MSF 566 - Financial Time Series Analysis
Lecture Notes

Andrew P. Acosta

April 20, 2018

Contents

List of Figures

List of Tables

Code Samples

1 Introduction

Primary Text Reading. Tsay (2005, chap. 1)

1.1 What is financial time series analysis?

In general, a time series is a sequence of data points, usually measured at successive times, spaced at specific intervals. Time series analysis comprises methods that seek to understand the context of the data points to answer questions such as, "What is this series telling me about whatever is generating this series?", or to make *forecasts* to predict future data points. Time series forecasting is the use of a model to forecast future events based on known past events: to forecast future data points before they occur. A good example might be forecasting a price of a share of stock based on its past performance.

Much of time series analysis is predicting future outcomes.

Financial time series analysis is concerned with asset valuation and volatility over time. The state of the world today (*e.g.* a stock price or interest rate) is affected in some way by the previous state, perhaps yesterday or two seconds ago. Similarly, the state of the world in the future is likely to be affected by the current state. This assumption allows some level of *financial forecasting* using various methods of time series analysis.

There are three major outcomes of time series analysis, graphical analysis, autocorrelation, and trend behavior. We will briefly discuss how each is explored.

Graphical analysis. When we view a time series in the form of time series plot, we may observe peaks and valleys. It may become apparent that a trend has developed by looking at the rising and falling of the line. Methods that analyze time series data graphically comprise *technical analysis* and a *Exploratory Data Analysis*.

Autocorrelation. We may want to measure the extend to which data points seem to rely upon previous points, or how random they seem to be. This type of analysis applies autocorrelation studies.

Trend behavior. Some financial time series exhibit trends which can be examined numerically. They may have some seasonal pattern, such that, for instance, their level will rise in summer and decline in winter.

1.2 Review of matrices, statistics and probability

1.2.1 Descriptive Statistics

This section is a brief review of inferential statistics and probability. There will also be some review of linear algebra, the mathematics of matrices.

A working knowledge of statistics and matrix math is assumed.

Descriptive statistics are used to describe the basic features of the data in a study. They provide simple summaries about the sample and the measures. Together with simple graphics analysis, they form the basis of virtually every quantitative analysis of data. Statistics such as mean, median, and standard deviation are descriptive.

Descriptive statistics are typically distinguished from *inferential statistics*. With descriptive statistics we are simply describing what is or what the data shows. With inferential statistics, we are trying to reach conclusions that extend beyond the immediate data

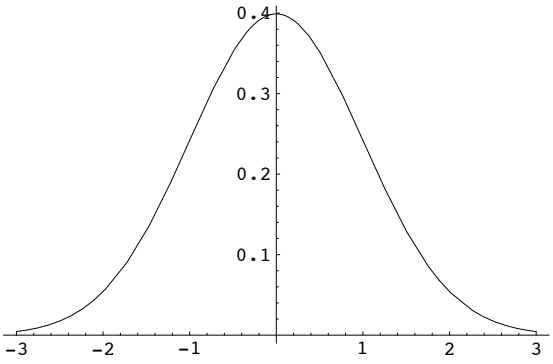

Figure 1.1: Normal Probability Density

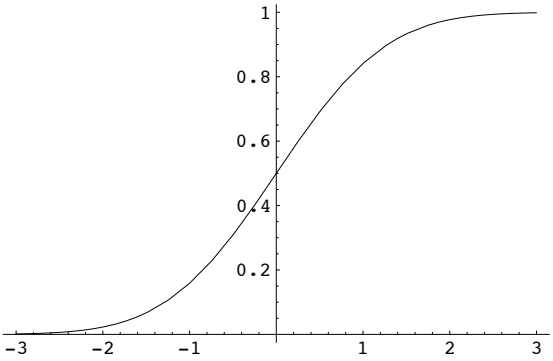

Figure 1.2: Normal Cumulative Distribution

alone. For instance, we use inferential statistics to try to infer from the sample data what the population might think. Or, we use inferential statistics to make judgments of the probability that an observed difference between groups is a dependable one or one that might have happened by chance in this study. Thus, we use inferential statistics to make inferences from our data to more general conditions; we use descriptive statistics simply to describe what's going on in our data.

Every time we try to describe a large set of observations with a single indicator we run the risk of distorting the original data or losing important detail. For this reason, statistics such as kurtosis and skewness are helpful in describing a frequency distribution with respect to a normal distribution.

The normal distribution is a histogram depicting the probability of a number with respect to its distance from a mean of zero across a range $(-\infty, \infty)$.

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \exp(-z^2/2), \quad -\infty < z < \infty \tag{1.1}$$

For instance, (1.1) is the equation that produces Figure 1.1. The cumulative distribution (1.2) is depicted in Figure 1.2.

$$\phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2), \quad -\infty < z < \infty \tag{1.2}$$

A great deal of statistical analysis comes from the normal distribution. Being able to calculate the values of specific ranges in the distribution is essential to understanding descriptive statistics.

1.2.2 Sampling and Validity

Sampling is the process of selecting units (*e.g.*, people, organizations) from a population of interest so that by studying the sample we may fairly generalize our results back to the population from which they were chosen. We begin by covering some of the key terms in sampling like "population" and "sampling frame." Then, because some types of sampling rely upon quantitative models, we talk about some of the statistical terms used in sampling.

Random Sample. Since it is not practical or even possible in many cases, to gather all data from a population, gathering a sample is preferred. However, the method by which a sample is chosen is very important and will affect the results of analysis. One of the easiest sampling methods is creating a *random sample*.

How do we select a simple random sample? Assume that we are doing some research in equity analysis using a specific price forecasting method. First, we have to get the sampling frame organized. To accomplish this, we have to define the universe of stocks that are eligible for study. Next, we actually draw the sample using some random number assigned to each stock. If we are studying 100 stocks in our sample of perhaps 5,000 stocks. Then, the sampling fraction is $f = n/N = 100/5000 = .02$ or 2%. To actually draw the sample, we have several options. We could print the list of 5,000 stocks, tear them into separate strips, put the strips in a box, and pull out the first 100. This mechanical procedure is tedious and the quality of the sample would depend on how thoroughly we mixed them up and how randomly we reached in.

A better procedure would be to use a random number generator that picks a number from 1 to 5,000, such as the following MATLAB code,

```
my_sample=floor(5000*rand(1,100));
```

However, this method does *not* guarantee that the numbers generated will be unique, but it is a start.

Simple random sampling is simple to accomplish and is easy to explain to others. Because simple random sampling is a fair way to select a sample, it is reasonable to generalize the results from the sample back to the population. Simple random sampling is not the most statistically efficient method of sampling and we may, just because of the luck of the draw, not get good representation of subgroups in a population. To deal with these issues, we have to turn to other sampling methods.

Systematic Sample. Another random selection method is to pick every kth item in a population. Here are the steps to achieve a systematic random sample,

1. number the units in the population from 1 to N
2. decide on the n (sample size) that we need

3. $k = N/n$ = the interval size
4. randomly select an integer between 1 to k
5. select every kth unit

All of this will be much clearer with an example. Assume that we have a population that only has $N = 100$ items in it and that we want to take a sample of $n = 20$. To use systematic sampling, the population must be listed in a random order. The sampling fraction would be $f = 20/100 = 20\%$. In this case, the interval size, k, is equal to $N/n = 100/20 = 5$. Now, we select a random integer from 1 to 5. In our example, imagine that we chose 4. Now, to select the sample, we start with the 4th unit in the list and take every kth unit (every 5th, because k=5). We would be sampling units 4, 9, 14, 19, ..., 100 and we would have 20 units in our sample.

For this to work, it is essential that the units in the population are randomly ordered, at least with respect to the characteristics we are measuring. Systematic random sampling is fairly easy to do. We only have to select a single random number to start things off. It may also be more precise when selecting the sample size n than a simple random sampling, which uses a random number generator that is not guaranteed to be unique.

Stratified Sample. This is a sample within a sample because a sample is drawn, and then a sample from within that is drawn. Stratified random sampling, also sometimes called *proportional* or *quota random sampling*, involves dividing our population into homogeneous subgroups and then taking a simple random sample in each subgroup.

In more formal terms, we divide the population into non-overlapping groups (*i.e.*, strata) $N_1, N_2, N_3, \ldots, N_i$, such that $N_1 + N_2 + N_3 + \cdots + N_i = N$. Then we select a simple random sample of $f = n/N$ in each strata.

There are several major reasons why we might prefer stratified sampling over simple random sampling. First, it assures that we will be able to represent not only the overall population, but also key subgroups of the population, especially small minority groups. If we want to be able to talk about subgroups, this may be the only way to effectively assure we will be able to.

If the subgroup is extremely small, we can use different sampling fractions, f within the different strata to randomly over-sample the small group (although we will have to weight the within-group estimates using the sampling fraction whenever we want overall population estimates). When we use the same sampling fraction within strata we are conducting proportionate stratified random sampling. When we use different sampling fractions in the strata, we call this *disproportionate stratified random sampling.*

Second, stratified random sampling will generally have more statistical precision than simple random sampling. This will only be true if the strata or groups are homogeneous. If they are, we expect that the variability within-groups is lower than the variability for the population as a whole. Stratified sampling capitalizes on that fact.

For example, we want to test a trading strategy that works well for stocks of consumer staples, automotive, and leisure industry. We have three populations. Let us say that the population of stocks for our study can be divided into three groups: consumer staples, automotive, and leisure. Furthermore, let us assume that both the automotive and the leisure stocks are relatively few within the population (10% and 5% respectively). If we

just did a simple random sample of $n = 100$ with a sampling fraction of 10%, we would expect by chance alone that we would only get 10 and 5 persons from each of our two smaller groups. And, by chance, we might get fewer than that.

If we stratify, we can do better. First, let us determine how many stocks we want to have in each group. Let us say we still want to take a sample of 100 from the population of 1,000 stocks. But we think that in order to say anything about subgroups we will need at least 25 cases in each group. So, we will sample 50 consumer staples, 25 automotive, and 25 leisure stocks. We know that 10% of the population, or 100 stocks, are automotive. If we randomly sample 25 of these, we have a within-stratum sampling fraction of $25/100 = 25\%$. Similarly, we know that 5% or 50 stocks are leisure. So our within-stratum sampling fraction will be $25/50 = 50\%$. Finally, by subtraction we know that there are 850 consumer staples stocks. Our within-stratum sampling fraction for them is $50/850 = 5.88\%$. Because the groups are more homogeneous within-group than across the population as a whole, we can expect greater statistical precision (less variance). And, because we stratified, we know we will have enough cases from each group to make meaningful subgroup inferences.

Stratified Sample			
Type	within pop.	sample size	within-stratum sample
consumer staples	85%	50	$50/850 = 5.88\%$
automotive	10%	25	$25/100 = 25\%$
leisure industry	5%	25	$25/50 = 50\%$

Table 1.1: Stratified Sample Selection

Convenience Sample. A convenience sample is one where the items sampled from the population are chosen for their easy access. For instance, a consumer survey conducted by asking the first 100 people who walk into a particular bank one morning is a convenience sample. Those not going to that bank on that morning are not part of the sample. This might be an important issue, or it might not be, depending upon the purpose of the survey. Choosing 100 stocks for analysis based upon the fact that the analyst has heard of them is also a convenience sample because it ignores the perhaps thousands of stocks unknown to the analyst.

Validity. *External validity* is related to generalizing. External validity refers to the approximate truth of conclusions that involve generalizations. In other words, external validity is the degree to which the conclusions in a study would hold for other persons in other places and at other times.

Validity refers to the approximate truth of propositions, inferences, or conclusions

In science there are two major approaches to how we provide evidence for a generalization. The first approach is the *Sampling Model*. In the sampling model, we start by identifying the population to generalize to. Then, we draw a fair sample from that population and conduct research with the sample. Finally, because the sample is representative of the population, one can automatically generalize results back to the population. However, there are several problems with this approach.

First, perhaps we do not know at the time of the study who we might ultimately like to generalize to. Second, we may not be easily able to draw a fair or representative sample.

Third, it is impossible to sample across all times that we might like to generalize to (like next year).

The second approach to generalizing is the *Proximal Similarity Model*. Under this model, we begin by thinking about different generalizability contexts and developing a theory about which contexts are more like our study and which are less so. For instance, we might imagine several settings that have people who are more similar to the people in our study or people who are less similar. This also holds for times and places.

When we place different contexts in terms of their relative similarities, we can call this implicit theoretical a gradient of similarity. Once we have developed this proximal similarity framework, we are able to generalize. How? We conclude that we can generalize the results of our study to other persons, places or times that are *more like* (that is, more proximally similar) to our study. Notice that here, we can never generalize with certainty – it is always a question of more or less similar.

Threats to External Validity. A threat to external validity is an explanation of how we might be wrong in making a generalization. For instance, we conclude that the results of our study (which was done in a specific place, with certain types of people, and at a specific time) can be generalized to another context (for instance, another place, with slightly different people, at a slightly later time). There are three major threats to external validity because there are three ways we could be wrong – people, places or times.

Our critics could argue that the results of our study are due to the unusual type of people who were in the study. Or, they could argue that it might only work because of the unusual place we did the study in (perhaps we did our educational study in a college town with lots of high-achieving educationally-oriented students). Or, one might suggest that we did our study in a peculiar time. For instance, if we did an interest rate study the week after the Federal Reserve issues the well-publicized results of the latest rate change, we might get different results than if we had done it the week before.

Improving External Validity. How can we improve external validity? One way, based on the sampling model, suggests that we do a good job of drawing a sample from a population. For instance, we should use random selection, if possible, rather than a *nonrandom* procedure.

A second approach would be to use the theory of proximal similarity more effectively. How? Perhaps we could do a better job of describing the ways our contexts and others differ, providing lots of data about the degree of similarity between various groups of people, places, and even times. We might even be able to map out the degree of proximal similarity among various contexts with a methodology like concept mapping. Perhaps the best approach to criticisms of generalizations is simply to show them that they are wrong – do the study in a variety of places, with different people and at different times.

External validity (ability to generalize) will be stronger the more we replicate our study.

Once we identify the theoretical and accessible populations, we have to get a list of the members of the accessible population. The listing of the accessible population from which we draw our sample is called the *sampling frame*. If we were doing a performance survey of S&P 500 stocks, then the stocks within the index would be our sampling frame. Getting that listing is simply a matter of going to a Bloomberg terminal and obtaining the list. Other studies may be as simple, such as studying the rate of inflation on the money supply, for instance. However, preparing sample data of credit default swaps, for which

the data is very difficult to obtain, also is difficult to compare to other swaps. This is a generally inaccessible population.

People often confuse what is meant by *random selection* with the idea of *random assignment*. You should make sure that you understand the distinction between random selection and random assignment. Random selection is the method by which the sample is selected, and random assignment is the method by which those samples are assigned to a test group or to a control group.

1.2.3 Inferential Statistics

With inferential statistics, we are trying to reach conclusions that extend beyond the immediate data alone. For instance, we use inferential statistics to try to infer from the sample data what the population might signify. Or, we use inferential statistics to make judgments of the probability that an observed difference between groups is a dependable one or one that might have happened by chance in this study. Thus, we use inferential statistics to make inferences from our data to more general conditions; we use descriptive statistics simply to describe what is going on in our data.

Inferential statistics help make predictions and judgements

Most of the major inferential statistics come from a general family of statistical models known as the General Linear Model. This includes the t test, Analysis of Variance (ANOVA), Analysis of Covariance (ANCOVA), regression analysis, and many of the multivariate methods like factor analysis, multidimensional scaling, cluster analysis, and discriminant function analysis.

The purpose of these statistical tests is to conduct an experiment. Is a certain procedure effective in forecasting rates? Is there a significant difference between two samples?

1.3 Statistical Inference

When we make some conclusion from a data sample we obtained, we draw an inference about a population based on some statistical inference method. We may be attempting to predict one of the following,

- point estimate: we are trying to determine a specific number
- interval estimate: we are tying to obtain a *confidence interval*
- hypothesis test: we run a statistical significance test to determine if a sample belongs in the population

1.3.1 Point Estimation

Point estimation involves the use of sample data to calculate a single value (known as a *statistic*) which is to serve as a "best guess" for an unknown (fixed or random) population *parameter*. One of the best known methods of point estimation is linear regression using the method of ordinary least squares. For example, Figure 1.3 is a linear regression through a scatterplot.

A statistic is drawn from a sample. A parameter is contained in the population.

In most types of regression analysis, linear or otherwise, we are getting a point estimate through a plane, or more simply, a line that predicts points along the line.

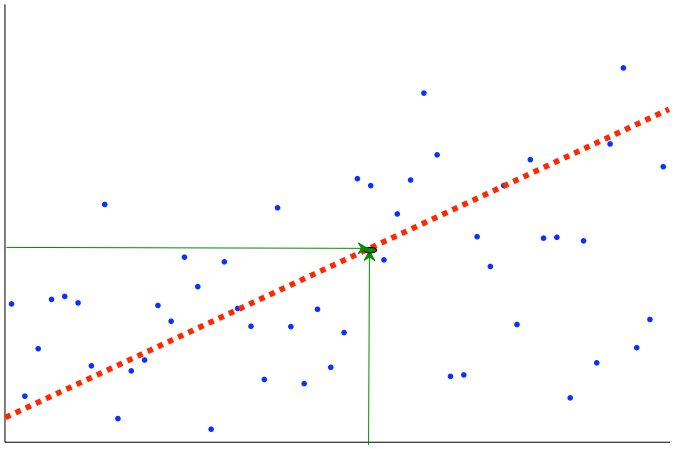

Figure 1.3: Scatterplot and OLS regression showing a single point estimate which is at the $(x,\ y)$ coordinates marked by arrows.

