Part 8: Dimension Reduction

Summary

Often data come to us in a high-dimensional form, meaning that each observation is actually best thought of as a vector of many numbers.

In some situations, it is crucial that lower-dimensional representations of these data are obtained. Here we present some approaches to doing that.

First, we motivated why this dimension reduction can be so important.

The Curse of Dimensionality

Despite the promise of nonparametric approaches, they suffer from a serious drawback: The amount of data required to fit nonparametric density estimates or regressions well increases exponentially with the dimension of the data.

This is called the Curse of Dimensionality.

To start, we ask: **What is the "dimension" of data?**

Standard data come to us in the form of numbers, but a single observation is often best thought of as a list of numbers.

For example, a yield curve shows how bond rates vary as a function of time to maturity. A single yield curve consists of several numbers, usually around ten rates. This count of how many numbers it takes to represent a yield curve is called its dimension. We would say that a single yield curve is "ten-dimensional."

Another example: Imagine our data consists of separate price time series for each of 100 stocks. Each series goes back 20 trading days. We would say in this case that we have "100 observations of 20-dimensional time series".

One place where the dimension of data often comes up is in the context of regression or supervised learning. If there are p predictors being used in a model for a response variable, we would say that we are using a "p-dimensional predictor vector."

Nonparametric regression is possible when the predictor vector is p-dimensional. For instance, if $p = 2$, a neighborhood can be thought of as a square centered on the target value. Local linear regression involves fitting a plane within this square.

Reconsider our options sample from the previous Part. The figure on the following page shows a two-dimensional local linear regression fit. The response is the "error" in the Black-Scholes price (relative to the ask price) and the predictors are the time to expiration and the strike price.

First, read in the data and fit the model:

```
> optionsdata = read.table("optionssample09302017.txt",
+                                  header=T, sep=",")
> optionsdata$bserr = optionsdata$bsval - optionsdata$ask
>
> holdlo2d = loess(bserr ~ timetoexpiry + strike,
+                         data=optionsdata, degree=1)
```

The function `expand.grid()` creates a two-dimensional grid of values from two vectors.

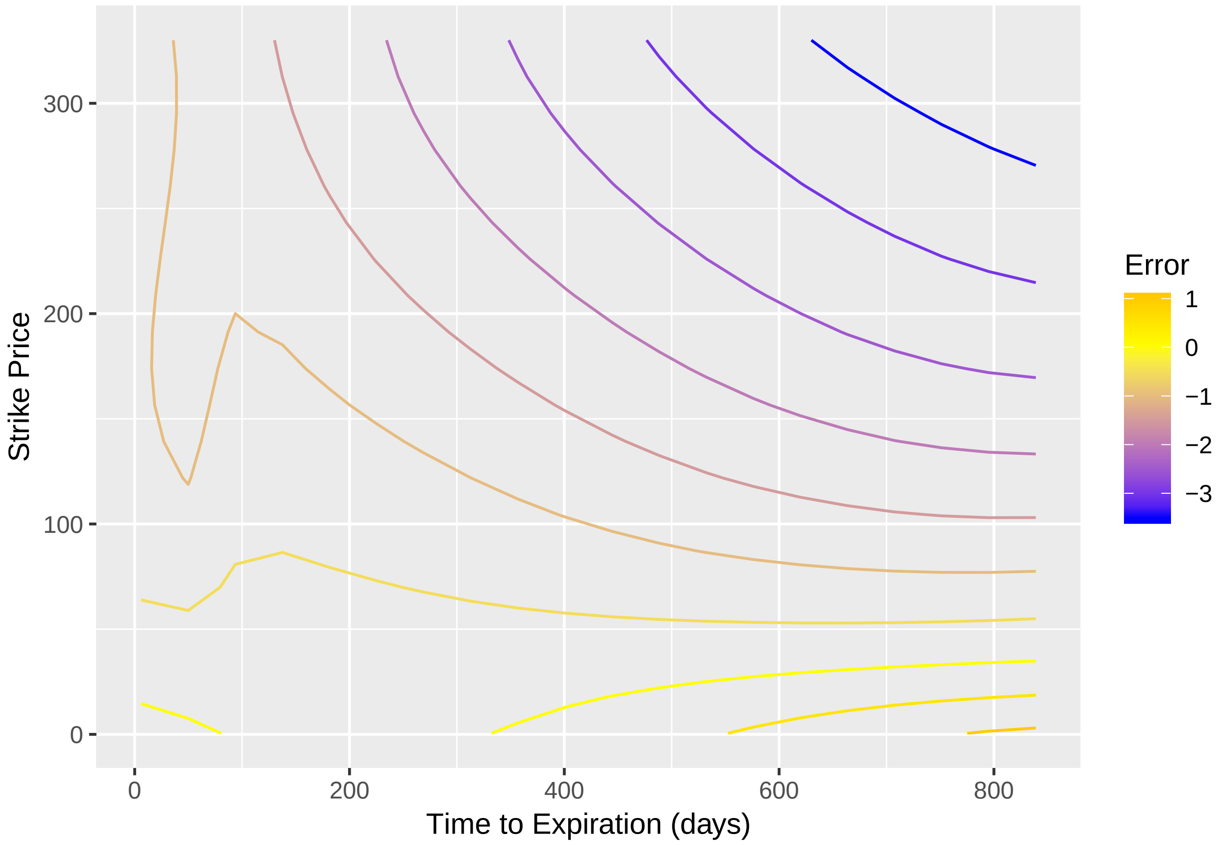
```
> fullgrid = expand.grid(
+   timetoexpiry=
+     seq(min(optionsdata$timetoexpiry),
+         max(optionsdata$timetoexpiry),length=20),
+   strike=seq(min(optionsdata$strike),
+              max(optionsdata$strike),length=20))
```

Now, make the predictions on this grid:

```
> fits = predict(holdlo2d,newdata=as.matrix(fullgrid))
```

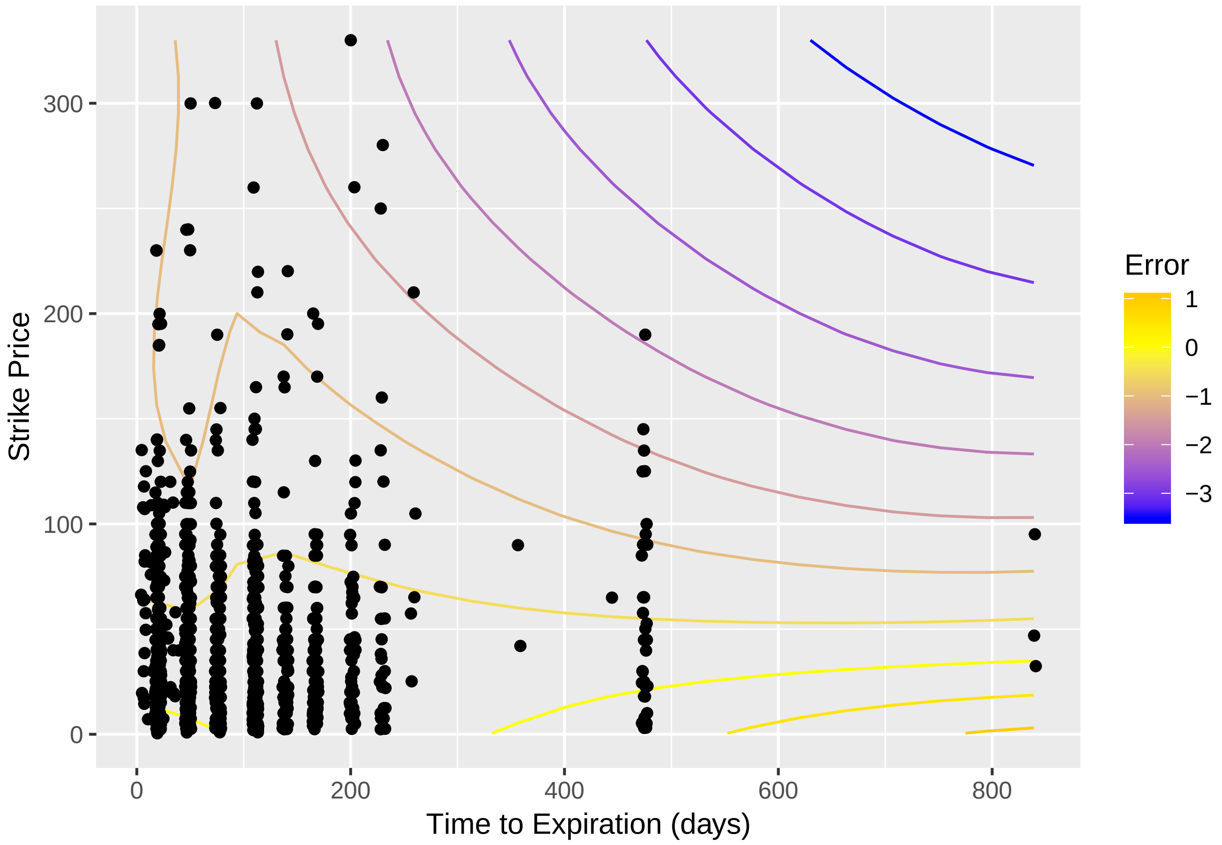
Finally, create the plot. Note the use of the syntax `color=..level..` in order to specify that the colors should reflect the different levels in the contour plot.

```
> fitframe = data.frame(cbind(fullgrid,fits))
>
> ggplot(fitframe, aes(x=timetoexpiry,y=strike,z=fits)) +
+   geom_contour(aes(color=..level..)) +
+   scale_color_gradient2(low="blue", high="red",
+                         mid="yellow",midpoint=0) +
+   labs(x="Time to Expiration (days)", y="Strike Price",
+        color="Error")
```


Strike Price
300
200
100
0
0
200
400
600
800
Time to Expiration (days)
Error
1
0
−1
−2
−3

1

Exercise: The next plot adds on the 1,000 observed predictor pairs used in this case. Sketch the neighborhood when the regression function is estimated at $(400, 200)$. What will have to happen in order for this estimate to have sufficiently low variance?


