, this would require more than 1800 gigabytes! There are various modifications of MDS [3] that try to overcome these complexity problems by sampling [15] or variations of the error measures [16, 17].

All these dimension-reduction methods generate scatter plots with abstract coordinate axes that do not correspond to attributes of the original data. In this way, the scatter plots of projection pursuit can even be extended and evaluated by the measures of interestingness proposed for projection pursuit and also other statistical measures and tests in order to select the most interesting visualizations [23].

4.3.2.5 Parallel Coordinates

So far, the visualization techniques for multidimensional data we have introduced here are based on scatter plots with abstract coordinate axes. Since coordinate axes should be drawn pairwise perpendicular and only two axes can satisfy this condition on a plane and three axes in 3D-space, we cannot use more than three axes for scatter plot-like visualizations. **Parallel coordinates** draw the coordinate axes parallel to each other, so that there is no limitation for the number of axes to be displayed. For a data object, a polyline is drawn connecting the values of the data object for the attributes on the corresponding axes.

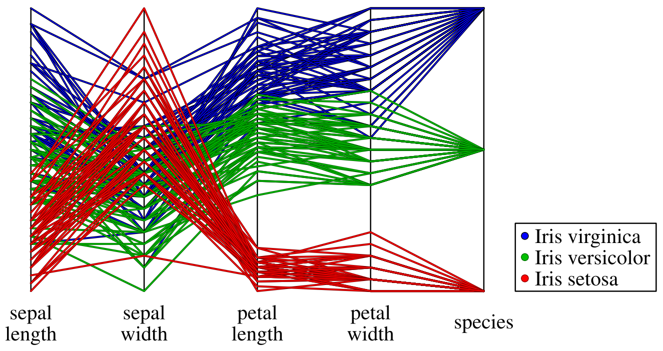


Fig. 4.19 Parallel coordinates plot for the Iris data set

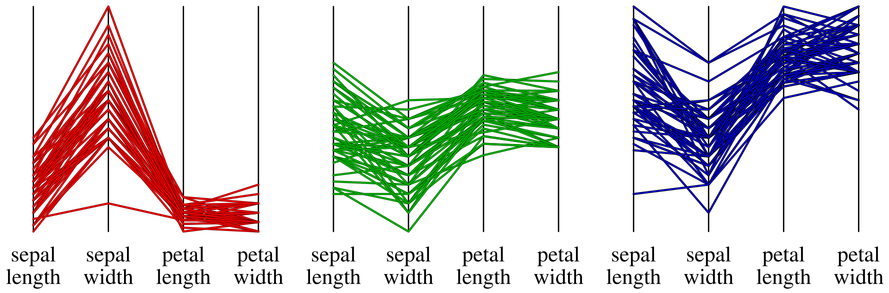


Fig. 4.20 Parallel coordinate plots for Iris setosa, Iris versicolor, and Iris virginica (*left* to *right*)

Figure 4.19 shows the plot of the Iris data set with parallel coordinates. The species is a categorical attribute and can only assume three possible values. The polylines for the different species are displayed with different colors. The plot clearly shows that the setosa has smaller values for the petal length and width than the other two species.

For larger data sets, it becomes more or less impossible to track the lines that correspond to a data object in parallel coordinates plots or even to discover general structures. It can be helpful to generate separate parallel coordinate plots for different subsets of the data. For instance, in the case of the Iris data set, we could generate a separate plot for each of the three species as shown in Fig. 4.20. In order to keep the three plots comparable, we have not rescaled the axes for each of the three plots. Normally, the axes will be scaled in such a way that the minimum and maximum of all values for the corresponding attribute of the displayed data are lowest and highest point of the axes.

Figure 4.21 shows the parallel coordinates plot for the "3D chessboard pattern" data set in Fig. 4.14. This plot is identical to the plot that we would obtain if we had not used the "3D chessboard pattern," but had filled the cube uniformly with data.

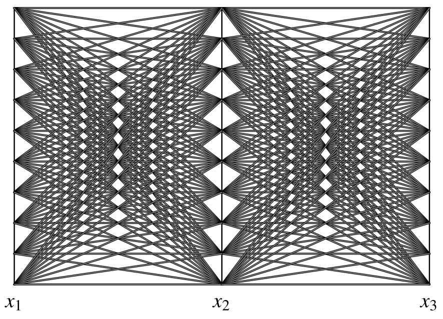


Fig. 4.21 Parallel coordinates plot for the data set shown in Fig. 4.14

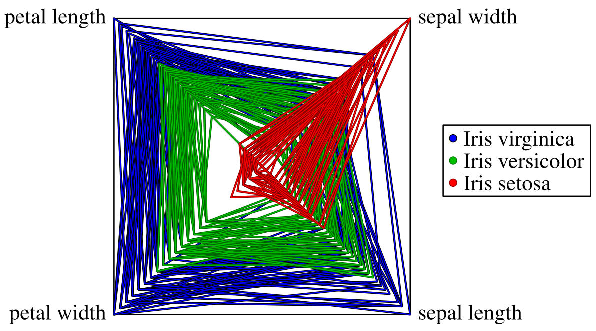


Fig. 4.22 Radar plot for the numerical attributes of the Iris data set

4.3.2.6 Radar and Star Plots

Radar plots are based on a similar idea as parallel coordinates with the difference that the coordinate axes are not drawn as parallel lines, but in a star-like fashion intersecting in one point. Figure 4.22 shows a radar plot for the four numerical attributes of the Iris data set. Due to the fact that radar plots sometimes resemble spider webs, they are also called **spider plots**.

Radar plots are only suited for smaller data sets. For such smaller data sets, it is sometimes better not to draw all data objects in the system of coordinate axes but to draw each data object separately, which is then called a **star plot**. A star plot for the numerical attributes of the Iris data set is shown in Fig. 4.23. The first 50 “stars” correspond to the objects from the species setosa, the next 50 to versicolor, and the last 50 to virginica. The star plot also shows clearly that setosa differs much from the two species.

