21:06 Thursday 17th March, 2016
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Chapter 16

Principal Components Analysis

In Chapter 14, we saw that kernel density estimation gives us, in principle, a consistent way of nonparametrically estimating joint distributions for arbitrarily many variables. We also saw (§14.4.2) that, like regression (§9.3), density estimation suffers from the curse of dimensionality — the amount of data needed grows exponentially with the number of variables. Moreover, this is not a flaw in kernel methods, but reflects the intrinsic difficulty of the problem.

Accordingly, to go forward in multivariate data analysis, we need to somehow lift the curse of dimensionality. One approach is to hope that while we have a large number p of variables, the data is really only q-dimensional, and $q \ll p$. The next few chapters will explore various ways of finding low-dimensional structure. Alternatively, we could hope that while the data really does have lots of dimensions, it also has lots of independent parts. At an extreme, if it had p dimensions but we knew they were all statistically independent, we'd just do p one-dimensional density estimates. Chapter 20 and its sequels are concerned with this second approach, of factoring the joint distribution into independent or conditionally-independent pieces.

Principal components analysis (PCA) is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more tractable, lower-dimensional form, without losing too much information. PCA is one of the simplest and most robust ways of doing such **dimensionality reduction**. The hope with PCA is that the data lie in, or close to, a low-dimensional linear subspace.

16.1 Mathematics of Principal Components

We start with p-dimensional vectors, and want to summarize them by projecting down into a q-dimensional subspace. Our summary will be the projection of the original vectors on to q directions, the **principal components**, which span the subspace.

There are several equivalent ways of deriving the principal components mathematically. The simplest one is by finding the projections which maximize the vari-

ance. The first principal component is the direction in space along which projections have the largest variance. The second principal component is the direction which maximizes variance among all directions orthogonal to the first. The k^{th} component is the variance-maximizing direction orthogonal to the previous $k-1$ components. There are p principal components in all.

Rather than maximizing variance, it might sound more plausible to look for the projection with the smallest average (mean-squared) distance between the original vectors and their projections on to the principal components; this turns out to be equivalent to maximizing the variance.

Throughout, assume that the data have been "centered", so that every variable has mean 0. If we write the centered data in a matrix $\mathbf{x}$, where rows are objects and columns are variables, then $\mathbf{x}^T\mathbf{x} = n\mathbf{v}$, where $\mathbf{v}$ is the covariance matrix of the data. (You should check that last statement!)

16.1.1 Minimizing Projection Residuals

We'll start by looking for a one-dimensional projection. That is, we have p-dimensional vectors, and we want to project them on to a line through the origin. We can specify the line by a unit vector along it, $\vec{w}$, and then the projection of a data vector $\vec{x_i}$ on to the line is $\vec{x_i} \cdot \vec{w}$, which is a scalar. (Sanity check: this gives us the right answer when we project on to one of the coordinate axes.) This is the distance of the projection from the origin; the actual coordinate in p-dimensional space is $(\vec{x_i} \cdot \vec{w})\vec{w}$. The mean of the projections will be zero, because the mean of the vectors $\vec{x_i}$ is zero:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{x_i} \cdot \vec{w})\vec{w} = \left(\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \cdot \vec{w}\right)\vec{w}v \tag{16.1}$$

If we try to use our projected or **image** vectors instead of our original vectors, there will be some error, because (in general) the images do not coincide with the original vectors. (When do they coincide?) The difference is the error or **residual** of the projection. How big is it? For any one vector, say $\vec{x_i}$, it's

$$\begin{aligned}
||\vec{x_i} - (\vec{w} \cdot \vec{x_i})\vec{w}||^2 &= \left(\vec{x_i} - (\vec{w} \cdot \vec{x_i})\vec{w}\right) \cdot \left(\vec{x_i} - (\vec{w} \cdot \vec{x_i})\vec{w}\right) && (16.2)\\
&= \vec{x_i} \cdot \vec{x_i} - \vec{x_i} \cdot (\vec{w} \cdot \vec{x_i})\vec{w} && (16.3)\\
&\quad -(\vec{w} \cdot \vec{x_i})\vec{w} \cdot \vec{x_i} + (\vec{w} \cdot \vec{x_i})\vec{w} \cdot (\vec{w} \cdot \vec{x_i})\vec{w}\\
&= ||\vec{x_i}||^2 - 2(\vec{w} \cdot \vec{x_i})^2 + (\vec{w} \cdot \vec{x_i})^2 \vec{w} \cdot \vec{w} && (16.4)\\
&= \vec{x_i} \cdot \vec{x_i} - (\vec{w} \cdot \vec{x_i})^2 && (16.5)
\end{aligned}$$

since $\vec{w} \cdot \vec{w} = ||\vec{w}||^2 = 1$.

Add those residuals up across all the vectors:

$$\begin{aligned}
MSE(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n} ||\vec{x_i}||^2 - (\vec{w} \cdot \vec{x_i})^2 && (16.6)\\
&= \frac{1}{n}\left(\sum_{i=1}^{n} ||\vec{x_i}||^2 - \sum_{i=1}^{n} (\vec{w} \cdot \vec{x_i})^2\right) && (16.7)
\end{aligned}$$

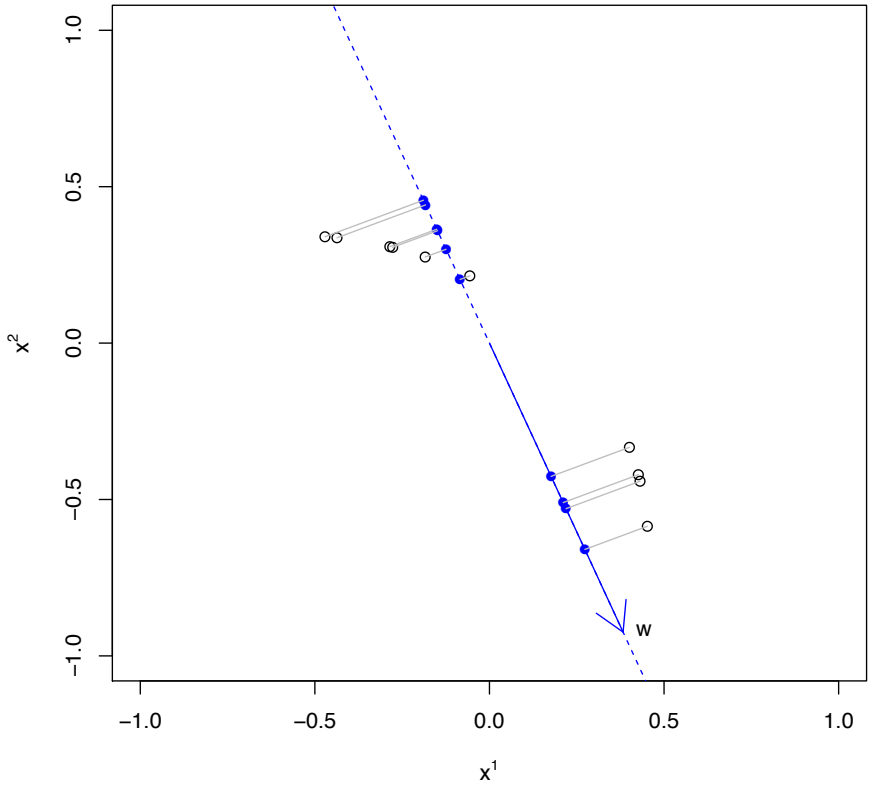


FIGURE 16.1: *Illustration of projecting data points $\vec{x}$ (black dots) on to an arbitrary line through the space (blue, dashed), represented by a unit vector $\vec{w}$ along the line (also blue but solid). The blue dots are the projections on to the blue line, $(\vec{x} \cdot \vec{w})\vec{w}$; the gray lines are the vector residuals, $\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}$. These are* not *the residuals from regressing one of the components of the data vector on the other.*

