(c) Recall from Section 11.5.2 that in the case of a single binary covariate, the log-rank test statistic should be identical to the score statistic for the Cox model. Conduct a log-rank test to determine whether there is a difference between the survival curves for the two groups. How does the p-value for the log-rank test statistic compare to the p-value for the score statistic for the Cox model from (b)?

12
Unsupervised Learning

Most of this book concerns *supervised learning* methods such as regression and classification. In the supervised learning setting, we typically have access to a set of p features $X_1, X_2, \ldots, X_p$, measured on n observations, and a response Y also measured on those same n observations. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.

This chapter will instead focus on *unsupervised learning*, a set of statistical tools intended for the setting in which we have only a set of features $X_1, X_2, \ldots, X_p$ measured on n observations. We are not interested in prediction, because we do not have an associated response variable Y. Rather, the goal is to discover interesting things about the measurements on $X_1, X_2, \ldots, X_p$. Is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations? Unsupervised learning refers to a diverse set of techniques for answering questions such as these. In this chapter, we will focus on two particular types of unsupervised learning: *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and *clustering*, a broad class of methods for discovering unknown subgroups in data.

12.1 The Challenge of Unsupervised Learning

Supervised learning is a well-understood area. In fact, if you have read the preceding chapters in this book, then you should by now have a good

G. James et al., *An Introduction to Statistical Learning*, Springer Texts in Statistics,
https://doi.org/10.1007/978-1-0716-1418-1_12

grasp of supervised learning. For instance, if you are asked to predict a binary outcome from a data set, you have a very well developed set of tools at your disposal (such as logistic regression, linear discriminant analysis, classification trees, support vector machines, and more) as well as a clear understanding of how to assess the quality of the results obtained (using cross-validation, validation on an independent test set, and so forth).

In contrast, unsupervised learning is often much more challenging. The exercise tends to be more subjective, and there is no simple goal for the analysis, such as prediction of a response. Unsupervised learning is often performed as part of an *exploratory data analysis*. Furthermore, it can be hard to assess the results obtained from unsupervised learning methods, since there is no universally accepted mechanism for performing cross-validation or validating results on an independent data set. The reason for this difference is simple. If we fit a predictive model using a supervised learning technique, then it is possible to *check our work* by seeing how well our model predicts the response Y on observations not used in fitting the model. However, in unsupervised learning, there is no way to check our work because we don't know the true answer—the problem is unsupervised.

exploratory data analysis

Techniques for unsupervised learning are of growing importance in a number of fields. A cancer researcher might assay gene expression levels in 100 patients with breast cancer. He or she might then look for subgroups among the breast cancer samples, or among the genes, in order to obtain a better understanding of the disease. An online shopping site might try to identify groups of shoppers with similar browsing and purchase histories, as well as items that are of particular interest to the shoppers within each group. Then an individual shopper can be preferentially shown the items in which he or she is particularly likely to be interested, based on the purchase histories of similar shoppers. A search engine might choose which search results to display to a particular individual based on the click histories of other individuals with similar search patterns. These statistical learning tasks, and many more, can be performed via unsupervised learning techniques.

12.2 Principal Components Analysis

Principal components are discussed in Section 6.3.1 in the context of principal components regression. When faced with a large set of correlated variables, principal components allow us to summarize this set with a smaller number of representative variables that collectively explain most of the variability in the original set. The principal component directions are presented in Section 6.3.1 as directions in feature space along which the original data are *highly variable*. These directions also define lines and subspaces that are *as close as possible* to the data cloud. To perform

principal components regression, we simply use principal components as predictors in a regression model in place of the original larger set of variables.

Principal components analysis (PCA) refers to the process by which principal components are computed, and the subsequent use of these components in understanding the data. PCA is an unsupervised approach, since it involves only a set of features $X_1, X_2, \ldots, X_p$, and no associated response Y. Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization (visualization of the observations or visualization of the variables). It can also be used as a tool for data imputation — that is, for filling in missing values in a data matrix.

principal components analysis

We now discuss PCA in greater detail, focusing on the use of PCA as a tool for unsupervised data exploration, in keeping with the topic of this chapter.

12.2.1 What Are Principal Components?

Suppose that we wish to visualize n observations with measurements on a set of p features, $X_1, X_2, \ldots, X_p$, as part of an exploratory data analysis. We could do this by examining two-dimensional scatterplots of the data, each of which contains the n observations' measurements on two of the features. However, there are $\binom{p}{2} = p(p-1)/2$ such scatterplots; for example, with $p = 10$ there are 45 plots! If p is large, then it will certainly not be possible to look at all of them; moreover, most likely none of them will be informative since they each contain just a small fraction of the total information present in the data set. Clearly, a better method is required to visualize the n observations when p is large. In particular, we would like to find a low-dimensional representation of the data that captures as much of the information as possible. For instance, if we can obtain a two-dimensional representation of the data that captures most of the information, then we can plot the observations in this low-dimensional space.

PCA provides a tool to do just this. It finds a low-dimensional representation of a data set that contains as much as possible of the variation. The idea is that each of the n observations lives in p-dimensional space, but not all of these dimensions are equally interesting. PCA seeks a small number of dimensions that are as interesting as possible, where the concept of *interesting* is measured by the amount that the observations vary along each dimension. Each of the dimensions found by PCA is a linear combination of the p features. We now explain the manner in which these dimensions, or *principal components*, are found.