4.4 Correlation Analysis

In the previous section, we have introduced methods to visualize the distribution of values for one, two, or even more attributes simultaneously. Scatter plots can uncover dependencies or correlations between two attributes. Instead of a purely visual approach, it is possible to compute measures of correlation between attributes

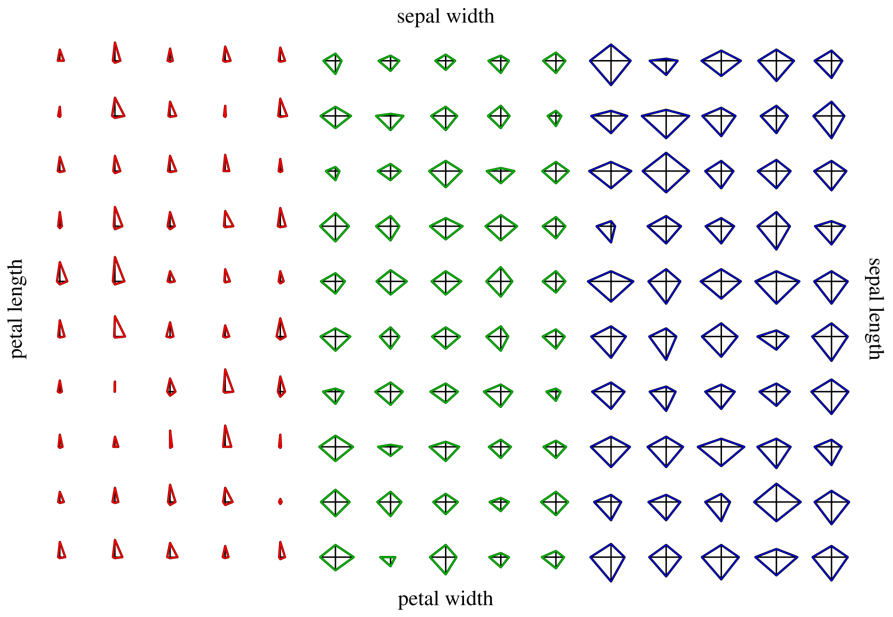


Fig. 4.23 Star plot for the numerical attributes of the Iris data set

Table 4.3 Pearson's correlation coefficients for the numerical attributes of the Iris data set

	Sepal length	Sepal width	Petal length	Petal width
Sepal length	1.000	−0.118	0.872	0.818
Sepal width	−0.118	1.000	−0.428	−0.366
Petal length	0.872	−0.428	1.000	0.963
Petal width	0.818	−0.366	0.963	1.000

to confirm expected dependencies or to discover unexpected correlations between attributes.

The (sample) **Pearson's correlation coefficient** is a measure for a linear relationship between two numerical attributes X and Y and is defined as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}, \tag{4.14}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the attributes X and Y, respectively, and s_x and s_y are the corresponding (sample) standard deviations. Pearson's correlation coefficient yields values between -1 and 1. The larger the absolute value of the Pearson correlation coefficient, the stronger the linear relationship between the two attributes. For $|r_{xy}| = 1$, the values of X and Y lie exactly on a line. For $r_{xy} = 1$, the line has a positive slope, for $r_{xy} = -1$ a negative one. Table 4.3 lists the Pearson correlation coefficients for the numerical attributes of the Iris data set.

Of course, the values in the diagonal must be equal to 1, since an attribute correlates fully with itself. When we plot an attribute against itself in a scatter plot, the points lie on a perfect line, the diagonal, so that Pearson's correlation coefficient must be 1 in this case. The matrix with the coefficients must also be symmetric.

It is also not surprising that the Pearson's correlation coefficient between the length and the petal width is very high. This means more or less that the leaves roughly keep their shape. A short leaf will also not be very broad, and a long leaf will be broader. It seems counterintuitive that there is more or less no correlation between the sepal length and the sepal width or even a very small negative correlation. When we take a look at the scatter plots in Fig. 4.11 on page 46, this can be explained easily. The negative correlation originates from the fact that we have the measurements from different species. Setosa has short but broad leaves compared to the other species. When we compute Pearson's correlation coefficient separately for the species for the sepal length and width, we obtain the values 0.743, 0.526, and 0.457 for Iris setosa, Iris versicolor, and Iris virginica, respectively. The correlation is not as high as for the petal length and width, but at least it is positive and not negative.

Pearson's correlation coefficient measures linear correlation. Even if there is a functional dependency between two attributes, but the function is nonlinear but monotone, Pearson's correlation coefficient will not be -1 or 1. It can even be far away from these values, depending on how much the function describing the functional relationship deviates from a line.

Rank correlation coefficients avoid this problem by ignoring the exact numerical values of the attributes and considering only the ordering of the values. Rank correlation coefficients intend to measure monotonous correlations between attributes where the monotonous function does not have to be linear.

Spearman's rank correlation coefficient or **Spearman's rho** is defined as

$$\rho = 1 - 6\frac{\sum_{i=1}^{n}(r(x_i) - r(y_i))^2}{n(n^2-1)}, \tag{4.15}$$

where $r(x_i)$ is the rank of value x_i when we sort the list $(x_1, \ldots, x_n)$. $r(y_i)$ is defined analogously.

Spearman's rho measures the sum of quadratic distances of ranks and scales this measure to the interval $[-1, 1]$. When the rankings of the x- and y-values are exactly in the same order, Spearman's rho will yield the value 1; if they are in reverse order, we will obtain the value -1.

Spearman's rho assumes that there are no ties, i.e., no two values of one attribute are equal. If two or more values coincide, their rank is not defined. When ties exist, the rank $r(x_i)$ is usually defined as the mean value of all ranks of consecutive coinciding values in the sorted list. So if we have the (already sorted) list of values 0.6, 1.2, 1.4, 1.4, 1.6 for an attribute, the corresponding ranks would be 1, 2, 3.5, 3.5, 4.

Kendall's tau rank correlation coefficient or simply **Kendall's tau** is not, like Spearman's rho, based on ranks, but rather on the comparison of the orders of pairs of values. Assuming that $x_i < x_j$, the two pairs (x_i, x_j) and (y_i, y_j) are called **concordant** if $y_i < y_j$, i.e., when the two pairs are in the same order. They are called **discordant** when they are in reverse order, which means that $y_i > y_j$.

Table 4.4 Spearman's rank correlation coefficients for the numerical attributes of the Iris data set

	Sepal length	Sepal width	Petal length	Petal width
Sepal length	1.000	−0.167	0.882	0.834
Sepal width	−0.167	1.000	−0.289	−0.289
Petal length	0.882	−0.289	1.000	0.938
Petal width	0.834	−0.289	0.938	1.000

Table 4.5 Kendall's tau for the numerical attributes of the Iris data set

	Sepal length	Sepal width	Petal length	Petal width
Sepal length	1.000	−0.077	0.719	0.655
Sepal width	−0.077	1.000	−0.186	−0.157
Petal length	0.719	−0.186	1.000	0.807
Petal width	0.655	−0.157	0.807	1.000

Kendall's tau is computed as

$$\tau_a = \frac{C - D}{\frac{1}{2}n(n-1)},$$

where C and D denote the numbers of concordant and discordant pairs, respectively:

$$C = |\{(i,j) \mid x_i < x_j \text{ and } y_i < y_j\}|, \tag{4.16}$$

$$D = |\{(i,j) \mid x_i < x_j \text{ and } y_i > y_j\}|. \tag{4.17}$$

Tables 4.4 and 4.5 contain the values for Spearman's rho and Kendal's tau for the numerical attributes of the Iris data set. The values are not identical but very similar to the ones in Table 4.3 for Pearson's correlation coefficient.