Strike Price
300
200
100
0
0
200
400
600
800
Time to Expiration (days)
Error
1
0
−1
−2
−3

1

This example shows the fundamental challenge of working with high-dimensional predictors: As the dimension increases, the available observations become more "spread out." Neighborhoods must be made larger in order to compensate and achieve estimates with acceptable variance.

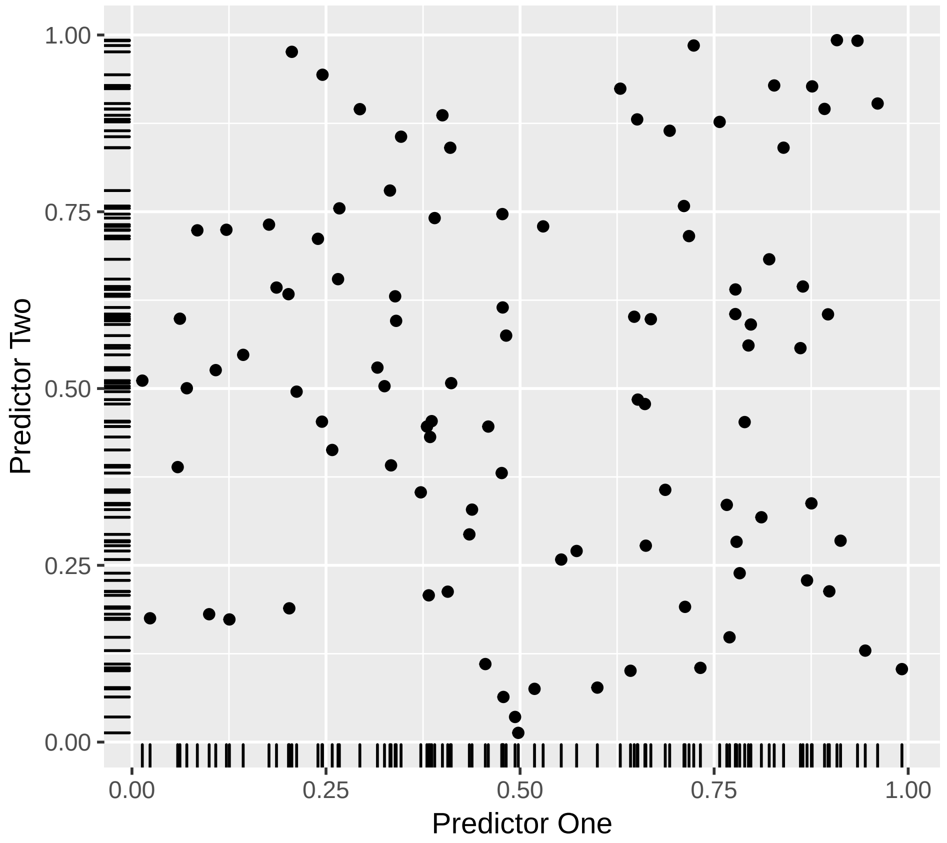
But, choosing a large neighborhood takes away a key feature of using non-parametric approaches: We want the estimate to be flexible, and not be based on too much "smoothing."

Consider the plot on the following slide.

Here, the sample size is 100. Each observation is two-dimensional, i.e., imagine these are the two predictors. These data are simulated from uniform distributions.

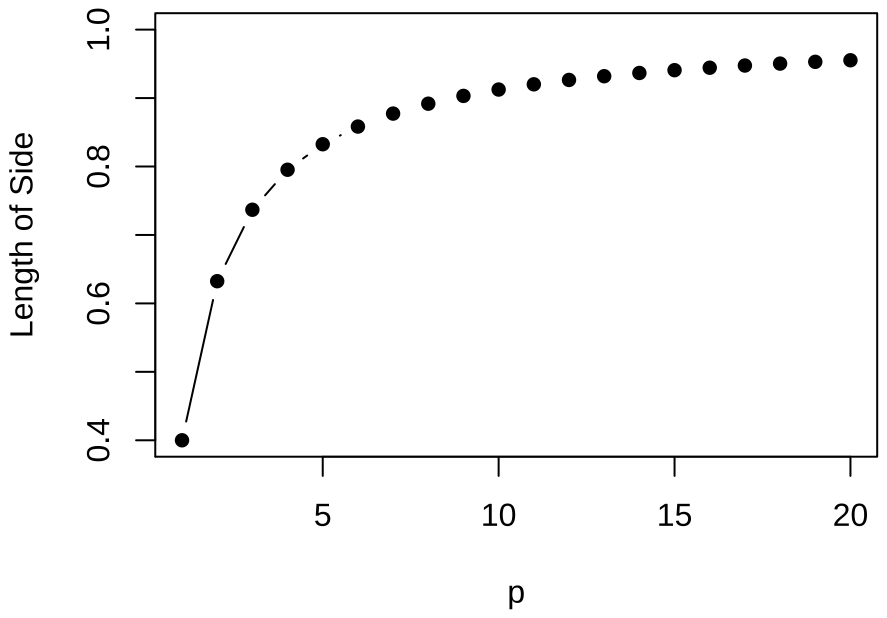
The "marks" shown in each margin depict the marginal distribution for each predictor. Note that in the margins, the data appear to be quite closely spaced. If we were to fit a nonparametric regression using just one predictor, neighborhoods could be chosen small.

But, in two dimensions, large gaps appear in the distribution of the observations. Neighborhoods will have to be chosen larger in order to capture enough data.


1.00
0.75
0.50
0.25
0.00
Predictor Two
0.00
0.25
0.50
0.75
1.00
Predictor One

1

Another way of thinking about it: Under the uniform setup, with a p-dimensional predictor, in order for a neighborhood to include proportion α of the data, the length of the sides of the neighborhood must be $\alpha^{1/p}$. This plot shows the case where $\alpha = 0.4$.



Exercise: Suppose that $p = 5$. Under our uniform setup, how long are the sides of a neighborhood that covers 40% of the data? What are the implications of this?

Exercise: Again assume our uniform predictor distribution. Define that an observation is "close to the boundary" if it is either less than 0.1 or greater than 0.9 for at least one predictor. What proportion of the observations are "close to the boundary?"

The curse of dimensionality motivates us to consider methods which create low-dimensional representations of data.

In many cases, data which are p-dimensional in their original form can be compressed to q dimensions, where $q \ll p$, with minimal loss of important information.

The aforementioned yield curves provide an important example. The daily yield curve consists of approximately 10 rates, but a yield curve takes a limited class of shapes. Shifts in the yield curve can be described using three numbers in most situations. Hence, a 10-dimensional object can be described compressed to three dimensions.

Web Scraping Example: Obtaining Treasury Yield Curves

Web scraping refers to automatically pulling data off of web sites.

Here we will illustrate an example of doing this for a data set of interest using the R package `rvest`.

Every implementation of web scraping is going to present its own challenges, but the general concepts presented here should be useful.

The data we will use in this example consist of daily treasury yield curves, found at the following URL:

https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yieldAll

which I have conveniently shortened to https://goo.gl/j97141

Data on this page go back to the start of 1990.

A yield curve is simply the relationship between the yield for the bonds of a single type and the time to maturity.

To get started, install the package `rvest` and load it up. Then use the function `read_html()` to read in the desired web page:

```
> fullYCweb = read_html("https://goo.gl/j97141")
```

The function `html_nodes()` allows ones to easily extract elements of the web page by specifying CSS selectors, which are identifiers that one can use to define formatting for elements of similar types.

See https://www.w3schools.com/cssref/trysel.asp for illustrative examples.

Exercise: Take a moment and study the source of the target web page. (All web browsers have an option that allows you to view the source.) Find where in the source the desired data resides.

A typical piece of the data table appears as follows:

```
<td class="text_view_data">7.83</td><td class="text_view_data">7.89</td>
```

In this example, `text_view_data` is a CSS selector. It is a "class," meaning that it is a selector shared across different elements. These are defined using an extra period, e.g., `.class_name` and this is how they should be referenced within `html_nodes()`.

So, the following command will extract all of the nodes which include that CSS selector:

```
> tvdnodes = html_nodes(fullYCweb, ".text_view_data")
```

You will note that the elements of `tvdnodes` contain extra, undesired information. Fortunately, `rvest` has another function, `html_text()` which strips away the unwanted wrappers.

```
> tableelements = html_text(tvdnodes)
> print(tableelements[1:10])
 [1] "01/02/90"                "\n\t\t\tN/A\n\t\t" "7.83"
 [4] "7.89"                    "7.81"                    "7.87"
 [7] "7.90"                    "7.87"                    "7.98"
[10] "7.94"
```

One clear problem is that the `N/A` values in the original table are now appearing with extra formatting characters.

The `grep()` function allows for a search over regular expressions. This is a large topic, but, briefly stated, regular expressions allow for searching for strings that meet flexible requirements.