The *first principal component* of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p \tag{12.1}$$

that has the largest variance. By *normalized*, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. We refer to the elements $\phi_{11}, \ldots, \phi_{p1}$ as the *loadings* of the first principal component; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11}\ \phi_{21}\ \ldots\ \phi_{p1})^T$. We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

loading

Given a $n \times p$ data set $\mathbf{X}$, how do we compute the first principal component? Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero). We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \cdots + \phi_{p1} x_{ip} \tag{12.2}$$

that has largest sample variance, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. In other words, the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1. \tag{12.3}$$

From (12.2) we can write the objective in (12.3) as $\frac{1}{n} \sum_{i=1}^{n} z_{i1}^2$. Since $\frac{1}{n} \sum_{i=1}^{n} x_{ij} = 0$, the average of the $z_{11}, \ldots, z_{n1}$ will be zero as well. Hence the objective that we are maximizing in (12.3) is just the sample variance of the n values of z_{i1}. We refer to $z_{11}, \ldots, z_{n1}$ as the *scores* of the first principal component. Problem (12.3) can be solved via an *eigen decomposition*, a standard technique in linear algebra, but details are outside of the scope of this book.[1]

score

eigen decomposition

There is a nice geometric interpretation for the first principal component. The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ defines a direction in feature space along which the data vary the most. If we project the n data points $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ themselves. For instance, Figure 6.14 on page 253 displays the first principal component loading vector (green solid line) on an advertising data set. In these data, there are only two features, and so the observations as well as the first principal component loading vector can be easily displayed. As can be seen from (6.19), in that data set $\phi_{11} = 0.839$ and $\phi_{21} = 0.544$.

After the first principal component Z_1 of the features has been determined, we can find the second principal component Z_2. The second prin-

[1] As an alternative to the eigen decomposition, a related technique called the singular value decomposition can be used. This will be explored in the lab at the end of this chapter.

cipal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance out of all linear combinations that are *uncorrelated* with Z_1. The second principal component scores $z_{12}, z_{22}, \ldots, z_{n2}$ take the form

$$z_{i2} = \phi_{12} x_{i1} + \phi_{22} x_{i2} + \cdots + \phi_{p2} x_{ip}, \tag{12.4}$$

where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, \ldots, \phi_{p2}$. It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be orthogonal (perpendicular) to the direction ϕ_1. In the example in Figure 6.14, the observations lie in two-dimensional space (since $p = 2$), and so once we have found ϕ_1, there is only one possibility for ϕ_2, which is shown as a blue dashed line. (From Section 6.3.1, we know that $\phi_{12} = 0.544$ and $\phi_{22} = -0.839$.) But in a larger data set with $p > 2$ variables, there are multiple distinct principal components, and they are defined in a similar manner. To find ϕ_2, we solve a problem similar to (12.3) with ϕ_2 replacing ϕ_1, and with the additional constraint that ϕ_2 is orthogonal to ϕ_1.[2]

Once we have computed the principal components, we can plot them against each other in order to produce low-dimensional views of the data. For instance, we can plot the score vector Z_1 against Z_2, Z_1 against Z_3, Z_2 against Z_3, and so forth. Geometrically, this amounts to projecting the original data down onto the subspace spanned by ϕ_1, ϕ_2, and ϕ_3, and plotting the projected points.

We illustrate the use of PCA on the `USArrests` data set. For each of the 50 states in the United States, the data set contains the number of arrests per 100,000 residents for each of three crimes: `Assault`, `Murder`, and `Rape`. We also record `UrbanPop` (the percent of the population in each state living in urban areas). The principal component score vectors have length $n = 50$, and the principal component loading vectors have length $p = 4$. PCA was performed after standardizing each variable to have mean zero and standard deviation one. Figure 12.1 plots the first two principal components of these data. The figure represents both the principal component scores and the loading vectors in a single *biplot* display. The loadings are also given in Table 12.1.

biplot

In Figure 12.1, we see that the first loading vector places approximately equal weight on `Assault`, `Murder`, and `Rape`, but with much less weight on `UrbanPop`. Hence this component roughly corresponds to a measure of overall rates of serious crimes. The second loading vector places most of its weight on `UrbanPop` and much less weight on the other three features. Hence, this component roughly corresponds to the level of urbanization of the state. Overall, we see that the crime-related variables (`Murder`, `Assault`, and `Rape`) are located close to each other, and that the `UrbanPop` variable is far from the

[2]On a technical note, the principal component directions ϕ_1, ϕ_2, $\phi_3, \ldots$ are the ordered sequence of eigenvectors of the matrix $\mathbf{X}^T\mathbf{X}$, and the variances of the components are the eigenvalues. There are at most $\min(n-1, p)$ principal components.

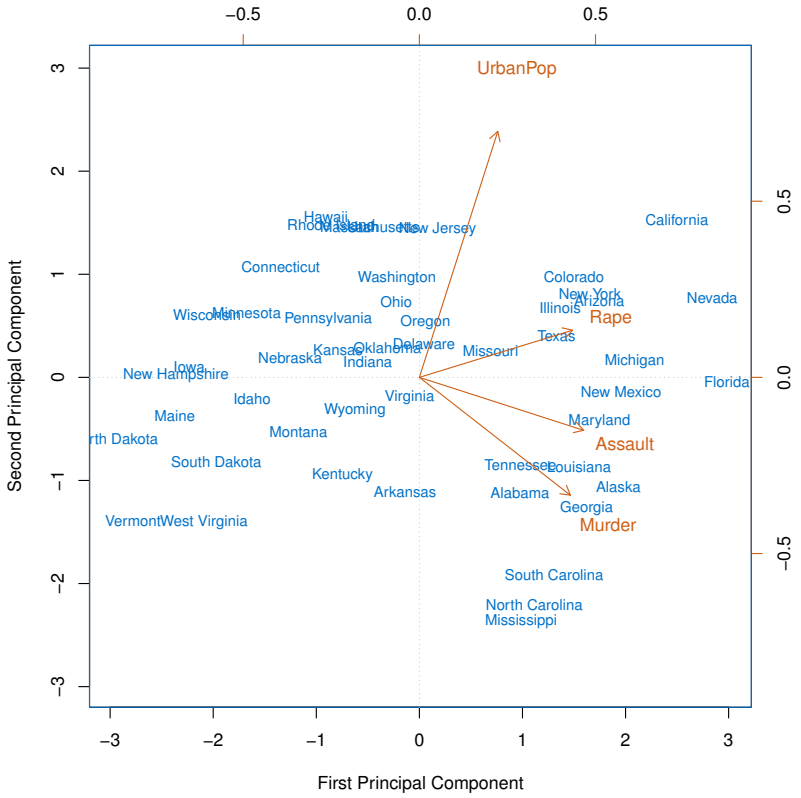

FIGURE 12.1. *The first two principal components for the* `USArrests` *data. The blue state names represent the scores for the first two principal components. The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for* `Rape` *on the first component is* 0.54*, and its loading on the second principal component* 0.17 *(the word* `Rape` *is centered at the point* $(0.54, 0.17)$*). This figure is known as a biplot, because it displays both the principal component scores and the principal component loadings.*