1.3.2 Interval Estimation

Interval estimation is the use of sample data to calculate an interval of probable values of an unknown population parameter. The most prevalent forms of interval estimation are *confidence intervals*. Because samples are taken from a population with a distribution, such as a normal distribution (Figure 1.1), it is possible that the sample does not accurately describe the population. If a point estimate is taken from one of the two tails of the distribution, it certainly would not represent the mean of the population.

An interval estimate provides some information about the accuracy of the sample and it represents a range of likely values. For instance, we can obtain a 95% confidence interval, meaning that we have a 95% confidence that our population mean μ is contained in our interval. To create the interval we calculate,

$$\begin{aligned} P\big(-z < \frac{\bar{Y}-\mu}{1/\sqrt{n}} < z\big) &= .95; z = 1.96 \qquad (1.3) \\ P\big(-1.96 < \frac{\bar{Y}-\mu}{1/\sqrt{n}} < 1.96\big) &= .95 \end{aligned}$$

which gives us an interval estimate of $[\bar{y} - 1.96/\sqrt{n}, \bar{y} + 1.96/\sqrt{n}]$, or $\bar{y} \pm 1.96/\sqrt{n}$.

1.3.3 Hypothesis Testing

A hypothesis test is a method of making statistical decisions about experimental data that answers the question of how well the findings fit the probability that chance factors alone might be responsible. This is done by asking a hypothetical question and performing a test that compares a sample to its population.

To set up a hypothesis test, we begin with a *null hypothesis*. For example, we want to test a cereal-filling machine at a factory (Levine, Stephan, Krehbiel, & Berenson, 2004,

pp. 332–358). Our test is to determine if the machine is pouring out the expected weight of 368 grams into each box of cereal.

We have a sample of 25 cereal boxes, and we want to test if the machine is filling properly based on the average weight from our sample. To do so, we must state a null hypothesis,

$$H_0 : \mu = 368,$$

and an alternative hypothesis as,

$$H_1 : \mu \neq 368.$$

The cereal in the 25 boxes is weighed, and its average (*mean* weight) is found to be 372.5 grams. We know that, on average, the machine has poured too much cereal, some boxes may have too much, others too little. The real question is, "Is 372.5 grams per box *close enough* to consider the cereal machine to be working?"

Before we can say what is close enough, we need to know how much variance to expect from the cereal-filling machine, and we need to establish a threshold of how much difference is acceptable. Because we are using only a sample, it is possible to make a testing error. We are given the population standard deviation $\sigma = 15$.

Hypothesis Errors. There are two kinds of hypothesis errors we can make,

Type I Error This occurs when rejecting a null hypothesis that should not have been rejected. The probability of a Type I error is defined as the *level of significance* which uses the symbol α.

Type II Error This occurs when not rejecting the null hypothesis when we should have. The probability of a Type II error is denoted by the symbol β.

	Actual State of Population	
Decision	H_0 is True	H_0 is False
Reject H_0	Type I Error Probability: α	Correct Decision Probability: $1 - \beta$
Do not Reject H_0	Correct Decision Probability: $1 - \alpha$	Type II Error Probability: β

Table 1.2: Hypothesis Testing Errors

Two-sided hypothesis test divides the α equally on both tails.

A hypothesis test can be one-sided or two-sided, depending upon the alternative hypothesis. A two-sided hypothesis is nondirectional, such as $H_1 : \theta \neq \theta_0$, which we see in Figure 1.4. A one-sided hypothesis is only concerned with one direction or the other. The alternative hypothesis $H_1 : \theta < \theta_0$ is a *left-directional test*, as seen in Figure 1.5, and $H_1 : \theta > \theta_0$ is a *right-directional test*.

Our cereal machine should not pour too much or too little cereal. We have established a two-sided hypothesis test. We will establish our level of significance of .05 or 5%.

Next, we need a *confidence coefficient*, which is denoted by $1 - \alpha$. In our cereal test, the confidence coefficient is .95. Our *region of non-rejection* therefore is .95, which implies

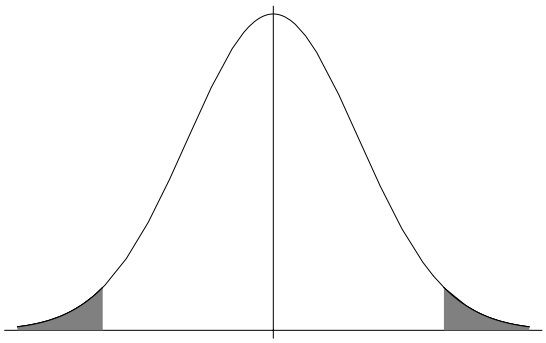

Figure 1.4: Two-Sided Region of Rejection

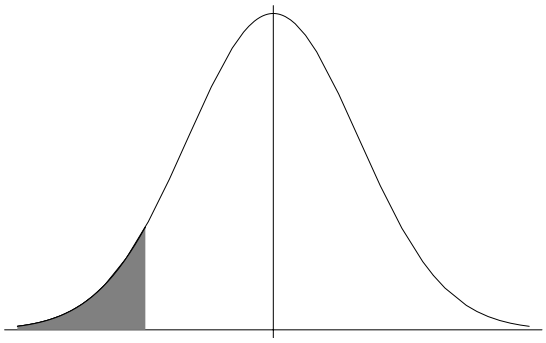

Figure 1.5: One-Sided Region of Rejection

that our *region of rejection* is .025 on either side of the distribution in Figure 1.4 for a total of .05. We must convert our rejection threshold value into a *critical value* by locating where on the normal distribution the region ending at .025 is located. We can obtain the critical value in MATLAB using, `cv=norminv((1-0.025),0,1)`, which gives us 1.96. Since this is a two-tailed test, our critical value is ± 1.96.

Now that we know where the sample mean should lie within the distribution, we need to calculate the *z-test statistic* which determines the sample mean's position relative to the population. The z-test statistic is obtained by,

$$\boxed{z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}.} \tag{1.4}$$

We have the numbers we need to proceed, $\bar{X} = 372.5, \mu = 368, \sigma = 15, n = 25$.

We can use (1.4) to generate a z-test statistic,

$$z = \frac{372.5 - 368}{15/\sqrt{25}} = 1.5.$$

Since our z-test statistic is within the region of non-rejection ($-1.96 < z < 1.96$), we state that there is insufficient evidence to claim that our sample mean is different from the population mean. The difference in the weight of the cereal is "close enough."

1.4 Linear Algebra

To perform matrix operations, We will be using MATLAB, and R, which allow us to perform matrix multiplication and other related operations, but it is important to remember is how to multiply two matrices. For example, matrix $\mathbf{A} \times \mathbf{B}$ where,

The size of a matrix is described by number of rows × columns

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix} \\ \mathbf{AB} &= \begin{bmatrix} 220 & 280 \\ 490 & 640 \end{bmatrix}, \end{aligned} \tag{1.5}$$

or, more generally,

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \\ \mathbf{AB} &= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix}. \end{aligned} \tag{1.6}$$

Only matrices of specific sizes may be multiplied.

The multiplication in (1.5) and (1.6) is allowed because $\mathbf{A}$ is a 2×3 matrix, $\mathbf{B}$ is a 3×2 matrix, and the number of columns in $\mathbf{A}$ must equal the number of rows in $\mathbf{B}$. The expression,

$$\mathbf{A}_{[\mathbf{m}\times\mathbf{n}]} \times \mathbf{B}_{[\mathbf{p}\times\mathbf{q}]}$$

requires that $n = p$. The resulting matrix will be of size $[m \times q]$.

It is important to note that the order the two matrices is multiplied is very important. The result will most likely be very different, if it can be computed at all.

The law of commutativity does *not* apply to matrices.

$$\mathbf{AB} \neq \mathbf{BA}$$

Often, some calculations require a matrix to be *transposed*. This is simply a rotation of columns to rows, as in the example,

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \\ \mathbf{A^T} &= \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix} \end{aligned}$$

Let us examine how MATLAB evaluates (1.5). A data structure, such as an array or matrix, is enclosed within brackets. The data elements may be comma-separated or separated by a space. A semicolon delimits a new row.

```
A=[1 2 3; 4 5 6];
B=[10 20; 30 40; 50 60];
A*B

ans =
   220   280
   490   640
```

Compare (1.5) to the MATLAB syntax above to see how the expression is coded. The semicolon at the end of a line suppresses printing of an expression to the Command Window, but the values are displayed in the Workspace.

2 Introduction to MATLAB and R

Primary Text Reading. `http://www.r-project.org/about.html`
`http://cran.r-project.org/manuals.html`
especially **An Introduction to R** and **R Data Import/Export**

2.1 Statistical Computing

Many analytical methods are highly complex, involving multiple calculations, formulas, and graphics.[1] Statistical computing, also known as computational statistics may also be used to refer to computationally-intensive statistical methods including resampling methods, Markov chain Monte Carlo methods, local regression, kernel density estimation and generalized additive models.

Both MATLAB and R are statistical computing environments; they both have broad support in the quantitative finance community as well as having a large number of libraries or packages that contain specialized functions. In class, we will attempt to be as platform-independent as possible, as well as trying to avoid proprietary data formats, such as Microsoft Excel. All data sets will be comma-delimited, or tab/space delimited so that they can be used by any editor and work equally well in MATLAB, R, or any other statistical software package.

2.1.1 MATLAB

To MATLAB, *everything* is a matrix, including a single value, a *scalar*. MATLAB has matrix math processing built in as well as a rich set of programming features for data access, graphical user interface, and mathematical computation and simulation. Despite its long list of features, MATLAB is very easy to use and can produce meaningful results in a short time span. MATLAB has robust file processing capability, meaning that very large datasets, perhaps amounting to millions of items, pose no difficulty. Having numerous functions already available and tested make MATLAB a good environment for rapid development of models. Writing programs in a language like C++ requires considerably more commitment of time and programming expertise.

There is a wide literature available for users of MATLAB and numerous examples of functions available through the worldwide web. There are hundreds of user contributions at MATLAB Central[2].

MATLAB has a rich set of functions for mathematics, statistics, data analysis, and graphics. Additionally, users may write their own scripts and functions in the form of *M-files*, the programming language of MATLAB. The latest technical documentation on MATLAB is available online[3]. Martinez and Martinez (2008) provides an excellent resource for statistics in MATLAB.

[1] A good resource for exploring more on statistical computing is the American Statistical Association website for **Statistics Computing and Graphics** at, `http://stat-computing.org/`

[2] The MATLAB Central File Exchange is an excellent source of code samples and ideas in various categories of MATLAB applications,
`http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do`

[3] `http://www.mathworks.com/access/helpdesk/help/techdoc/index.html`

This is an example of the Normal Probability Density Function (1.1) in MATLAB. *This is a simple example, and a more complex version already exists in the MATLAB library of functions.*

```
function [p] = MynormPDF(z)
% Normal Probability Density Function
p = 1 / sqrt(2.0 * pi) * exp(-z.^2 / 2.0);
```

To call this function, entering `MynormPDF(0)` into the MATLAB Command Window would display `ans = 0.3989`.

The next example demonstrates how to read Excel workbook data into a MATLAB matrix variable, q. An example of the data is represented in Table 2.1. The file name is `QQQQ.xls` and the data we want is in the Excel sheet named "Daily". Then, we will extract the first 100 items from column 2 of the dataset [1:100, 2], and calculate its standard deviation and its mean. The last line compares element [21, 2] with the 100-item mean that we just calculated. If $q[21, 2] > mean(q[1 : 100, 2])$ then it will return 1 (for *true*), otherwise, it returns 0 (for *false*).

```
q=xlsread('/filepath/data/QQQQ.xls', 'Daily');
mnth=q(1:100,2)
std(mnth)
mean(mnth)
q(21,2)>mean(mnth)
```

Stock Price Data Imported into MATLAB						
Date	Open	High	Low	Close	Volume	Adj. Close
38115	48.03	48.46	47.9	48.21	97037800	48.21
38114	48.24	48.7	48.06	48.4	128058700	48.4
38113	48.94	49.23	47.87	48.04	140008300	48.04
⋮	⋮	⋮	⋮	⋮	⋮	⋮
36959	37.61	37.67	37.08	37.52	97043600	37.14

Table 2.1: Stock Price Finance Data Imported into MATLAB

2.1.2 R Language

The R environment is a language for statistical computing and advanced graphics. It is a GNU project which means it can be downloaded for free.[4] When we refer to the R statistical environment, for convenience we will identify it as, "R language" to mean the entire system of statement syntax, graphics, and packages. Two excellent learning resources are R Development Core Team (2008), the online R documentation, and Crawley (2007), which contains numerous programming examples.

To demonstrate some basic functionality, we will load a file from Ruey Tsay's teaching website, and plot a time series.

[4]To download the latest version of the R environment and any documentation and packages, go to `http://cran.r-project.org/`.

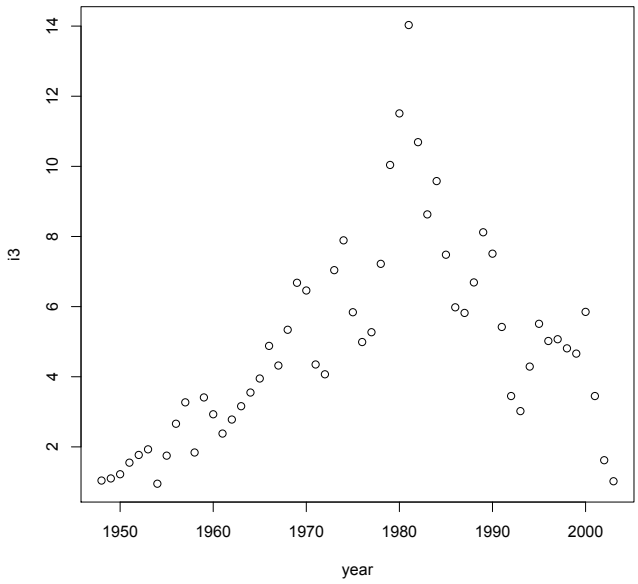

Figure 2.1: Interest Rate Time Series Plot in R

```
# Read in a Ruey Tsay dataset from a web page.
filename="http://faculty.chicagogsb.edu/ruey.tsay/teaching/
     fts2/d-ibmvwewsp6203.txt"
rets<-read.table(filename)
with(rets,{retsts<-ts(V3); plot(retsts,ylab='Returns')})
```

Some of the data that we will be working with has header information, a title at the top of each column. To load these data, we simply append, `header=T`. This example loads a tab-delimited file named `intdef.txt`, generates some summary statistics, and creates the plot in Figure 2.1.

```
# Read a data file with a header.
intrt<-read.table('intdef.txt', header=T)
summary(intrt)
with(intrt,sd(i3))
with(intrt,plot(year,i3))
```

For a simple side-by-side comparison of some basic functions of both MATLAB and R, see Table 2.2.

2.2 Loading Data

MATLAB. There are a few ways to load data. The easiest is `load`. To create an array variable name `plainTS` from a single-column text file `plain.txt`, we need the statement,

```
plainTS=load('/path/plain.txt');
```

Function	MATLAB	R
Expansion	Toolboxes	Packages
Programming	M code	R code
File Input	`load`	`read.table`
Getting Help	`help plot`	`?plot`
Matrices	rows, then columns	columns, then rows
Change Directory	`cd('/path')`	`setwd('/path')`

Table 2.2: Basic Functions in MATLAB and R

The semicolon at the end of the statement suppresses any output to the screen. Leaving out the semicolon would display each line of text as it read into MATLAB. If our data set had multiple columns which were separated by spaces or tabs, we could use the same syntax, and we would still have a single variable, but with multiple columns. To refer to a single item in the array, such as the 27th observation, we address it `plainTS(27)`. If we had multiple columns, we want to refer to column 2, we simply use `plainTS(:,2)`.

Importing delimited data sets requires `dlmread`, and is not much more difficult. To read only the numerical data from the five-column tab-delimited file, `SP-MidCap4.txt`

```
sp_mid=dlmread('/path/SP-MidCap4.txt', '\t', 0, 1);
```

The `'\t'` is the delimiter, which in our case is the tab character. To use comma-delimited text, we replace that with ',' or whichever marker we require. We can specify the starting column and row number of the data. MATLAB uses a zero-based index, so to get *all rows* but columns 1 to whichever is the last column, we enter 0 and 1, respectively. Once loaded, we have a variable named `sp_mid`. To refer to the 120th row, and 3rd column data item, `sp_mid(120,3)`.

Now try to load `SP-MidCap.csv` which is comma-delimited and has the first row consisting of column names. We do not want to import the first column, which are non-numerical dates, nor do we want the column names, since they too are non-numerical.

```
sp_mid=dlmread('/path/SP-MidCap.csv', ',', 1, 1);
```

There are other methods in MATLAB for loading data and writing results back to a file, but these two will work fine for most purposes. We have also only concerned ourselves with numerical data at this point. We address date conversions in Section 2.3.