Rank correlation coefficients like Spearman's rho and Kendall's tau depend only on the order (ranks) of the values and are therefore more robust against extreme outliers than Pearson's correlation coefficient.

For categorical attributes, these correlation coefficients are not applicable. Instead, one can carry out independence tests like the χ^2 test for independence, as it is described in the appendix in Sect. A.4.3.6. This test can also be used for numerical attributes when they are discretized by some binning strategy.

4.5 Outlier Detection

Since it has an intuitive, though imprecise meaning, we have used the term outlier already before without giving a precise definition. Actually there is no formally precise definition of outliers. An **outlier** is simply a value or data object that is far away or very different from all or most of the other data.

Outliers can be a hint to data quality problems as they will be discussed in the next section. Outliers can correspond to erroneous data coming from wrong measurements or typing mistakes when data are entered manually. Erroneous data should be corrected, or, if this is not possible, they should be excluded from the data set before further analysis steps are carried out. However, outliers can also be correct data that differ from the rest of the data just by chance or for other reasons like special exceptional situations. Even if outliers are correct data, it might be worthwhile to exclude them from the data set for further analysis and to consider them separately. Some data analysis methods are robust against outliers and more or less not influenced by them, whereas other are extremely sensitive to outliers and might produce incoherent or nonsense results, just because of the presence of one or a few outliers. As a simple example, consider the median and the mean value. If one replaces the largest value in a sample by an extremely large value or even infinity, the median will not change. But the mean value will tend to infinity, even though only one value in the sample goes to infinity.

4.5.1 Outlier Detection for Single Attributes

For a categorical attribute, one can consider the finite set of values. An outlier is a value that occurs with a frequency extremely lower than the frequency of all other values. However, in some cases, this might be actually the target of our analysis. If we want to set up an automatic quality control system and want to train a classifier, classifying the parts as correct or with failures based on measurements of the produced parts, we will probably have so many correct parts in comparison the ones with failures that we would consider them as outliers. However, removing these "outliers" from the data set would actually make it impossible to achieve our original goal to derive a classifier from the data set that can identify the parts with failures.

For numerical attributes, outlier detection is more difficult. We have already classified certain data points in a boxplot as outliers. However, the definition of outliers in a boxplot does not take the number of data into account, so that for larger data sets, boxplots will usually contain points marked as outliers. We have seen this already in the boxplots in Fig. 4.6, both showing samples from a standard normal distribution. The left boxplot in Fig. 4.6 for a sample of size $n = 1000$ contains eight outliers corresponding to what we expect theoretically, namely seven points outside the whiskers. As mentioned before, for a normal distribution, we can expect roughly 0.7% points to be marked as outliers in a boxplot.

For asymmetric distributions, boxplots tend to contain more outliers. For example, in boxplots for samples from an exponential distribution with $\lambda = 1$ as they are shown in Fig. 4.8, we would expect roughly 4.8% points marked as outliers. Heavy-tailed distributions tend to show more outliers in a boxplot, whereas for a sample from a uniform distribution, we would expect no outliers at all, no matter how large the sample size is.

Even if we try to adjust the definition of outliers according to the sample size, the above examples show that what we consider as an outlier depends strongly on the underlying distribution from which the data are sampled. Therefore, statistical tests for outliers are usually based on assumptions about the underlying distribution, although we might not know from which distribution the data are sampled.

The standard assumption for outlier tests for continuous attributes is that the underlying distribution is a normal distribution. **Grubb's test** is a test for outliers for normal distributions taking the sample size into account. It is based on the statistics

$$G = \frac{\max\{|x_i - \bar{x}| \,|\, 1 \le i \le n\}}{s}, \tag{4.18}$$

where $x_1, \ldots, x_n$ is the sample, $\bar{x}$ is its mean value, and s is its empirical standard deviation. For a given significance level α, the null hypothesis that the sample does not contain outliers is rejected if

$$G > \frac{n-1}{\sqrt{n}} \sqrt{\frac{t^2_{1-\alpha/(2n),n-2}}{n-2+t^2_{1-\alpha/(2n),n-2}}}, \tag{4.19}$$

where $t_{1-\alpha/(2n),n-2}$ denotes the $(1-\alpha/(2n))$-quantile of the t-distribution with $(n-2)$ degrees of freedom.

For the Iris data set, Grubb's test yields the p-values[4] 0.92, 0.13, 1.0, and 1.0 for the sepal length and width and the petal length and width, respectively. Even the lowest p-value does not indicate that there is an outlier. Note that this is the case although the assumption that the attributes follow normal distributions is not realistic at all as can be seen from the scatter plot in Fig. 4.11 on page 46. The attributes for each species separately might follow normal distributions, but not the attributes considered for all species together.

When Grubb's test with its assumption of normal distribution is applied to very skewed and heavy-tailed distributions like the exponential distribution, an outlier will almost always be indicated.

4.5.2 *Outlier Detection for Multidimensional Data*

Outlier detection in multidimensional data is usually not based on specific assumptions on the distribution of the data and is not carried out in the sense of statistical tests. Visualization techniques provide a simple method for outlier detection in multidimensional data. Scatter plots can be used when only two attributes are considered. In the right scatter plot in Fig. 4.12 on page 47, we had added two outliers. The one in the upper left corner would not have been detected when only single

[4]For Grubb's test, the null hypothesis is that there are no outliers. Then the point in the sample with the largest distance to the mean is considered. In the case of Grubb's test, the p-value is the probability that in a sample of size n, such a large or even large deviation from the mean would occur. For a more formal and general definition of p-values, see also Appendix A.

attributes are considered. Instead of using projections to two attributes, one can also use dimension-reduction methods like PCA or multidimensional scaling in order to identify outliers in the corresponding plots as in Figs. 4.16 and 4.18.

There are many approaches for finding outliers in multidimensional data based on clustering the data and defining those data objects as outliers that cannot be assigned reasonably to any cluster [18, 19]. There are also distance-based [13, 14], density-based [4], and projection-based methods [1] for outlier detection.