The first summation doesn't depend on $\vec{w}$, so it doesn't matter for trying to minimize the mean squared residual. To make the MSE small, what we must do is make the second sum big, i.e., we want to maximize

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2 \tag{16.8}$$

which we can see is the sample mean of $(\vec{w}\cdot\vec{x_i})^2$. The (sample) mean of a square is always equal to the square of the (sample) mean plus the (sample) variance:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2 = \left(\frac{1}{n}\sum_{i=1}^{n}\vec{x_i}\cdot\vec{w}\right)^2 + \widehat{\sigma^2}(\vec{w}\cdot\vec{x_i}) \tag{16.9}$$

Since we've just seen that the mean of the projections is zero, minimizing the residual sum of squares is equivalent to maximizing the variance of the projections.

(Of course in general we don't want to project on to just one vector, but on to multiple principal components. If those components are orthogonal and have the unit vectors $\vec{w}_1, \vec{w}_2, \ldots \vec{w}_k$, then the image of x_i is its projection into the space spanned by these vectors,

$$\sum_{j=1}^{k}(\vec{x_i}\cdot\vec{w_j})\vec{w_j} \tag{16.10}$$

The mean of the projection on to each component is still zero. If we go through the same algebra for the mean squared error, it turns [Exercise 1] out that the cross-terms between different components all cancel out, and we are left with trying to maximize the sum of the variances of the projections on to the components.)

16.1.2 Maximizing Variance

Accordingly, let's maximize the variance! Writing out all the summations grows tedious, so let's do our algebra in matrix form. If we stack our n data vectors into an $n \times p$ matrix, $\mathbf{x}$, then the projections are given by $\mathbf{xw}$, which is an $n \times 1$ matrix. The variance is

$$\begin{aligned}
\widehat{\sigma^2}(\vec{w}\cdot\vec{x_i}) &= \frac{1}{n}\sum_i (\vec{x_i}\cdot\vec{w})^2 && (16.11)\\
&= \frac{1}{n}(\mathbf{xw})^T(\mathbf{xw}) && (16.12)\\
&= \frac{1}{n}\mathbf{w}^T\mathbf{x}^T\mathbf{xw} && (16.13)\\
&= \mathbf{w}^T\frac{\mathbf{x}^T\mathbf{x}}{n}\mathbf{w} && (16.14)\\
&= \mathbf{w}^T\mathbf{vw} && (16.15)
\end{aligned}$$

We want to chose a unit vector $\vec{w}$ so as to maximize $\widehat{\sigma^2}(\vec{w} \cdot \vec{x_i})$. To do this, we need to make sure that we only look at unit vectors — we need to constrain the maximization. The constraint is that $\vec{w} \cdot \vec{w} = 1$, or $\mathbf{w}^T\mathbf{w} = 1$. To enforce this constraint, we introduce a Lagrange multiplier λ (Appendix H.5) and do a larger unconstrained optimization:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \lambda) &\equiv \mathbf{w}^T\mathbf{v}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1) && (16.16) \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= \mathbf{w}^T\mathbf{w} - 1 && (16.17) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= 2\mathbf{v}\mathbf{w} - 2\lambda\mathbf{w} && (16.18) \end{aligned}$$

Setting the derivatives to zero at the optimum, we get

$$\begin{aligned} \mathbf{w}^T\mathbf{w} &= 1 && (16.19) \\ \mathbf{v}\mathbf{w} &= \lambda\mathbf{w} && (16.20) \end{aligned}$$

Thus, the desired vector $\mathbf{w}$ is an eigenvector of the covariance matrix $\mathbf{v}$, and the maximizing vector will be the one associated with the largest eigenvalue λ. This is good news, because finding eigenvectors is something which can be done comparatively rapidly, and because eigenvectors have many nice mathematical properties, which we can use as follows.

We know that $\mathbf{v}$ is a $p \times p$ matrix, so it will have at most p different eigenvectors. We know that $\mathbf{v}$ is a covariance matrix, so it is symmetric, and then linear algebra tells us that the eigenvectors must be orthogonal to one another. Again because $\mathbf{v}$ is a covariance matrix, it is a **positive matrix**, in the sense that $\vec{x} \cdot \mathbf{v}\vec{x} \geq 0$ for any $\vec{x}$. This tells us that the eigenvalues of $\mathbf{v}$ must all be ≥ 0.

The eigenvectors of $\mathbf{v}$ are the **principal components** of the data. We know that they are all orthogonal to each other from the previous paragraph, so together they span the whole p-dimensional space. The first principal component, i.e. the eigenvector which goes the largest value of λ, is the direction along which the data have the most variance. The second principal component, i.e. the second eigenvector, is the direction orthogonal to the first component with the most variance. Because it is orthogonal to the first eigenvector, their projections will be uncorrelated. In fact, projections on to all the principal components are uncorrelated with each other. If we use q principal components, our weight matrix $\mathbf{w}$ will be a $p \times q$ matrix, where each column will be a different eigenvector of the covariance matrix $\mathbf{v}$. The eigenvalues will give the total variance described by each component. The variance of the projections on to the first q principal components is then $\sum_{i=1}^{q} \lambda_i$.

16.1.3 More Geometry; Back to the Residuals

Suppose that the data really are q-dimensional. Then $\mathbf{v}$ will have only q positive eigenvalues, and $p - q$ zero eigenvalues. If the data fall near a q-dimensional subspace, then $p - q$ of the eigenvalues will be nearly zero.

If we pick the top q components, we can define a projection operator $\mathbf{P}_q$. The images of the data are then $\mathbf{x}\mathbf{P}_q$. The **projection residuals** are $\mathbf{x} - \mathbf{x}\mathbf{P}_q$ or $\mathbf{x}(\mathbf{I} - \mathbf{P}_q)$. (Notice that the residuals here are vectors, not just magnitudes.) If the data really are q-dimensional, then the residuals will be zero. If the data are *approximately* q-dimensional, then the residuals will be small. In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 \equiv \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{j=1}^{p} \lambda_j} \tag{16.21}$$

just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.