Here we simply use the following command to return all entries in `tableelements` that contain `N/A`:

```
> grep("N/A", tableelements)
```

So, the following command will clean things up:

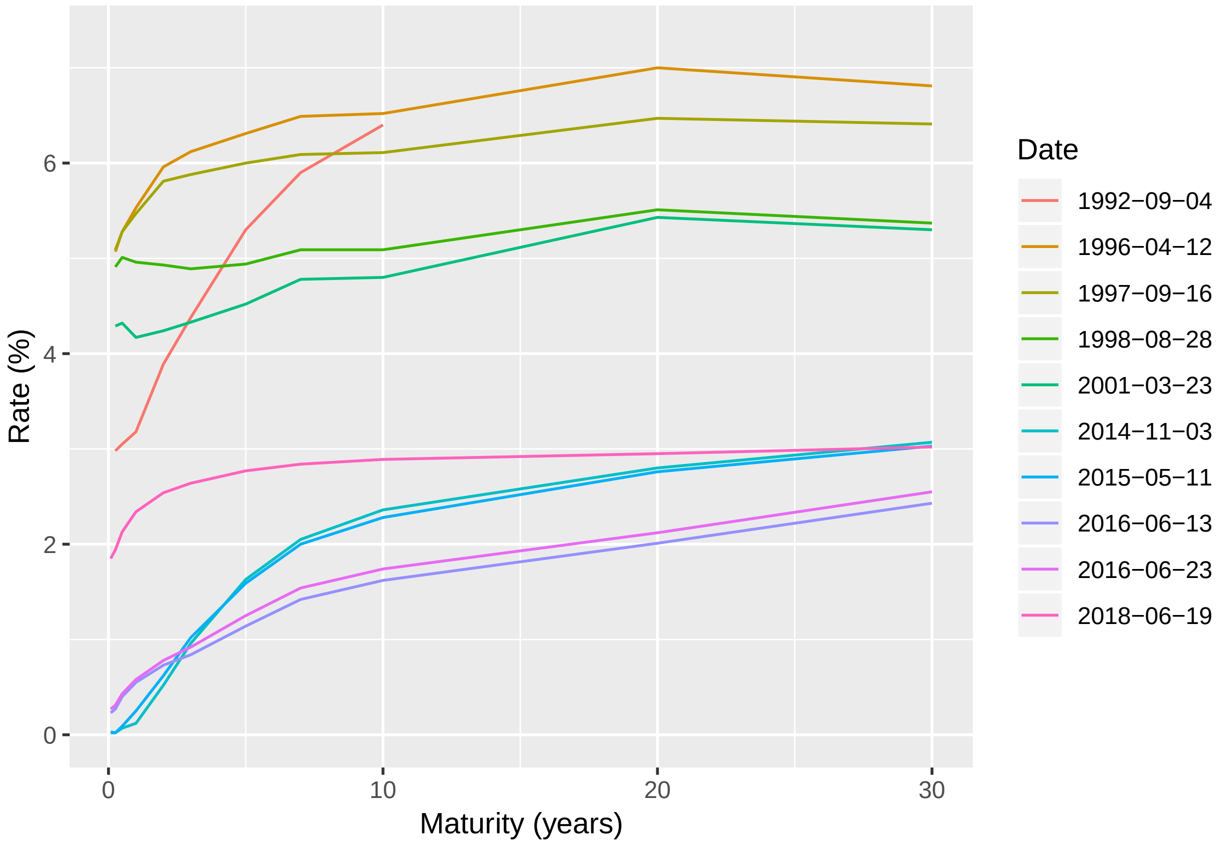
```
> tableelements[grep("N/A", tableelements)] = NA
```

Now let's get the table into a proper format:

```
> YCdata = matrix(tableelements, ncol=12,byrow=TRUE)
> YCdata = data.frame(YCdata, stringsAsFactors=FALSE)
> names(YCdata) = c("Date","1mo","3mo","6mo","1yr",
+                 "2yr","3yr","5yr","7yr","10yr",
+                 "20yr","30yr")
> YCdata$Date = as.Date(YCdata$Date,format="%m/%d/%y")
> YCdata[,2:12] = apply(YCdata[,2:12],2,as.numeric)
```

The package `reshape2` has some nice functions for adjusting data frames to get them into a desired form. Here, in order to create a plot we need the rates in a "long" form, with one rate per line. The `melt()` function will do this:

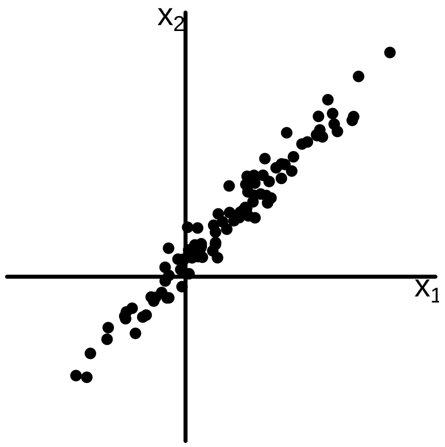
```
> library(reshape2)
> YCmelt = melt(YCdata, id.vars="Date")
> maturities = c(1/12,3/12,6/12,1,2,3,5,7,10,20,30)
> YCmelt$maturity =
+     maturities[as.integer(YCmelt$variable)]
> YCmeltsub =
+     YCmelt[YCmelt$Date %in% sample(YCmelt$Date, size=10), ]
> ggplot(YCmeltsub, aes(x=maturity,y=value,
+         color=factor(Date)) +
+   geom_line() +
+   labs(x="Maturity (years)", y="Rate (%)", color="Date")
```


Rate (%)
Maturity (years)
0
2
4
6
0
10
20
30
Date
1992−09−04
1996−04−12
1997−09−16
1998−08−28
2001−03−23
2014−11−03
2015−05−11
2016−06−13
2016−06−23
2018−06−19

1

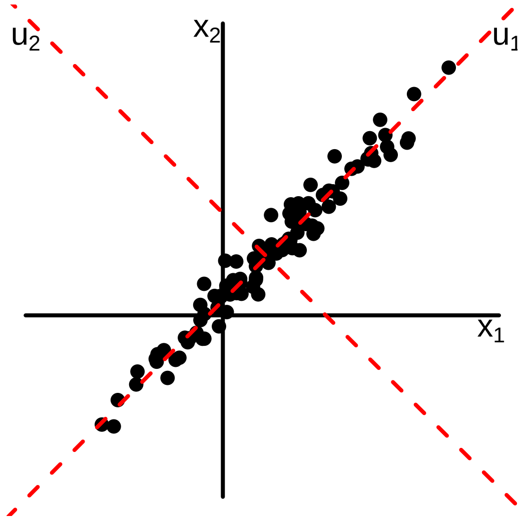
Principal Components Analysis

A standard, classic tool for dimension reduction is principal components analysis (PCA). It has some drawbacks, mostly in that it is best suited to find **linear** structure in data, but in the right situation it can be a very effective tool. First, consider the synthetic data shown in the following figure.



Exercise: Comment on the structure seen in the pair of variables shown in the previous figure. Would you say that the current pair of axes is the most “natural” way to represent the data?

The figure below shows the same data, with a new (u_1, u_2) coordinate system shown as the dashed lines.



These axes seem to be a more natural representation of the data. In particular, we note that the u_1 axis captures the major source of variability in the data. These axes are formed from the principal components of the data.

Formal Setup of PCA

Suppose we have observed n vectors $\mathbf{x}_i$, each of which is of dimension p:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T, \qquad i = 1, 2, \ldots, n.$$

For example, $\mathbf{x}_i$ could be a single yield curve, with $p \approx 10$.

Consider the collection of all possible linear combinations of the original variables formed by different choices of $\mathbf{u}$:

$$v_i = \sum_{j=1}^{p} u_j (x_{ij} - \overline{x}_j), \quad i = 1, 2, \ldots, n$$

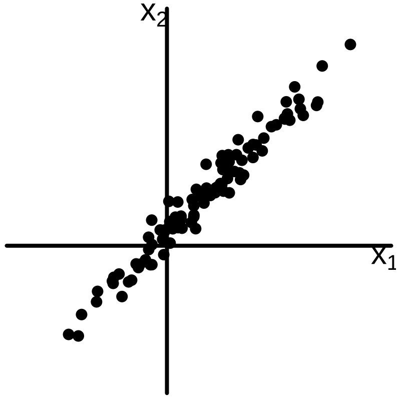
where $\overline{x}_j$ is the mean of the j^{th} variable, subject to

$$\sum_{j=1}^{p} u_j^2 = 1.$$

Hence, we can think of each choice of $\mathbf{u}$ as being a different **direction** centered on the sample mean.

The new measurements $v_1, v_2, \ldots, v_n$ are the **projections** of the original observations onto this new direction.

We now ask: What choice of $\mathbf{u}$ maximizes the sample variance of the resulting numbers $v_1, v_2, \ldots, v_n$?



The Math:

If $\mathbf{A}$ is a symmetric, positive definite matrix then the choice of $\mathbf{u}$ that maximizes $\mathbf{u}^T\mathbf{A}\mathbf{u}$ subject to $\mathbf{u}^T\mathbf{u}$ is the first eigenvector of $\mathbf{A}$.

Recall that

$$V(\mathbf{u}^T\mathbf{x}) = \mathbf{u}^T\boldsymbol{\Sigma}\mathbf{u}$$

where $V(\mathbf{x}) = \boldsymbol{\Sigma}$.

We approximate $\boldsymbol{\Sigma}$ using the sample covariance matrix $\widehat{\boldsymbol{\Sigma}}$.

The leading eigenvector of $\widehat{\boldsymbol{\Sigma}}$ is the first principal component.

The additional principal components $\mathbf{u}_2, \mathbf{u}_3, \ldots, \mathbf{u}_p$ are found by finding successive eigenvectors of $\widehat{\Sigma}$.

Some comments:

A new coordinate system is being constructed in which the p-dimensional data are represented. The center of this coordinate system is the sample mean of the data, but the axes are the principal components $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$.

When expressed in this new coordinate system, each of the dimensions have sample correlation of zero.

The amount of "variance explained" by each principal component is less than each of the previous principal components.