other three. This indicates that the crime-related variables are correlated with each other—states with high murder rates tend to have high assault and rape rates—and that the `UrbanPop` variable is less correlated with the other three.

We can examine differences between the states via the two principal component score vectors shown in Figure 12.1. Our discussion of the loading vectors suggests that states with large positive scores on the first component, such as California, Nevada and Florida, have high crime rates, while states like North Dakota, with negative scores on the first component, have

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

TABLE 12.1. *The principal component loading vectors, ϕ_1 and ϕ_2, for the* USArrests *data. These are also displayed in Figure 12.1.*

low crime rates. California also has a high score on the second component, indicating a high level of urbanization, while the opposite is true for states like Mississippi. States close to zero on both components, such as Indiana, have approximately average levels of both crime and urbanization.

12.2.2 Another Interpretation of Principal Components

The first two principal component loading vectors in a simulated three-dimensional data set are shown in the left-hand panel of Figure 12.2; these two loading vectors span a plane along which the observations have the highest variance.

In the previous section, we describe the principal component loading vectors as the directions in feature space along which the data vary the most, and the principal component scores as projections along these directions. However, an alternative interpretation for principal components can also be useful: principal components provide low-dimensional linear surfaces that are *closest* to the observations. We expand upon that interpretation here.[3]

The first principal component loading vector has a very special property: it is the line in p-dimensional space that is *closest* to the n observations (using average squared Euclidean distance as a measure of closeness). This interpretation can be seen in the left-hand panel of Figure 6.15; the dashed lines indicate the distance between each observation and the line defined by the first principal component loading vector. The appeal of this interpretation is clear: we seek a single dimension of the data that lies as close as possible to all of the data points, since such a line will likely provide a good summary of the data.

The notion of principal components as the dimensions that are closest to the n observations extends beyond just the first principal component. For instance, the first two principal components of a data set span the plane that is closest to the n observations, in terms of average squared Euclidean distance. An example is shown in the left-hand panel of Figure 12.2. The first three principal components of a data set span

[3]In this section, we continue to assume that each column of the data matrix $\mathbf{X}$ has been centered to have mean zero—that is, the column mean has been subtracted from each column.

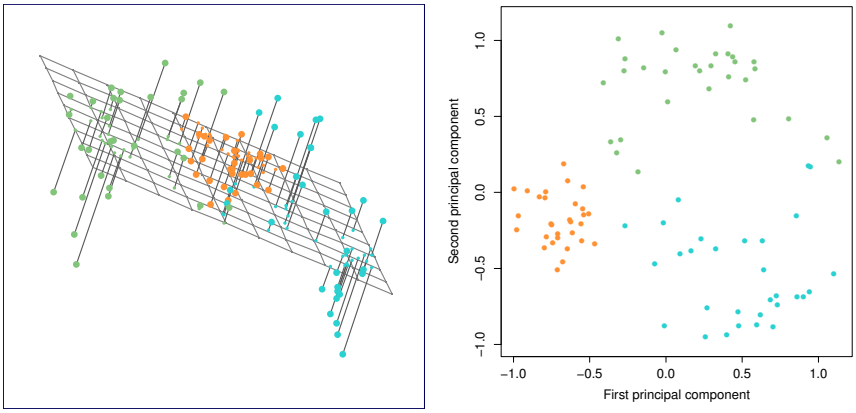

FIGURE 12.2. *Ninety observations simulated in three dimensions. The observations are displayed in color for ease of visualization.* Left: *the first two principal component directions span the plane that best fits the data. The plane is positioned to minimize the sum of squared distances to each point.* Right: *the first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane.*

the three-dimensional hyperplane that is closest to the n observations, and so forth.

Using this interpretation, together the first M principal component score vectors and the first M principal component loading vectors provide the best M-dimensional approximation (in terms of Euclidean distance) to the ith observation x_{ij}. This representation can be written as

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}. \tag{12.5}$$

We can state this more formally by writing down an optimization problem. Suppose the data matrix $\mathbf{X}$ is column-centered. Out of all approximations of the form $x_{ij} \approx \sum_{m=1}^{M} a_{im}b_{jm}$, we could ask for the one with the smallest residual sum of squares:

$$\underset{\mathbf{A}\in\mathbb{R}^{n\times M},\mathbf{B}\in\mathbb{R}^{p\times M}}{\text{minimize}} \left\{ \sum_{j=1}^{p}\sum_{i=1}^{n} \left(x_{ij} - \sum_{m=1}^{M} a_{im}b_{jm} \right)^2 \right\}. \tag{12.6}$$

Here, $\mathbf{A}$ is a $n \times M$ matrix whose (i, m) element is a_{im}, and $\mathbf{B}$ is a $p \times M$ element whose (j, m) element is b_{jm}.