The $q = 1$ case is especially instructive. We know that the residual vectors are all orthogonal to the projections. Suppose we ask for the first principal component of the residuals. This will be the direction of largest variance which is perpendicular to the first principal component. In other words, it will be the second principal component of the data. This suggests a recursive algorithm for finding all the principal components: the k^{th} principal component is the leading component of the residuals after subtracting off the first $k-1$ components. In practice, it is faster to get all the components at once from $\mathbf{v}$'s eigenvectors, but this idea is correct in principle.

This is a good place to remark that if the data really fall in a q-dimensional subspace, then $\mathbf{v}$ will have only q positive eigenvalues, because after subtracting off those components there will be no residuals. The other $p - q$ eigenvectors will all have eigenvalue 0. If the data cluster around a q-dimensional subspace, then $p - q$ of the eigenvalues will be very small, though how small they need to be before we can neglect them is a tricky question.[1]

Projections on to the first two or three principal components can be visualized; there is no guarantee, however, that only two or three dimensions really matter. Usually, to get an R^2 of 1, you need to use all p principal components.[2] How many principal components you should use depends on your data, and how big an R^2 you need. Sometimes, you can get better than 80% of the variance described with just two or three components. Sometimes, however, to keep a lot of the original variance you need to use almost as many components as you had dimensions to start with.

16.1.3.1 Scree Plots

People sometimes like to make plots of the eigenvalues, in decreasing order. Ideally, one starts with a few big eigenvalues, and then sees a clear drop-off to a remainder of

[1] Be careful when $n < p$. Any two points define a line, and three points define a plane, etc., so if there are fewer data points than variables, it is *necessarily* true that the fall on a low-dimensional subspace. In §16.4.1, we represent stories in the New York *Times* as vectors with $p \approx 440$, but $n = 102$. Finding that only 102 principal components keep all the variance is not an empirical discovery but a mathematical artifact.

[2] The exceptions are when some of your variables are linear combinations of the others, so that you don't really have p *different* variables, or when, as just mentioned, $n < p$.

small, comparatively negligible eigenvalues. These diagrams are called **scree plots**[3]. (Some people make similar plots, but show $1-R^2$ versus the number of components, rather than the individual eigenvalues.) Folklore recommends find the "base of the cliff" or "elbow" in the plot, the place where the number eigenvalues decrease dramatically and then level off to the right, and then retaining that number of components. This folklore appears to be based on nothing more than intuition, and offers no recommendation for what to do when there is no clear cliff or elbow in the scree plot.

16.1.4 Statistical Inference, or Not

You may have noticed, and even been troubled by, the fact that I have said nothing at all in this chapter like "assume the data are drawn at random from some distribution", or "assume the different rows of the data frame are statistically independent". This is because no such assumption is required for principal components. *All* it does is say "these data can be summarized using projections along these directions". It says nothing about the larger population or stochastic process the data came from; it doesn't even suppose there *is* a larger population or stochastic process. This is part of why §16.1.3 was so wishy-washy about the right number of components to use.

However, we could *add* a statistical assumption and see how PCA behaves under those conditions. The simplest one is to suppose that the data come IIDly from a distribution with covariance matrix $\mathbf{v}_0$. Then the sample covariance matrix $\mathbf{v} \equiv n^{-1}\mathbf{x}^T\mathbf{x}$ will converge on $\mathbf{v}_0$ as $n \to \infty$. Since the principal components are smooth functions of $\mathbf{v}$ (namely its eigenvectors), they will tend to converge as n grows[4]. So, along with that additional assumption about the data-generating process, PCA does make a prediction: in the future, the principal components will look like they do now.

We could always add stronger statistical assumptions; in fact, Chapter 17 will look at what happens when our assumptions essentially amount to "the data lie on a low-dimensional linear subspace, plus noise". Even this, however, turns out to make PCA a not-very-attractive estimate of the statistical structure.

16.2 Example 1: Cars

Enough math; let's work an example. The data[5] consists of 388 cars from the 2004 model year, with 18 features. Eight features are binary indicators; the other 11 fea-

[3]The small loose rocks one finds at the base of cliffs or mountains are called "scree"; the metaphor is that one starts with the big eigenvalues at the top of the hill, goes down some slope, and then finds the scree beneath it, which is supposed to be negligible noise. Those who have had to cross scree fields carrying heavy camping backpacks may disagree about whether it can really be ignored.

[4]There is a wrinkle if $\mathbf{v}_0$ has "degenerate" eigenvalues, i.e., two or more eigenvectors with the same eigenvalue. Then any linear combination of those vectors is also an eigenvector, with the same eigenvalue (Exercise 2.) For instance, if $\mathbf{v}_0$ is the identity matrix, then every vector is an eigenvector, and PCA routines will return an essentially arbitrary collection of mutually perpendicular vectors. Generically, however, any arbitrarily small tweak to $\mathbf{v}_0$ will break the degeneracy.

[5]On the course website; from `http://www.amstat.org/publications/jse/datasets/04cars.txt`, with incomplete records removed.

Variable	Meaning
Sports	Binary indicator for being a sports car
SUV	Indicator for sports utility vehicle
Wagon	Indicator
Minivan	Indicator
Pickup	Indicator
AWD	Indicator for all-wheel drive
RWD	Indicator for rear-wheel drive
Retail	Suggested retail price (US$)
Dealer	Price to dealer (US$)
Engine	Engine size (liters)
Cylinders	Number of engine cylinders
Horsepower	Engine horsepower
CityMPG	City gas mileage
HighwayMPG	Highway gas mileage
Weight	Weight (pounds)
Wheelbase	Wheelbase (inches)
Length	Length (inches)
Width	Width (inches)

TABLE 16.1: *Features for the 2004 cars data.*

```
Sports, SUV, Wagon, Minivan, Pickup, AWD, RWD, Retail,Dealer,Engine,
  Cylinders,Horsepower,CityMPG,HighwayMPG,Weight,Wheelbase,Length,Width
Acura 3.5 RL,0,0,0,0,0,0,0,43755,39014,3.5,6,225,18,24,3880,115,197,72
Acura MDX,0,1,0,0,0,1,0,36945,33337,3.5,6,265,17,23,4451,106,189,77
Acura NSX S,1,0,0,0,0,0,1,89765,79978,3.2,6,290,17,24,3153,100,174,71
```

TABLE 16.2: *The first few lines of the 2004 cars data set. [[TODO: replace with actual function evaluation?]]*

tures are numerical (Table 16.1). All of the features except `Type` are numerical. Table 16.2 shows the first few lines from the data set. PCA only works with numerical variables, so we have ten of them to play with.

[[ATTN: Data is now 10 years old; update?]]

There are *two* R functions for doing PCA, `princomp` and `prcomp`, which differ in how they do the actual calculation.[6] The latter is generally more robust, so we'll just use it.

```
cars04 = read.csv("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/cars-fixed04.dat")
cars04.pca = prcomp(cars04[, 8:18], scale. = TRUE)
```

The second argument to `prcomp` tells it to first scale all the variables to have variance 1, i.e., to standardize them. You should experiment with what happens with

[6] `princomp` actually calculates the covariance matrix and takes its eigenvalues. `prcomp` uses a different technique called "singular value decomposition".

this data when we don't standardize.