In many situations, we will choose some cutoff $q \ll p$ such that we will only retain the first q axes in our new coordinate system. This will be because these first q components "capture" most of the variability in the data.

For instance, in the toy example above, most of the variability in the data is "captured" by the first axis $\mathbf{u}_1$.

In typical applications p will be very large, and a low-dimensional representation will be of great value.

Back to the Yield Curves

Our previous inspection of some yield curves suggested that these curves share a common shape, and that it does not require ten numbers to describe changes in this shape from day to day.

This is an ideal situation to consider dimension reduction. The simplicity of the shapes of the curves suggests that PCA is worth trying.

For this analysis we will use the yield curves from 2010 to the present. We need to extract the relevant rates to run the PCA. Note that there are 11 rates over this time period.

```
> YCrates = YCdata[YCdata$Date > "2010-01-01", -1]
```

There is one row that is filled with zeros:

```
> YCrates = YCrates[-which(apply(YCrates,1,sum) == 0),]
```

We want to characterize shifts in the yield curves, so we transform into the day-to-day change in the rates:

```
> YCshifts = apply(YCrates,2,diff)
```

The function `princomp()` runs the PCA. The `subset` argument is used to skip over the rows with NA.

```
> pcaout = princomp(YCshifts,
+                subset=complete.cases(YCshifts))
```

1 The following output is useful:

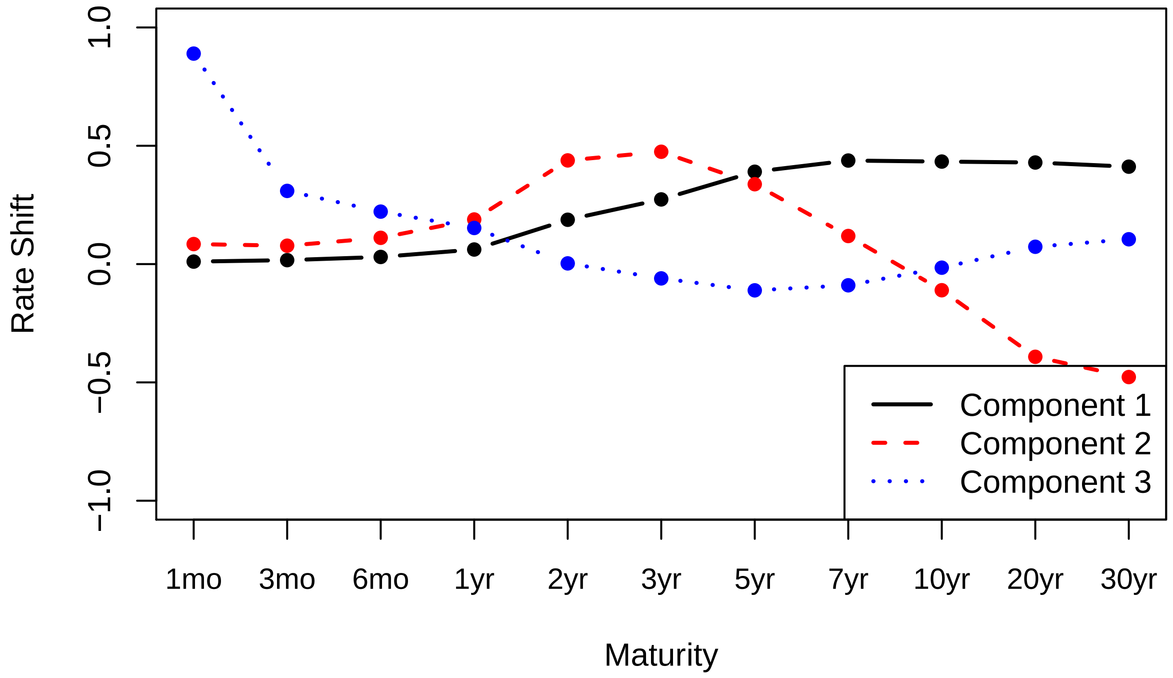
```
> summary(pcaout)
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4
Standard deviation     0.1132046 0.03442868 0.02476141 0.01669611
Proportion of Variance 0.8253307 0.07633800 0.03948671 0.01795273
Cumulative Proportion  0.8253307 0.90166871 0.94115543 0.95910815
                           Comp.5      Comp.6      Comp.7      Comp.8
Standard deviation     0.01431992 0.011679546 0.009897267 0.009073786
Proportion of Variance 0.01320631 0.008785207 0.006308566 0.005302457
Cumulative Proportion  0.97231446 0.981099670 0.987408236 0.992710693
                            Comp.9     Comp.10     Comp.11
Standard deviation     0.006819678 0.005906935 0.005637764
Proportion of Variance 0.002995214 0.002247111 0.002046982
Cumulative Proportion  0.995705907 0.997953018 1.000000000
```

2 This shows the proportion of variance explained by each component.

The first three components account for approximately 95% of the variability in the shifts in the yield curves.

The actual components (directions) are called the loadings. Look at `pcaout$loadings` to see these vectors. The following plot shows the first three components/loadings.

The interpretation is as follows: The day-to-day change in the yield curves can mostly be modelled as a linear combination of the three curves shown in this figure. The weight placed on each component (called the score) will change from day to day.


Rate Shift
1.0
0.5
0.0
−0.5
−1.0
1mo
3mo
6mo
1yr
2yr
3yr
5yr
7yr
10yr
20yr
30yr
Maturity
Component 1
Component 2
Component 3

1

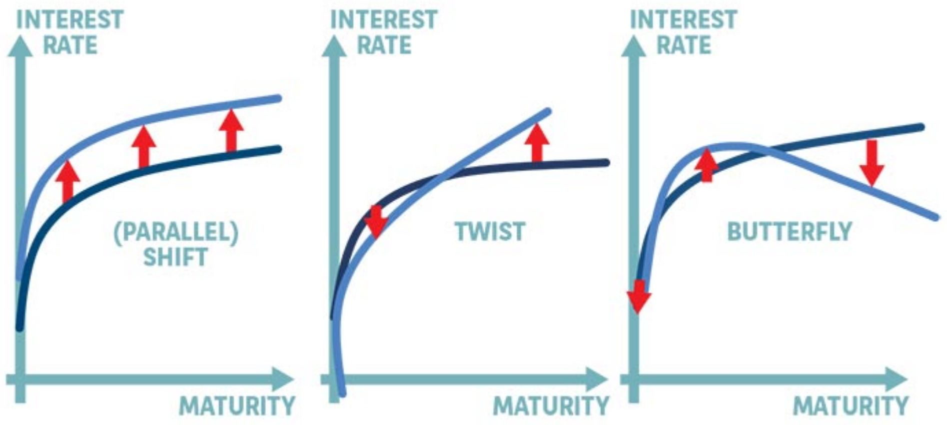


Figure 1: Figure from advisoranalyst.com

The scores give a representation of the shifts in yield curves. In fact, this is related to a common way to characterize how the yield curve has changed. The first three components are commonly given the names “parallel shift,” “twist,” and “butterfly.”

The `predict()` function, when applied to the output of `princomp()` will return the position of yield curve shift in this new representation. Note that R is a little picky on the format. The new shift must come in as a data frame with column names the same as used in the data.

```
> newshift = matrix(c(0.00, 0.00, 0.01, 0.01, 0.01, 0.07, 0.05, 0.05,
+                      0.03, 0.03, 0.04), nrow = 1)
> newshift = data.frame(newshift)
> names(newshift) = colnames(YCshifts)
> predict(pcaout, newshift)
        Comp.1     Comp.2       Comp.3      Comp.4      Comp.5      Comp.6
[1,] 0.1060112 0.02753681 -0.005937651 0.003564653 -0.00352043 0.003759287
           Comp.7      Comp.8        Comp.9     Comp.10     Comp.11
[1,] -0.005728047 -0.03691347 -0.0005428808 0.009590538 0.006370914
```

Scaling Variables for PCA

In cases where the variables being incorporated into PCA are on different scales, it is crucial that the variables be standardized prior to the analysis. It is customary to scale variables so that each has sample mean of zero, and sample variance of one.

This is equivalent to finding the eigenvectors of the correlation matrix instead of the covariance matrix. In fact, with function `princomp()`, one can simply use argument `cor = T` to achieve this.

Nonlinear Dimension Reduction

Consider the figure on the following slide. In this case, an individual "observation" is an image of a face. These faces are mapped into a two-dimensional space. The faces along the bottom show how the faces evolve along the line in the upper right of the figure.

This illustrates the use of nonlinear dimension reduction on high-dimensional data. PCA would not have been a good choice in this case because individual images cannot be approximated by a linear combination of a "basis" faces.

Exercise: What is the utility of the representation shown in the previous figure?