R Environment. Just as we explored one method of loading tab-delimited data and one method for comma-delimited data in MATLAB, we will discuss two ways in the R environment. One advantage is that we may load column names with our data, something we are unable to do in MATLAB. We will load all data, *including column names*, from the five-column tab-delimited file, `SP-MidCap3.txt` using `read.table`.

```
mid400<-read.table('/path/SP-MidCap3.txt', header=T)
```

We have two ways of using the column names from a data set. The first method is easier, but can result in name collision when multiple data sets are loaded. Now that we have `SP-MidCap3.txt` stored into a variable named `mid400`, we can assign columnn names using `attach(mid400)`. To see the names assigned to columns, we can use `names(mid400)`.

Name collision is the result of two processes or data sets using the same name.

We can refer directly to the column of data by its name, such as when we want the mean of the column `Close`, by typing `mean(Close)`. The problem is that these column names are global, which is generally a bad idea in the event that we want other data with the same name.

The second method of using the column names from a data set is preferred, and is only slightly more complex. We do not use the `attach` statement, but instead refer to a column by variable name and column name using `with`. We can compute the mean of the column `Close`, by typing `with(mid400, mean(Close))`. We still have the column names in the variable, and we can view the first row of data by typing `mid400[1,]`.

Reading comma-separated data is as easy as reading table data – the tab or space delimited files we just discussed. This time, we use the statement, `read.csv`.

```
mid400<-read.csv('/path/SP-MidCap.csv', header=T)
```

Once the data set is loaded, all functions work exactly the same as before.

2.3 Date Handling

Date Formatting. Most computer systems store a date as the number of days or number of minutes since a specific date. Some Unix systems store the date as number of days since January 1, 1970, while some software may use a different date. This "epoch date" may cause confusion when reading and writing dates into a statistical package. For instance, what is the date value of 733671? That depends on which system created that date. This particular date value came from MATLAB, where it represents September 20, 2008, but in Excel (with 1904 Date System enabled) it is September 21, 3912, and it is September 20, 3908 with 1904 Date System disabled.

Be aware of the computer epoch date of the system you import and export into.

The spreadsheet software in OpenOffice for X11 uses a date value of 1 for December 31, 1899 (*which is technically January 0, 1900*), but Microsoft Excel uses January 1, 1904, the Lotus 1-2-3 "standard." In Excel, the "1904 Date System" can be turned off, so that date value of 1 becomes January 1, 1900.

MATLAB. Dates must be stored as numbers in MATLAB, which measures the number of days since January 1, 0. There are functions to convert a string date into a number. For example the date, 20 September 2008, could be entered a few ways, such as,

```
my_date=datenum('2008-09-20', 'yyyy-mm-dd')
my_date=datenum('20-Sep-2008')
```

which stores the number 733671 in the variable `my_date`. This makes date arithmetic easier, but, that number does not tell us much once stored since we like seeing months, days, and years. We can convert that number to a date string using `datestr(my_date)`. If we want the month, day, and year as a number, we have `[yr,mo,day]=datevec(my_date)`, which populated three variables as described in the left-hand side.

R Environment. Our data set `mid400` has a column named `Date` which is formatted `yyyy-mm-dd`. This works well for date conversion and performing date arithmetic. To convert the item into a "time" data type, we need `as.POSIXlt` so that converting the date in column 1, row 1 is done by `as.POSIXlt(mid400[1,1])`. To add 7 days to that date is as easy as `as.POSIXlt(mid400[1,1]) + (24*60*60*7)`.

R also has a full set of objects that can display and manipulate dates.

```
today<-Sys.Date()
next.week<-today+7
format(next.week, "%d %b %Y")
```

2.4 Statistical Operations

MATLAB. There is a wide variety of statistical functions available in MATLAB. As a very brief introduction, we will look at some basic descriptive statistics. Using our `sp_mid` data set, we compute the mean and standard deviation of the set, using `mean` and `std`, respectively. We can summarize the entire data set, column-by-column with `mean(sp_mid)`, which returns a matrix that has one row, and has a mean for each column.

If, for example, we want the standard deviation of the most recent 200 opening prices (column 1 of the data), we need to specify the range using `std(sp_mid(1:200, 1))`.

R Environment. There is also an extensive variety of statistical functions available in R. As a very brief introduction, we will look at some basic descriptive statistics. Using our `mid400` data set, we compute the mean and standard deviation of the set, using `mean` and `sd`, respectively. We can summarize the entire data set, column-by-column with `mean(mid400)`, which returns the mean for each column, except for `Date`, which is not numeric.

If, for example, we want the standard deviation of the most recent 200 opening prices (column 2 of the data), we need to specify the range using `sd(mid400[1:200, 2])`.

2.5 Functions

MATLAB. Sometimes, the built-in functionality of MATLAB is not enough to handle specific requirements. In order to extend the environment, users may write functions. Here is an example of a simple moving average.

```
function y = movavg(x,p)
% Simple Moving Average written as a MATLAB demo
% by Andrew P. Acosta
% This works with a COLUMN of numbers only.
y=zeros(size(x));
for n=p:size(x,1)
    y(n)=mean(x(n-p+1:n));
end
```

The elements of a function are the word `function` as the first keyword, followed by the variable(s) returned by the function. A MATLAB function can return more than one variable, provided that the list is enclosed in brackets (*e.g.* `[x,y,z]`). Comments in

MATLAB begin with a percent sign (`%`). The contents of the comments placed between the first line of the function and the first line of instructions will display when a user types `help` followed by the function name. It is a good way of documenting user-defined code.

What follows is the actual work of the function. The return variable(s) must be set somewhere in this segment. The three-day moving average for the first ten observations is called by, `movavg(sp_mid(1:10,1), 3)`.

R Environment. Since R also is a programming language, there is a specific syntax for writing functions. Here is an example of a simple moving average.

```
movavg<-function(x,p) {
    # Simple Moving Average written as an R demo
    # by Andrew P. Acosta
    y<-numeric(length(x))
    for(n in p:length(x)) {
        y[n]<-mean(x[(n-p+1):n])
        }
    y
}
```

The elements of a function are the word `function` after assignment to the return variable. Comments in R begin with a pound or hash symbol sign (`#`).

What follows is the actual work of the function. The return variable must be set somewhere in this segment. The three-day moving average for the first ten observations is called by, `movavg(mid400[1:10,2], 3)`.

2.6 Graphing

MATLAB. There is an extensive collection of graphical features in MATLAB. One of the most used functions is `plot`, which creates an X,Y plot which is depicted in Figure 2.2.

To work with the data set as a MATLAB time series, we will explore the `timeseries` and `tscollection` objects.

R Environment. It is no surprise that R also has an extensive collection of graphical feaures. One of the most used functions is `plot`, which creates an X,Y plot which is depicted in Figure 2.3.

To work with the data set as an R time series, we will explore the `ts` object.

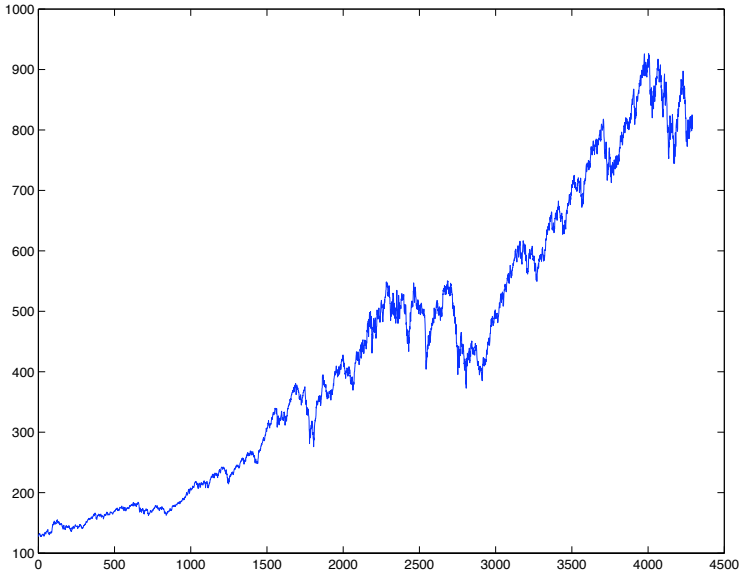

Figure 2.2: Plot of `SP-MidCap-sort.csv` Data Set using MATLAB statement `plot(sp_mid(:,1))`.

2.7 Technical Issues

As with any data, there could be some level of work required to "clean" the data, or to make it presentable for analysis. This cleanup effort will depend greatly upon the quality and source of the data, as well as the type of analysis and software.

Some issues arise at the source of financial time series data, and generally become typical problems solved through information technology. Often, they are simple matters of parsing a text file, rearranging rows and columns, or validating source data. Such rudimentary tasks, as well as more complex pre-analytical text formatting, are solved by scripting languages like Perl or Python.

Be aware that some financial time series data may have the most recent data first, yet your graphics and numerical analysis expects oldest data first. In this case, you simply need to reverse the order of the observations in the data set.

Some data conversion is more complicated. For example, if a process is receiving XML-formatted data, it is worth knowing how the data are described by the format. The same applies to reading and writing HTML data. MATLAB and R have capabilities of handling formatted data, but it is still important to know the underlying formatting issues.

In other cases, there may be missing data, or data that are simply incorrect. How would you react to a price series such as {23.38, 26.73, **95.69**, 26.55, 26.47, 26.63}? Do you explain the 95.69 that seems to be out of place? Does it belong there? Is it an error? How do you know? How do you locate these kinds of numbers in a large data set?

Even if the data is correct on some superficial level, there remain several other issues such as: of normalizing a variable's range, proper sampling methods, and capturing patterns. Pyle (1999) writes in detail about these concerns as well as many others.

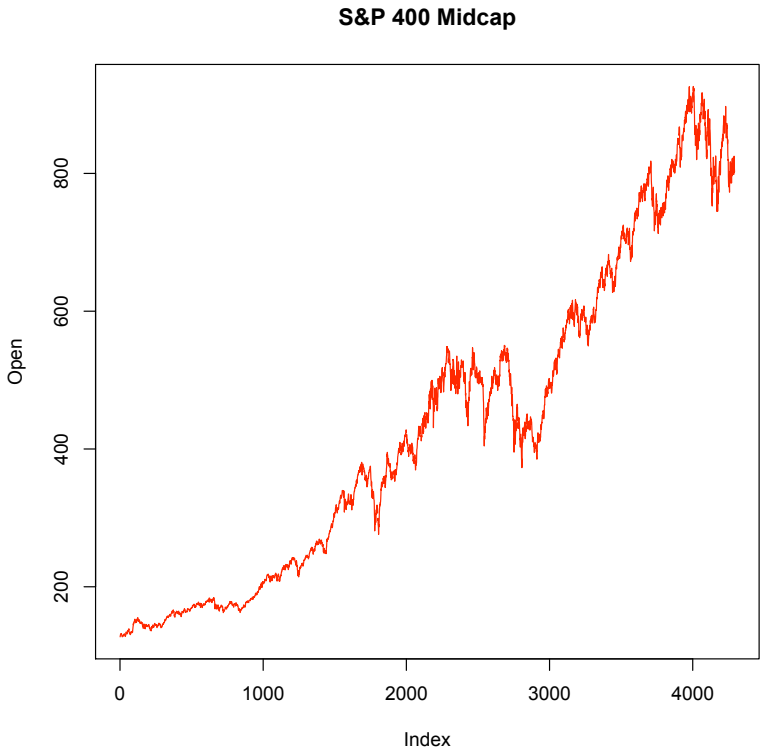

Figure 2.3: Plot of `SP-MidCap-sort.csv` Data Set using R programming. `with(mid400, plot(Open, type="l", col="red", ylab="Open", main="S&P 400 Midcap"))`.

No doubt, you will encounter errors in data, or require them to be modified in some way before or after your analysis. Being able to do this is vital, and the responsibility should not be relegated to a lesser importance.

3 Getting Started with Financial Time Series

Primary Text Reading. Tsay (2005, chap. 1)

3.1 Asset Returns Over Time

Let P_t be the price of an asset at time t, and we assume no dividends are paid, then the one-period simple gross return is,

$$1 + R_t = \frac{P_t}{P_{t-1}} \quad \text{which is} \quad P_t = P_{t-1}(1 + R_t). \tag{3.1}$$

The one-period simple return is simply the difference in value between two observations divided by its initial value,

$$\begin{aligned} R_t &= \frac{P_t}{P_{t-1}} - 1 \\ &= \frac{P_t - P_{t-1}}{P_{t-1}}. \end{aligned} \tag{3.2}$$

The multiperiod simple return takes into account a series of one-period simple returns, thus making it the product of all asset return observations,

$$\begin{aligned} 1 + R_t[k] = \frac{P_t}{P_{t-k}} &= \frac{P_t}{P_{t-1}} \times \frac{P_{t-1}}{P_{t-2}} \times \cdots \times \frac{P_{t-k+1}}{P_{t-k}} \\ &= (1 + R_t)(1 + R_{t-1}) \cdots (1 + R_{t-k+1}) \\ &= \prod_{j=0}^{k-1} (1 + R_{t-j}). \end{aligned} \tag{3.3}$$

The k-period simple gross return is just the product of the k one-period simple gross returns involved. This is a compound return. The k-period simple net return is $R_t[k] = (P_t - P_{t-k})/P_{t-k}$.

Day	Price
1	37.84
2	38.49
3	37.12
4	37.60
5	36.30

Table 3.1: Simple Return Closing Prices

Using Table 3.1, what is the simple return from day 1 to day 2?

$$R_2 = \frac{38.49 - 37.84}{37.84} = 0.017.$$

What is the simple return from day 1 to day 5?

$$R_5(4) = \frac{36.30 - 37.84}{37.84} = -0.041.$$

Continuous Compounding. Extending (3.2), if we can continually reduce the number of compounding periods, we have a continuous function of e,

$$A = Ce^{r \times n}. \tag{3.4}$$

For example, let us take a currency amount, $C = 100$ at a continuously compounded rate, $r = .05$ for 3 years, $n = 3$.

$$A = 100 \exp(.05 \times 3) = 116.1834$$

If we take the basic function for interest payment for rate r and compound to n periods per year, we have,

$$\left(1 + \frac{r}{n}\right)^n.$$

For a single interest payment in year, $n = 1$. Two payments per year, $n = 2$, *etc.* As we increase the compounding frequency n, we approach a limit.

$$\begin{aligned}
1.05 &= \left(1 + \frac{.05}{1}\right)^1 \\
1.05063 &= \left(1 + \frac{.05}{2}\right)^2 \\
1.05095 &= \left(1 + \frac{.05}{4}\right)^4 \\
1.05116 &= \left(1 + \frac{.05}{12}\right)^{12} \\
1.05127 &= \left(1 + \frac{.05}{365}\right)^{365} \\
\\
1.05127 &= \exp(.05)
\end{aligned}$$

Proof. Increase the frequency of compounding n while applying $1 + \frac{1}{n}$.

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

□

Continuously Compounded Returns. The logarithm of the simple gross return of an asset is the continuously compounded return, also known as the *log return*,

$$r_t = \ln(1 + R_t) = \ln \frac{P_t}{P_t - 1} = p_t - p_t - 1, \tag{3.5}$$

where $p_t = \ln(P_t)$.

Multiperiod Log Returns. The sum of continuously compounded one-period returns is the multiperiod return.

$$\begin{aligned} r_t(k) &= \ln[1 + R_t(k)] \\ &= \ln[(1 + R_t)(1 + R_{t-1}) \cdots (1 + R_{t-k+1})] \\ &= \ln(1 + R_t) + \ln(1 + R_{t-1}) + \cdots + \ln(1 + R_{t-k+1}) \\ &= r_t + r_{t-1} + \cdots + r_{t-k+1}. \end{aligned}$$

Examples. What is the *log return* from day 1 to day 2?

$$r_2 = \ln(38.49) - \ln(37.84) = 0.017.$$

What is the *log return* from day 1 to day 5?

$$r_5(4) = \ln(36.3) - \ln(37.84) = -0.042.$$

3.2 Moments of Random Variables

The ℓth moment of a continuous random variable X is defined as

$$m_\ell^{'} = E(X^\ell) = \int_{-\infty}^{\infty} x^\ell f(x) dx \tag{3.6}$$

where E is the expectation and $f(x)$ is the probability density function of X. The first moment is called the *mean* or *expectation* of X. It measures the central location of the distribution. We denote the mean of X by μ_x. The ℓth central moment of X is

$$m_\ell = E[(X - \mu_x)^\ell] = \int_{-\infty}^{\infty} (x - \mu_x)^\ell f(x) dx \tag{3.7}$$

Tsay (2005, p. 8). With a random variable $X = \{x_1, \ldots, x_T\}$ of T observations, we can compute the sample mean,

$$\hat{\mu_x} = \frac{1}{T} \sum_{t=1}^{T} x_t \tag{3.8}$$

sample variance,

$$\hat{\sigma}_x^2 = \frac{1}{T-1} \sum_{t=1}^{T} (x_t - \hat{\mu}_x)^2 \tag{3.9}$$

sample skewness,

$$\hat{S}(x) = \frac{1}{(T-1)\hat{\sigma}_x^3} \sum_{t=1}^{T} (x_t - \hat{\mu}_x)^3 \tag{3.10}$$

sample kurtosis,

$$\hat{K}(x) = \frac{1}{(T-1)\hat{\sigma}_x^4} \sum_{t=1}^{T} (x_t - \hat{\mu}_x)^4. \tag{3.11}$$

Mean and variance of returns are important because they communicate long-term return and risk, respectively. The symmetry of the distribution has important implications in holding short or long financial positions and in risk management. Skew and kurtosis are important to volatility forecasting, efficiency in estimation and tests.