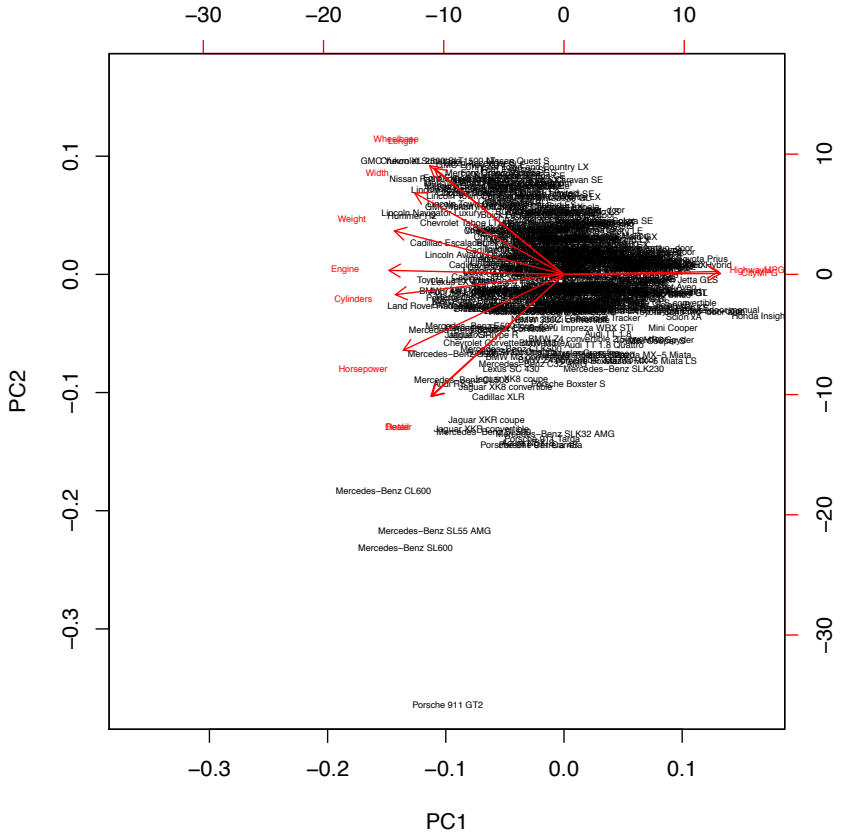
We can now extract the loadings or weight matrix from the `cars04.pca` object. For comprehensibility I'll just show the first two components.

```
round(cars04.pca$rotation[, 1:2], 2)
##              PC1   PC2
## Retail     -0.26 -0.47
## Dealer     -0.26 -0.47
## Engine     -0.35  0.02
## Cylinders  -0.33 -0.08
## Horsepower -0.32 -0.29
## CityMPG     0.31  0.00
## HighwayMPG  0.31  0.01
## Weight     -0.34  0.17
## Wheelbase  -0.27  0.42
## Length     -0.26  0.41
## Width      -0.30  0.31
```

This says that all the variables *except* the gas-mileages have a negative projection on to the first component. This means that there is a negative correlation between mileage and everything else. The first principal component tells us about whether we are getting a big, expensive gas-guzzling car with a powerful engine, or whether we are getting a small, cheap, fuel-efficient car with a wimpy engine.

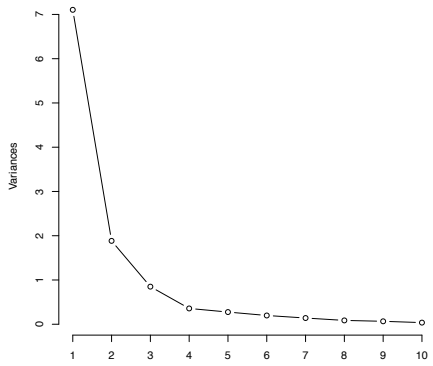
The second component is a little more interesting. Engine size and gas mileage hardly project on to it at all. Instead we have a contrast between the physical size of the car (positive projection) and the price and horsepower. Basically, this axis separates mini-vans, trucks and SUVs (big, not so expensive, not so much horsepower) from sports-cars (small, expensive, lots of horse-power).

To check this interpretation, we can use a useful tool called a **biplot**, which plots the data, along with the projections of the original variables, on to the first two components (Figure 16.2). Notice that the car with the lowest value of the second component is a Porsche 911, with pick-up trucks and mini-vans at the other end of the scale. Similarly, the highest values of the first component all belong to hybrids.




```
biplot(cars04.pca, cex = 0.4)
```

FIGURE 16.2: *"Biplot" of the 2004 cars data. The horizontal axis shows projections on to the first principal component, the vertical axis the second component. Car names are written at their projections on to the components (using the coordinate scales on the top and the right). Red arrows show the projections of the original variables on to the principal components (using the coordinate scales on the bottom and on the left).*




```
plot(cars04.pca, type = "l", main = "")
```

FIGURE 16.3: *Scree plot of the 2004 cars data: the eigenvalues of the principal components, in decreasing order. Each eigenvalue is the variance along that component. Folklore suggests adding components until the plot levels off, or goes past an "elbow" — here this might be 2 or 3 components.*

16.3 Example 2: The United States *circa* 1977

R contains a built-in data file, `state.x77`, with facts and figures for the various states of the USA as of about 1977: population, per-capita income, the adult illiteracy rate, life expectancy, the homicide rate, the proportion of adults with at least a high-school education, the number of days of frost a year, and the state's area. While this data set is almost as old as I am, it still makes a convenient example, so let's step through a principal components analysis of it.

Since the variables all have different, incomparable scales, it's not a bad idea to scale them to unit variance before finding the components[7]:

```
state.pca <- prcomp(state.x77, scale. = TRUE)
```

The biplot and the scree plot (Figure 16.4) look reasonable.

With this reasonable-looking PCA, we might try to interpret the components.

```
signif(state.pca$rotation[, 1:2], 2)
##               PC1    PC2
## Population  0.130  0.410
## Income     -0.300  0.520
## Illiteracy  0.470  0.053
## Life Exp   -0.410 -0.082
## Murder      0.440  0.310
## HS Grad    -0.420  0.300
## Frost      -0.360 -0.150
## Area       -0.033  0.590
```

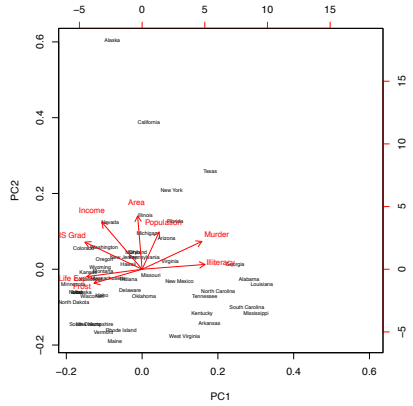
The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state. The second component is positively algined with area, income, population, high school graduation and murder, and negativey aligned, weakly, with cold weather and life expectancy. The first component thus separates short-lived, violent, ill-educated, poor warm states from those with the opposite qualities. The second component separates big, rich, educated, violent states from those which are small (in land or people), poor, less educated, and less violent.