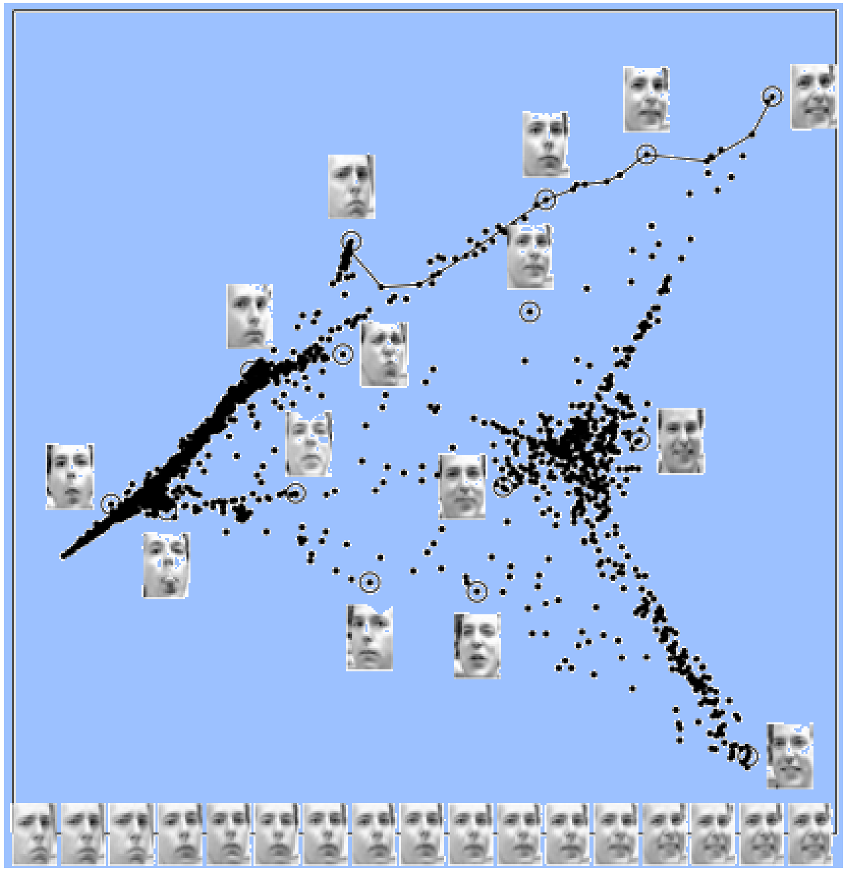
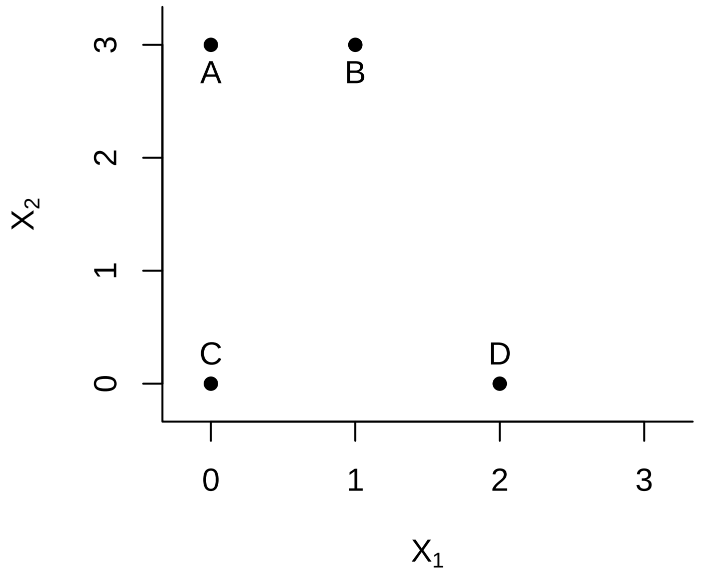


Figure 2: From Roweis and Saul (2000), *Science*

1 **Multidimensional Scaling**

2 Consider a synthetic case in which we have four points in two-dimensions,
3 labelled as in the following figure.

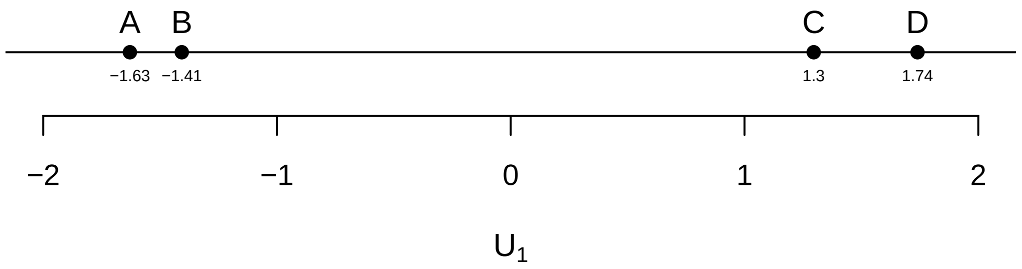
4 

Imagine calculating all of the pairwise distances between the four points shown in the above figure.

Then ask: **What one-dimensional representation of the data preserves those distances to the greatest extent possible?**

The answer is addressed by multidimensional scaling (MDS).

The result, using the classic approach, is shown in the figure below.



Exercise: What is the practical motivation behind trying to find a low-dimensional representation that preserves distances?

In R, classic multidimensional scaling is implemented through function `cmdscale()`.

Note in the example below how the `dist()` function is used in conjunction with `cmdscale()`. The argument `k` controls how many dimensions should be returned.

```
> mdsout = cmdscale(
+   dist(YCshifts[complete.cases(YCshifts),]), k=2)
```

In this example, each "observation" is a single shift in the yield curve. We calculate the distances between all pairs of yield curves, and then seek the two-dimensional mapping that preserves these distances to the extent possible.

Exercise: Compare the results from multidimensional scaling to those found using PCA. What conclusion can you draw?

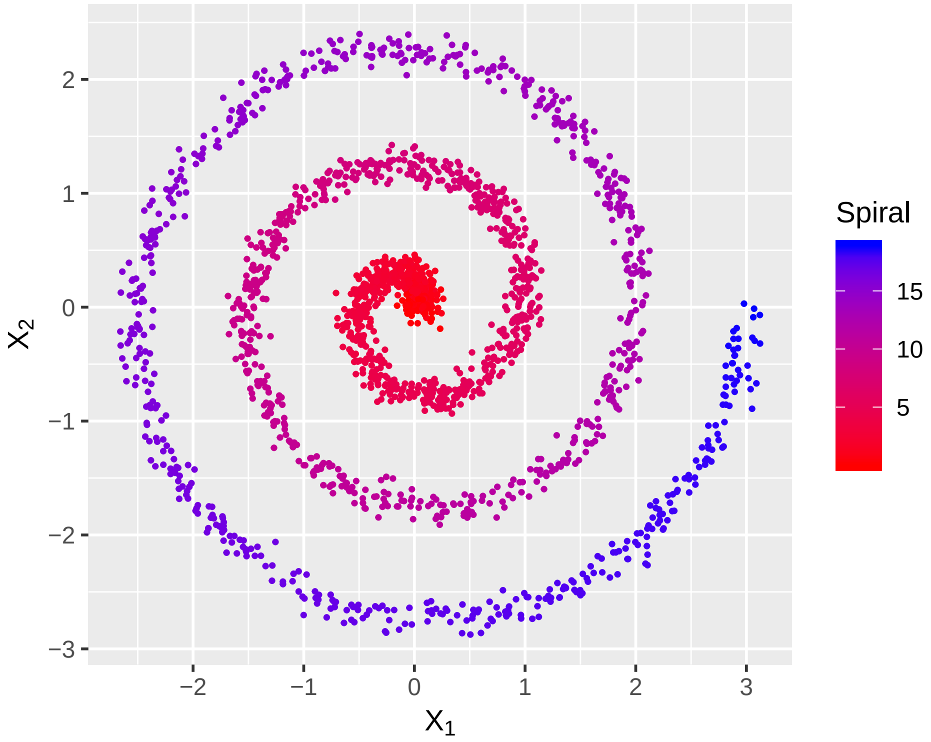
The criterion minimized by MDS can be written as follows:

$$\text{Stress}(x_1, x_2, \ldots, x_n) = \left(\sum_{i \neq j} (d_{ij} - \Delta(x_i, x_j))^2 \right)^{1/2}$$

where d_{ij} is the "true" distance between observations i and j, and $\Delta(x_i, x_j)$ is the distance in the lower-dimensional mapping.

Although in "classical" MDS the distances utilized are Euclidean, the general concept of "metric" MDS can use any distance d_{ij} which is a properly defined distance metric.

Consider a synthetic data set where points lie on a spiral:



Exercise: Does Euclidean distance provide a meaningful measure of "dissimilarity" for the data depicted in the spiral example?

Some approaches to dimension reduction attempt to construct an improved measure of “distance” or “dissimilarity” and incorporate that into metric MDS. The above spiral example is a (classic) illustration of why that can be necessary. (Imagine drawing a new pair of axes onto the spiral example. Can you find any axis system that will yield a useful representation of the data?)

Consider the following observation: Although Euclidean distance is not a useful **global** measure of dissimilarity, it is useful on **local** scales. In other words, Euclidean distance could be used to identify two observations which are very similar.

ISOMAP

Isomap is an approach to nonlinear dimension reduction built on MDS with a well-chosen geodesic distance between points. The steps:

1. **Connect all "neighboring" observations.** This can be achieved by finding all observations within distance ϵ or by find the K nearest neighbors of each observation.

2. **Assign weights to connected observations.** The weight is equal to the Eucldean distance separating them.

3. **Calculate the geodesic distance between all pairs of points.** The shortest path between points using the provided weights.

