

SANJIV RANJAN DAS

DATA SCIENCE: THEORIES, MODELS, ALGORITHMS, AND ANALYTICS

S. R. DAS

Copyright © 2013, 2014, 2016 Sanjiv Ranjan Das

PUBLISHED BY S. R. DAS

[HTTP://ALGO.SCU.EDU/~SANJIVDAS/](http://ALGO.SCU.EDU/~SANJIVDAS/)

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this book except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This printing, July 2016

THE FUTURE IS ALREADY HERE; IT'S JUST NOT VERY EVENLY DISTRIBUTED.

– WILLIAM GIBSON

THE PUBLIC IS MORE FAMILIAR WITH BAD DESIGN THAN GOOD DESIGN. IT IS, IN EFFECT, CONDITIONED TO PREFER BAD DESIGN, BECAUSE THAT IS WHAT IT LIVES WITH. THE NEW BECOMES THREATENING, THE OLD REASSURING.

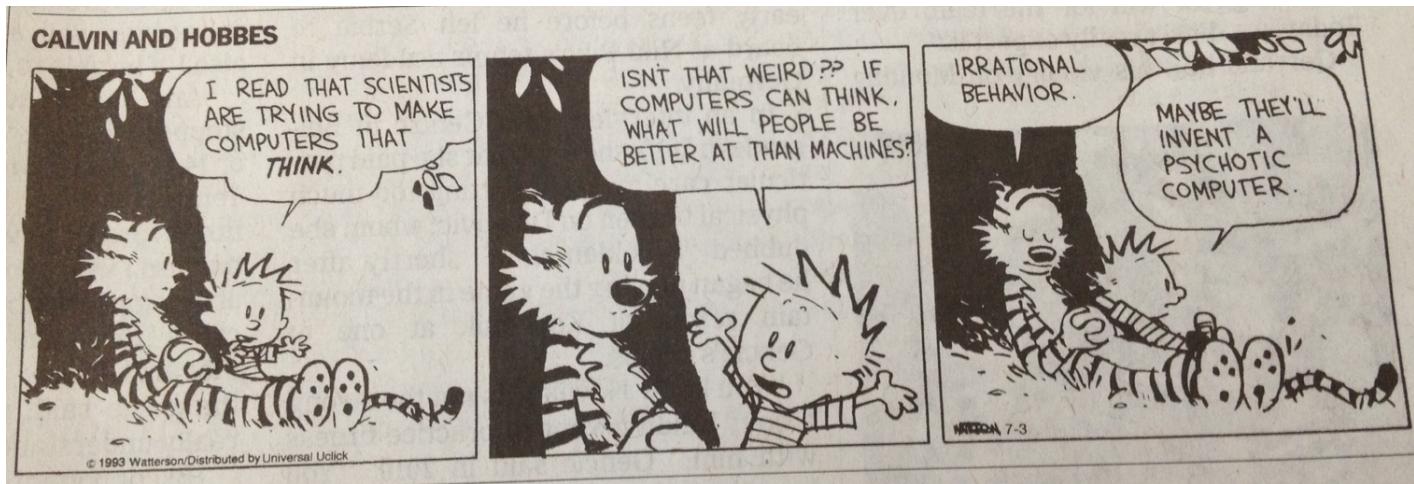
– PAUL RAND

IT SEEMS THAT PERFECTION IS ATTAINED NOT WHEN THERE IS NOTHING LEFT TO ADD, BUT WHEN THERE IS NOTHING MORE TO REMOVE.

– ANTOINE DE SAINT-EXUPÉRY

...IN GOD WE TRUST, ALL OTHERS BRING DATA.