4.6 Missing Values

In the ideal case, all attributes of all objects in the data table have well-defined values. However, in real data sets, one has to deal with **missing values** quite often. The occurrence of missing values can have different causes. A sensor might be broken leading to a missing value. People might have refused or forgotten to answer a question in a questionnaire, or an attribute might not be applicable for a certain object. For instance, the attribute *pregnant* with the two possible values *yes* and *no* does not make sense for men. Of course, one could always enter the value *no* for the attribute *pregnant*. But this might lead to a grouping of men with nonpregnant women.

Especially in a customer database, missing values will be quite common. Attributes like *age*, *sex*, and *email address* might be available for some customers, but probably not for all customers.

In the best case, missing values are indicated in a unique way. A symbol like "?" or null entry might be used for missing value. Sometimes a special numeric value is used to indicate missing values. For example, the value -1 could indicate a missing value for a counting attribute (like number of products bought by a customer in a certain period, number of children, etc.). Since the value -1 can never occur in a counting attribute, it cannot be misunderstood as meaningful (nonmissing) value. One must, however, take this into account when simple characteristics like the mean value or the median of an attribute are computed. If one simply adds up all values—including the -1's for the missing values—and divide it by the number of values to obtain the mean value, the result might even be negative if there are many missing values. If we know the number of children for only very few of our customers, we might find out in this way that the average number of children of our customers is -0.3, which is, of course, complete nonsense.

Sometimes, missing values are not even marked in a specific way. We have seen already an example in Fig. 2.2 on page 18 where a missing value of the customer's birth date was replaced by a default value that could also be just a usual value.

As another example, consider an anemometer, a sensor for measuring the wind speed. Such an anemometer is equipped with a small wheel that is turned by the wind, and from the speed of the wheel the speed of the wind can be deduced. Sometimes, the wheel might be blocked for some reason. In this case, the sensor will constantly yield the value zero for the wind speed. This value zero is also a possible wind speed although it is very improbable in most geographical regions that

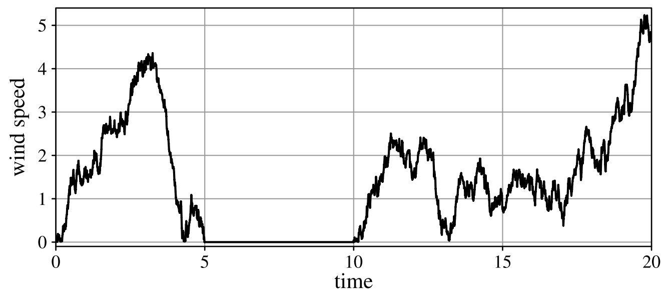


Fig. 4.24 Measured wind speeds with a period of a jammed sensor

the wind speed is absolutely zero over a longer time. When we see a diagram like in Fig. 4.24, it is very probable that during the period between 5 and 10, the wind speed was not zero, but that there are missing values in this period due to a jammed anemometer.

It is very important to identify such hidden missing values. Otherwise, the further analysis of the data can be completely mislead by such erroneous values.

When there are missing values, one should also take into account how missing values enter the data set. The simplest and most common assumption about missing values is that they are **missing completely at random** (**MCAR**) or are **observed at random** (**OAR**). This means that special circumstances or special values in the data lead to higher or lower chances for missing values. One can imagine that one has printed out the data table on a large sheet of paper with no missing values and then someone has dropped accidentally some random spots of ink on the paper so that some values cannot be read anymore.

In a more formal way, the situation missing completely at random can be described in the following way. We consider the random variable X for which we might have some missing entries in the data set. The random variable X itself does not have missing values. The corresponding value is just not available in our data set. The random variable X_{miss} can assume the value 0 for a missing values and 1 for a nonmissing value for the random variable X of interest. This means that for $X_{\text{miss}} = 1$, we see the true value of X in our data set, and for $X_{\text{miss}} = 0$, instead of the true value of X, we see a missing value. We also consider the random vector Y representing all attributes except X in our data set. The situation observed at random means that X_{miss} is independent of X and Y, i.e.,

$$P(X_{\text{miss}}) = P(X_{\text{miss}}|X, Y) \qquad \text{(MCAR)}.$$

Consider a sensor for the outside air temperature X whose battery might run out of energy once in a while, leading to missing values of the missing completely at random. This is the best case for missing values. It can, for instance, be concluded that the unknown missing values follow the same distribution as the known values of X.

The situation **missing at random** (**MAR**) is more complicated but might still be manageable with appropriate techniques. The probability for a missing value depends on the value of the other attributes Y but is conditionally independent[5] of the true value of X given Y:

$$P(X_{\text{miss}}|Y) = P(X_{\text{miss}}|X, Y) \qquad \text{(MAR)}.$$

To illustrate what missing at random means, consider the above example of a sensor for the outside air temperature with the battery running out of energy once in a while. But assume now that the staff responsible for changing the batteries does not change the batteries when it is raining. Therefore, missing values for the temperature are more probable when it is raining, so that the occurrence of missing values depends on other attributes, in this case the attribute for rain or amount of rain.

Note that the occurrence of a missing value is in this case not independent of the value of the variable X, it is only conditionally independent given the attribute for the rain. Usually, the outside temperatures are lower when it is raining, so that missing values correlate with lower temperatures. In contrast to missing completely at random, here the missing values of X do not follow the same distribution as the measured values of X. If we compute the average temperature based only on the measured values, we would obviously overestimate the average temperature. Therefore, special care has to be taken when missing values occur at random, but not completely at random.

The worst case are **nonignorable missing values**, where the occurrence of missing values directly depends on the true value, and the dependence cannot be resolved by other attributes. Consider a bad sensor for the outside temperature that always fails when there is frost. This would mean that we have no information about temperatures below 0°C. In the case of missing at random, we might still be able to obtain information from the data set about the missing values. For the nonignorable values, we have no chance to make correct guesses, at least based on the available data.

How can we distinguish between the three different types of missing values? First of all, the knowledge and background information about why missing values occur should be taken into account. It is also possible to try to distinguish the two cases observed at random and missing at random based on the given data. If an attribute X has missing values and one wants to find out of which type the missing values are, one way would be to apply independence tests checking directly whether $P(X_{\text{miss}}) = P(X_{\text{miss}}|Y)$. Usually, it is not feasible to apply a test that checks the independence of X_{miss} against the multivariate attribute Y standing for all other attributes, but only against single attributes. Applying independence test of X_{miss} against each other attribute Y will not uncover multiple dependencies, but if even these tests indicate a dependency already, then we can be quite sure that the values of X are not missing completely at random.

Another way to approach the problem of distinguishing between missing completely at random and missing at random is the following procedure:

1. Turn the considered attribute X into a binary attribute, replacing all measured values by the values *yes* and all missing values by the value *no*.

[5]A formal definition of conditional independence is given in Sect. A.3.3.4 in the appendix.