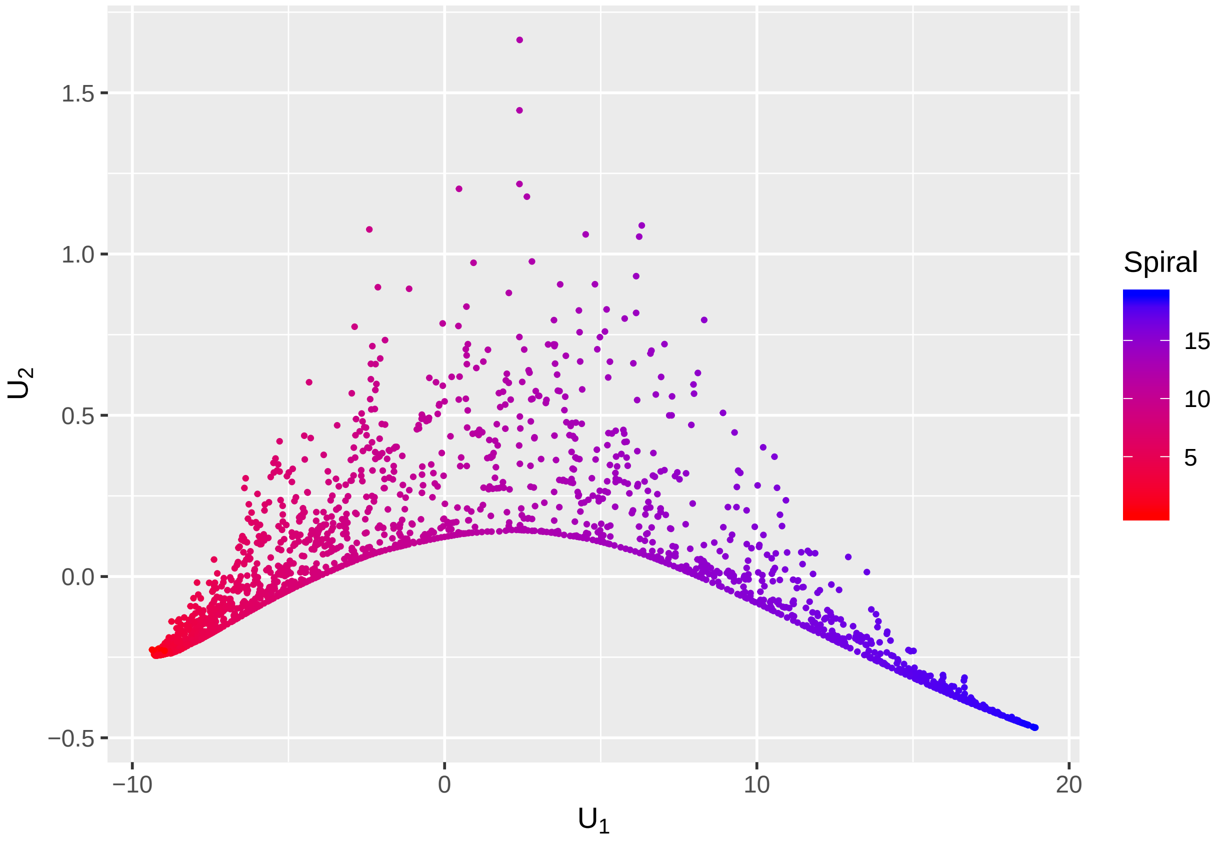
4. **Use MDS on the resulting distance matrix.**

The following slides show the result of applying the Isomap algorithm to the spiral example.

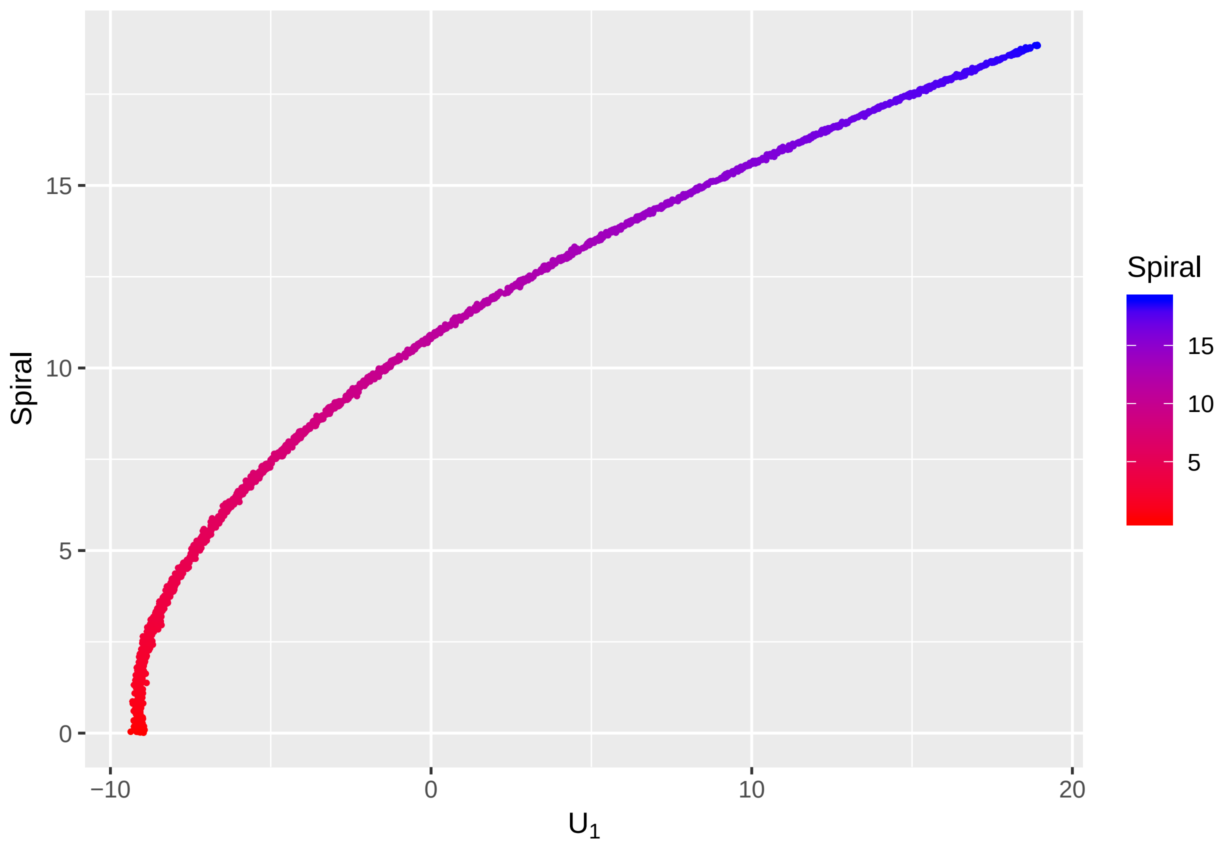
In this case $K = 10$ is utilized. Note that choosing K too small creates the risk of having a "fragmented" graph in which there are observations with no connecting path.

The first graph shows the new two-dimensional representation created by Isomap. The color indicates the position along the spiral. We observe that U_1 maps well to the spiral position.

This is confirmed by the following plot. Note that there is no reason to expect or prefer that there be a linear relationship between U_1 and the spiral position. The strictly monotonic relationship is sufficient.



1



1

Stock Example

In this example we will consider a sample of 1000 randomly chosen NYSE stocks. For each, a time series of their closing price is obtained, covering 30 trading days from August 18, 2017 to September 29, 2017.

Read in the data:

```
> stocksample = read.table("stocksample.txt", header=T,
+                          sep="\t", comment.char="")
```

Scale each time series to have mean zero and variance one, so that they can be fairly compared.

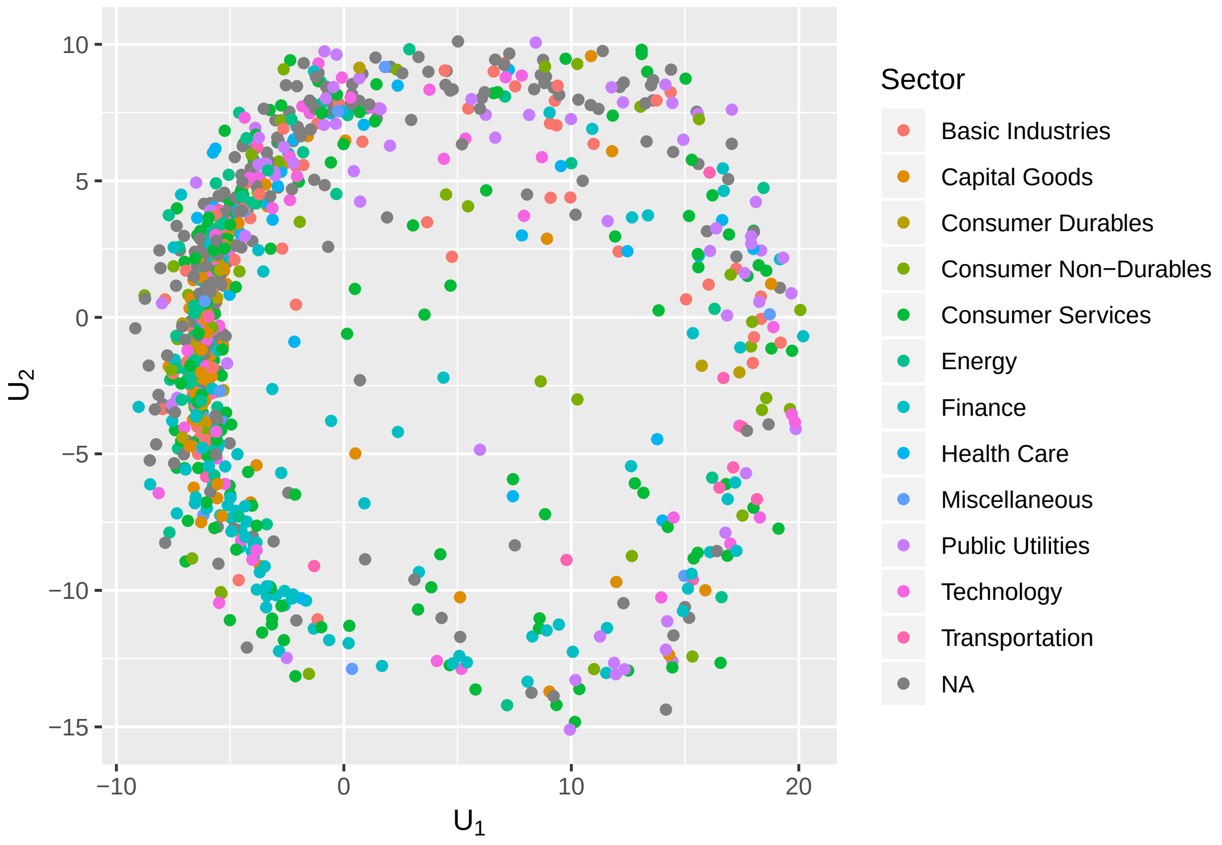
```
> stocksamplescl = apply(stocksample[,5:34],1,scale)
```