– WILLIAM EDWARDS DEMING



Acknowledgements: I am extremely grateful to the following friends, students, and readers (mutually non-exclusive) who offered me feedback on these chapters. I am most grateful to John Heineke for his constant feedback and continuous encouragement. All the following students made helpful suggestions on the manuscript: Sumit Agarwal, Kevin Aguilar, Sankalp Bansal, Sivan Bershan, Ali Burney, Monalisa Chati, Jian-Wei Cheng, Chris Gadek, Karl Hennig, Pochang Hsu, Justin Ishikawa, Ravi Jagannathan, Alice Yehjin Jun, Seoyoung Kim, Ram Kumar, Federico Morales, Antonio Piccolboni, Shaharyar Shaikh, Jean-Marc Soumet, Rakesh Sountharajan, Greg Tseng, Dan Wong, Jeffrey Woo.

Contents

1	<i>The Art of Data Science</i>	25
1.1	<i>Volume, Velocity, Variety</i>	27
1.2	<i>Machine Learning</i>	29
1.3	<i>Supervised and Unsupervised Learning</i>	30
1.4	<i>Predictions and Forecasts</i>	30
1.5	<i>Innovation and Experimentation</i>	31
1.6	<i>The Dark Side</i>	31
1.6.1	<i>Big Errors</i>	31
1.6.2	<i>Privacy</i>	32
1.7	<i>Theories, Models, Intuition, Causality, Prediction, Correlation</i>	37
2	<i>The Very Beginning: Got Math?</i>	41
2.1	<i>Exponentials, Logarithms, and Compounding</i>	41
2.2	<i>Normal Distribution</i>	43
2.3	<i>Poisson Distribution</i>	43
2.4	<i>Moments of a continuous random variable</i>	44
2.5	<i>Combining random variables</i>	45
2.6	<i>Vector Algebra</i>	45
2.7	<i>Statistical Regression</i>	48
2.8	<i>Diversification</i>	49
2.9	<i>Matrix Calculus</i>	50
2.10	<i>Matrix Equations</i>	52

3	<i>Open Source: Modeling in R</i>	55
	3.1 <i>System Commands</i>	55
	3.2 <i>Loading Data</i>	56
	3.3 <i>Matrices</i>	58
	3.4 <i>Descriptive Statistics</i>	59
	3.5 <i>Higher-Order Moments</i>	61
	3.6 <i>Quick Introduction to Brownian Motions with R</i>	61
	3.7 <i>Estimation using maximum-likelihood</i>	62
	3.8 <i>GARCH/ARCH Models</i>	64
	3.9 <i>Introduction to Monte Carlo</i>	66
	3.10 <i>Portfolio Computations in R</i>	71
	3.11 <i>Finding the Optimal Portfolio</i>	72
	3.12 <i>Root Solving</i>	75
	3.13 <i>Regression</i>	77
	3.14 <i>Heteroskedasticity</i>	81
	3.15 <i>Auto-regressive models</i>	83
	3.16 <i>Vector Auto-Regression</i>	86
	3.17 <i>Logit</i>	90
	3.18 <i>Probit</i>	94
	3.19 <i>Solving Non-Linear Equations</i>	95
	3.20 <i>Web-Enabling R Functions</i>	97
4	<i>MoRe: Data Handling and Other Useful Things</i>	103
	4.1 <i>Data Extraction of stocks using quantmod</i>	103
	4.2 <i>Using the merge function</i>	109
	4.3 <i>Using the apply class of functions</i>	114
	4.4 <i>Getting interest rate data from FRED</i>	114
	4.5 <i>Cross-Sectional Data (an example)</i>	117
	4.6 <i>Handling dates with lubridate</i>	121
	4.7 <i>Using the data.table package</i>	124
	4.8 <i>Another data set: Bay Area Bike Share data</i>	128
	4.9 <i>Using the plyr package family</i>	130