It can be shown that for any value of M, the columns of the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ that solve (12.6) are in fact the first M principal components score and loading vectors. In other words, if $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ solve (12.6), then

$\hat{a}_{im} = z_{im}$ and $\hat{b}_{jm} = \phi_{jm}$.[4] This means that the smallest possible value of the objective in (12.6) is

$$\sum_{j=1}^{p}\sum_{i=1}^{n}\left(x_{ij} - \sum_{m=1}^{M} z_{im}\phi_{jm}\right)^2. \tag{12.7}$$

In summary, together the M principal component score vectors and M principal component loading vectors can give a good approximation to the data when M is sufficiently large. When $M = \min(n-1, p)$, then the representation is exact: $x_{ij} = \sum_{m=1}^{M} z_{im}\phi_{jm}$.

12.2.3 The Proportion of Variance Explained

In Figure 12.2, we performed PCA on a three-dimensional data set (left-hand panel) and projected the data onto the first two principal component loading vectors in order to obtain a two-dimensional view of the data (i.e. the principal component score vectors; right-hand panel). We see that this two-dimensional representation of the three-dimensional data does successfully capture the major pattern in the data: the orange, green, and cyan observations that are near each other in three-dimensional space remain nearby in the two-dimensional representation. Similarly, we have seen on the `USArrests` data set that we can summarize the 50 observations and 4 variables using just the first two principal component score vectors and the first two principal component loading vectors.

We can now ask a natural question: how much of the information in a given data set is lost by projecting the observations onto the first few principal components? That is, how much of the variance in the data is *not* contained in the first few principal components? More generally, we are interested in knowing the *proportion of variance explained* (PVE) by each principal component. The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

proportion of variance explained

$$\sum_{j=1}^{p}\mathrm{Var}(X_j) = \sum_{j=1}^{p}\frac{1}{n}\sum_{i=1}^{n}x_{ij}^2, \tag{12.8}$$

and the variance explained by the mth principal component is

$$\frac{1}{n}\sum_{i=1}^{n}z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2. \tag{12.9}$$

[4] Technically, the solution to (12.6) is not unique. Thus, it is more precise to state that any solution to (12.6) can be easily transformed to yield the principal components.

Therefore, the PVE of the mth principal component is given by

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}. \tag{12.10}$$

The PVE of each principal component is a positive quantity. In order to compute the cumulative PVE of the first M principal components, we can simply sum (12.10) over each of the first M PVEs. In total, there are $\min(n-1, p)$ principal components, and their PVEs sum to one.

In Section 12.2.2, we showed that the first M principal component loading and score vectors can be interpreted as the best M-dimensional approximation to the data, in terms of residual sum of squares. It turns out that the variance of the data can be decomposed into the variance of the first M principal components plus the mean squared error of this M-dimensional approximation, as follows:

$$\underbrace{\sum_{j=1}^{p}\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2}_{\text{Var. of data}} = \underbrace{\sum_{m=1}^{M}\frac{1}{n}\sum_{i=1}^{n} z_{im}^2}_{\text{Var. of first } M \text{ PCs}} + \underbrace{\frac{1}{n}\sum_{j=1}^{p}\sum_{i=1}^{n}\left(x_{ij} - \sum_{m=1}^{M} z_{im}\phi_{jm}\right)^2}_{\text{MSE of } M\text{-dimensional approximation}} \tag{12.11}$$

The three terms in this decomposition are discussed in (12.8), (12.9), and (12.7), respectively. Since the first term is fixed, we see that by maximizing the variance of the first M principal components, we minimize the mean squared error of the M-dimensional approximation, and vice versa. This explains why principal components can be equivalently viewed as minimizing the approximation error (as in Section 12.2.2) or maximizing the variance (as in Section 12.2.1).

Moreover, we can use (12.11) to see that the PVE defined in (12.10) equals

$$1 - \frac{\sum_{j=1}^{p}\sum_{i=1}^{n}\left(x_{ij} - \sum_{m=1}^{M} z_{im}\phi_{jm}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2} = 1 - \frac{\text{RSS}}{\text{TSS}},$$

where TSS represents the total sum of squared elements of $\mathbf{X}$, and RSS represents the residual sum of squares of the M-dimensional approximation given by the principal components. Recalling the definition of R^2 from (3.17), this means that we can interpret the PVE as the R^2 of the approximation for $\mathbf{X}$ given by the first M principal components.

In the `USArrests` data, the first principal component explains 62.0 % of the variance in the data, and the next principal component explains 24.7 % of the variance. Together, the first two principal components explain almost 87 % of the variance in the data, and the last two principal components explain only 13 % of the variance. This means that Figure 12.1 provides a pretty accurate summary of the data using just two dimensions. The PVE of each principal component, as well as the cumulative PVE, is shown in Figure 12.3. The left-hand panel is known as a *scree plot*, and will be discussed later in this chapter.

scree plot

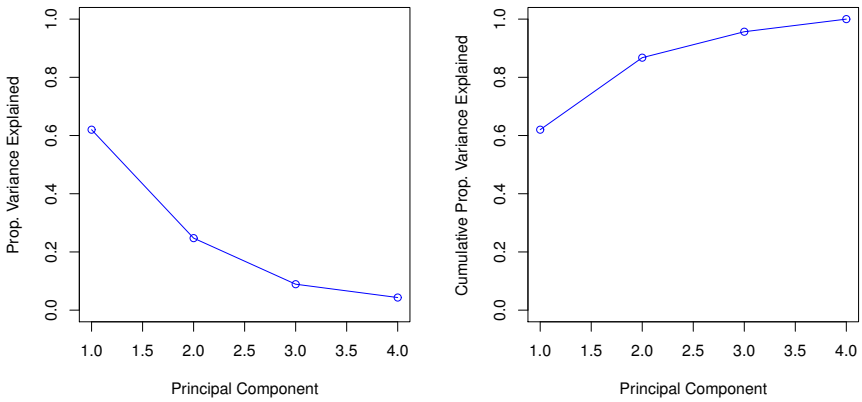

FIGURE 12.3. Left: *a scree plot depicting the proportion of variance explained by each of the four principal components in the* `USArrests` *data.* Right: *the cumulative proportion of variance explained by the four principal components in the* `USArrests` *data.*

12.2.4 More on PCA

Scaling the Variables

We have already mentioned that before PCA is performed, the variables should be centered to have mean zero. Furthermore, *the results obtained when we perform PCA will also depend on whether the variables have been individually scaled* (each multiplied by a different constant). This is in contrast to some other supervised and unsupervised learning techniques, such as linear regression, in which scaling the variables has no effect. (In linear regression, multiplying a variable by a factor of c will simply lead to multiplication of the corresponding coefficient estimate by a factor of $1/c$, and thus will have no substantive effect on the model obtained.)