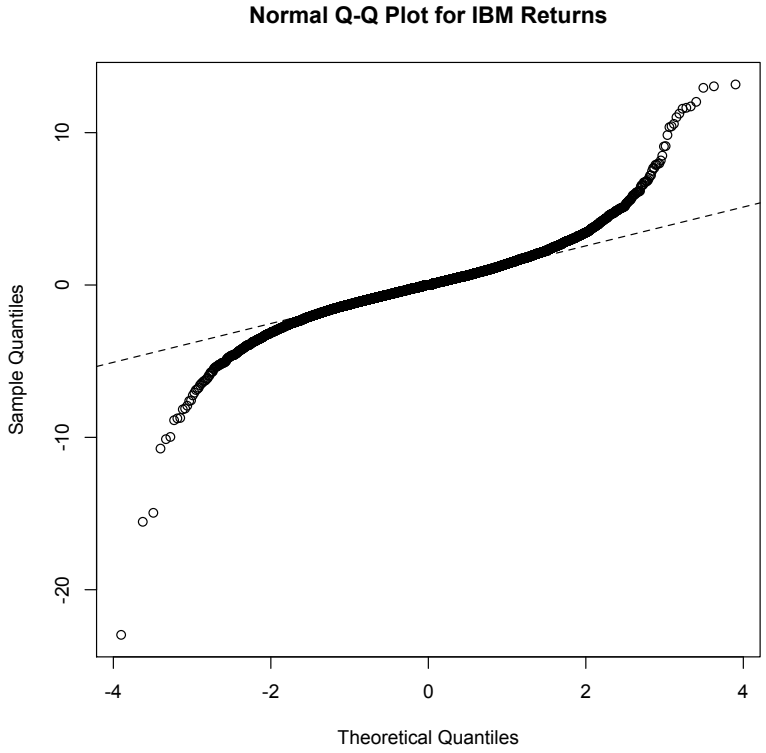

Figure 3.1: By looking at a quantile-quantile plot of our data set, we see that returns are not normally distributed. The tails deviate sharply from normality.

Examining Distribution of Returns. Using R, we load in the data set and examine column 2, which is the IBM returns data. To compare the returns to a normal distribution, we look at Figure 3.1. The returns plotted over time are depicted in Figure 3.2.

```
rets<-read.table("d-ibmvwewsp6203")
attach(rets)
retsts<-ts(V2)
plot(retsts,ylab="Returns")

qqnorm(V2,main="Normal Q-Q Plot for IBM Returns")
qqline(V2,lty=2)

plot(rets[,1]/10000, ibm, ylab="IBM Returns", xlab="Year")
```

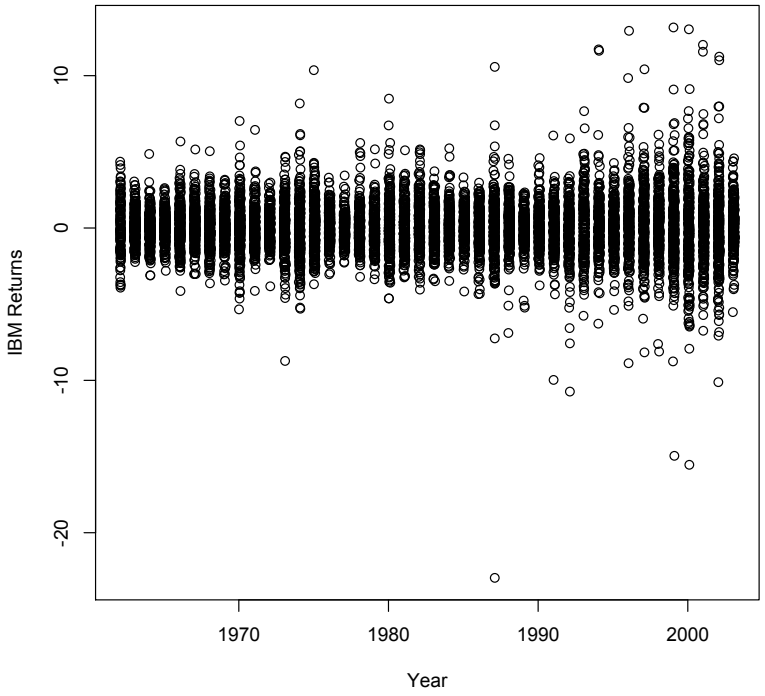

Figure 3.2: IBM returns become more varied over time, deviating further from zero.

Normal Distribution. For the sake of simplicity, returns are often considered to be *normally distributed*. However, the lower bound of a simple is -1, total loss of the asset. The normal distribution has no lower bound. Also, multiperiod returns (*the product of one-period returns,* (3.3) *is a multiperiod return*) of a normally distributed returns are themselves not normally distributed. Additionally, asset returns observed in reality have positive excess kurtosis, and do not fit the normal distribution.

Lognormal Distribution. Instead of assuming a normal distribution, we could assume that the log returns r_t of an asset are independent and identically distributed (*iid*) as normal with mean μ and variance σ^2, which makes simple returns iid lognormal random variables, and mean and variance are,

$$E(R_t) = \exp\left(\mu + \frac{\sigma^2}{2}\right) - 1, \quad \text{Var}(R_t) = \exp(2\mu + \sigma^2)[\exp(\sigma^2) - 1].$$

The mean and variance of the log returns r_t when m_1 and m_2 are the mean and variance of the simple return R_t

$$E(r_t) = \ln\left(\frac{m_1 + 1}{\sqrt{1 + m_2/(1 + m_1)^2}}\right), \quad \text{Var}(r_t) = \ln\left(1 + \frac{m_2}{(1 + m_1)^2}\right).$$

4 Linear Time Series Analysis

Primary Text Reading. Tsay (2005, chap. 2)

A collection of asset returns, such as the log returns of an asset (3.5) is a *linear time series*. Some of the elements of this analysis are: stationarity, dynamic dependence, autocorrelation function, modeling, and, forecasting. Models include:

1. simple autoregressive (AR)
2. simple moving average (MA)
3. mixed autoregressive moving-average (ARMA)
4. seasonal models
5. unit-root nonstationary
6. regression models with time series errors

4.1 Stationarity

Strict stationarity occurs when joint distributions $P(X \cap Y)$ are time-invariant, or in other words, $X : (r_{t_1}, \ldots, r_{t_k})$ is identical to $Y : (r_{t_{1+t}}, \ldots, r_{t_{k+t}})$ for all t. A weak stationarity exists if the mean of r_t and the covariance between r_t and $r_{t-\ell}$ are time-invariant, where ℓ is an arbitrary integer. What this means in practice is that asset values fluctuate with constant variation around a fixed level so that we can make inferences about future observations. The mean, or expectation, of returns is therefore $\mu = E(r_t)$ and the variance of returns is $\text{Var}(r_t) = E[(r_t - \mu)^2]$.

4.2 Correlation and ACF

The correlation coefficient between two random variables X and Y is

$$\boxed{\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y},}$$

where μ_X and μ_Y are expected values of X and Y respectively, and σ_X and σ_Y are its standard deviations. Thus, the sample correlation is

$$\begin{aligned} r_{xy} &= \frac{\sum x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y} \\ &= \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\,\sqrt{n\sum y_i^2 - (\sum y_i)^2}}. \end{aligned} \tag{4.1}$$

Autocorrelation Function (ACF). When we want to examine the linear dependence between r_t and its previous values $r_{t-\ell}$ we are interested in the lag-ℓ *autocorrelation* of the series r_t. The extent to which the series exhibits autocorrelation can help us determine how

Existence of serial correlations implies that the return is predictable, indicating market inefficiency.

much of today's asset value has been determined by previous values. We could perform a hypothesis test for zero serial correlations, which would imply market efficiency,

$$\begin{aligned} H_0 : r_{XY} &= 0 \\ H_a : r_{XY} &\neq 0. \end{aligned}$$

Our null hypothesis H_0 is that this market is efficient, and our alternative hypothesis H_a is that this market is *not* efficient. Some sources of serial correlations found in financial time series are,

- Nonsynchronous trading (Section 7)
- Bid-ask bounce (Section 7)
- Risk premium, *etc.* (Section 5)

Thus, significant sample ACF does not necessarily imply market inefficiency.

Portmanteau test. To look for autocorrelation, we will run a *portmanteau test*, which tests whether any of a group of autocorrelations of a time series are different from zero. Two portmanteau tests we will use are the Box-Pierce test and the Ljung-Box test. The Ljung-Box test can be defined as
H_0: The data are random.
H_a: The data are not random.

The Ljung-Box test statistic is calculated as

$$Q_{LB} = n(n+2)\sum_{k=1}^{s} r_k^2/(n-k). \tag{4.2}$$

where,
n = number of observations
s = number of coefficients to test autocorrelation
r_k = autocorrelation coefficient (for lag k).

If the sample value of Q_{LB} exceeds the critical value of a chi-square distribution with s degrees of freedom, then at least one value of r is statistically different from zero at the specified significance level. The null hypothesis is that none of the autocorrelation coefficients up to lag s is different from zero.

We will use R to examine autocorrelation graphically, and then to perform a test for randomness, or lack of serial correlation of data.

```
rets<-read.table("aapl.txt")
s1<-acf(rets$Log.Return, lag=15, main="Autocorrelation of AAPL")
s1$acf
Box.test(rets, lag=15)
Box.test(rets, lag=15, type="Ljung")
```

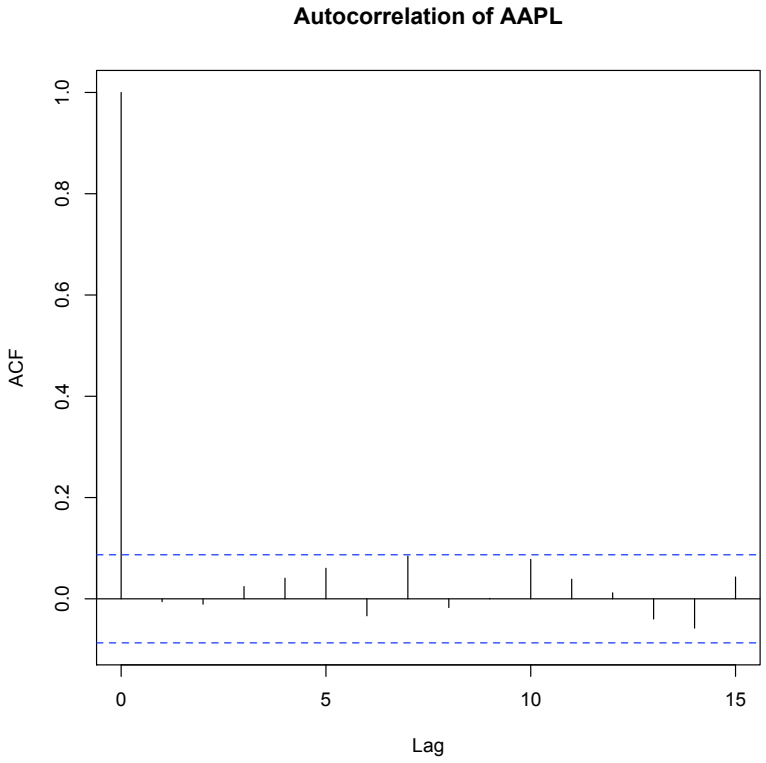

Figure 4.1: ACF function of AAPL log returns with 15-day lag.

The dotted lines in Figure 4.1 represent the 95% confidence interval of the ACF function. Notice that the vertical lines do not cross the dotted line boundary, which would indicate statistical significance from zero.

The first `Box.test` gives us the *Box-Pierce test* of our log returns with χ-squared = 14.7735, df = 15, p-value = 0.4679. The second `Box.test` gives us the *Ljung-Box test*, χ-squared = 15.0983, df = 15, p-value = 0.4444.

Using R to generate the appropriate χ-squared statistic to determine the critical region, `qchisq(p=0.4444, df=15)`, we get 13.60595. The hypothesis of randomness is rejected if $Q_{LB} > \chi^2(p, df)$. In this case, since our p-value is not more extreme than our α of 0.05, we do not reject the hypothesis that the data are random.

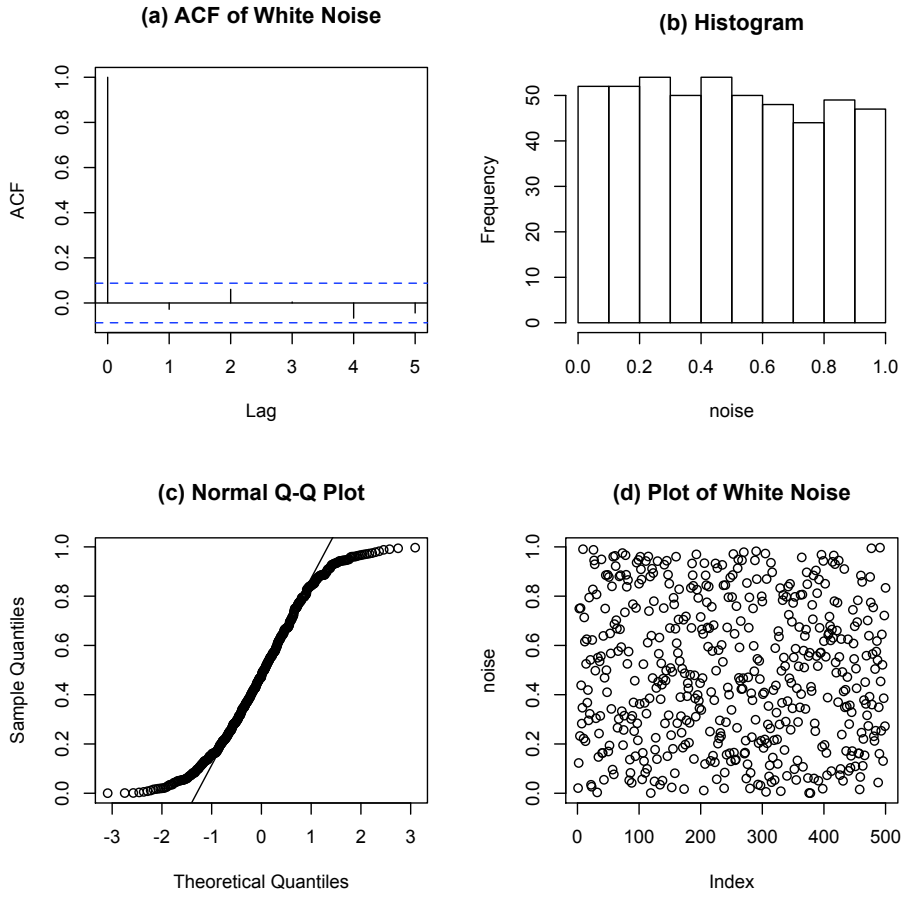

Figure 4.2: White noise has an ACF of zero as seen in (a). We see in (b) a nearly uniform distribution, and in (c) we observe that the tails are not normally distributed. The values are plotted in (d).

White Noise. A time series r_t that is a sequence of iid random variables with a finite mean and variance is *white noise*. The serial correlation of any two points in r_t, regardless of lag, is zero, thus ACF values of white noise is zero. Using R to demonstrate, we can see the results in Figure 4.2. In some circumstances, log returns of asset values can be considered white noise. With this assumption, we can create linear time series models.

```
noise=runif(5000) # Random numbers from Uniform Distribution
layout(rbind(c(1,2), c(3,4)))
acf(noise,lag=5, main="(a) ACF of White Noise")
hist(noise, main="(b) Histogram")
qqnorm(noise, main="(c) Normal Q-Q Plot")
qqline(noise)
plot(noise, main="(d) Plot of White Noise")
```

Thus, we can implement a linear time series as

$$r_t = \mu + \sum_{i=0}^{\infty} \psi_i a_{t-i} \tag{4.3}$$

where μ is the mean of r_t, $\psi_0 = 1$, and $\{a_t\}$ is a sequence of iid random variables.

4.3 Simple Autoregressive Models

Within a linear time series, there may exist a certain random movement, described as white noise a_t with a mean of zero and variance of σ_a^2. This allows us to create a simple model

$$r_t = \phi_0 + \phi_1 r_{t-1} + a_t, \tag{4.4}$$

which follows the form of simple linear regression, where r_t is the dependent variable, and r_{t-1} is the independent variable. Equation (4.4) is the autoregressive (AR) model of order 1, also known as AR(1). We can generalize (4.4) to the AR(p) model

$$r_t = \phi_0 + \phi_1 r_{t-1} + \cdots + \phi_p r_{t-p} + a_t. \tag{4.5}$$

AR Models. We assume that (4.4) has weak stationarity, and $E(r_t) = \mu$, $\text{Var}(r_t) = \gamma_0$, and $\text{Cov}(r_t, r_{t-j}) = \gamma_j$, where μ and γ_0 are constant and γ_j is a function of j, not t. This gives us a weakly stationary AR(1) model

$$\text{Var}(r_t) = \gamma_0 = \frac{\sigma^2}{1 - \phi_1^2}, \text{ and } \gamma_\ell = \phi_1 \gamma_{\ell-1}, \text{ for } \ell > 0$$

using the results of

$$\gamma_\ell = \begin{cases} \phi_1 \gamma_1 + \sigma_a^2 & \text{if } \ell = 0, \\ \phi_1 \gamma_{\ell-1} & \text{if } \ell > 0, \end{cases}$$

where we use $\gamma_\ell = \gamma_{-\ell}$.