5	<i>Being Mean with Variance: Markowitz Optimization</i>	135
	5.1 <i>Quadratic (Markowitz) Problem</i>	135
	5.1.1 <i>Solution in R</i>	137
	5.2 <i>Solving the problem with the quadprog package</i>	138
	5.3 <i>Tracing out the Efficient Frontier</i>	140
	5.4 <i>Covariances of frontier portfolios: r_p, r_q</i>	141
	5.5 <i>Combinations</i>	142
	5.6 <i>Zero Covariance Portfolio</i>	143
	5.7 <i>Portfolio Problems with Riskless Assets</i>	143
	5.8 <i>Risk Budgeting</i>	145
6	<i>Learning from Experience: Bayes Theorem</i>	149
	6.1 <i>Introduction</i>	149
	6.2 <i>Bayes and Joint Probability Distributions</i>	151
	6.3 <i>Correlated default (conditional default)</i>	152
	6.4 <i>Continuous and More Formal Exposition</i>	153
	6.5 <i>Bayes Nets</i>	156
	6.6 <i>Bayes Rule in Marketing</i>	159
	6.7 <i>Other Applications</i>	162
	6.7.1 <i>Bayes Models in Credit Rating Transitions</i>	162
	6.7.2 <i>Accounting Fraud</i>	162
	6.7.3 <i>Bayes was a Reverend after all...</i>	162
7	<i>More than Words: Extracting Information from News</i>	165
	7.1 <i>Prologue</i>	165
	7.2 <i>Framework</i>	167
	7.3 <i>Algorithms</i>	169
	7.3.1 <i>Crawlers and Scrapers</i>	169
	7.3.2 <i>Text Pre-processing</i>	172
	7.3.3 <i>The tm package</i>	175
	7.3.4 <i>Term Frequency - Inverse Document Frequency (TF-IDF)</i>	178
	7.3.5 <i>Wordclouds</i>	180
	7.3.6 <i>Regular Expressions</i>	181

7.4	<i>Extracting Data from Web Sources using APIs</i>	184
7.4.1	<i>Using Twitter</i>	184
7.4.2	<i>Using Facebook</i>	187
7.4.3	<i>Text processing, plain and simple</i>	190
7.4.4	<i>A Multipurpose Function to Extract Text</i>	191
7.5	<i>Text Classification</i>	193
7.5.1	<i>Bayes Classifier</i>	193
7.5.2	<i>Support Vector Machines</i>	198
7.5.3	<i>Word Count Classifiers</i>	200
7.5.4	<i>Vector Distance Classifier</i>	201
7.5.5	<i>Discriminant-Based Classifier</i>	202
7.5.6	<i>Adjective-Adverb Classifier</i>	204
7.5.7	<i>Scoring Optimism and Pessimism</i>	205
7.5.8	<i>Voting among Classifiers</i>	206
7.5.9	<i>Ambiguity Filters</i>	206
7.6	<i>Metrics</i>	207
7.6.1	<i>Confusion Matrix</i>	207
7.6.2	<i>Precision and Recall</i>	208
7.6.3	<i>Accuracy</i>	209
7.6.4	<i>False Positives</i>	209
7.6.5	<i>Sentiment Error</i>	210
7.6.6	<i>Disagreement</i>	210
7.6.7	<i>Correlations</i>	210
7.6.8	<i>Aggregation Performance</i>	211
7.6.9	<i>Phase-Lag Metrics</i>	213
7.6.10	<i>Economic Significance</i>	215
7.7	<i>Grading Text</i>	215
7.8	<i>Text Summarization</i>	216
7.9	<i>Discussion</i>	219
7.10	<i>Appendix: Sample text from Bloomberg for summarization</i>	221