2. Build a classifier with now binary attribute X as the target attribute and use all other attributes for the prediction of the class values *yes* and *no*.
3. Determine the misclassification rate. The misclassification rate is the proportion of data objects that are not assigned to the correct class by the classifier.

In the case of missing values of the type observed at random, the other attributes should not provide any information, whether X has a missing value or not. Therefore, the misclassification rate of the classifier should not differ significantly from pure guessing, i.e., if there are 10% missing values for the attribute X, the misclassification rate of the classifier should not be much smaller than 10%. If, however, the misclassification rate of the classifier is significantly better than pure guessing, this is an indicator that there is a correlation between missing values for X and the values of the other attributes. Therefore, the missing values for X might not be of the type observed at random but of the type missing at random or, even worse, nonignorable. Note that it is in general not possible to distinguish the case nonignorable from the other two cases based on the data only.

4.7 A Checklist for Data Understanding

In this chapter, we have provided a selection of techniques for data understanding. It is not necessary to go through all of them in the phase of data understanding, but rather to keep all of them in mind.

- First of all, one should keep in mind that there are general and specific goals for data understanding. One important part of data understanding is to get an idea of the data quality. There are standard data quality problems like syntactic accuracy which are easy to check.
- Outliers are another problem, and there are various methods to support the identification of outliers. Apart from methods exclusively designed for outlier detection as they have been discussed in Sect. 4.5, there are especially visualization techniques like boxplots, histograms, scatter plots, projections based on PCA and MDS that can help to find outliers but are also useful for other purposes.
- Missing values are another concern of data quality. When there are explicit missing values, i.e., entries that are directly marked as missing, then one should still try to find out of which type—OAR, MAR, or nonignorable—they are. This can sometimes be derived from domain knowledge, but also based on classification methods as described in the previous section. We should also be aware of the possibility of hidden missing values that are not explicitly marked as missing. The simplest case might be hidden missing values that have a default value. Histograms might help to identify candidates for such hidden missing values when there are unusual peaks as in the example in Fig. 2.2 on page 18. However, there is no standard test or technique to identify possible hidden missing values. Therefore, whenever we see something unexpected in the data, hidden missing values of a specific type might be one explanation.

- Apart from these data quality issues, data understanding should also help to discover new or confirm expected dependencies or correlations between attributes. Techniques like the ones mentioned in Sect. 4.4 are one way to solve this task. Apart from this, scatter plots can show correlations between pairs of attributes.
- Specific application dependent assumptions—for instance, the assumption that a specific attribute follows a normal distribution—should also be checked during data understanding.
- Representativeness of the data cannot always be checked just based on the data, but we have to compare the statistics with our expectations. If we suspect that there is a change in a numerical attribute over time, we can compare histograms or boxplots for different time periods. We can do the same with bar charts for categorical attributes.

No matter what the specific application dependent purposes of data understanding are, here is a checklist what we consider as a must for the data understanding phase:

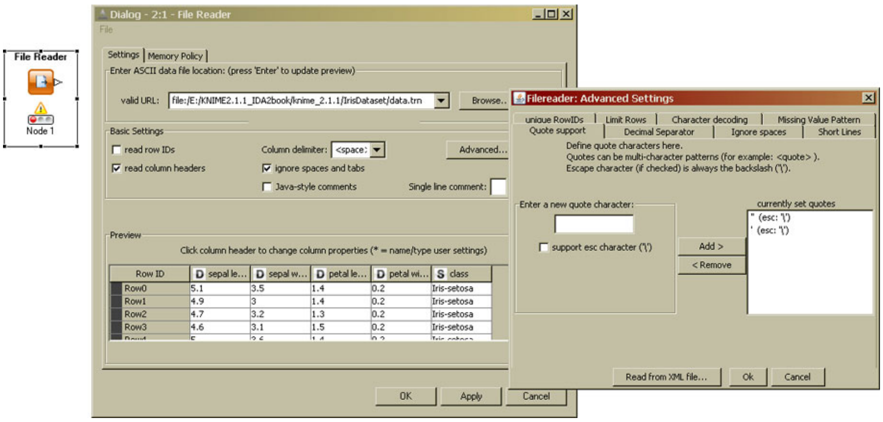
- Check the distributions for each attribute whether there are unusual or unexpected properties like outliers. Are the domains or ranges correct? Do the medians of numerical attributes look correct? This should be done based on
  - histograms and boxplots for continuous attributes and
  - bar charts for categorical attributes.
- Check correlations or dependencies between pairs of attributes with scatter plots which should be density-based for larger data sets. For small numbers of attributes, inspect scatter plots for all pairs of attributes. For higher numbers of attributes, do not generate scatter plots for all pairs, but only for those ones where independence or a specific dependency is expected. Generate in addition scatter plots for some randomly chosen pairs.

To cite Tukey [24] again, if problems in later phases of the data analysis process occur that could have been discovered or even avoided in an early plotting and looking at the data, there is absolutely no excuse for this failure.

Of course, one should also exploit the other methods described in this section for these and other more specific purposes.

4.8 Data Understanding in Practice

A key issue in data understanding (and actually the entire data analysis process) often causes considerable headaches: the loading of the data into the analysis tool of choice. One of the strengths of KNIME is its versatility in terms of powerful file importing nodes and database connectors. R, on the other hand offers the entire breadth of analysis and visualizations, although they are often not all that intuitive to use.



Enter ASCII data file location: (press 'Enter' to update preview)

valid URL: file:/E:/KNIME2.1.1_IDA2book/knime_2.1.1/IrisDataset/data.trn

Basic Settings

☐ read row IDs
☑ read column headers
Column delimiter: <space>
☑ ignore spaces and tabs
☐ Java-style comments
Single line comment:

Preview

Click column header to change column properties (* = name/type user settings)

Row ID	D sepal le...	D sepal w...	D petal le...	D petal wi...	S class
Row0	5.1	3.5	1.4	0.2	Iris-setosa
Row1	4.9	3	1.4	0.2	Iris-setosa
Row2	4.7	3.2	1.3	0.2	Iris-setosa
Row3	4.6	3.1	1.5	0.2	Iris-setosa

Define quote characters here.
Quotes can be multi-character patterns (for example: <quote>).
Escape character (if checked) is always the backslash ('\').

Enter a new quote character:
☐ support esc character ('\')

currently set quotes
" (esc: '\')
' (esc: '\')

Fig. 4.25 The dialog of the file reader node offers manifold options to control the reading of diverse file formats

4.8.1 Data Understanding in KNIME

Data Loading Data understanding starts with loading the data: KNIME offers a "File Reader" node which hides an entire file import wizard in its dialog. Manifold options allow one to choose the underlying character encoding, column types, and separators and escape characters, to name just a few. The ability to quickly check in the preview tab if the selected options fit the given file make it possible to smoothly find suitable settings also for complex files. Figure 4.25 shows the dialog of this node together with the "expert tab" which hides another set of options.