For instance, Figure 12.1 was obtained after scaling each of the variables to have standard deviation one. This is reproduced in the left-hand plot in Figure 12.4. Why does it matter that we scaled the variables? In these data, the variables are measured in different units; `Murder`, `Rape`, and `Assault` are reported as the number of occurrences per 100,000 people, and `UrbanPop` is the percentage of the state's population that lives in an urban area. These four variables have variances of 18.97, 87.73, 6945.16, and 209.5, respectively. Consequently, if we perform PCA on the unscaled variables, then the first principal component loading vector will have a very large loading for `Assault`, since that variable has by far the highest variance. The right-hand plot in Figure 12.4 displays the first two principal components for the `USArrests` data set, without scaling the variables to have standard deviation one. As predicted, the first principal component loading vector places almost all of its weight on `Assault`, while the second principal component

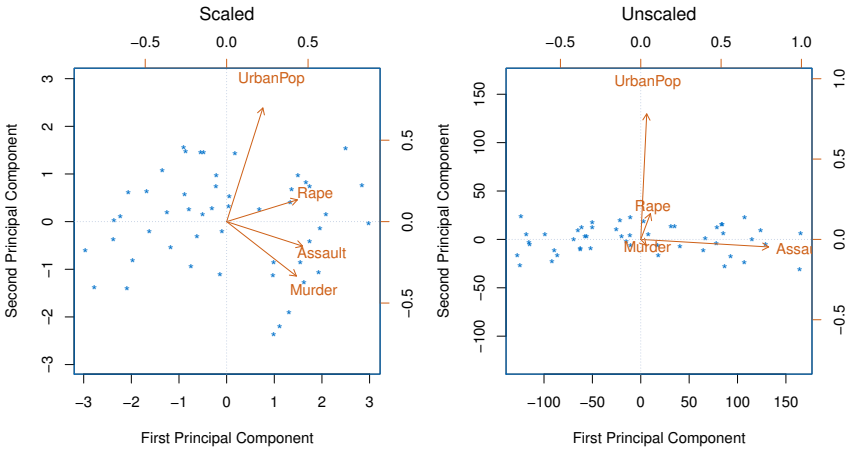

FIGURE 12.4. *Two principal component biplots for the* `USArrests` *data.* Left: *the same as Figure 12.1, with the variables scaled to have unit standard deviations.* Right: *principal components using unscaled data.* `Assault` *has by far the largest loading on the first principal component because it has the highest variance among the four variables. In general, scaling the variables to have standard deviation one is recommended.*

loading vector places almost all of its weight on `UrpanPop`. Comparing this to the left-hand plot, we see that scaling does indeed have a substantial effect on the results obtained.

However, this result is simply a consequence of the scales on which the variables were measured. For instance, if `Assault` were measured in units of the number of occurrences per 100 people (rather than number of occurrences per 100,000 people), then this would amount to dividing all of the elements of that variable by 1,000. Then the variance of the variable would be tiny, and so the first principal component loading vector would have a very small value for that variable. Because it is undesirable for the principal components obtained to depend on an arbitrary choice of scaling, we typically scale each variable to have standard deviation one before we perform PCA.

In certain settings, however, the variables may be measured in the same units. In this case, we might not wish to scale the variables to have standard deviation one before performing PCA. For instance, suppose that the variables in a given data set correspond to expression levels for p genes. Then since expression is measured in the same "units" for each gene, we might choose not to scale the genes to each have standard deviation one.

Uniqueness of the Principal Components

Each principal component loading vector is unique, up to a sign flip. This means that two different software packages will yield the same principal component loading vectors, although the signs of those loading vectors may differ. The signs may differ because each principal component loading vector specifies a direction in p-dimensional space: flipping the sign has no effect as the direction does not change. (Consider Figure 6.14—the principal component loading vector is a line that extends in either direction, and flipping its sign would have no effect.) Similarly, the score vectors are unique up to a sign flip, since the variance of Z is the same as the variance of $-Z$. It is worth noting that when we use (12.5) to approximate x_{ij} we multiply z_{im} by ϕ_{jm}. Hence, if the sign is flipped on both the loading and score vectors, the final product of the two quantities is unchanged.

Deciding How Many Principal Components to Use

In general, a $n \times p$ data matrix $\mathbf{X}$ has $\min(n-1, p)$ distinct principal components. However, we usually are not interested in all of them; rather, we would like to use just the first few principal components in order to visualize or interpret the data. In fact, we would like to use the smallest number of principal components required to get a *good* understanding of the data. How many principal components are needed? Unfortunately, there is no single (or simple!) answer to this question.

We typically decide on the number of principal components required to visualize the data by examining a *scree plot*, such as the one shown in the left-hand panel of Figure 12.3. We choose the smallest number of principal components that are required in order to explain a sizable amount of the variation in the data. This is done by eyeballing the scree plot, and looking for a point at which the proportion of variance explained by each subsequent principal component drops off. This drop is often referred to as an *elbow* in the scree plot. For instance, by inspection of Figure 12.3, one might conclude that a fair amount of variance is explained by the first two principal components, and that there is an elbow after the second component. After all, the third principal component explains less than ten percent of the variance in the data, and the fourth principal component explains less than half that and so is essentially worthless.