Construct a distance matrix between the time series, and then run the Isomap procedure with $k = 5$:

```
> stockdistmat = dist(t(stocksamplescl))
> isooutstocks = isomap(stockdistmat, k=5)
> stocksample = cbind(stocksample, isooutstocks$points[,1:5])
```

Now create a plot looking at the first two dimensions, coloring the points by sector:

```
> ggplot(stocksample, aes(x=stocksample[,35],y=stocksample[,36],
+                         color=sector)) + geom_point() +
+   labs(x=expression(U[1]), y=expression(U[2]), color="Sector")
```


10
5
0
−5
−10
−15
−10
0
10
20
U_2
U_1
Sector
Basic Industries
Capital Goods
Consumer Durables
Consumer Non−Durables
Consumer Services
Energy
Finance
Health Care
Miscellaneous
Public Utilities
Technology
Transportation
NA

1

Exercise: Consider the following R commands. Use this to explore the structure found in this low-dimensional representation. Can you find a relationship between position in this two-dimensional space and shape of the time series?

```
> plot(stocksample[,35],stocksample[,36],pch=16)
> identify(stocksample[,35],stocksample[,36])
```

Diffusion Map

A criticism of Isomap is that it is computationally intensive to calculate the geodesic distances. Also, Isomap is not as robust to noise in the data as are some other approaches, including the one described next, diffusion map.

This approach is again built on MDS, but the underlying distance between pairs of points is calculated from random walks constructed on the observed data. Steps in this random walk are taken from observation to observation based on proximity in Euclidean distance (typically), but only neighbors at short distances are candidates for stepping to.

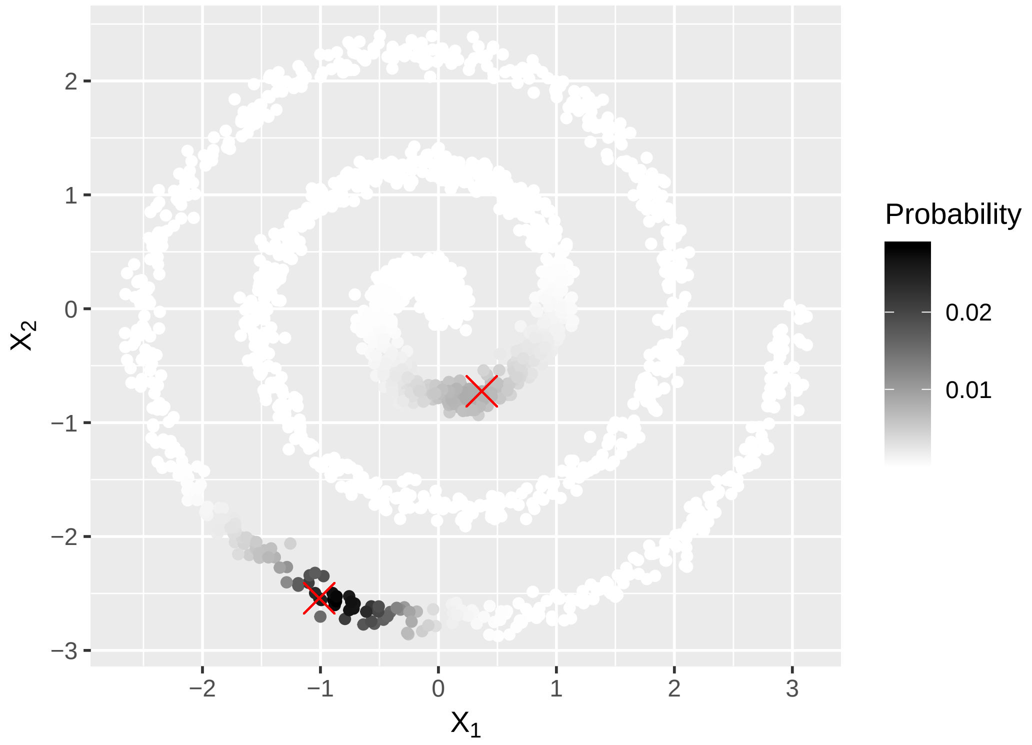
If the random walk "diffuses" over enough steps, the result is a distribution over the observations. Comparing distributions between different starting points gives a natural measure of global proximity.

The figure on the next slide illustrates the idea.

Two observations in the data set are marked with "X." For either X, imagine a random walk that takes ten steps away from that observation. Steps are small, i.e., the probability of stepping to an observation decreases rapidly as a function of its distance from the current position in the walk.

The ending position of the random walk is, of course, random. The shading shows the probability that the random walk ends at various observations.

The dissimilarity of these distributions is a natural way of quantifying the distance between them.



1

Technical Details on Diffusion Map

The process starts by constructing a similarity measure between pairs of observations. There is some flexibility in how this is done, but a standard approach is quantify the similarity between observations i and j as

$$s_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\epsilon)$$

where ϵ is a tuning parameter.

Next, the probability of stepping from observation i to any other observation j is

$$p_{ij} = s_{ij} \Big/ \sum_k s_{ik}$$

Note that this scaling will force $\sum_j p_{ij} = 1$.

This completely defines the behaviour of the random walk.

The choice of ϵ is crucial. If it is chosen too small, then steps of the random walk will likely move only to other observations which are very similar, and, in particular, remain at the same observation. (A "step" can be to the same observation.)

If ϵ is chosen too large, then it will be possible to step to observations which are actually quite dissimilar, and the result will be a mapping that does not adhere to the lower dimensional structure of interest.

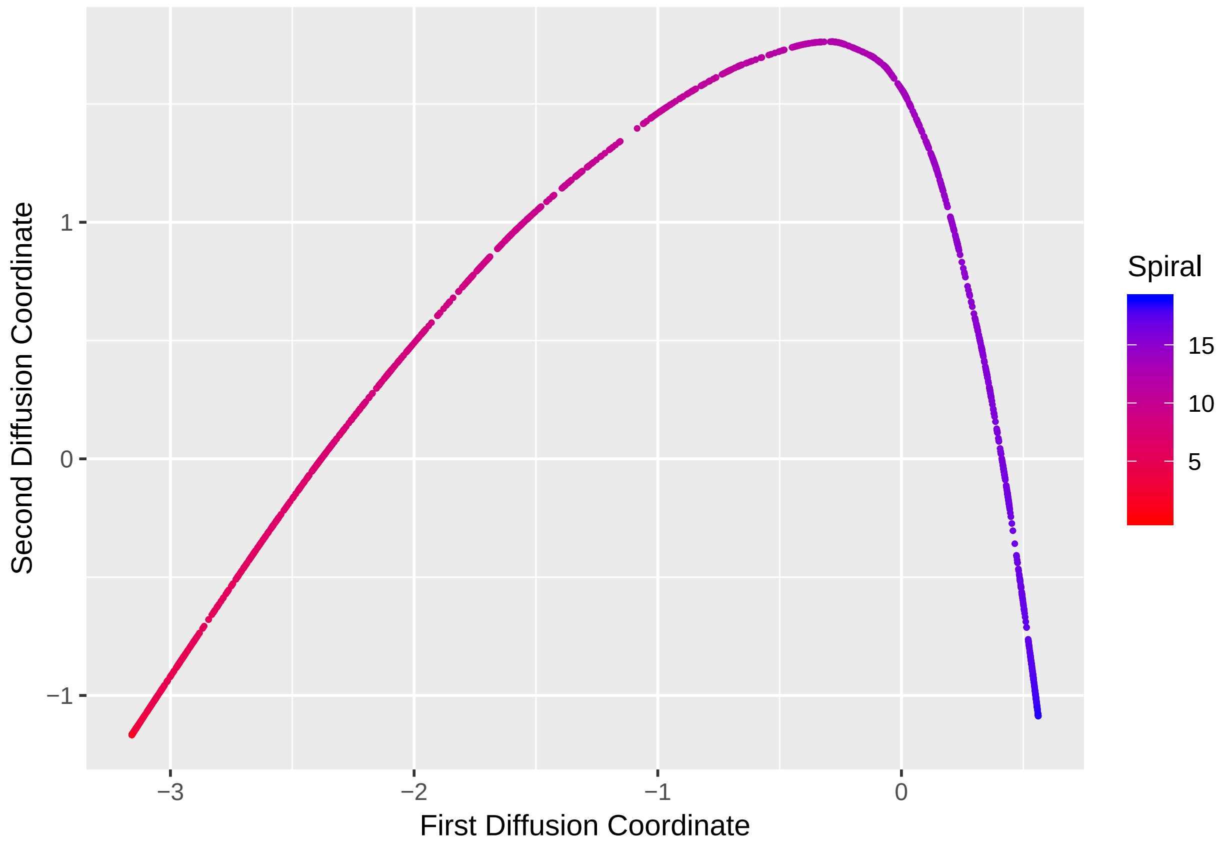
There is no "right" choice of ϵ. Ultimately, different values should be tried, and the chosen value should be that which yields a useful low-dimensional representation.

The slides on the next page show the results when running diffusion map on the spiral example.