KNIME also allows one to read data from specialized file formats, such as the Weka ARFF format or the compressed KNIME table format. For these formats, specialized nodes are available in the "IO" category as well.

Reading data from databases is often neglected in stand-alone analytic tools. KNIME offers flexible connectivity to access databases of various types by specifying the corresponding JDBC driver (Java Database Connectivity) in addition to the table location and name. For basic data filtering, specialized nodes are also available, which allow one to do, e.g., column and row filtering also within the database, without loading the data into KNIME explicitly. This saves time for larger databases if only a much smaller subset is to be analyzed. Figure 4.26 shows a part of a workflow reading data from a database and filtering some rows and columns before reading the data into KNIME itself.

Data Types KNIME supports all basic data types, such as string, integers, and numbers, but also date and time types and nominal values. In addition, various extensions add the ability to process images, molecular structures, sequences, and textual data. The repository of type extension is constantly growing as well. For a first glance at the data read it into KNIME and in order to check domain and nominal value range, add the "Statistics" and "Value Counter" node. The first one computes

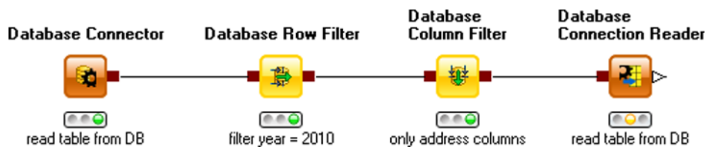
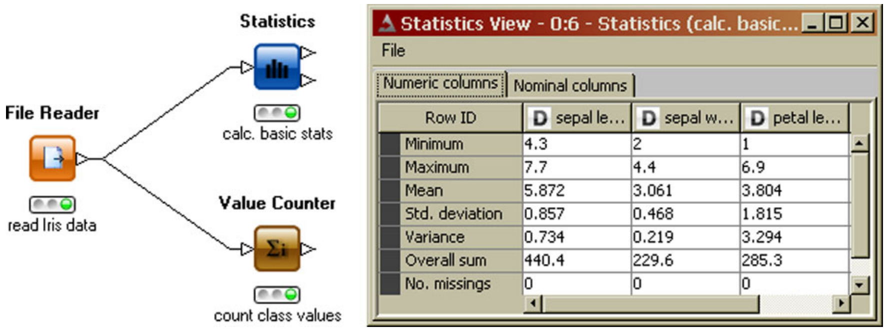


Fig. 4.26 Specialized nodes for database access allow one to filter rows and columns using database routines before loading the data into KNIME



Row ID	D sepal le...	D sepal w...	D petal le...
Minimum	4.3	2	1
Maximum	7.7	4.4	6.9
Mean	5.872	3.061	3.804
Std. deviation	0.857	0.468	1.815
Variance	0.734	0.219	3.294
Overall sum	440.4	229.6	285.3
No. missings	0	0	0

Fig. 4.27 Looking at basic information of the data's attributes often helps one to spot outliers or other errors in the data

basic statistical measures (minimum, maximum, mean, standard deviation, variance, overall sum), counts the number of missing values for numerical attributes, and lists the most and least frequent values for nominal values. The "Value Counter" node operates on one nominal attribute and lists the occurrence frequencies of each value. Figure 4.27 shows an example flow and the view of the statistics node for the iris data.

Visualization Checking basic descriptors of your data is important, but, as pointed out earlier in this chapter, looking at visualizations often gives dramatically better insights into your data. KNIME offers all of the visualizations described in Sect. 4.3: histograms, boxplots (also conditional), scatter and scatter matrix plots, and parallel coordinates. Principal Component Analysis and Multidimensional Scaling methods are also available and can be used to calculate new coordinates for, e.g., scatter plot views. All of these views make use of color, shape, and size information whenever possible. Such additional **visual dimensions** are added by using "Color/Shape/Size Manager" nodes as the additional advantage that all subsequent nodes are using the same color/shape/size information. Figure 4.28 shows an example of a KNIME workflow with a number of views connected to a workflow which assigns color and shape based on selected columns.

KNIME has been developed to also support explorative data analysis and therefore most of the views allow for interactivity, or **visual brushing**. This means that points (or other visual objects) representing records in the underlying database can be marked and will then be marked in other, related views as well. KNIME allows

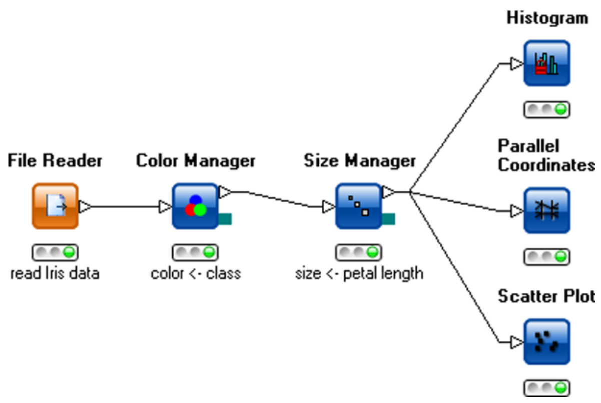
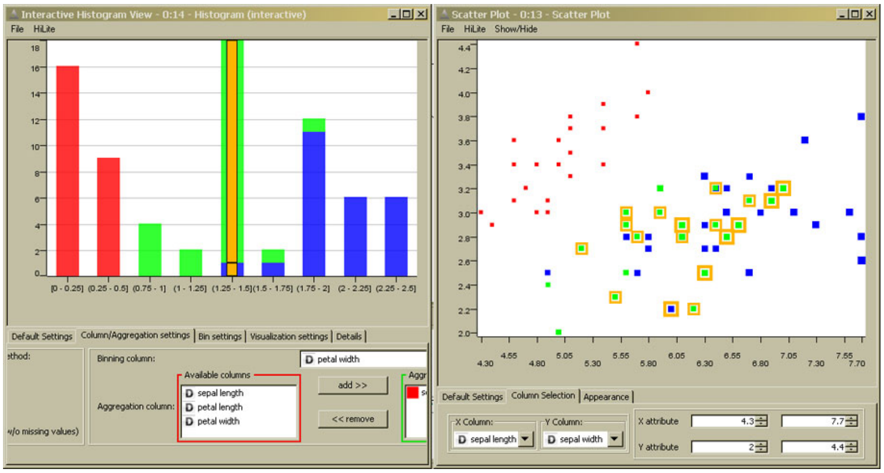