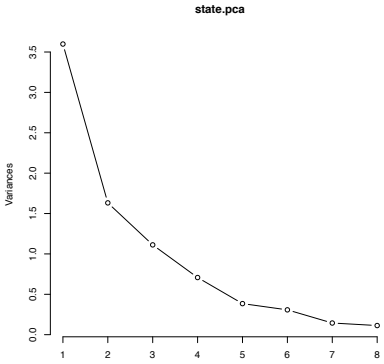
Since each data point has a geographic location, we can make a map, where the sizes of the symbols for each state vary with their projection on to the first principal component. This suggests that the component is something we might call "southernness" — more precisely, the contrast between the South and the rest of the nation[8]. I will leave making a map of the second component as an exercise[9].

[7] You should try re-running all this with `scale.=FALSE`, and ponder what the experience tells you about the wisdom of advice like "maximize R^2, or even "minimize the approximation error".

[8] The correlation between the first component and an indicator for being in the Confederacy is 0.8; for being a state which permitted slavery when the Civil War began, 0.78.

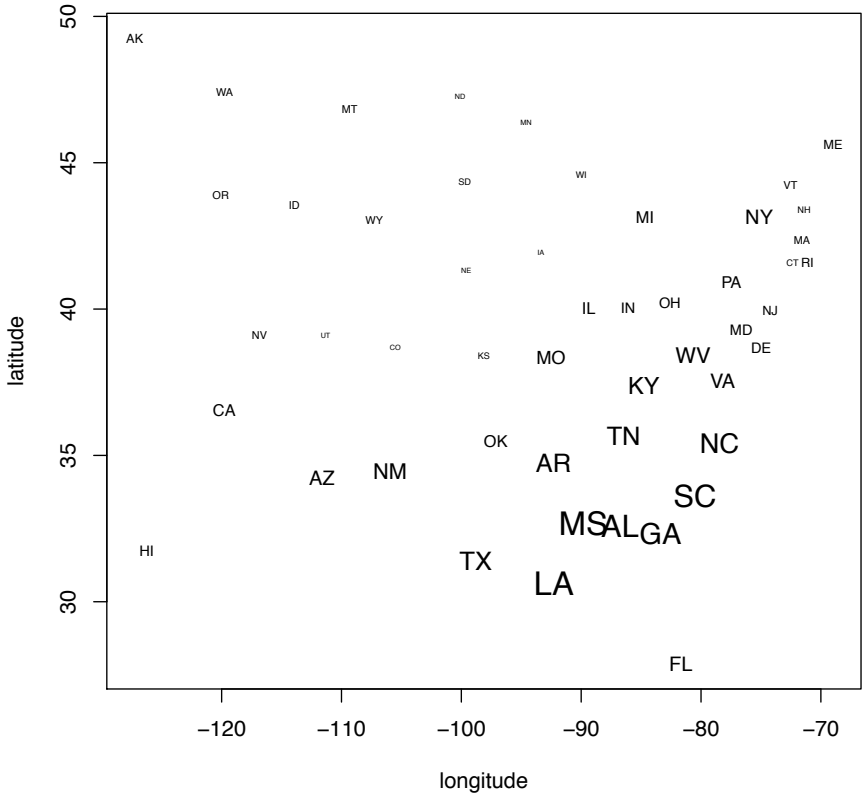
[9] §17.9.1 has more on this example.






```
biplot(state.pca, cex = c(0.5, 0.75))
plot(state.pca, type = "l")
```

FIGURE 16.4: *Biplot and scree plot for the PCA of* `state.x77`.




```
plot.states_scaled <- function(sizes, min.size = 0.4, max.size = 2, ...) {
    plot(state.center, type = "n", ...)
    out.range = max.size - min.size
    in.range = max(sizes) - min(sizes)
    scaled.sizes = out.range * ((sizes - min(sizes))/in.range)
    text(state.center, state.abb, cex = scaled.sizes + min.size)
    invisible(scaled.sizes)
}
plot.states_scaled(state.pca$x[, 1], min.size = 0.3, max.size = 1.5, xlab = "longitude",
    ylab = "latitude")
```

FIGURE 16.5: *The US states, plotted in their geographic locations, with symbol size varying with the projection of the state on to the first principal component. This suggests the component is something we might call "southernness".*

16.4 Latent Semantic Analysis

Information retrieval systems (like search engines) and people doing computational text analysis often represent documents as what are called **bags of words**: documents are represented as vectors, where each component counts how many times each word in the dictionary appears in the text. This throws away information about word order, but gives us something we can work with mathematically. Part of the representation of one document might look like:

```
 a abandoned abc ability able about above abroad absorbed absorbing abstract
43         0   0       0    0    10     0      0        0         0        1
```

and so on through to "zebra", "zoology", "zygote", etc. to the end of the dictionary. These vectors are very, very large! At least in English and similar languages, these bag-of-word vectors have three outstanding properties:

1. Most words do not appear in most documents; the bag-of-words vectors are very **sparse** (most entries are zero).

2. A small number of words appear many times in almost all documents; these words tell us almost nothing about what the document is about. (Examples: "the", "is", "of", "for", "at", "a", "and", "here", "was", etc.)

3. Apart from those hyper-common words, most words' counts are correlated with some but not all other words; words tend to come in bunches which appear together.

Taken together, this suggests that we do not really get a lot of value from keeping around *all* the words. We would be better off if we could project down a smaller number of new variables, which we can think of as combinations of words that tend to appear together in the documents, or not at all. But this tendency needn't be absolute — it can be partial because the words mean slightly different things, or because of stylistic differences, etc. This is *exactly* what principal components analysis does.

To see how this can be useful, imagine we have a collection of documents (a **corpus**), which we want to search for documents about agriculture. It's entirely possible that many documents on this topic don't actually contain the *word* "agriculture", just closely related words like "farming". A simple search on "agriculture" will miss them. But it's very likely that the occurrence of these related words is well-correlated with the occurrence of "agriculture". This means that all these words will have similar projections on to the principal components, and will be easy to find documents whose principal components projection is like that for a query about agriculture. This is called **latent semantic indexing**.