4.3.1 Partial Autocorrelation Function (PACF)

Before working with an AR(p) time series, it is necessary to find out what p represents. This process is called *order determination* of an AR model, and we work our out in terms of order p,

$$\begin{aligned}
r_t &= \phi_{0,1} + \phi_{1,1} r_{t-1} + e_{1t}, \\
r_t &= \phi_{0,2} + \phi_{1,2} r_{t-1} + \phi_{2,2} r_{t-2} + e_{2t}, \\
r_t &= \phi_{0,3} + \phi_{1,3} r_{t-1} + \phi_{2,3} r_{t-2} + \phi_{3,3} r_{t-3} + e_{3t}, \\
&\vdots \qquad\qquad \vdots \\
r_t &= \phi_{0,n} + \phi_{1,n} r_{t-1} + \phi_{n+1,n+2} r_{t-1-n} + \cdots + e_{nt}.
\end{aligned}$$

These models are solved as multiple linear regressions with the lag period increasing with each equation. Thus, we can see the added contribution of each lag coefficient. We compare ACF and PACF in Figure 4.3.

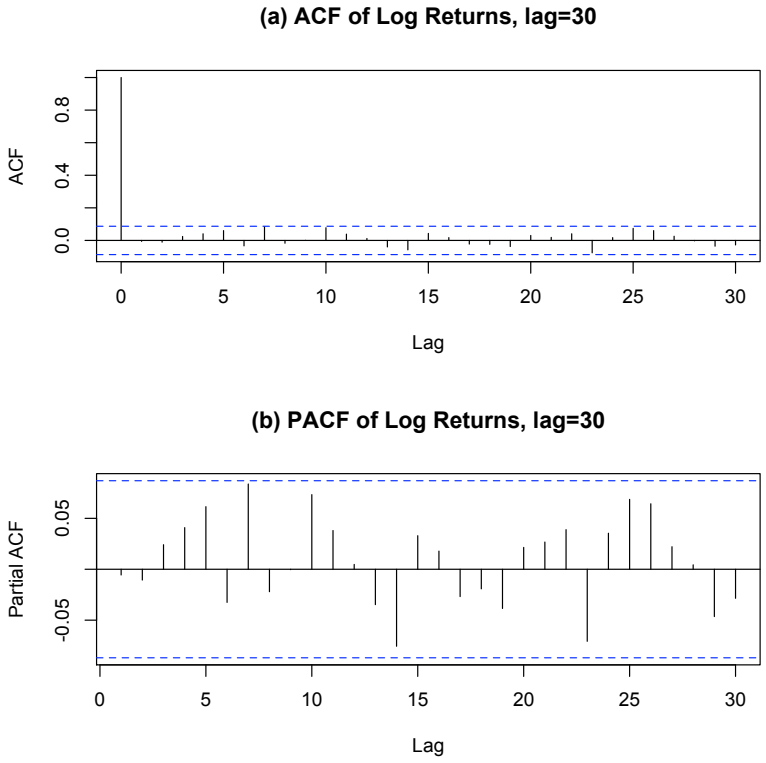

Figure 4.3: (a) ACF function of AAPL log returns with 30-day lag. (b) PACF function of AAPL log returns with 30-day lag.

4.3.2 Information Criterion Function

Another way to determine the order p of an AR process is by using a likelihood function such as the *Akaike information criterion (AIC)*, which measures goodness of fit of a model, and penalizes excessive use of parameters k.

$$AIC = \frac{-2}{n} \ln(\text{likelihood}) + \frac{2}{n} \times k, \tag{4.6}$$

where the likelihood function is evaluated at the maximum likelihood estimates,
n = sample size,
k = number of parameters.

4.3.3 Goodness of Fit

A conventional statistic to measure goodness of fit of a stationary model is r^2,

$$r^2 = 1 - \frac{\sum_{t=p+1}^{T} \hat{a}_t^2}{\sum_{t=p+1}^{T} (r_t - \bar{r})^2} \tag{4.7}$$

where $\bar{r} = \left(\sum Tt = p + 1r_t\right)/(T-p)$. Yet, this only applies to a stationary time series. As a replacement, the *adjusted* r^2 is offered,

$$\begin{aligned} \text{Adj } r^2 &= 1 - \frac{\text{Variance of residuals}}{\text{Variance of } r_t} \\ &= 1 - \frac{\hat{\sigma}_a^2}{\hat{\sigma}_r^2}. \end{aligned}$$

4.3.4 Forecasting

Much of the purpose of financial time series is forecasting. The AR(p) model in (4.5) can be used to model a forecast. If we at time index h and we forecast $r_{h+\ell}$, where $\ell \geq 1$, then h is our *forecast origin* and ℓ is our *forecast horizon*. Let $\hat{r}_h(\ell)$ be the forecast of $r_{h+\ell}$ using the minimum squared error loss function and F_h is the collection of information at the forecast origin h. Then, the forecast $\hat{r}_k(\ell)$ is chosen such that

$$E\{[r_{h+\ell} - \hat{r}_h(\ell)]^2 | F_h\} \leq \min_g E[(r_{h+\ell} - g)^2 | F_h],$$

where g is a function of he information available at time h (inclusive), that is, a function of F_h. We referred to $\hat{r}_h$ as the ℓ-step ahead forecast of r_t, at the forecast origin h.

The ϕ parameters are to be estimated from the data.

1-Step Ahead Forecast. Using (4.5), we see

$$r_{h+1} = \phi_0 + \phi_1 r_h + \cdots + \phi_p r_{h+1-p} + a_{h+1}.$$

Under the minimum squared error loss function, the point forecast of r_{h+1} given that $F_h = \{r_h, r_{h-1}, \ldots\}$ is the conditional expectation

$$\hat{r}_h(1) = E(r_{h+1}|F_h) = \phi_0 + \sum_{i=1}^{p} \phi_i r_{h+1-i},$$

and the associated forecast error is

$$e_h(1) = r_{h+1} - \hat{r}_h(1) = a_{h+1}.$$

The process can be extended to 2-step ahead and beyond by simple extension. The conditional expectation and forecast error are also adjusted accordingly.

Multistep Ahead Forecast. A generalization of 1-step ahead forecast is an ℓ-step ahead forecast

$$\hat{r}_h(\ell) = \phi_0 + \sum_{i=1}^{p} \phi_i \hat{r}_h(\ell - i), \tag{4.8}$$

where $\hat{r}_h(i) = r_{h+i}$ if $i \leq 0$. This forecast can be computed recursively using forecasts $\hat{r}_h(i)$ for $i = \{1, \ldots, \ell - 1\}$. The ℓ-step ahead forecast error is $e_h(\ell) = r_{h+\ell} - \hat{r}_h(\ell)$. It can be shown that for a stationary AR(p) model, $\hat{r}_h(\ell)$ converges to $E(r_t)$ as $\ell \to \infty$, meaning that for such a series long-term point forecast approaches its unconditional mean. This property is *mean reversion*, and appears in literature, notably in interest rate studies.

In an AR(1) model, the speed of mean reversion is measured by the *half-life* defined as $k = \ln(0.5/|\phi_1|)$, which is defined as the number of periods required for the magnitude of the forecast to become one-half of the forecast origin. The variance of the forecast error approaches the unconditional variance of r_t.

We see in Figure 4.4 a plot of 176 observations of U.S. GNP changes. In panel (a), it appears that although the changes fluctuate, they tend to return to a central point indicated by the red dashed line, a point of mean reversion. The lag is changed from lag-1 to lag-2, and we observe a slight upward drift, which is consistent with our seeing that panel (a) has a mean slightly greater than zero (0.007741, actually). Next, we look for the presence of autocorrelation. The R code to examine GNP changes and ACF is very simple.

```
gnpdata=read.table("q-gnp4791.txt")
GNP=gnpdata[,1]
par(mfcol=c(2,2)) # put 4 plots on one page
plot(GNP,type="l",main="(a)")
abline(mean(GNP),0, col="red",lty=4)
plot(GNP[1:175], GNP[2:176], main="(c)")
plot(GNP[1:174], GNP[3:176], main="(b)")
acf(GNP, lag=12, main="(d)")
```

4.4 Simple Moving-Average Models

Another type of financial time series model is the *moving average* (MA) model. Initially, an MA model is understood as an AR model of infinite order

$$r_t = \phi_0 + \phi_1 r_{t-1} + \cdots + \phi_\infty r_\infty.$$

This is unrealistic because we have an infinite number of parameters. We will replace it with an MA(ℓ) model of finite order, specifically, ℓ

$$X_t = \varepsilon_t + \sum_{i=1}^{\ell} \theta_i \varepsilon_{t-i}.$$

MA Models. MA models are always weakly stationary because they are finite linear combinations of a white noise sequence for which the first two moments are time-invariant. To discover the order of an MA model, we may use an ACF function. We see from Figure 4.3 what the 30-day lag looks like with AAPL log returns. Using that lag, we plot the log returns, its mean, and a 30-day moving average is superimposed in blue in Figure 4.5.

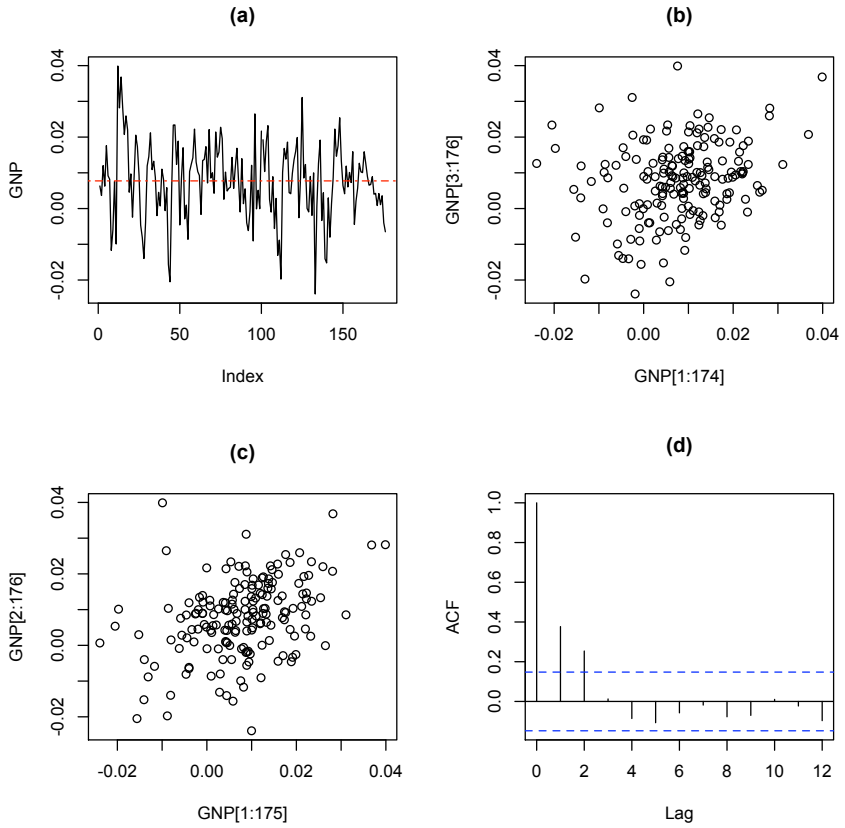

Figure 4.4: After plotting changes in U.S. GNP (a), we compare lag-1 (b), to lag-2 (c), and then see that the 12-period ACF function (d) finds autocorrelation at lag-1 and lag-2.

4.5 Simple ARMA Models

ARMA models are used with volatility modeling more than with returns series.

Having briefly discussed both AR and MA models, we discover that a model of multiple order may be required to perform adequate modeling and forecasting. A remedy for this is the autoregressive moving-average (ARMA) model, which is a combined model that keeps the number of parameters small. It is also important to understand ARMA models as a building block to GARCH models, which are discussed in Section 5.2.

A time series r_t follows an ARMA(1,1) model if it satisfies

$$r_t - \phi_1 r_{t-1} = \phi_0 + a_t - \theta_0 + a_t - \theta_1 a_{t-1}, \tag{4.9}$$

where $\{a_t\}$ is a white noise series. The left-hand side of (4.9) is the AR component and the right-hand side is the MA component. The constant term is ϕ_0. To avoid cancellation of both sides, which would make (4.9) a white noise series, we require that $\phi_1 \neq \theta_1$.

ARMA(1,1) Models. Since ARMA(1,1) models are generalizations of AR(1) models with modifications to allow inclusion of a MA(1) component, we already know much about these models, such as the stationarity condition is the same. There is a notable differences,

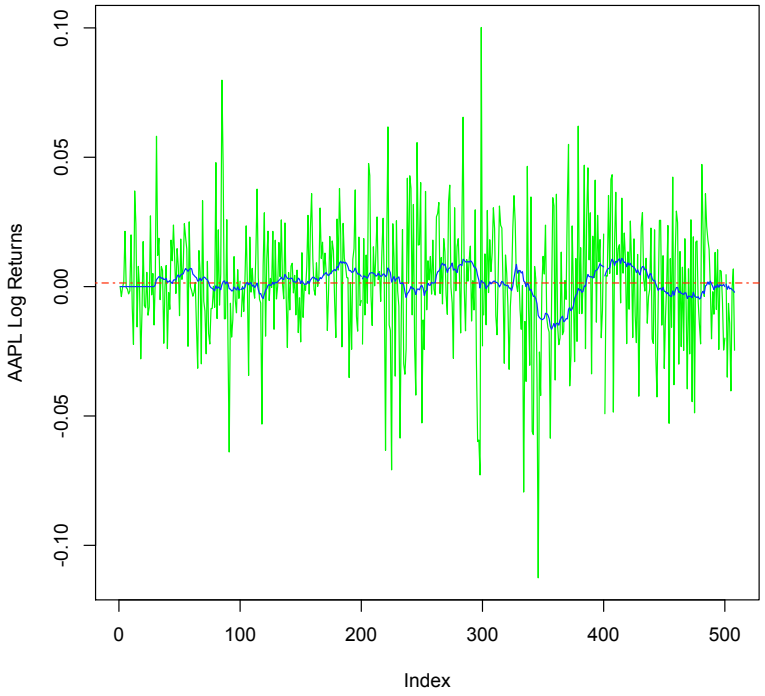

Figure 4.5: AAPL log return plotted in green. The mean plotted in red and dashed line. The 30-day moving average is superimposed in blue.

the ACF of an ARMA(1,1) model looks like a AR(1) model except that the exponential decay starts with lag 2 instead of lag 1. The general form of an ARMA(p, q) model is

$$r_t = \phi_0 + \sum_{i=1}^{p} \phi_i r_{t-1} + a_t - \sum_{i=1}^{q} \theta_i a_{t-i},$$

where $\{a_t\}$ is a white noise series and p and q are non-negative integers.

Using ACF and PACF are not very useful in determining the order of an ARMA model. Tsay and Tiao (1984) propose an alternative approach, which is known as the *extended autocorrelation function* (EACF). This function is used to specify the order of an ARMA process. The aim is to derive the MA component from a consistent estimate of the AR component of an ARMA model.

Forecasting with ARMA. A 1-step ahead forecast r_{h+1} of an ARMA model is

$$\hat{r}_h(1) = E(r_{h+1}|F_h) = \phi_0 + \sum_{i=1}^{p} \phi_i r_{h+1-i} - \sum_{i=1}^{q} \theta_i a_{h+1-i},$$

where h is the forecast origin, and F_h is the available information.

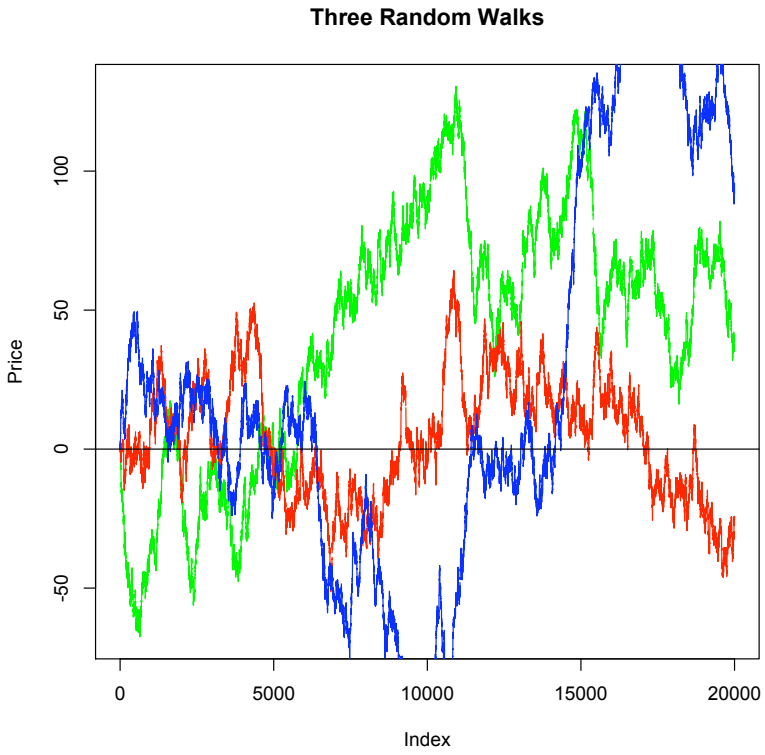

Figure 4.6: Three random walks, with no drift, yet they seem to converge.