Fig. 4.28 Views in KNIME are simply connected to the data pipeline as well. Adding nodes which set additional visual properties allows to control color, shape and size in the subsequent views



Binning column: petal width

Aggregation column: Available columns: sepal length, petal length, petal width

X Column: sepal length

Y Column: sepal width

X attribute 4.3 7.7

Y attribute 2 4.4

Fig. 4.29 Interactive views in KNIME allow one to propagate selections to other views within the same workflow. The points falling into the selected bar in the histogram are automatically selected by KNIME's Hiliting mechanism

for this selection process (called *hiliting*) to propagate along the data pipeline as long as a meaningful translation between records (or rows in the table) is possible. For instance, selecting a rule will hilite all points contained within that rule in all other views. Visual brushing is a powerful tool to allow interactive exploration of data since it allows one to quickly select interesting elements in one view and see details about the selected elements in other views, for instance, the underlying customer data, the images covered by a rule, or the molecular structures that seem to look like outliers in a scatter plot. Figure 4.29 illustrates the hiliting mechanism in KNIME. We see an interactive histogram of one attribute and a scatter plot depicting two other attributes. One of the bars in the histogram was selected, and the corresponding points in the scatter plot are now marked as well.

4.8.2 Data Understanding in R

4.8.2.1 Histograms

Histograms are generated by the function `hist`. The simplest way to create a histogram is to just use the corresponding attribute as an argument of the function `hist`, and R will automatically determine the number of bins for the histogram based on Sturge's rule. In order to generate the histogram for the petal length of the Iris data set, the following command is sufficient:

```
> hist(iris$Petal.Length)
```

The partition into bins can also be specified directly. One of the parameters of `hist` is `breaks`. If the bins should cover the intervals $[a_0, a_1)$, $[a_1, a_2), \dots, [a_{k-1}, a_k]$, then one can simply create a vector in R containing the values a_i and assign it to `breaks`. Note that a_0 and a_k should be the minimum and maximum values of the corresponding attribute. If we want the boundaries for the bins at $1.0, 3.0, 4.5, 4.0, 6.1$, then we would use

```
> hist(iris$Petal.Length,breaks=c(1.0,3.0,4.5,4.0,6.9))
```

to generate the histogram. Note that in the case of bins with different length, the heights of the boxes in the histogram do not show the relative frequencies. The areas of the boxes are chosen in such a way that they are proportional to the relative frequencies.

4.8.2.2 Boxplots

A boxplot for a single attribute is generated by

```
> boxplot(iris$Petal.Length)
```

yielding the boxplot for the petal length of the Iris data set. Instead of a single attribute, we can hand over more than one attribute

```
> boxplot(iris$Petal.Length,iris$Petal.Width)
```

to show the boxplots in the same plot. We can even use the whole data set as an argument to see the boxplots of all attributes in one plot:

```
> boxplot(iris)
```

In this case, categorical attributes will be turned into numerical attributes by coding the values of the categorical attribute as $1, 2, \dots$, so that these boxplots are also shown but do not really make sense.

In order to include the notches in the boxplots, we need to set the parameter `notch` to true:

```
> boxplot(iris,notch=TRUE)
```

If one is interested in the precise values of the boxplot like the median, etc., one can use the `print`-command:

```
> print(boxplot(iris$Sepal.Width))
$stats
     [,1]
[1,]  2.2
[2,]  2.8
[3,]  3.0
[4,]  3.3
[5,]  4.0

$n
[1] 150

$conf
         [,1]
[1,] 2.935497
[2,] 3.064503

$out
[1] 4.4 4.1 4.2 2.0

$group
[1] 1 1 1 1

$names
[1] "1"
```

The first five values are the minimum, the first quartile, the median, the third quartile, and the maximum value of the attribute, respectively. n is the number of data. Then come the boundaries for the confidence interval for the notch, followed by the list of outliers. The last values `group` and `names` only make sense when more than one boxplot is included in the same plot. Then `group` is needed to identify to which attribute the outliers in the list of outliers belong. `names` just lists the names of the attributes.

4.8.2.3 Scatter Plots

A scatter plot of the petal width against petal length of the Iris data is obtained by

```
> plot(iris$Petal.Width,iris$Petal.Length)
```

All scatter plots of each attribute against each other in one diagram are created with

```
> plot(iris)
```

If symbols representing the values for some categorical attribute should be included in a scatter plot, this can be achieved by

```
> plot(iris$Petal.Width,iris$Petal.Length,
       pch=as.numeric(iris$Species))
```

where in this example the three types of Iris are plotted with different symbols.

If there are some interesting or suspicious points in a scatter plot and one wants to find out which data records these are, one can do this by

```
> plot(iris$Petal.Width,iris$Petal.Length)
> identify(iris$Petal.Width,iris$Petal.Length)
```

and then clicking on the points. The index of the corresponding records will be added to the scatter plot. To finish selecting points, press the ESCAPE-key.

Jitter can be added to a scatter plot in the following way:

```
> plot(jitter(iris$Petal.Width),
       jitter(iris$Petal.Length))
```

Intensity plots and density plots with hexagonal binning, as they are shown Fig. 4.9, can be generated by

```
> plot(iris$Petal.Width,iris$Petal.Length,
       col=rgb(0,0,0,50,maxColorValue=255),
       pch=16)
```

and

```
> library(hexbin)
> bin<-hexbin(iris$Petal.Width,
              iris$Petal.Length,
              xbins=50)
> plot(bin)
```

respectively, where the library `hexbin` does not come along with the standard version of R and needs to be installed as described in the appendix on R. Note that such plots are not very useful for such a small data sets like the Iris data set.

For three-dimensional scatter plots, the library `scatterplots3d` is needed and has to be installed first:

```
> library(scatterplot3d)
> scatterplot3d(iris$Sepal.Length,
                iris$Sepal.Width,
                iris$Petal.Length)
```

4.8.2.4 Principal Component Analysis

PCA can be carried out with R in the following way:

```
> species <- which(colnames(iris)=="Species")
> iris.pca <- prcomp(iris[,-species],
                  center=T,scale=T)
> print(iris.pca)
Standard deviations:
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation:
                    PC1         PC2
Sepal.Length  0.5210659 -0.37741762
Sepal.Width  -0.2693474 -0.92329566
Petal.Length  0.5804131 -0.02449161
Petal.Width   0.5648565 -0.06694199
                    PC3        PC4
              0.7195664  0.2612863
             -0.2443818 -0.1235096
             -0.1421264 -0.8014492
             -0.6342727  0.5235971

> summary(iris.pca)
Importance of components:
                        PC1   PC2    PC3     PC4
Standard deviation     1.71 0.956 0.3831 0.14393
Proportion of Variance 0.73 0.229 0.0367 0.00518
Cumulative Proportion  0.73 0.958 0.9948 1.00000

> plot(predict(iris.pca))
```

For the Iris data set, it is necessary to exclude the categorical attribute *Species* from PCA. This is achieved by the first line of the code and calling `prcomp` with `iris[,-species]` instead of `iris`.