However, this type of visual analysis is inherently *ad hoc*. Unfortunately, there is no well-accepted objective way to decide how many principal components are *enough*. In fact, the question of how many principal components are enough is inherently ill-defined, and will depend on the specific area of application and the specific data set. In practice, we tend to look at the first few principal components in order to find interesting patterns in the data. If no interesting patterns are found in the first few principal components, then further principal components are unlikely to be of interest. Conversely, if the first few principal components are interesting, then

we typically continue to look at subsequent principal components until no further interesting patterns are found. This is admittedly a subjective approach, and is reflective of the fact that PCA is generally used as a tool for exploratory data analysis.

On the other hand, if we compute principal components for use in a supervised analysis, such as the principal components regression presented in Section 6.3.1, then there is a simple and objective way to determine how many principal components to use: we can treat the number of principal component score vectors to be used in the regression as a tuning parameter to be selected via cross-validation or a related approach. The comparative simplicity of selecting the number of principal components for a supervised analysis is one manifestation of the fact that supervised analyses tend to be more clearly defined and more objectively evaluated than unsupervised analyses.

12.2.5 Other Uses for Principal Components

We saw in Section 6.3.1 that we can perform regression using the principal component score vectors as features. In fact, many statistical techniques, such as regression, classification, and clustering, can be easily adapted to use the $n \times M$ matrix whose columns are the first $M \ll p$ principal component score vectors, rather than using the full $n \times p$ data matrix. This can lead to *less noisy* results, since it is often the case that the signal (as opposed to the noise) in a data set is concentrated in its first few principal components.

12.3 Missing Values and Matrix Completion

Often datasets have missing values, which can be a nuisance. For example, suppose that we wish to analyze the `USArrests` data, and discover that 20 of the 200 values have been randomly corrupted and marked as missing. Unfortunately, the statistical learning methods that we have seen in this book cannot handle missing values. How should we proceed?

We could remove the rows that contain missing observations and perform our data analysis on the complete rows. But this seems wasteful, and depending on the fraction missing, unrealistic. Alternatively, if x_{ij} is missing, then we could replace it by the mean of the jth column (using the non-missing entries to compute the mean). Although this is a common and convenient strategy, often we can do better by exploiting the correlation between the variables.

In this section we show how principal components can be used to *impute* the missing values, through a process known as *matrix completion*. The

impute
imputation
matrix completion

completed matrix can then be used in a statistical learning method, such as linear regression or LDA.

This approach for imputing missing data is appropriate if the missingness is random. For example, it is suitable if a patient's weight is missing because the battery of the electronic scale was flat at the time of his exam. By contrast, if the weight is missing because the patient was too heavy to climb on the scale, then this is not missing at random; the missingness is informative, and the approach described here for handling missing data is not suitable.

missing at random

Sometimes data is missing by necessity. For example, if we form a matrix of the ratings (on a scale from 1 to 5) that n customers have given to the entire Netflix catalog of p movies, then most of the matrix will be missing, since no customer will have seen and rated more than a tiny fraction of the catalog. If we can impute the missing values well, then we will have an idea of what each customer will think of movies they have not yet seen. Hence matrix completion can be used to power *recommender systems*.

recommender systems

Principal Components with Missing Values

In Section 12.2.2, we showed that the first M principal component score and loading vectors provide the "best" approximation to the data matrix $\mathbf{X}$, in the sense of (12.6). Suppose that some of the observations x_{ij} are missing. We now show how one can both impute the missing values and solve the principal component problem at the same time. We return to a modified form of the optimization problem (12.6),

$$\underset{\mathbf{A}\in\mathbb{R}^{n\times M},\mathbf{B}\in\mathbb{R}^{p\times M}}{\text{minimize}} \left\{ \sum_{(i,j)\in\mathcal{O}} \left(x_{ij} - \sum_{m=1}^{M} a_{im} b_{jm} \right)^2 \right\}, \tag{12.12}$$

where $\mathcal{O}$ is the set of all *observed* pairs of indices (i, j), a subset of the possible $n \times p$ pairs.

Once we solve this problem:

- we can estimate a missing observation x_{ij} using $\hat{x}_{ij} = \sum_{m=1}^{M} \hat{a}_{im}\hat{b}_{jm}$, where $\hat{a}_{im}$ and $\hat{b}_{jm}$ are the (i, m) and (j, m) elements, respectively, of the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ that solve (12.12); and

- we can (approximately) recover the M principal component scores and loadings, as we did when the data were complete.

It turns out that solving (12.12) exactly is difficult, unlike in the case of complete data: the eigen decomposition no longer applies. But the sim-

Algorithm 12.1 *Iterative Algorithm for Matrix Completion*

1. Create a complete data matrix $\tilde{\mathbf{X}}$ of dimension $n \times p$ of which the (i, j) element equals

$$\tilde{x}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in \mathcal{O} \\ \bar{x}_j & \text{if } (i,j) \notin \mathcal{O}, \end{cases}$$

   where $\bar{x}_j$ is the average of the observed values for the jth variable in the incomplete data matrix $\mathbf{X}$. Here, $\mathcal{O}$ indexes the observations that are observed in $\mathbf{X}$.

2. Repeat steps (a)–(c) until the objective (12.14) fails to decrease:

   (a) Solve

$$\underset{\mathbf{A}\in\mathbb{R}^{n\times M},\mathbf{B}\in\mathbb{R}^{p\times M}}{\text{minimize}} \left\{ \sum_{j=1}^{p} \sum_{i=1}^{n} \left(\tilde{x}_{ij} - \sum_{m=1}^{M} a_{im} b_{jm} \right)^2 \right\} \tag{12.13}$$

   by computing the principal components of $\tilde{\mathbf{X}}$.