4.6 Unit-Root Nonstationarity

Several types of financial time series are nonstationary, such as interest rates, foreign exchange currency rates, and asset price series because they have no fixed level of price. This condition is all referred to as *unit-root nonsationarity*. The best known example of this is a random-walk model.

A random walk is a time series $\{p_t\}$ that satisfies the condition

$$p_t = p_{t-1} + a_t, \tag{4.10}$$

where p_0 is the starting value and $\{a_t\}$ is white noise.

Using a simple R statement, `plot(cumsum(rnorm(20000)),type="l")` used three times, we create three random walks that seem to converge in Figure 4.6. One obvious problem is that the price time series drops below zero on several occasions. A nominal price or rate will never be below zero, but a *real rate*, one compared to another over time, may become negative.

If we treat (4.10) as a kind of AR(1) model, then the coefficient of p_{t-1} is one (*or unity*), which does not satisfy the weak stationarity condition of an AR(1) model. Therefore, a random-walk series is a unit-root nonstationary time series. This kind of series is not predictable or mean-reverting.

The MA representation of (4.10) is

$$p_t = a_t + a_{t-1} + a_{t-2} + \cdots,$$

which may continue indefinitely. The ℓ-step ahead forecast error is

$$e_h(\ell) = a_{h+\ell} + \cdots + a_{h+1},$$

which means that $\text{Var}[e_h(\ell)] = \ell\sigma^2$, and it diverges to infinity as $\ell \to \infty$. Again, this model is not predictable because the usefulness of the point forecast $\hat{p}_h(\ell)$ diminishes as ℓ increases.

Random Walk with Drift. Observing markets, it becomes clear that the log return series exhibits a positive mean, which would indicate some form of drift.

$$p_t = \mu + p_{t-1} + a_t, \tag{4.11}$$

where $\mu = E(p_t - p_{t-1})$ and $\{a_t\}$ is white noise. The μ term is vital to the model because it represents the time trend of the log price p_t, and is referred to as the *drift*.

4.6.1 Trend-Stationary Time Series

A model that is related to random walk with drift is the trend-stationary time series model

$$p_t = \beta_0 + \beta_1 t + r_t$$

where r_t is a stationary time series, such as a stationary AR(p) series. We observe that p_t grows in time on a linear scale β_1.

4.6.2 Unit-Root Nonstationary Models

Modifying an ARMA model by allowing the AR polynomial to have 1 as a characteristic root, the model becomes the autoregressive integrated moving-average (ARIMA) model. A common way to handle unit-root nonstationarity is by *differencing*.

Differencing. Price time series are seen as nonstationary, but log returns are stationary. A way to transform a non-stationary model into a stationary one is by *differencing*. The first difference series of y_t is

$$c_t = y_t - y_{t-1}.$$

Also, if s_t follows an ARMA(p, q) model, then y_t is an ARIMA($p, 2, q$) process.

Unit-Root Test. To test whether a log price series p_t follows random walk or random walk with drift, we use

$$\begin{aligned} p_t &= \phi_1 p_{t-1} + e_t, && \text{(4.12a)} \\ &= \phi_0 + \phi_1 p_{t-1} + e_t && \text{(4.12b)} \end{aligned}$$

where e_t is an error term comprising white noise, and we may using either (4.12a) or (4.12b) to examine the unit testing problem, which is set up as the *Dickey-Fuller Test.*

$$\begin{aligned} H_0 &: \phi_1 = 1 \\ H_a &: \phi_1 < 1 \end{aligned}$$

The test statistic is a t-ratio of least square estimate of ϕ_1. For instance, the least square method for (4.12a) is

$$\hat{\phi}_1 = \frac{\sum_{t=1}^{T} p_{t-1}p_t}{\sum_{t=1}^{T} p_{t-1}^2}, \qquad \hat{\sigma}_e^2 = \frac{\sum_{t=1}^{T}(p_t - \hat{\phi}_1 p_{t-1})^2}{T-1},$$

where $p_0 = 0$ and T is the sample size. The Dickey-Fuller test t-ratio is

$$DF \equiv t\text{-ratio} = \frac{\hat{\phi}_1 - 1}{\text{std}(\hat{\phi}_1)} = \frac{\sum_{t=1}^{T} p_{t-1}e_t}{\hat{\sigma}_e \sqrt{\sum_{t=1}^{T} p_{t-1}^2}}$$

When modeling economic time series, it is better to use ARIMA(p, d, q) than the simpler (4.12b), although AR(p) models are also often used.

Testing ARIMA Models. The three components (p, d, q) of ARIMA are the AR order p, the degree of differencing d, and the MA order q. Using the R functions `arima` and `AIC` we will test several combinations of an ARIMA(p, d, q) model and get an AIC value for each. The smaller the AIC, the better the model fits the data.

```
# Test AR order p by itself: ARIMA(p,0,0)
model100<-arima(aapl$Close,order=c(1,0,0))
model200<-arima(aapl$Close,order=c(2,0,0))
AIC(model100, model200)

# Test the MA order q by itself: ARIMA(0,0,q)
model001<-arima(aapl$Close,order=c(0,0,1))
model002<-arima(aapl$Close,order=c(0,0,2))
AIC(model001,model002)

# Now hold AR constant, change MA
model100<-arima(aapl$Close,order=c(1,0,0))
model101<-arima(aapl$Close,order=c(1,0,1))
model102<-arima(aapl$Close,order=c(1,0,2))
AIC(model100,model101,model102)

# Test the degree of differencing d
model100<-arima(aapl$Close,order=c(1,0,0))
model110<-arima(aapl$Close,order=c(1,1,0))
AIC(model100,model110)
```

4.7 Seasonal Models

We can use R to create a decomposition of the AAPL time series, which displays the data, seasonal pattern, trend, and the remainder. The decomposition is displayed in Figure 4.7. There are four factors that influence time series as seen in Table 4.1. The R code follows.

```
aaplts<-ts(aapl$Close,start=c(2006,1),frequency=250)
decomp<-stl(aaplts,"periodic")
plot(decomp)
```

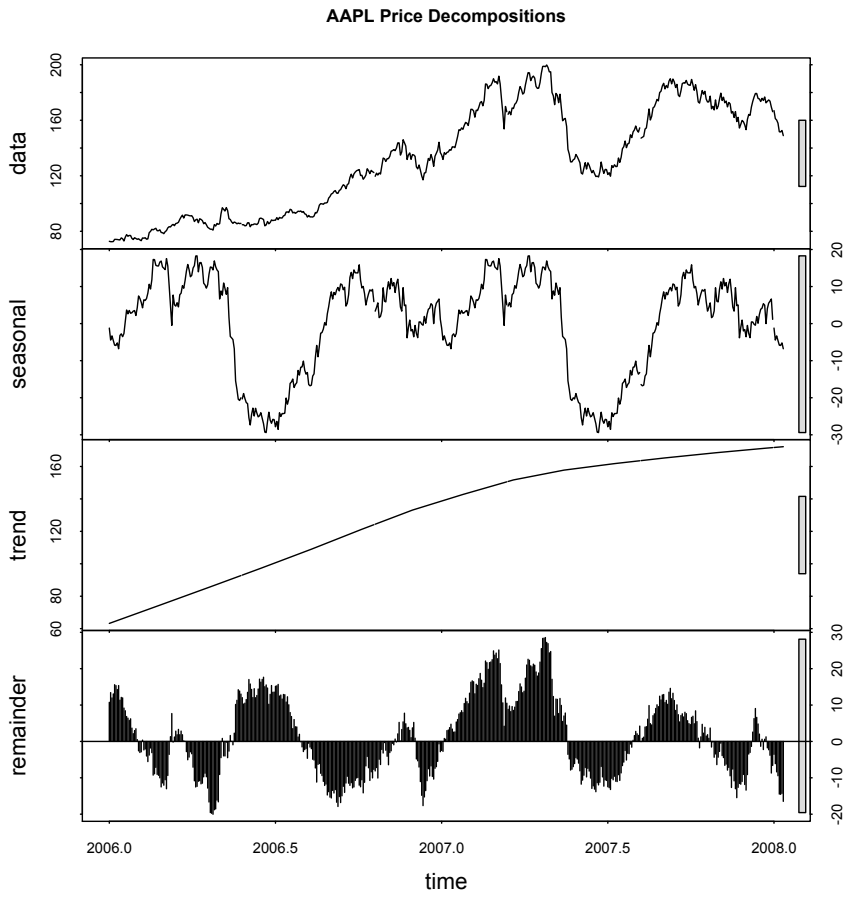

Figure 4.7: AAPL Price Series Decomposition

4.7.1 Seasonal Differencing with Dummy Variables

We will take data from the 2004 *Economic Report of the President* which contain many variables, but we are only concerned with, *i3* the 3-month T-bill rate, *inf* the annual inflation rate, and *def* the federal budget deficit as a percentage of GDP. We want to calculate β values for the equation,

$$\widehat{i3_t} = \beta_0 + \beta_1 inf_t + \beta_2 def_t. \tag{4.13}$$

We load the MATLAB data file `econrep` which has a matrix named `intdef`. The remaining steps to get our β values are required to perform matrix division.

```
load econrep
i3=intdef(:,2);
inf=intdef(:,3);
def=intdef(:,6);
A=[ones(size(inf)) inf def];
x=A\i3
```

Factor	Definition	Duration
Trend	Overall persistent long-term movement	Years
Seasonal	Regular pattern of periodic fluctuations per year	Within 12 mo. (weekly, annually, *etc.*)
Cyclical	Repeating up/down swings	Several (2–10) years
Irregular	Erratic, random variation	Non-repeating

Table 4.1: Factors Influencing Time Series

```
x =
    1.7333
    0.6059
    0.5131
```

which allows us to fill in constants for (4.13),

$$\widehat{i3_t} = 1.7333 + 0.6059 \times inf_t + 0.5131 \times def_t.$$

Next, we want to determine the r^2 value of this regression equation. However, we want to obtain an *adjusted* r^2,

$$r^2_{adj} = 1 - \Big((1 - r^2)\frac{n-1}{n-k-1}\Big), \tag{4.14}$$

where n is sample size, and k is degrees of freedom in the regression. This allows us to compare regressions with different numbers of independent variables. Adjusting r^2 in (4.14) allows us to account for the number of parameters used in the regression.

```
yhat=x(1) + x(2).*inf + x(3).*def;
sst=sum((i3-mean(i3)).^2);
sse=sum((yhat-mean(i3)).^2);
ssr=sum((yhat-i3).^2);
r_squared=sse/sst;
n=size(i3,1);
k=size(x,1)-1;
adj_r2=1-((1-r_squared).*(n-1)/(n-k-1))
```

We see that our $r^2 = 0.6021$ and adjusted $r^2 = 0.5871$.

Dummy Variables. Binary variables are also known as *dummy variables*. In order to include an observed qualitative factor, we need to pick a variable that indicates what is represented by the number 1. Choosing $female = 1$ is more clear than $gender = 1$, which tells us nothing about what the value of 1 represents.

Ordinal Variables. Some qualitative variables are *ordinal*, that is, they measure a factor that has more than two values. If we want to use season of the year as an independent variable, we need four values (Winter, Spring, Summer, Fall). It makes little sense to assign

Season	Variables
Winter	$\mathbf{S_1 = 1}; S_2 = 0; S_3 = 0; S_4 = 0$
Spring	$S_1 = 0; \mathbf{S_2 = 1}; S_3 = 0; S_4 = 0$
Summer	$S_1 = 0; S_2 = 0; \mathbf{S_3 = 1}; S_4 = 0$
Fall	$S_1 = 0; S_2 = 0; S_3 = 0; \mathbf{S_4 = 1}$

Table 4.2: Ordinal Variable Setting

$S = 1$ to Winter, and $S = 2$ to Spring because Spring does not represent any quantity that is double the value of Winter. Instead, we may assign a variable for each season (Table 4.2), so that we have,

$$\hat{y} = \beta_0 + \delta_1 S_1 + \delta_2 S_2 + \delta_3 S_3 + \delta_4 S_4 + \epsilon.$$

One value they can add to time series analysis is in the form of an *event study*. The variable can indicate the period in which a specific event occurred. If we add a dummy variable to (4.13) that indicates if Congress has enacted a certain law,

$$\widehat{i3_t} = \beta_0 + \beta_1 inf_t + \beta_2 def_t + \beta_3 d_t,$$

then, with that dummy variable d_t we can observe the effect of the law by measuring the estimators with the variable and without it.

Index Numbers. To construct an event study, we must often work with an *index number*. The Consumer Price Index (CPI), "is a measure of changes in prices for purchases made by urban consumers only" (Rogers, 1998, p. 91). Essentially, it is the price of a specific basket of consumer goods at a specific point in time. The Bureau of Labor Statistics publishes a national number as well as several regions CPI numbers.

An index number such as the CPI-U[5] means nothing on its own; the August 2008 level is 219.086, and is -0.4 percent lower than in the previous month. However, its value comes from comparing it to a *base period* which has a *base value*. Our August 2008 CPI-U has a base period of 1982–1984, and a base value of 100. From this point, we can determine the change in price level for consumers over a span of time. Additionally, we can adjust *nominal dollars* into *real dollars* by factoring inflation as measured by CPI changes.

Having made any adjustments using index numbers to our time series data, we may use them in time series regressions without distorting their value.

4.7.2 Trend and Seasonality

Referring back to Figure 4.7, we see that econometric models calculate the level, trend and seasonality and combine them to arrive at a forecast. It is not possible to understand the nature of trend and seasonality simply by viewing the data in the graphical screen. Decomposing time series data into trend and seasonality provides insights into the patterns

[5]The U.S. Department of Labor publishes a monthly "All Urban Consumers" Consumer Price Index (CPI-U) based on U.S. city average of a basket of consumer goods. The CPI website is located at `http://www.bls.gov/cpi/home.htm`

of these components. This helps us choose the right forecast strategy or in setting the parameters for a particular strategy.

Isolating the trend component requires neutralizing the seasonal effect in the data. If the data has a 12 period seasonality, computing a 12 period *moving average* neutralizes the effect of seasonality. With a history of 24–36 months, we can calculate the trend for 13–25 months. Viewing the trend graphically provides insights into the way trend is changing over the analysis period. We can assess from this graph if the short term trend is in line with the long term trend. The short term trend is relevant for operational decisions like trading whereas the long term trend would be relevant for annual budgeting and other capital expenditure decisions.

Since the time series is a combination of trend, seasonality and randomness, subtracting the trend from the time series will leave us with seasonality and randomness. The seasonal patterns can be viewed graphically at different aggregate levels. If the time series has seasonality, one will be viewing recurring patterns at periodic intervals. If there is no seasonality, there will not be any recurring patterns but only random patterns.

5 Conditional Heteroskedastic Models

Primary Text Reading. Tsay (2005, chap. 3)

Up to this point, we have focused on modeling assets returns or forecasting their prices. When analyzing financial time series, we may also want to analyze the *volatility* or, the conditional standard deviation, of the asset. Since volatility is not directly experienced, we must find a substitute for it. We can look at an option pricing model that uses volatility as an input.

The *implied volatility* of an option contract is the volatility implied by the market price of the option based on an option pricing model. It is the volatility that, given a particular pricing model, yields a theoretical value for the option equal to the current market price. Implied volatility, a forward-looking measure, differs from historical volatility because the latter is calculated from known past prices of a security.

One model, which requires conditional volatility as one of its inputs, is the Black-Scholes option pricing model,

$$C(S,T) = S\Phi(d_1) - Ke^{-rT}\Phi(d_2) \tag{5.1a}$$
$$P(S,T) = Ke^{-rT}\Phi(-d_2) - S\Phi(-d_1) \tag{5.1b}$$

where,

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = d_1 - \sigma\sqrt{T}$$
$$\Phi(\cdot) = \text{c.d.f. of the Normal Distribution, (1.2).}$$

Using this closed-form solution, we can obtain the price for either a call option (5.1a) or a put option (5.1b)[6]. Similarly, if we have the option price, and all other inputs except the volatility, we can calculate the implied volatility by entering a value that would satisfy the equation. However, to do so may be relying upon the model (and its unrealistic assumptions) too much.

Volatility. It is a difficult task to measure volatility of an asset return. Although the volatility may exist in a continuous form, with no sudden jumps, it may remain low for a period and then shift upward. Depending upon the asset, and whether it rose or fell in price, a major price change in the underlying security may affect volatility differently. To be certain, volatility changes, and it can be rapid and difficult to observe directly.

To measure volatility, we need to model the changes in variance σ_t^2 in the asset return time series. The various conditional heteroskedastic models differ in how this variance changes in time. The evolution of σ_t^2 is deterministic in some models, such as the GARCH model, yet *stochastic volatility* models exist also.

Different volatility models make different assumptions about σ_t^2.

[6] A *call option* is a contract that gives the holder the right to **buy** a certain quantity of an underlying security from the writer of the option, at a specified price (the strike price) up to a specified date (the expiration date). A *put option* is a contract that gives the holder the right to **sell** a certain quantity of an underlying security to the writer of the option, at a specified price up to a specified date.