To see why this is *indexing*, think about what goes into coming up with an index for a book by hand. Someone draws up a list of topics and then goes through the book noting all the passages which refer to the topic, and maybe a little bit of what they say there. For example, here's the start of the entry for "Agriculture" in the index to Adam Smith's *The Wealth of Nations*:

> AGRICULTURE, the labour of, does not admit of such subdivisions as manufactures, 6; this impossibility of separation, prevents agriculture from improving equally with manufactures, 6; natural state of, in a new colony, 92; requires more knowledge and experience than most mechanical professions, and yet is carried on without any restrictions, 127; the terms of rent, how adjusted between landlord and tenant, 144; is extended by good roads and navigable canals, 147; under what circumstances pasture land is more valuable than arable, 149; gardening not a very gainful employment, 152–3; vines the most profitable article of culture, 154; estimates of profit from projects, very fallacious, *ib.*; cattle and tillage mutually improve each other, 220; ...

and so on. (Agriculture is an important topic in *The Wealth of Nations*.) It's asking a lot to hope for a computer to be able to do something like this, but we could at least hope for a list of pages like "$6, 92, 126, 144, 147, 152 - -3, 154, 220, \ldots$". One could imagine doing this by treating each page as its own document, forming its bag-of-words vector, and then returning the list of pages with a non-zero entry for "agriculture". This will fail: only two of those nine pages actually contains that word, and this is pretty typical. On the other hand, they are full of words strongly correlated with "agriculture", so asking for the pages which are most similar in their principal components projection to that word will work great.[10]

At first glance, and maybe even second, this seems like a wonderful trick for extracting meaning, or **semantics**, from pure correlations. Of course there are also all sorts of ways it can fail, not least from spurious correlations. If our training corpus happens to contain lots of documents which mention "farming" and "Kansas", as well as "farming" and "agriculture", latent semantic indexing will not make a big distinction between the relationship between "agriculture" and "farming" (which is genuinely semantic, about the meaning of the words) and that between "Kansas" and "farming" (which reflects non-linguistic facts about the world, and probably wouldn't show up in, say, a corpus collected from Australia).

Despite this susceptibility to spurious correlations, latent semantic indexing is an *extremely* useful technique in practice, and the foundational papers (Deerwester *et al.*, 1990; Landauer and Dumais, 1997) are worth reading.

16.4.1 Principal Components of the New York *Times*

To get a more concrete sense of how latent semantic analysis works, and how it reveals semantic information, let's apply it to some data. The accompanying R file and R workspace contains some news stories taken from the New York *Times* Annotated Corpus (Sandhaus, 2008), which consists of about 1.8 million stories from the *Times*, from 1987 to 2007, which have been hand-annotated by actual human beings with standardized machine-readable information about their contents. From this corpus, I have randomly selected 57 stories about art and 45 stories about music, and turned them into a bag-of-words data frame, one row per story, one column per word; plus an indicator in the first column of whether the story is one about art or one about

[10] Or it should anyway; I haven't actually done the experiment with this book.

music.[11] The original data frame thus has 102 rows, and 4432 columns: the categorical label, and 4431 columns with counts for every distinct word that appears in at least one of the stories.[12]

The PCA is done as it would be for any other data:

[[ATTN: Why isn't loading from the course website working?]]

```
load("~/teaching/ADAfaEPoV/data/pca-examples.Rdata")
nyt.pca <- prcomp(nyt.frame[, -1])
nyt.latent.sem <- nyt.pca$rotation
```

We need to omit the first column in the first command because it contains categorical variables, and PCA doesn't apply to them. The second command just picks out the matrix of projections of the variables on to the components — this is called `rotation` because it can be thought of as rotating the coordinate axes in feature-vector space.

Now that we've done this, let's look at what the leading components are.

```
signif(sort(nyt.latent.sem[, 1], decreasing = TRUE)[1:30], 2)
##       music        trio     theater   orchestra   composers       opera
##       0.110       0.084       0.083       0.067       0.059       0.058
##    theaters           m    festival        east     program           y
##       0.055       0.054       0.051       0.049       0.048       0.048
##      jersey     players   committee      sunday        june     concert
##       0.047       0.047       0.046       0.045       0.045       0.045
##    symphony       organ     matinee   misstated instruments           p
##       0.044       0.044       0.043       0.042       0.041       0.041
##         X.d       april      samuel        jazz     pianist     society
##       0.041       0.040       0.040       0.039       0.038       0.038
signif(sort(nyt.latent.sem[, 1], decreasing = FALSE)[1:30], 2)
##       she       her        ms         i      said    mother    cooper
##    -0.260    -0.240    -0.200    -0.150    -0.130    -0.110    -0.100
##        my  painting   process paintings        im        he       mrs
##    -0.094    -0.088    -0.071    -0.070    -0.068    -0.065    -0.065
##        me  gagosian       was   picasso     image sculpture      baby
##    -0.063    -0.062    -0.058    -0.057    -0.056    -0.056    -0.055
##   artists      work    photos       you    nature    studio       out
##    -0.055    -0.054    -0.051    -0.051    -0.050    -0.050    -0.050
##      says      like
##    -0.050    -0.049
```

These are the thirty words with the largest positive and negative projections on to the first component.[13] The words with positive projections are mostly associated

[11] Actually, following standard practice in language processing, I've normalized the bag-of-word vectors so that documents of different lengths are comparable, and used "inverse document-frequency weighting" to de-emphasize hyper-common words like "the" and emphasize more informative words. See the lecture notes for data mining if you're interested.

[12] If we were trying to work with the complete corpus, we should expect at least 50000 words, and perhaps more.

[13] Which direction is positive and which negative is of course arbitrary; basically it depends on internal choices in the algorithm.

with music, those with negative components with the visual arts. The letters "m" and "p" show up with music because of the combination "p.m", which our parsing breaks into two single-letter words, and because stories about music give show-times more often than do stories about art. Personal pronouns appear with art stories because more of those quote people, such as artists or collectors.[14]

What about the second component?

```
signif(sort(nyt.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
##         art      museum      images     artists   donations     museums
##       0.150       0.120       0.095       0.092       0.075       0.073
##    painting         tax   paintings   sculpture     gallery  sculptures
##       0.073       0.070       0.065       0.060       0.055       0.051
##     painted       white    patterns      artist      nature     service
##       0.050       0.050       0.047       0.047       0.046       0.046
##  decorative        feet     digital      statue       color    computer
##       0.043       0.043       0.043       0.042       0.042       0.041
##       paris         war collections     diamond       stone     dealers
##       0.041       0.041       0.041       0.041       0.041       0.040
signif(sort(nyt.latent.sem[, 2], decreasing = FALSE)[1:30], 2)
##          her          she      theater        opera           ms
##       -0.220       -0.220       -0.160       -0.130       -0.130
##            i         hour   production         sang     festival
##       -0.083       -0.081       -0.075       -0.075       -0.074
##        music      musical        songs        vocal    orchestra
##       -0.070       -0.070       -0.068       -0.067       -0.067
##           la      singing      matinee  performance         band
##       -0.065       -0.065       -0.061       -0.061       -0.060
##       awards    composers         says           my           im
##       -0.058       -0.058       -0.058       -0.056       -0.056
##         play     broadway       singer       cooper performances
##       -0.056       -0.055       -0.052       -0.051       -0.051
```

Here the positive words are about art, but more focused on acquiring and trading ("collections", "dealers", "donations", "dealers") than on talking with artists or about them. The negative words are musical, specifically about musical theater and vocal performances.