   (b) For each element $(i, j) \notin \mathcal{O}$, set $\tilde{x}_{ij} \leftarrow \sum_{m=1}^{M} \hat{a}_{im} \hat{b}_{jm}$.

   (c) Compute the objective

$$\sum_{(i,j)\in\mathcal{O}} \left(x_{ij} - \sum_{m=1}^{M} \hat{a}_{im} \hat{b}_{jm} \right)^2. \tag{12.14}$$

3. Return the estimated missing entries $\tilde{x}_{ij}$, $(i, j) \notin \mathcal{O}$.

ple iterative approach in Algorithm 12.1, which is demonstrated in Section 12.5.2, typically provides a good solution.[5][6]

We illustrate Algorithm 12.1 on the `USArrests` data. There are $p = 4$ variables and $n = 50$ observations (states). We first standardized the data so each variable has mean zero and standard deviation one. We then randomly selected 20 of the 50 states, and then for each of these we randomly set one of the four variables to be missing. Thus, 10% of the elements of the data matrix were missing. We applied Algorithm 12.1 with $M = 1$ principal component. Figure 12.5 shows that the recovery of the missing elements

[5]This algorithm is referred to as "Hard-Impute" in Mazumder, Hastie, and Tibshirani (2010) "Spectral regularization algorithms for learning large incomplete matrices", published in *Journal of Machine Learning Research*, pages 2287–2322.

[6]Each iteration of Step 2 of this algorithm decreases the objective (12.14). However, the algorithm is not guaranteed to achieve the global optimum of (12.12).

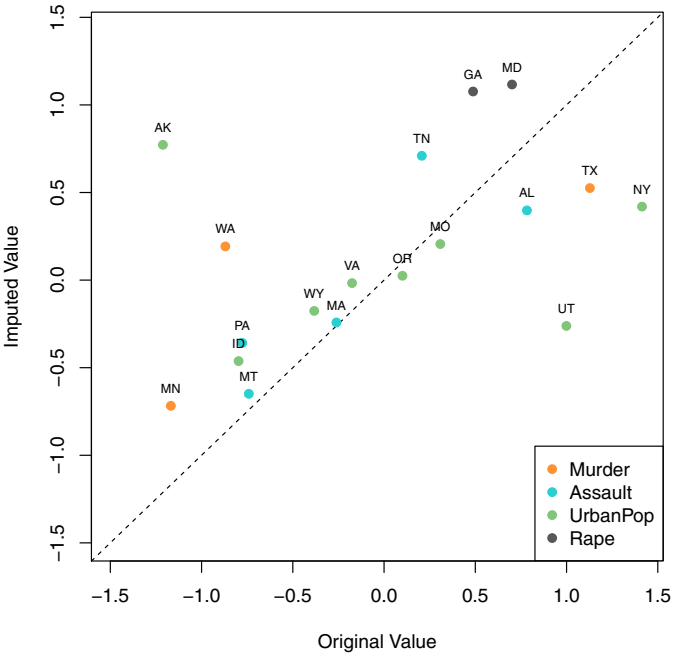

FIGURE 12.5. *Missing value imputation on the* `USArrests` *data. Twenty values (10% of the total number of matrix elements) were artificially set to be missing, and then imputed via Algorithm 12.1 with* $M = 1$. *The figure displays the true value* x_{ij} *and the imputed value* $\hat{x}_{ij}$ *for all twenty missing values. For each of the twenty missing values, the color indicates the variable, and the label indicates the state. The correlation between the true and imputed values is around* 0.63.

is pretty accurate. Over 100 random runs of this experiment, the average correlation between the true and imputed values of the missing elements is 0.63, with a standard deviation of 0.11. Is this good performance? To answer this question, we can compare this correlation to what we would have gotten if we had estimated these 20 values using the *complete* data — that is, if we had simply computed $\hat{x}_{ij} = z_{i1}\phi_{j1}$, where z_{i1} and ϕ_{j1} are elements of the first principal component score and loading vectors of the complete data.[7] Using the complete data in this way results in an average correlation of 0.79 between the true and estimated values for these 20 elements, with a standard deviation of 0.08. Thus, our imputation method does worse than the method that uses all of the data (0.63 ± 0.11 versus 0.79 ± 0.08), but its performance is still pretty good. (And of course, the

[7]This is an unattainable gold standard, in the sense that with missing data, we of course cannot compute the principal components of the complete data.

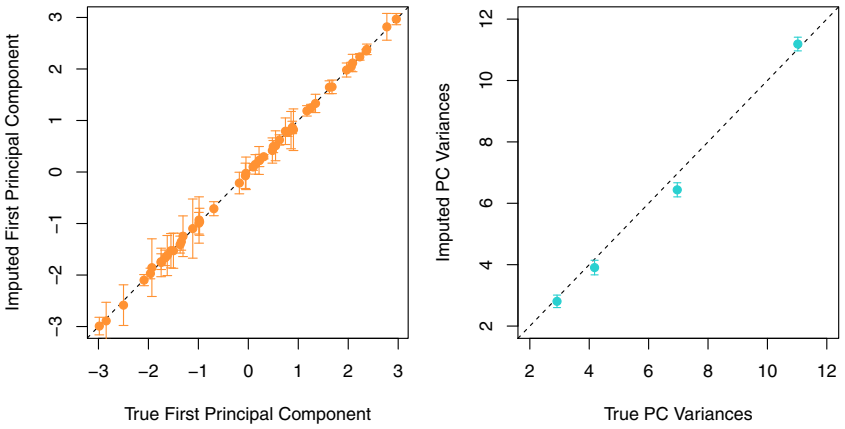

FIGURE 12.6. *As described in the text, in each of 100 trials, we left out 20 elements of the* `USArrests` *dataset. In each trial, we applied Algorithm 12.1 with* $M = 1$ *to impute the missing elements and compute the principal components.* Left: *For each of the 50 states, the imputed first principal component scores (averaged over 100 trials, and displayed with a standard deviation bar) are plotted against the first principal component scores computed using all the data.* Right: *The imputed principal component loadings (averaged over 100 trials, and displayed with a standard deviation bar) are plotted against the true principal component loadings.*

method that uses all of the data cannot be applied in a real-world setting with missing data.)