5.1 Building a Volatility Model

The basic structure of volatility modeling is

$$r_t = \mu_t + a_t, \quad \mu_t = \phi_0 + \sum_{i=1}^{k} \beta_i x_{it} + \sum_{i=1}^{p} \phi_i r_{t-i} - \sum_{i=1}^{q} \theta_i a_{t-i},$$

and volatility models are concerned with time-evolution of

$$\sigma_t^2 = \text{Var}\left(r_t | F_{t-1}\right) = \text{Var}\left(a_t | F_{t-1}\right).$$

Volatility model building requires four steps,

1. Create a mean equation and test for serial dependence in the financial time series, and calibrate an AR or ARMA model to remove autocorrelation. For many asset returns, this is as simple as removing the sample mean from the data, and adding qualitative variables, as described in Section 4.7.1.
2. Test for ARCH effects with the residuals from the previous step.
3. If ARCH effects are statistically significant, choose a volatility model, and perform a joint estimation of the mean and volatility equations.
4. Refine the model if needed.

5.1.1 ARCH Effects

If we denote $a_t = r_t - \mu_t$ as the residuals of the mean equation, then the squared series a_t^2 is used to check for conditional heteroskedacity. These are the *ARCH effects*. There are two different tests available. We can apply the Ljung-Box statistic $Q(m)$ to $\{a_t^2\}$ using as the null hypothesis that the first m lags are zero. Another test is the *Lagrange multiplier test*, which is equivalent to F statistic for testing $\alpha_i = 0 \quad (i = 1, \ldots, m)$ in a linear regression

$$a_t^2 = \alpha_0 + \alpha_1 \alpha_{t-1}^2 + \cdots + \alpha_m \alpha_{t-m}^2 + e_t, \quad t = m+1, \ldots, T,$$

where e_t is the error term, m is a specific positive integer, and T is the sample size. The null hypothesis is $H_0 : \alpha_1 = \cdots a_m = 0$. Next, we need to define SSR_0 and SSR_1

$$\begin{aligned} SSR_0 &= \sum_{t=m+1}^{T} (a_t^2 - \bar{\omega})^2, \\ SSR_1 &= \sum_{t=m+1}^{T} \hat{e}_t^2, \end{aligned}$$

where $\bar{\omega} = (1/T) \sum_{t=1}^{T} a_t^2$ is the sample mean of a_t^2 and $\hat{e}^t$ is the least squares residual of the prior linear regression. The F-test is then

$$F = \frac{(SSR_0 - SSR_1)/m}{SSR_1/(T - 2m - 1)}.$$

5.1.2 ARCH Model

An *autoregressive conditional heteroskedasticity* (ARCH) model considers the variance of the current error term to be a function of the variances of the previous time period's error terms (Engle, 1982). ARCH relates the error variance to the square of a previous period's error. It is employed commonly in modeling financial time series that exhibit time-varying volatility clustering, which are periods of swings followed by periods of relative calm. The shock a_t of an asset return is serially uncorrelated, but dependent; and the dependence can be described by a simple quadratic function of its lagged values.

Specifically, let ϵ_t denote the returns (or return residuals, net of a mean process) and assume that $\epsilon_t = \sigma_t z_t$, where $z_t \sim iid\ N(0,1)$ and where the series σ_t^2 are modeled by

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2$$

and where $\alpha_0 > 0$ and $\alpha_i \geq 0,\ i > 0$.

5.1.3 ARCH(q) Model Specification

An ARCH(q) model can be estimated using ordinary least squares. A methodology to test for the lag length of ARCH errors using the Lagrange multiplier test was proposed by Engle (1982). Estimate the best fitting AR(q) model

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_q y_{t-q} + \epsilon_t = a_0 + \sum_{i=1}^{q} a_i y_{t-i} + \epsilon_t$$

Obtain the squares of the error $\hat{\epsilon}^2$ and regress them on a constant and q lagged values,

$$\hat{\epsilon}_t^2 = \hat{\alpha}_0 + \sum_{i=1}^{q} \hat{\alpha}_i \hat{\epsilon}_{t-i}^2$$

where q is the length of ARCH lags.

The null hypothesis is that, in the absence of ARCH effects, we have $\alpha_i = 0$ for all $i = \{1, \ldots, q\}$. The alternative hypothesis is that, in the presence of ARCH effects, at least one of the estimated α_i coefficients must be significant. In a sample of T residuals under the null hypothesis of no ARCH errors, the test statistic TR^2 follows χ^2 distribution with q degrees of freedom. If TR^2 is greater than the chi-square table value, we reject the null hypothesis and conclude there is an ARCH effect in the ARMA model. If TR^2 is smaller than the chi-square table value, we accept the null hypothesis.

Model Building. Using the R library `FinTS`[7] (Graves, 2008), we can perform the Lagrange Multiplier (LM) test for autoregressive conditional heteroscedasticity (ARCH).

```
data(m.intc7303)
intcLM <- ArchTest(log(1+as.numeric(m.intc7303)), lag=12)
# Matches answer on Tsay (p. 102)
```

[7] Rmetrics has many other excellent functions in the `fSeries` package (Wuertz et al., 2007c). A description of the many packages available from Rmetrics is at `http://www.rmetrics.org/rmetricsPackages.htm` and are available on CRAN.

Drawbacks of ARCH. The ARCH model assume that positive and negative shocks have the same effect on volatility because it depends on the square of the previous shocks. This is not realistic. The ARCH model is not very helpful in finding the source of variation within a financial time series. Also, the model responds slower to large isolated shocks.

5.2 GARCH

By comparison, ARCH models are simple, although the number of required parameters makes it difficult to calibrate. Next, we examine generalizations of ARCH to model volatility. If an ARMA model is assumed for the error variance, the model is a generalized autoregressive conditional heteroskedasticity model (Bollerslev, 1986). In such a case, the GARCH(m, s) model, where m is the order of the GARCH terms σ^2 and s is the order of the ARCH terms ϵ^2, is given by

$$\begin{aligned}\sigma_t^2 &= \alpha_0 + \alpha_1\epsilon_{t-1}^2 + \cdots + \alpha_s\epsilon_{t-s}^2 + \beta_1\sigma_{t-1}^2 + \cdots + \beta_m\sigma_{t-p}^2 \\ &= \alpha_0 + \sum_{i=1}^{q}\alpha_i\epsilon_{t-i}^2 + \sum_{i=1}^{m}\beta_i\sigma_{t-i}^2.\end{aligned}$$

So that, the GARCH(m, s) model is

$$a_t^2 = \alpha_0 + \sum_{i=1}^{\max(m,s)}(\alpha_i + \beta_i)a_{t-i}^2 + \eta_t - \sum_{j=1}^{s}\beta_j\eta_{t-j}. \tag{5.2}$$

5.2.1 GARCH(m, s) Model Specification

The lag length p of a GARCH process is established in three steps,

1. Estimate the best fitting AR(q) model

$$y_t = a_0 + a_1y_{t-1} + \cdots + a_qy_{t-q} + \epsilon_t = a_0 + \sum_{i=1}^{q}a_iy_{t-i} + \epsilon_t$$

2. Compute and plot the autocorrelations of ϵ^2 by

$$\rho = \frac{\sum_{t=i+1}^{T}(\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)(\hat{\epsilon}_{t-1}^2 - \hat{\sigma}_{t-1}^2)}{\sum_{t=1}^{2}(\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)^2}$$

3. The asymptotic, large sample, standard deviation of $\rho(i)$ is $T^{-\frac{1}{2}}$. Individual values that are larger than this indicate GARCH errors. To estimate the total number of lags, use the Ljung-Box test until the value of the these are less than a 5% significant. The Ljung-Box Q-statistic follows χ^2 distribution with n degrees of freedom if the squared residuals ϵ_t^2 are uncorrelated. It is recommended to consider up to $T/4$ values of n. The null hypothesis states that there are ARCH or GARCH errors. Therefore, rejecting the null means that there are no such errors in the conditional variance.

Specifying the order of a GARCH model is not easy, which is why lower order models such as GARCH(1,1), GARCH(1,2), or GARCH(2,1) are usually used. As long as the starting volatility is known, the conditional maximum likelihood method will work.

Forecasting with GARCH. Since volatility is not directly observable, testing and calibrating GARCH methods can be difficult. Often, one of the best ways to test a GARCH model for forecasting is to compare in-sample and out-of-sample results of volatility forecasts $\sigma_h^2(\ell)$ compared to shock $a_{h+\ell}^2$.

Forecasting with (5.2) can employ a similar method used when forecasting (4.9). Pursue the first parameter and seek the lowest AIC value, then hold that parameter constant while changing the second parameter. Sometimes, it may be helpful to go back and change the first parameter again after obtaining the second.

We can use a few R functions from the `fArma` library (Wuertz et al., 2007a), `fGarch` library (Wuertz et al., 2007b), and the `fSeries` library (Wuertz et al., 2007c) to examine GARCH models. The fit is displayed in Figure 5.1.

```
spxm<-read.csv("SPX-M.csv", header=T)
spxts<-ts(spxm$Adj.Close, start=c(1950,1), frequency=12)
plot(spxts)

spxG<-garchFit(~garch(1, 1), data = spxts)
plot(spxG) # There are 13 different charts to choose from.
spxG@title
```

With these functions, we can model the monthly volatility of the S&P 500 Index with `fGarch` by starting with a GARCH(1,1) model. The `garchFit` function generates several statistics among its results,

```
Parameter Matrix:
                      U            V     params includes
    mu     -3.644376e+03     3644.376   364.4376     TRUE
    omega   1.968285e-01 19682849.836 19682.8498     TRUE
    alpha1  1.000000e-08        1.000     0.1000     TRUE
    gamma1 -1.000000e+00        1.000     0.1000    FALSE
    beta1   1.000000e-08        1.000     0.8000     TRUE
    delta   0.000000e+00        2.000     2.0000    FALSE
    skew    1.000000e-01       10.000     1.0000    FALSE
    shape   1.000000e+00       20.000     4.0000    FALSE
```

Ultimately, after several iterations of changes in the log likelihood ("LLH"), we have a Hessian matrix,

$$\begin{bmatrix} & \mu & \omega & \alpha_1 & \beta_1 \\ \mu & 1.10102573 & & & \\ \omega & 0.07283437 & 0.16546865 & & \\ \alpha_1 & -0.28749874 & 0.98697087 & 311.6009165 & \\ \beta_1 & -11.94005197 & 0 & 0 & -1.136868e+11 \end{bmatrix}.$$

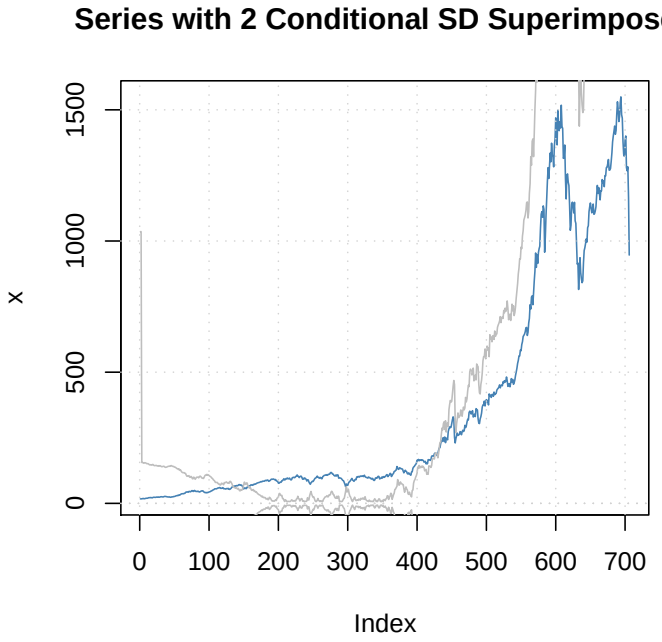

Figure 5.1: Monthly S&P 500 GARCH Estimate with Two Conditional SD Superimposed

We can examine the results graphically with 13 different types of graphs using `plot(spxG)`. Between this and the statistics generated from the GARCH model, we can modify the parameter set (m, s) for best fit.

```
Make a plot selection (or 0 to exit):

 1:   Time Series                                  2:   Conditional SD
 3:   Series with 2 Conditional SD Superimposed    4:   ACF of Observations
 5:   ACF of Squared Observations                  6:   Cross Correlation
 7:   Residuals                                    8:   Conditional SDs
 9:   Standardized Residuals                      10:   ACF of Standardized Residuals
11:   ACF of Squared Standardized Residuals       12:   Cross Correlation between r^2 and r
13:   QQ-Plot of Standardized Residuals
```

5.2.2 Integrated GARCH Model

IGARCH is a unit-root GARCH model.

When AR polynomial of (5.2) has a unit root, then we have an IGARCH model. The IGARCH(1,1) model

$$a_t = \sigma_t \epsilon_t, \quad \sigma_t^2 = \alpha_0 + \beta_1 \sigma_{t-1}^2 + (1 - \beta_1) a_{t-1}^2, \tag{5.3}$$

where $\{\epsilon_t\}$ is $1 > \beta_1 > 0$. An IGARCH(1,1) model can be estimated by exponential smoothing methods when $\alpha_0 = 0$.

5.2.3 GARCH in Mean Model

GARCH-M is a "GARCH in the mean" model because it adds the heteroskedastic term directly into the equation. It can be written as

$$\begin{aligned} r_t &= \mu + c\sigma_t^2 + a_t, \quad a_t = \sigma_t \epsilon_t, \\ \sigma_t^2 &= \alpha_0 + \alpha_1 \alpha_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \end{aligned} \tag{5.4}$$

where μ and c are constants. The c parameter is the *risk premium parameter*. A $c > 0$ indicates that the return has a positive relationship to its volatility.

5.2.4 Exponential GARCH Model

EGARCH was created to allow for asymmetric effects in asset returns. Positive and negative asset returns may not be equally symmetric, a problem not addressed by previous models. We begin with the weighted innovation

$$g(\epsilon_t) = \theta \epsilon_t + \gamma \left[|\epsilon_t| - E(|\epsilon_t|) \right],$$

where θ and γ are real constants. Both ϵ_t and $|\epsilon_t| - E(g(|\epsilon_t|)$ are zero-mean ($E[g(\epsilon_t)]$) iid sequences with continuous distributions. A better way to see the asymmetry is

$$g(\epsilon_t) = \begin{cases} (\theta + \gamma)\epsilon_t - \gamma E(|\epsilon + t|) & \text{if } \epsilon_t \geq 0, \\ (\theta - \gamma)\epsilon_t - \gamma E(|\epsilon + t|) & \text{if } \epsilon_t < 0. \end{cases}$$

An EGARCH(m, s) model can be written as

$$a_t = \sigma_t \epsilon_t, \quad \ln(\sigma_t^2) = \alpha_0 + \frac{1 + \beta_1 B + \cdots + \beta_{s-1} B^{s-1}}{1 - \alpha_1 B - \cdots - \alpha_m B^m} g(\epsilon_t - 1), \tag{5.5}$$

where α_0 is a constant, B is the *back-shift* or lag operator such that $Bg(\epsilon_t) = g(\epsilon_{t-1})$, and $1 + \beta_1 B + \cdots + \beta_{s-1} B^{s-1}$ and $1 - \alpha_1 B - \cdots - \alpha_m B^m$ are polynomials with zeros outside the unit circle[8] and have no common factors. The evolution of the conditional variance of a_t is still contained in (5.5) by using ARMA parameterization.

Alternative EGARCH Model. An alternative form for EGARCH(m, s) model is

$$\ln(\sigma_t^2) = \alpha_0 + \sum_{i=1}^{s} a_i \frac{|a_{t-i}| + \gamma_i a_{t-i}}{\sigma_{t-i}} + \sum_{j=1}^{m} \beta_j \ln(\sigma_{t-j}^2). \tag{5.6}$$

Notice in (5.6) that,

- a positive a_{t-i} contributes $\alpha_i(1 + \gamma_i)|\epsilon_{t-i}|$ to the log volatility,
- a negative a_{t-i} contributes $\alpha_i(1 - \gamma_i)|\epsilon_{t-i}|$,

where $\epsilon_{t-i} = a_{t-i}/\sigma_{t-i}$. The leverage effect of a_{t-i} is the γ_i parameter.

[8] To be *outside the unit circle* means to have absolute values of the zeros greater than 1.

5.2.5 Conditional Heteroskedasticity (CHARMA) Model

The next model is CHARMA, which uses random coefficients to produce conditional heteroskedasticity. A CHARMA model is defined as

$$r_t = \mu_t + a_t, \quad a_t = \delta_{1t}a_{t-1} + \delta_{2t}a_{t-2} + \cdots + \delta_{mt}a_{t-m} + \eta_t, \tag{5.7}$$

where $\{\eta_t\}$ is a Gaussian (normally distributed) white noise series with mean zero and variance $\sigma_\eta^2, \{\delta_t\} = \{(\delta_{1t}, \ldots, a_{mt})'\}$ is a sequence of iid random vectors with mean zero and non-negative definite covariance matrix $\mathbf{\Omega}$, and $\{\delta_t\}$ is independent of $\{\eta_t\}$.

The conditional variance of at of the CHARMA model (5.7) is

$$\begin{aligned} \sigma_t^2 &= \sigma_\eta^2 + \mathbf{a}_{t-1}^{'}\mathrm{Cov}(\delta)\mathbf{a}_{t-1} \\ &= \sigma_t^2 + (a, \ldots, a)\mathbf{\Omega}(a, \ldots, a). \end{aligned} \tag{5.8}$$

5.3 Stochastic Volatility Model

If we model an underlying security's volatility as a random process, which is governed by state variables such as the price level of a security, there is a tendency of volatility to revert to some long-run mean value, and the variance of the volatility process itself.