The parameter settings `center=T`, `scale=T`, where `T` is just a short form of `TRUE`, mean that z-score standardization is carried out for each attribute before applying PCA.

The function `predict` can be applied in the above-described way to obtain the transformed data from which the PCA was computed. If the computed PCA transformation should be applied to another data set `x`, this can be achieved by

```
> predict(iris.pca,newdata=x)
```

where `x` must have the same number of columns as the data set from which the PCA has been computed. In this case, `x` must have four columns which must be numerical. `predict` will compute the full transformation, so that the above command will also yield transformed data with four columns.

4.8.2.5 Multidimensional Scaling

MDS requires the library MASS which is not included in the standard version of R and needs installing. First, a distance matrix is needed for MDS. Identical objects leading to zero distances are not admitted. Therefore, if there are identical objects in a data set, all copies of the same object except one must be removed. In the Iris data set, there is only one pair of identical objects, so that one of them needs to be removed. The Species is not a numerical attribute and will be ignored for the distance.

```
> library(MASS)
> x <- iris[-102,]
> species <- which(colnames(x)=="Species")
> x.dist <- dist(x[,-species])
> x.sammon <- sammon(x.dist,k=2)
> plot(x.sammon$points)
```

$k = 2$ means that MDS should reduce the original data set to two dimensions.

Note that in the above example code no normalization or z-score standardization is carried out.

4.8.2.6 Parallel Coordinates, Radar, and Star Plots

Parallel coordinates need the library MASS. All attributes must be numerical. If the attribute *Species* should be included in the parallel coordinates, one can achieve this in the following way:

```
> library(MASS)
> x <- iris
> x$Species <- as.numeric(iris$Species)
> parcoord(x)
```

Star and radar plots are obtained by the following two commands:

```
> stars(iris)
> stars(iris,locations=c(0,0))
```

4.8.2.7 Correlation Coefficients

Pearson's, Spearman's, and Kendall's correlation coefficients are obtained by the following three commands:

```
> cor(iris$Sepal.Length,iris$Sepal.Width)
> cor.test(iris$Sepal.Length,iris$Sepal.Width,
           method="spearman")
> cor.test(iris$Sepal.Length,iris$Sepal.Width,
           method="kendall")
```

4.8.2.8 Grubb's Test for Outlier Detection

Grubb's test for outlier detection needs the installation of the library `outliers`:

```
> library(outliers)
> grubbs.test(iris$Petal.Width)
```

References

1. Aggarwal, C., Yu, P.: Outlier detection for high dimensional data. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2001, Santa Barbara, CA), pp. 37–46. ACM Press, New York (2001)
2. Anderson, E.: The irises of the Gaspe Penisula. Bull. Am. Iris Soc. **59**, 2–5 (1935)
3. Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer, Berlin (1997)
4. Breunig, M., Kriegel, H.-P., Ng, R., Sander, J.: LOF: identifying density-based local outliers. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2000, Dallas, TX), pp. 93–104. ACM Press, New York (2000)
5. Cook, D., Buja, A., Cabrera, J.: Projection pursuit indices based on orthonormal function expansion. J. Comput. Graph. Stat. **2**, 225–250 (1993)
6. Diaconis, P., Freedman, D.: Asymptotics of graphical projection pursuit. Ann. Stat. **17**, 793–815 (1989)
7. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(2), 179–188 (1936)
8. Freedman, D., Diaconis, P.: On the histogram as a density estimator: L_2 theory. Z. Wahrscheinlichkeitstheor. Verw. Geb. **57**, 453–476 (1981)
9. Friedman, J.: Exploraory projection pursuit. J. Am. Stat. Assoc. **82**, 249–266 (1987)
10. Friedman, J., Tukey, J.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. **C-23**, 881–890 (1974)
11. Hall, P.: On polynomial-based projection indices for exploratory projection pursuit. Ann. Stat. **17**, 589–605 (1989)
12. Huber, P.: Projection pursuit. Ann. Stat. **13**, 435–475 (1985)
13. Knorr, E., Ng, R.: Algorithms for mining distance-based outliers in large datasets. In: Proc. 24th Int. Conf. on Very Large Data Bases (VLDB 1998, New York, NY), pp. 392–403. Morgan Kaufmann, San Mateo (1998)
14. Knorr, E., Ng, R., Tucakov, V.: Distance-based outliers: algorithms and applications. Very Large Data Bases **8**, 237–253 (2000)
15. Morrison, A., Ross, G., Chalmers, M.: Fast multidimensional scaling through sampling. Inf. Vis. **2**, 68–77 (2003)
16. Pekalska, E., Ridder, D., Duin, R., Kraaijveld, M.: A new method of generalizing Sammon mapping with application to algorithm speed-up. In: Proc. 5th Annual Conf. Advanced School for Computing and Imaging, pp. 221–228, Delft, Netherlands (1999)
17. Rehm, F., Klawonn, F., Kruse, R.: Mds_{polar}: a new approach for dimension reduction to visualize high dimensional data. In: Advances in Intelligent Data Analysis, vol. VI, pp. 316–327. Springer, Berlin (2005)
18. Rehm, F., Klawonn, F., Kruse, R.: A novel approach to noise clustering for outlier detection. Soft. Comput. **11**, 489–494 (2007)
19. Santos-Pereira, C., Pires, A.: Detection of outliers in multivariate data: a method based on clustering and robust estimators. In: Proc. 5th Annual Conference of the Advanced School for Computing and Imaging, pp. 291–296. Physica, Berlin (2002)
20. Scott, S.: On optimal and data-based histograms. Biometrika **66**, 605–610 (1979)

21. Scott, D.: Sturges' rule. In: Wiley Interdisciplinary Reviews: Computational Statistics, vol. 1, pp. 303–306. Wiley, Chichester (2009)
22. Sturges, H.: The choice of a class interval. J. Am. Stat. Assoc. **21**, 65–66 (1926)
23. Tschumitschew, K., Klawonn, F.: Veda: statistical tests for finding interesting visualisations. In: Knowledge-Based and Intelligent Information and Engineering Systems 2009, Part II, pp. 236–243. Springer, Berlin (2009)
24. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley, Reading (1977)