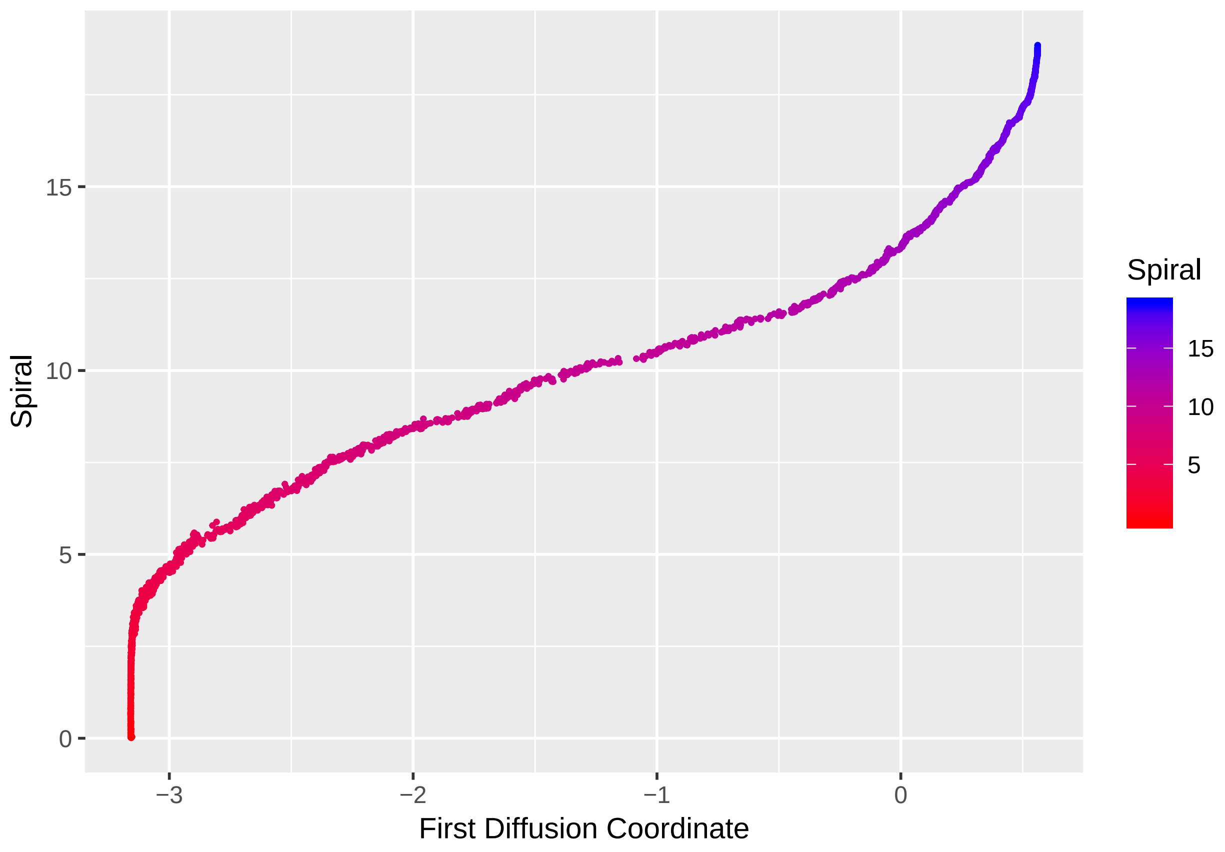
In this case, $\epsilon = 0.05$, and 10-step random walks are utilized.

The random walk is much more robust to noise in the data than is the geodesic distance. One way to see this is to note that small perturbations in the position of any observation will have very little effect on the diffusion distances, but could potentially affect geodesic distances a great deal.

Also, there are computational shortcuts that make calculating the diffusion map quick. Much of this is based on the theory of Markov chains.


Second Diffusion Coordinate
First Diffusion Coordinate
1
0
−1
−3
−2
−1
0
Spiral
15
10
5

1


Spiral
0
5
10
15
−3
−2
−1
0
First Diffusion Coordinate
Spiral
15
10
5

1

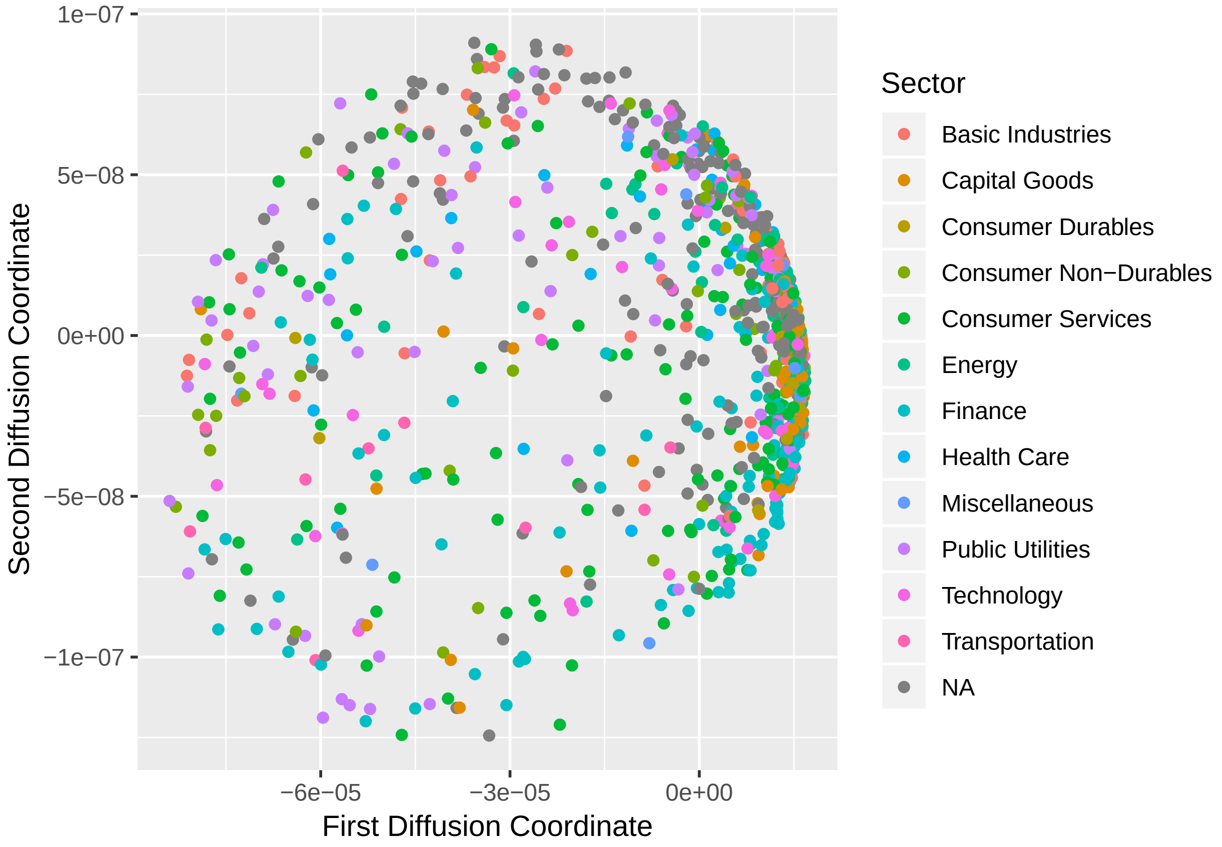
Return to the Stock Example

Diffusion map is implemented in the package `diffusionMap` via the function `diffuse()`. The arguments `eps.val` and `t` set the values of ϵ and the length of the random walk, respectively.

```
> stockdiffmap = diffuse(stockdistmat, eps.val=50, t=10)
```

The component `X` of the output contains the coordinates in diffusion space.

```
> stocksample$dmap1 = stockdiffmap$X[,1]
> stocksample$dmap2 = stockdiffmap$X[,2]
```


1e−07
5e−08
0e+00
−5e−08
−1e−07
−6e−05
−3e−05
0e+00
First Diffusion Coordinate
Second Diffusion Coordinate
Sector
Basic Industries
Capital Goods
Consumer Durables
Consumer Non−Durables
Consumer Services
Energy
Finance
Health Care
Miscellaneous
Public Utilities
Technology
Transportation
NA

1

Exercise: Again, explore the space to see how the shape of the time series varies over diffusion space.

The Options Example

Let's now revisit the options sample previously described, and look at the four quantities that we would consider as "predictors" in a model for the price.

```
> optionsdatasub = optionsdata[,c(2,3,9,10)]
```

To construct a distance metric in this case, we need to adjust for the fact that the four variables are not comparable. Here, we introduce the Mahalanobis Distance as a useful approach. The distance between two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$(\mathbf{x}_i - \mathbf{x}_j)^T \,\widehat{\Sigma}^{-1}(\mathbf{x}_i - \mathbf{x}_j)$$

where $\widehat{\Sigma}$ is the sample covariance matrix of the $\mathbf{x}$'s.

Exercise: Explain why the weighting by the inverse covariance matrix is a good approach.

We can calculate the Mahalanobis distances as follows, using the function `distances()` from the package of the same name:

```
> optdist = distance_matrix(distances(optionsdatasub,
+                                      normalize="mahalanobize"))
```

Note that the function `distance_matrix()` extracts the distance matrix from the output of `distances()`.

Now we use Isomap and store the mappings:

```
> optiso = isomap(optdist,k=10)
> optionsdata$iso1 = optiso$points[,1]
> optionsdata$iso2 = optiso$points[,2]
> optionsdata$iso3 = optiso$points[,3]
```

Exercise: Construct plots to compare the Isomap-constructed representation with the four input quantities, and also with the ask price. Be sure to look at the following plot:

```
> ggplot(optionsdata, aes(x=iso1,y=iso3,color=log10(ask))) +
+   geom_point(size=1) +
+   scale_color_continuous(low="blue",high="red") +
+   labs(x=expression(U[1]), y=expression(U[2]), color="Log Ask")
```