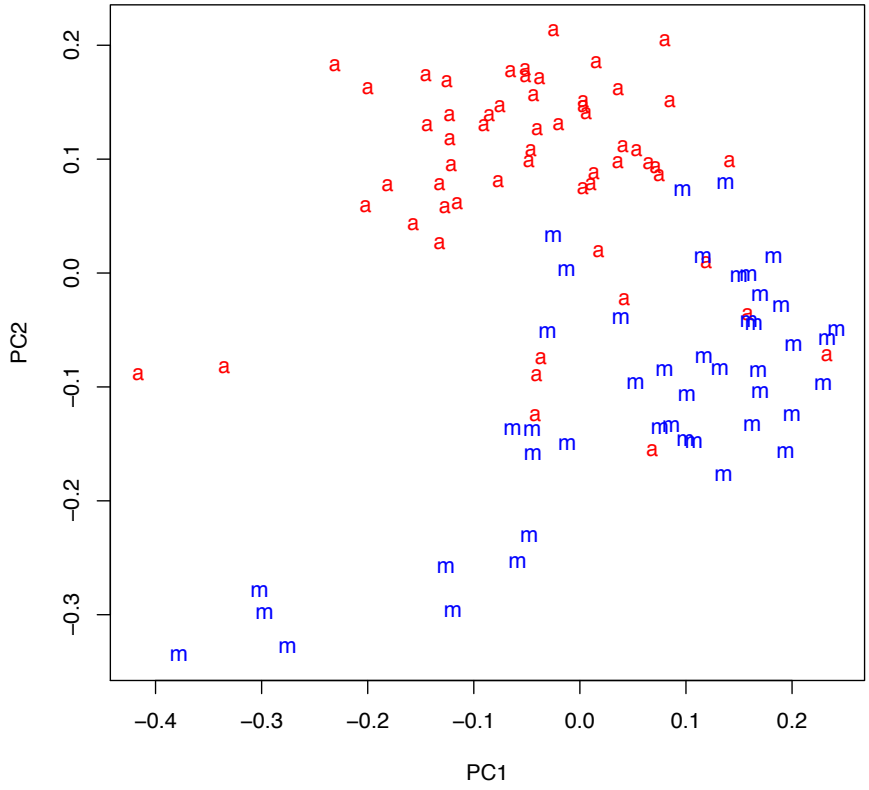
I could go on, but by this point you get the idea.

16.5 PCA for Visualization

Let's try displaying the *Times* stories using the principal components (Figure 16.6).

Notice that even though we have gone from 4431 dimensions to 2, and so thrown away a lot of information, we could draw a line across this plot and have most of the art stories on one side of it and all the music stories on the other. If we let ourselves use the first four or five principal components, we'd still have a thousand-fold savings in dimensions, but we'd be able to get almost-perfect separation between

[14] You should check out these explanations for yourself. The raw stories are part of the R workspace.




```
plot(nyt.pca$x[, 1:2], pch = ifelse(nyt.frame[, "class.labels"] == "music",
    "m", "a"), col = ifelse(nyt.frame[, "class.labels"] == "music", "blue",
    "red"))
```

FIGURE 16.6: *Projection of the* Times *stories on to the first two principal components. Music stories are marked with a blue "m", art stories with a red "a".*

the two classes. This is a sign that PCA is really doing a good job at summarizing the information in the word-count vectors, and in turn that the bags of words give us a lot of information about the meaning of the stories.

The figure also illustrates the idea of **multidimensional scaling**, which means finding low-dimensional points to represent high-dimensional data by preserving the distances between the points. If we write the original vectors as $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n$, and their images as $\vec{y}_1, \vec{y}_2, \ldots \vec{y}_n$, then the MDS problem is to pick the images to minimize the difference in distances:

$$\sum_i \sum_{j \neq i} \left(||\vec{y}_i - \vec{y}_j|| - ||\vec{x}_i - \vec{x}_j|| \right)^2 \tag{16.22}$$

This will be small if distances between the image points are all close to the distances between the original points. PCA accomplishes this precisely because $\vec{y}_i$ is itself close to $\vec{x}_i$ (on average).

16.6 PCA Cautions

Trying to guess at what the components might mean is a good idea, but like many good ideas it's easy to go overboard. Specifically, once you attach an idea in your mind to a component, and especially once you attach a *name* to it, it's very easy to forget that those are names and ideas you made up; to **reify** them, as you might reify clusters. Sometimes the components actually do measure real variables, but sometimes they just reflect patterns of covariance which have many different causes. If I did a PCA of the same variables but for, say, European cars, I might well get a similar first component, but the second component would probably be rather different, since SUVs are much less common there than here.

A more important example comes from population genetics. Starting in the late 1960s, L. L. Cavalli-Sforza and collaborators began a huge project of mapping human genetic variation — of determining the frequencies of different genes in different populations throughout the world. (Cavalli-Sforza *et al.* (1994) is the main summary; Cavalli-Sforza has also written several excellent popularizations.) For each point in space, there are a very large number of variables, which are the frequencies of the various genes among the people living there. Plotted over space, this gives a map of that gene's frequency. What they noticed (unsurprisingly) is that many genes had similar, but not identical, maps. This led them to use PCA, reducing the huge number of variables (genes) to a few components. Results look like Figure 16.7. They interpreted these components, very reasonably, as signs of large population movements. The first principal component for Europe and the Near East, for example, was supposed to show the expansion of agriculture out of the Fertile Crescent. The third, centered in steppes just north of the Caucasus, was supposed to reflect the expansion of Indo-European speakers towards the end of the Bronze Age. Similar stories were told of other components elsewhere.

Unfortunately, as Novembre and Stephens (2008) showed, spatial patterns like this are what one should expect to get when doing PCA of any kind of spatial data with local correlations, because that essentially amounts to taking a Fourier transform, and picking out the low-frequency components.[15] They simulated genetic diffusion processes, without any migration or population expansion, and got results that looked very like the real maps (Figure 16.8). This doesn't mean that the stories of the maps *must be* wrong, but it does undercut the principal components as evidence for those stories.

16.7 Further Reading

Principal components goes back at least to Hotelling in the 1930s, if not to Karl Pearson around 1900. It has been rediscovered many times in many fields, so it is also known as the Karhunen-Loève transformation, the Hotelling transformation, [[ATTN: Get exact citations.]]

[15] Remember that PCA re-writes the original vectors as a weighted sum of new, orthogonal vectors, just as Fourier transforms do. When there is a lot of spatial correlation, values at nearby points are similar, so the low-frequency modes will have a lot of amplitude, i.e., carry a lot of the variance. So first principal components will tend to be similar to the low-frequency Fourier modes.