Stochastic volatility models are one approach to resolve a shortcoming of some option pricing models that assume a fixed volatility during the life of the option contract. Because these option pricing models assume that the underlying volatility is constant over the life of the derivative, and unaffected by the changes in the price level of the underlying, they often misprice the option value. By assuming that the volatility of the underlying price is a stochastic process rather than a constant, it becomes possible to model derivatives more accurately.

The basic form of stochastic volatility is derived from Brownian motion, one of the simplest continuous-time stochastic processes, and is often used to model asset price movement.

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{5.9}$$

The maximum likelihood estimator to estimate the constant volatility σ, for given asset price time series S_t, at different times t_i, is

$$\hat{\sigma}^2 = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{(\ln S_{t_i} - \ln S_{t_{i-1}})^2}{t_i - t_{i-1}} \right) - \frac{1}{n} \frac{(\ln S_{t_n} - \ln S_{t_0})^2}{t_n - t_0}.$$

GARCH Estimation. The difficulty of estimating stochastic volatility is made slightly easier with GARCH methods, which assumes that the randomness of the variance process varies with the variance. The standard GARCH(1,1) model has this form for the variance differential

$$d\nu_t = \theta(\omega - \nu_t)dt + \xi \nu_t dB_t.$$

All stochastic volatility models have stochastic second moments (variance), but not all models that have stochastic second moments are called stochastic volatility models. Although stochastic volatility and GARCH are often presented as competing models, GARCH may help explain changes in volatility.

Both the ARCH/GARCH and stochastic volatility models derive their randomness from a white noise process. The difference is that an ARCH/GARCH process depends on just one white noise W. That white noise directly determines innovations in the ARCH/GARCH process while also indirectly determining innovations in its second moments. Despite the conventional description of (5.9), stochastic volatility models generally depend on two white noise series, V and W.

Heston model. Another stochastic volatility model used for describing the evolution of the volatility of an underlying asset is the *Heston model*. It assumes that the volatility of the asset is not constant, nor even deterministic, but follows a random process. Assuming that the asset price S_t is determined by a stochastic process,

$$dS_t = \mu S_t\, dt + \sqrt{\nu_t} S_t\, dW_t^S.$$

The instantaneous variance is

$$d\nu_t = \alpha(\theta - \nu_t)dt + \xi\sqrt{\nu_t}\, dW_t^\nu,$$

where, μ is the asset rate of return, θ is the long vol, or long run average price volatility; as t tends to infinity, the expected value of ν_t tends to α, α is the rate at which ν_t reverts to θ, and ξ is the volatility of the volatility (or, *wiggle of the wiggle*), which determines the variance of ν_t.

6 Nonlinear Models

Primary Text Reading. Tsay (2005, chap. 4) and Franses and van Dijk (2000)

6.1 Simple Nonlinear Models

Nonlinear data exists in financial time series, especially in volatility and high-frequency data. We focus now on simple nonlinear models and neural networks.

Bilinear Model. A way to include rate-change points or other nonlinear events is to use a bilinear model such as

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} - \sum_{j=1}^{q} \theta_j a_{t-j} + \sum_{i=1}^{m}\sum_{j=1}^{s} \beta_{ij} x_{t-i} a_{t-j} + a_t, \tag{6.1}$$

where p, q, m, and s are non-negative integers.

Some properties of bilinear models allow the modeling of nonlinear phenomenon with a minimum of parameters.

6.1.1 Threshold Autoregressive Model

Adding some asymmetry to (6.1), we have the threshold autoregressive model, which is a piecewise linear model in the threshold space. For example, a 2-regime AR(1) model

$$x_t = \begin{cases} -1.5x_{t-1} + a_t, & \text{if } x_{t-1} < 0, \\ 0.5x_{t-1} + a_t, & \text{if } x_{t-1} \geq 0, \end{cases}$$

where the a_t's are iid $N(0,1)$.

Here the delay is 1 time period, x_{t-1} is the threshold variable, and the threshold is 0. The threshold divides the x_{t-1} space into two regimes with Regime 1 denoting $x_{t-1} < 0$.

Special features of this model are, (a) asymmetry in rising and declining patterns, (*more observations are positive than are negative*) (b) the mean of x_t is not zero even though there is no constant term in the model, and (c) the lag-1 coefficient may be greater than 1 in absolute value.

6.1.2 Markov Switching Model

Another example of a two-regime model, but with aperiodic switching is a two-state *Markov autoregressive switching* (MSA) model is

$$x_t = \begin{cases} c_1 + \sum_{i=1}^{p} \phi_{1,i} x_{t-i} + a_{1t}, & \text{if } s_t = 1, \\ c_2 + \sum_{i=1}^{p} \phi_{2,i} x_{t-i} + a_{2t}, & \text{if } s_t = 2, \end{cases} \tag{6.2}$$

where s_t assumes values in $\{1, 2\}$ and is a first-order Markov chain with transition probabilities, $P(s_t = 2|s_{t-1} = 1) = w_1, \quad P(s_t = 1|s_{t-1} = 2) = w_2$, and where $0 \leq w_1 \leq 1$ is the probability of switching out State 1 from time $t-1$ to time t. A large w_1 means that it is easy to switch out State 1, that is, it cannot stay in State 1 for long. The inverse, $1/w_1$, is the expected duration (*number of time periods*) to stay in State 1. A similar idea applies to w_2.

6.1.3 Nonparametric Methods

There may be times when obtaining adequate information about the distribution will be impossible. In that case, we must rely upon nonparametric methods to perform estimation. They are very data dependent and may lead to over fitting of a model.

Kernel density estimators *discover* the data's distribution.

A major element of nonparametric methods is *smoothing*. To do this, we need a density estimator, which is the construction of an estimate, based on observed data, of an unobservable underlying probability density function. The unobservable density function is thought of as the density according to which a large population is distributed; the data are usually thought of as a random sample from that population.

In MATLAB, kernel density estimation is implemented through the `ksdensity` function. The R language can perform this estimation using `density` function and the `kde2d` function, for two-dimensional kernel density estimation.

Kernel Regression. Kernel regression is a non-parametric method to estimate the conditional expectation of a random variable. Its objective is to find a non-linear relationship between two random variables x and y, such that

$$E(y|x) = f(x)$$

where f is a non-parametric function. The relationship may also be expressed

$$E(y|x) = \int yf(y|x)dy = \int y\frac{f(x,y)}{f(x)}dy.$$

A common method is the Nadaraya-Watson kernel regression, which is an estimate m of a locally weighted average, using a kernel as a weighting function

$$\widehat{m}(x) = \frac{\sum_{t=1}^{T} K_h(x - x_t)y_t}{\sum_{t=1}^{T} K_h(x - x_t)}.$$

6.1.4 Neural Networks

Neural networks are non-linear statistical data modeling tools, often used to model complex relationships between inputs and outputs or to find meaningful patterns in data. Because of this, a neural network is a semi-parametric approach to data analysis. Neural networks learn by example. The analyst gathers representative data, and then invokes training algorithms to automatically learn the structure of the data. It has a network structure having,

- Input layer – This is the observed data that we are trying to discover a relationship to. The aim is to find how known inputs and possibly known outputs are related.
- Hidden layer – The relationships between value pairs and the weights applied to them. Since the relationships may be complex, and be linked many ways through *nodes*, there may multiple hidden layers. Sometimes, hidden layer is referred to in data mining literature as both a singular layer and as all layers collectively.
- Nodes – The mechanism that links layers together. Some neural network will have multiple hidden layers, and the way their layers are linked (through their nodes) will determine redundancy and search efficiency.

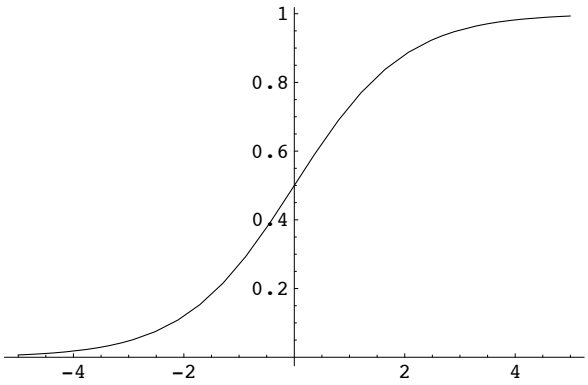

Figure 6.1: Plot of logistic function in (6.3)

- Output layer – The resulting pairing from the input layer that have been paired through the hidden layer. Often, the analyst is uncertain how the pairing occurred, which runs the risk that out-of-sample data may behave differently, and thus, create a different hidden layer with different weightings.

The *activation function*, which starts the network, is a logistic function,

$$\ell(z) = \frac{\exp(z)}{1 + \exp(z)} \tag{6.3}$$

This function is plotted in Figure 6.1 subject to $[-5, 5]$. Next, the network requires the Heaviside (or threshold) function, which allows switching between regimes,

$$H(z) = \begin{cases} 1 & \text{if } z > 0, \\ 0 & \text{if } z \leq 0. \end{cases}$$

A neural network having a hidden node works through,

$$x_j = f_j(\alpha_j + \sum_{i \to j} w_{ij} x_i)$$

where $f_j(\cdot)$ is an activation function which is typically taken to be the logistic function (6.3). Then, α_j is called the *bias*, the summation $i \to j$ means summing over all input nodes feeding to j, and w_{ij} are the weights. The output node,

$$y = f_o(\alpha_o + \sum_{j \to o} w_{jo} x_j),$$

where the activation function $f_o(\cdot)$ is either linear or a Heaviside function. By a Heaviside function, we mean $f_o(z) = 1$ if $z > 0$ and $f_o(z) = 0$, otherwise. Thus, the general form is,

$$y = f_o\left[a_o + \sum_{j \to o} w_{jo} f_j(\alpha_j + \sum_{i \to j} w_{ij} x_i)\right] \tag{6.4}$$

Supervised Learning. The task of knowledge discovery in databases (KDD) allows the analyst to examine data and its relationship to functional forms in two ways. The first, is through *supervised learning*. We have a set of data pairs (x, y), $x \in X$, $y \in Y$ and the goal is to find a function f within a class of functions that relates to the pairs. This "supervised" method requires us to infer how the mapping implied by the data and the cost function is related to the mismatch between our mapping and the data.

Unsupervised learning. By contrast, through *unsupervised learning* we are given some data x, and a cost function $c(\cdot)$ which is minimized with respect to any function of x and the network's output, f. The cost function is determined by the task formulation. In this method, we have little, if any, idea how the data are related. We let the data make the connection for us.

Training and Forecasting. Divide the data into training and forecasting subsamples. In the training subsample, build a few network systems. The forecasting is based on the accuracy of out-of-sample forecasts to select the "best" network.

6.2 Complex Nonlinear Models

6.2.1 Cosine Seasonality

In some markets, seasonal energy markets, for instance, there may be at least two seasonal factors (Pilipovic, 2007). The cosine function can capture these factors, requiring relatively few parameters – the magnitude of the seasonal factor, and the time of year when it occurs. Here is a function that capture two different periodicities,

$$\overbrace{\beta_1 \cos(2\pi(T - t_1^C))}^{annual} + \underbrace{\beta_2 \cos(4\pi(T - t_2^C))}_{semi-annual} \tag{6.5}$$

We can model two different seasonal patterns in (6.5) by representing t_1^C with β_1 magnitude, as the annual center, and t_2^C with β_2 magnitude, as the semi-annual center.

6.2.2 Exponential Seasonality

Using repetitive exponential functions is another way to model seasonality to capture a more narrow rise and fall of the financial time series. This is different from the cosine function, in which the seasonal factor affects the entire curve. With exponential seasonality, we can apply local treatment,

$$\beta_1 \exp\left(-\gamma_1 \big(rfc(T - t_1^C)\big)^2\right) + \beta_2 \exp\left(-\gamma_2 \big(rfc(T - t_2^C)\big)^2\right), \tag{6.6}$$

where the function rfc is annually repetitive, and returns the annualized time to or from the closest center t_i^C for seasonal factor i. The two seasonal factors in (6.6) each has a center at t^C, a magnitude β, and a width of the seasonal peak defined by the "decay" coefficient γ.

7 High-Frequency Data Analysis

Primary Text Reading. Tsay (2005, chap. 5)

7.1 Nonsynchronous Trading

Market microstructure: Why is it important?

1. Important in market design & operation, e.g. to compare different markets (NYSE vs NASDAQ)
2. To study price discovery, liquidity, volatility, etc.
3. To understand costs of trading
4. Important in learning the consequences of institutional arrangements on observed processes, e.g.
   - Nonsynchronous trading
   - Bid-ask bounce
   - Impact of changes in tick size, after-hour trading, etc.
   - Impact of daily price limits (many foreign markets)

Nonsynchronous trading: Key implication: may induce serial correlations even when the underlying returns are iid.

Setup: log returns r_t are iid (μ, σ^2) For each time index t, P(no trade) $= \pi$. Cannot observe r_t if there is no trade. What is the observed log return series r_t^o? It turns out r_t^o is given in Tsay Eq. (5.1),

7.2 Bid-Ask Spread

7.3 Transaction Data

7.4 Price Change Models

7.4.1 Ordered Probit Model

7.4.2 Decomposition Model

7.5 Duration Models

7.5.1 ACD Model

7.5.2 Simulation

7.5.3 Estimation

7.6 Nonlinear Duration Models

8 Continuous-Time Models

Primary Text Reading. Tsay (2005, chap. 6)

Discrete-Time Models

8.1 Wiener Process

Here is some R code for generating a Wiener process.

```
n<-3000
epsi<-rnorm(n,0,1)
w=cumsum(epsi)/sqrt(n)
plot(w, type='l')
```

This is quite similar to the R code that generated Figure 4.6
`plot(cumsum(rnorm(20000)),type="l")`.

8.2 Itô's Lemma

Itô's lemma starts with an *Itô process*, an adapted stochastic process which can be expressed as the sum of an integral with respect to Brownian motion and an integral with respect to time,

$$X_t = X_0 + \int_0^t \sigma_s \, dB_s + \int_0^t \mu_s \, ds. \tag{8.1}$$

where B is a Brownian motion and it is required that σ is a predictable B-integrable process, and μ is predictable and Lebesgue integrable, and any twice continuously differentiable function f on the real numbers, then $f(X)$ is also an Itô process satisfying

$$\begin{aligned} df(X_t) &= f'(X_t)\, dX_t + \frac{1}{2} f''(X_t)\sigma_t^2 \, dt \\ &= f'(X_t)\sigma_t \, dB_t + \left(f'(X_t)\mu_t + \frac{1}{2} f''(X_t)\sigma_t^2 \right) dt. \end{aligned}$$

which by Itô's lemma

$$\begin{aligned} df(t, X_t) &= \dot{f}(t, X_t)\, dt + f'(t, X_t)\, dX_t + \frac{1}{2} f''(t, X_t)\sigma_t^2 \, dt, \\ &= \dot{f}(t, X_t)\, dt + f'(t, X_t)(\mu_t \, dt + \sigma_t \, dB_t) + \frac{1}{2} f''(t, X_t)\sigma_t^2 \, dt. \end{aligned} \tag{8.2}$$

The lemma result (8.2) is the stochastic calculus counterpart of the chain rule of Newtonian calculus, and it can be thought of as the Taylor series expansion and retaining the second order term related to the stochastic component change.

8.3 Stochastic Differentials

8.4 Stochastic Integrals

8.5 Jump Diffusion Model

References

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*, 307–327.

Crawley, M. J. (2007). *The R Book*. Hoboken, NJ: Wiley.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of variance of United Kingdom inflation. *Econometrica*, *50*, 987–1008.

Franses, P. H., & van Dijk, D. (2000). *Non-linear time series models in empirical finance*. Cambridge, UK: Cambridge University Press.

Graves, S. (2008). FinTS: Companion to Tsay (2005) Analysis of financial time series [Computer software manual]. Retrieved from `http://www.r-project.org` (R package version 0.3-6)

Levine, D. M., Stephan, D., Krehbiel, T. C., & Berenson, M. L. (2004). *Statistics for managers using Microsoft Excel* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Martinez, W. L., & Martinez, A. R. (2008). *Computational statistics with MATLAB* (2nd ed.). New York: Chapman & Hall/CRC.

Pilipovic, D. (2007). *Energy risk: Valuing and managing energy derivatives* (2nd ed.). New York: McGraw-Hill.

Pyle, D. (1999). *Data preparation for data mining*. San Francisco: Morgan-Kaufmann.

R Development Core Team. (2008). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org` (ISBN 3-900051-07-0)

Rogers, M. R. (1998). *Handbook of key economic indicators* (2nd ed.). New York: McGraw-Hill.

Tsay, R. S. (2005). *Analysis of financial time series* (2nd ed.). Hoboken, NJ: Wiley.

Tsay, R. S., & Tiao, G. C. (1984). Consistent estimates of autoregressive parameters and extended sample autocorrelation function for stationary and nonstationary ARMA models. *Journal of the American Statistical Association*, *79*(385), 84–96.

Wuertz, D., et al. (2007a). fArma: Rmetrics - ARMA time series modelling [Computer software manual]. Retrieved from `http://www.rmetrics.org` (R package version 260.72)

Wuertz, D., et al. (2007b). fGarch: Rmetrics - autoregressive conditional heteroskedastic modelling [Computer software manual]. Retrieved from `http://www.rmetrics.org` (R package version 260.72)

Wuertz, D., et al. (2007c). fSeries: Rmetrics - financial time series objects [Computer software manual]. Retrieved from `http://www.rmetrics.org` (R package version 260.73)