Figure 12.6 further indicates that Algorithm 12.1 performs fairly well on this dataset.

We close with a few observations:

- The `USArrests` data has only four variables, which is on the low end for methods like Algorithm 12.1 to work well. For this reason, for this demonstration we randomly set at most one variable per state to be missing, and only used $M = 1$ principal component.
- In general, in order to apply Algorithm 12.1, we must select M, the number of principal components to use for the imputation. One approach is to randomly leave out a few additional elements from the matrix, and select M based on how well those known values are recovered. This is closely related to the validation-set approach seen in Chapter 5.

Recommender Systems

Digital streaming services like Netflix and Amazon use data about the content that a customer has viewed in the past, as well as data from other

	Jerry Maguire	Oceans	Road to Perdition	A Fortunate Man	Catch Me If You Can	Driving Miss Daisy	The Two Popes	The Laundromat	Code 8	The Social Network	⋯
Customer 1	•	•	•	•	4	•	•	•	•	•	⋯
Customer 2	•	•	3	•	•	•	3	•	•	3	⋯
Customer 3	•	2	•	4	•	•	•	•	2	•	⋯
Customer 4	3	•	•	•	•	•	•	•	•	•	⋯
Customer 5	5	1	•	•	4	•	•	•	•	•	⋯
Customer 6	•	•	•	•	•	2	4	•	•	•	⋯
Customer 7	•	•	5	•	•	•	•	3	•	•	⋯
Customer 8	•	•	•	•	•	•	•	•	•	•	⋯
Customer 9	3	•	•	•	5	•	•	1	•	•	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

TABLE 12.2. *Excerpt of the Netflix movie rating data. The movies are rated from 1 (worst) to 5 (best). The symbol • represents a missing value: a movie that was not rated by the corresponding customer.*

customers, to suggest other content for the customer. As a concrete example, some years back, Netflix had customers rate each movie that they had seen with a score from 1–5. This resulted in a very big $n \times p$ matrix for which the (i, j) element is the rating given by the ith customer to the jth movie. One specific early example of this matrix had $n = 480{,}189$ customers and $p = 17{,}770$ movies. However, on average each customer had seen around 200 movies, so 99% of the matrix had missing elements. Table 12.3 illustrates the setup.

In order to suggest a movie that a particular customer might like, Netflix needed a way to impute the missing values of this data matrix. The key idea is as follows: the set of movies that the ith customer has seen will overlap with those that other customers have seen. Furthermore, some of those other customers will have similar movie preferences to the ith customer. Thus, it should be possible to use similar customers' ratings of movies that the ith customer has not seen to predict whether the ith customer will like those movies.

More concretely, by applying Algorithm 12.1, we can predict the ith customer's rating for the jth movie using $\hat{x}_{ij} = \sum_{m=1}^{M} \hat{a}_{im}\hat{b}_{jm}$. Furthermore, we can interpret the M components in terms of "cliques" and "genres":

- $\hat{a}_{im}$ represents the strength with which the ith user belongs to the mth clique, where a *clique* is a group of customers that enjoys movies of the mth genre;
- $\hat{b}_{jm}$ represents the strength with which the jth movie belongs to the mth *genre*.

Examples of genres include Romance, Western, and Action.

Principal component models similar to Algorithm 12.1 are at the heart of many recommender systems. Although the data matrices involved are typically massive, algorithms have been developed that can exploit the high level of missingness in order to perform efficient computations.

12.4 Clustering Methods

clustering

Clustering refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set. When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other. Of course, to make this concrete, we must define what it means for two or more observations to be *similar* or *different*. Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

For instance, suppose that we have a set of n observations, each with p features. The n observations could correspond to tissue samples for patients with breast cancer, and the p features could correspond to measurements collected for each tissue sample; these could be clinical measurements, such as tumor stage or grade, or they could be gene expression measurements. We may have a reason to believe that there is some heterogeneity among the n tissue samples; for instance, perhaps there are a few different *unknown* subtypes of breast cancer. Clustering could be used to find these subgroups. This is an unsupervised problem because we are trying to discover structure—in this case, distinct clusters—on the basis of a data set. The goal in supervised problems, on the other hand, is to try to predict some outcome vector such as survival time or response to drug treatment.

Both clustering and PCA seek to simplify the data via a small number of summaries, but their mechanisms are different:

- PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance;
- Clustering looks to find homogeneous subgroups among the observations.

Another application of clustering arises in marketing. We may have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people. Our goal is to perform *market segmentation* by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product. The task of performing market segmentation amounts to clustering the people in the data set.

Since clustering is popular in many fields, there exist a great number of clustering methods. In this section we focus on perhaps the two best-known clustering approaches: *K-means clustering* and *hierarchical clustering*. In K-means clustering, we seek to partition the observations into a pre-specified number of clusters. On the other hand, in hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n. There are advantages and disadvantages to each of these clustering approaches, which we highlight in this chapter.

K-means clustering

hierarchical clustering

dendrogram

In general, we can cluster observations on the basis of the features in order to identify subgroups among the observations, or we can cluster features on the basis of the observations in order to discover subgroups among the features. In what follows, for simplicity we will discuss clustering observations on the basis of the features, though the converse can be performed by simply transposing the data matrix.

12.4.1 K-Means Clustering

K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters. To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters. Figure 12.7 shows the results obtained from performing K-means clustering on a simulated example consisting of 150 observations in two dimensions, using three different values of K.

The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$. The idea behind K-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible. The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other. Hence we want to solve the problem

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}. \tag{12.15}$$