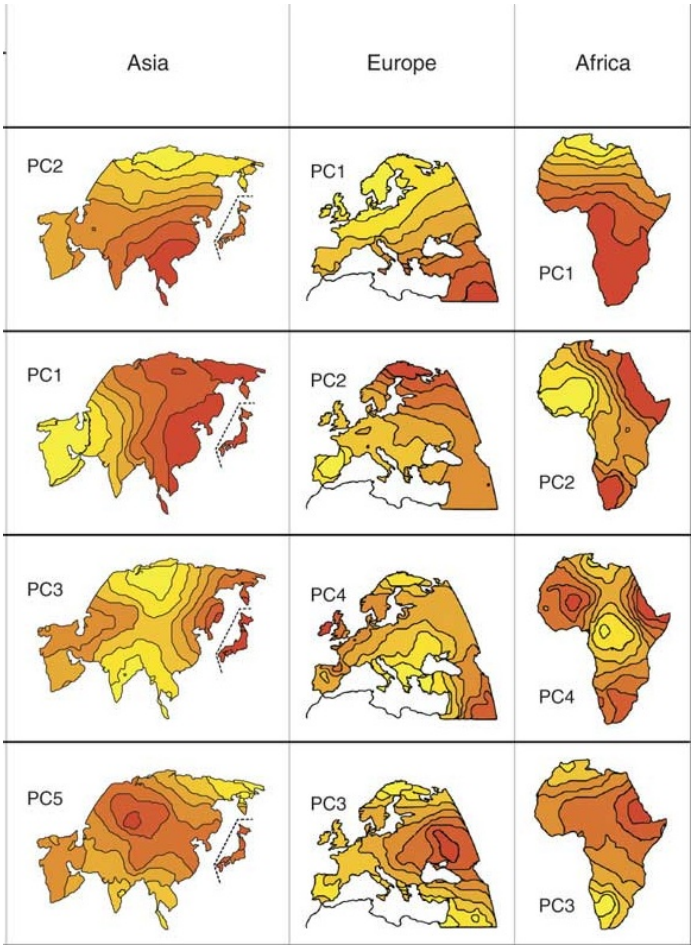


FIGURE 16.7: *Principal components of genetic variation in the old world, according to Cavalli-Sforza* et al. *(1994), as re-drawn by Novembre and Stephens (2008).*

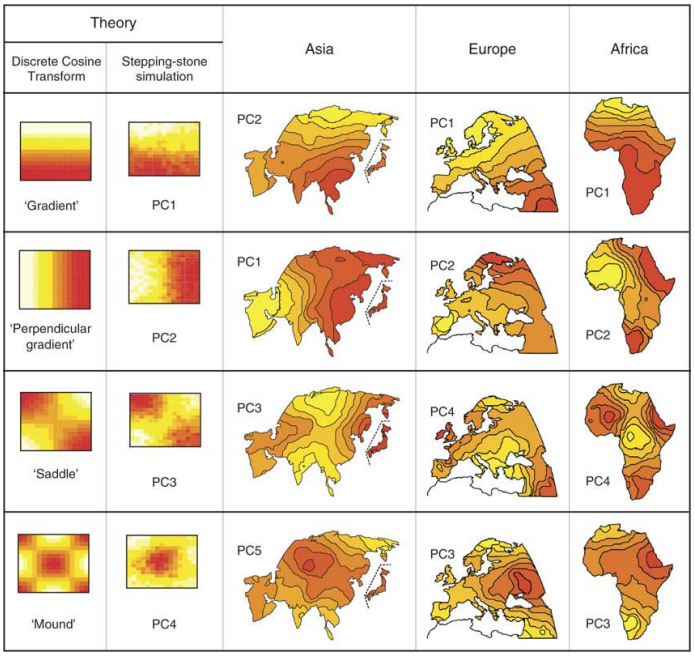


FIGURE 16.8: *How the PCA patterns can arise as numerical artifacts (far left column) or through simple genetic diffusion (next column). From Novembre and Stephens (2008).*

the method of empirical orthogonal functions, and singular value decomposition[16]. Many statistical presentations start with the idea of maximizing the variance kept; this seems less well-motivated to me than trying to find the best-approximating linear subspace.

As I said above, PCA is an example of a data analysis method which involves only approximation, with no statistical inference or underlying probabilistic model. Chapters 17 and 19 describe two (rather-different looking) statistical models which both imply that the data should lie in a linear subspace plus noise. Alternatively, chapter 18 introduces methods for approximating data with low-dimensional curved manifolds, rather than linear subspaces.

Latent semantic analysis, or latent semantic indexing, goes back to Deerwester *et al.* (1990). Hand *et al.* (2001) has a good discussion, setting it in the context of other data-analytic methods, and avoiding some of the more extravagant claims made on its behalf (Landauer and Dumais, 1997).

§16.5 just scratches the surface of the vast literature on multidimensional scaling, the general goal of which is to find low-dimensional, easily-visualized representations which are somehow faithful to the geometry of high-dimensional spaces. Much of this literature, including the name "multidimensional scaling", comes from psychology. For a brief introduction with references, I recommend Hand *et al.* (2001).

[[TODO: Expand on references to MDS.]]

Concerns about interpretation and reification[17] are rarely very far away whenever people start using methods for finding hidden structure, whether they're just approximation methods or they attempt proper statistical inference. We will touch on them again in chapters 17 and 19. In general, statisticians seem to find it easier to say what's wrong or dubious about other analysts' interpretations of their components or latent constructs, than to explain what's right about their own interpretations; certainly I do.

16.8 Exercises

1. Suppose that instead of projecting on to a line, we project on to a q-dimensional subspace, defined by q orthogonal length-one vectors $\vec{w}_1, \ldots \vec{w}_q$. We want to show that minimizing the mean squared error of the projection is equivalent to maximizing the sum of the variances of the scores along these q directions.
   (a) Write $\mathbf{w}$ for the matrix forms by stacking the $\vec{w}_i$. Prove that $\mathbf{w}^T\mathbf{w} = \mathbf{I}_q$.
   (b) Find the matrix of q-dimensional scores in terms of $\mathbf{x}$ and $\mathbf{w}$. *Hint:* your answer should reduce to $\vec{x}_i \cdot \vec{w}_1$ when $q = 1$.
   (c) Find the matrix of p-dimensional approximations based on these scores in terms of $\mathbf{x}$ and $\mathbf{w}$. *Hint:* your answer should reduce to $(\vec{x}_i \cdot \vec{w}_1)\vec{w}_1$ when $q = 1$.

[16] Strictly speaking, singular value decomposition is a matrix algebra trick which is used in the most common algorithm for PCA.

[17] I have not been able to find out where the term "reification" comes from; some claim that it is Marxist in origin, but it's used by the early and decidedly non-Marxist Thomson (1939), so I doubt that.

(d) Show that the MSE of using the vectors $\vec{w}_1, \ldots \vec{w}_q$ is the sum of two terms, one of which depends only on $\mathbf{x}$ and not $\mathbf{w}$, and the other depends only on the scores along those directions (and not otherwise on what those directions are). *Hint:* look at the derivation of Eq. 16.5, and use Exercise 1a.

(e) Explain in what sense minimizing projection residuals is equivalent to maximizing the sum of variances along the different directions.

2. Suppose that $\mathbf{u}$ has two eigenvectors, $\vec{w}_1$ and $\vec{w}_2$, with the same eigenvalue a. Prove that any linear combination of $\vec{w}_1$ and $\vec{w}_2$ is also an eigenvector of $\mathbf{u}$, and also has eigenvalue a.