8	<i>Virulent Products: The Bass Model</i>	227
	8.1 <i>Introduction</i>	227
	8.2 <i>Historical Examples</i>	227
	8.3 <i>The Basic Idea</i>	228
	8.4 <i>Solving the Model</i>	229
	8.4.1 <i>Symbolic math in R</i>	231
	8.5 <i>Software</i>	233
	8.6 <i>Calibration</i>	234
	8.7 <i>Sales Peak</i>	236
	8.8 <i>Notes</i>	238
9	<i>Extracting Dimensions: Discriminant and Factor Analysis</i>	241
	9.1 <i>Overview</i>	241
	9.2 <i>Discriminant Analysis</i>	241
	9.2.1 <i>Notation and assumptions</i>	242
	9.2.2 <i>Discriminant Function</i>	242
	9.2.3 <i>How good is the discriminant function?</i>	243
	9.2.4 <i>Caveats</i>	244
	9.2.5 <i>Implementation using R</i>	244
	9.2.6 <i>Confusion Matrix</i>	248
	9.2.7 <i>Multiple groups</i>	249
	9.3 <i>Eigen Systems</i>	250
	9.4 <i>Factor Analysis</i>	252
	9.4.1 <i>Notation</i>	252
	9.4.2 <i>The Idea</i>	253
	9.4.3 <i>Principal Components Analysis (PCA)</i>	253
	9.4.4 <i>Application to Treasury Yield Curves</i>	257
	9.4.5 <i>Application: Risk Parity and Risk Disparity</i>	260
	9.4.6 <i>Difference between PCA and FA</i>	260
	9.4.7 <i>Factor Rotation</i>	260
	9.4.8 <i>Using the factor analysis function</i>	261

10	<i>Bidding it Up: Auctions</i>	265
	<i>10.1 Theory</i>	265
	<i>10.1.1 Overview</i>	265
	<i>10.1.2 Auction types</i>	266
	<i>10.1.3 Value Determination</i>	266
	<i>10.1.4 Bidder Types</i>	267
	<i>10.1.5 Benchmark Model (BM)</i>	267
	<i>10.2 Auction Math</i>	268
	<i>10.2.1 Optimization by bidders</i>	269
	<i>10.2.2 Example</i>	270
	<i>10.3 Treasury Auctions</i>	272
	<i>10.3.1 DPA or UPA?</i>	272
	<i>10.4 Mechanism Design</i>	274
	<i>10.4.1 Collusion</i>	275
	<i>10.4.2 Clicks (Advertising Auctions)</i>	276
	<i>10.4.3 Next Price Auctions</i>	278
	<i>10.4.4 Laddered Auction</i>	279
11	<i>Truncate and Estimate: Limited Dependent Variables</i>	283
	<i>11.1 Introduction</i>	283
	<i>11.2 Logit</i>	284
	<i>11.3 Probit</i>	287
	<i>11.4 Analysis</i>	288
	<i>11.4.1 Slopes</i>	288
	<i>11.4.2 Maximum-Likelihood Estimation (MLE)</i>	292
	<i>11.5 Multinomial Logit</i>	293
	<i>11.6 Truncated Variables</i>	297
	<i>11.6.1 Endogeneity</i>	299
	<i>11.6.2 Example: Women in the Labor Market</i>	301
	<i>11.6.3 Endogeneity – Some Theory to Wrap Up</i>	303

12	<i>Riding the Wave: Fourier Analysis</i>	305
12.1	<i>Introduction</i>	305
12.2	<i>Fourier Series</i>	305
12.2.1	<i>Basic stuff</i>	305
12.2.2	<i>The unit circle</i>	305
12.2.3	<i>Angular velocity</i>	306
12.2.4	<i>Fourier series</i>	307
12.2.5	<i>Radians</i>	307
12.2.6	<i>Solving for the coefficients</i>	308
12.3	<i>Complex Algebra</i>	309
12.3.1	<i>From Trig to Complex</i>	310
12.3.2	<i>Getting rid of a_0</i>	311
12.3.3	<i>Collapsing and Simplifying</i>	311
12.4	<i>Fourier Transform</i>	312
12.4.1	<i>Empirical Example</i>	314
12.5	<i>Application to Binomial Option Pricing</i>	315
12.6	<i>Application to probability functions</i>	316
12.6.1	<i>Characteristic functions</i>	316
12.6.2	<i>Finance application</i>	316
12.6.3	<i>Solving for the characteristic function</i>	317
12.6.4	<i>Computing the moments</i>	318
12.6.5	<i>Probability density function</i>	318
13	<i>Making Connections: Network Theory</i>	321
13.1	<i>Overview</i>	321
13.2	<i>Graph Theory</i>	322
13.3	<i>Features of Graphs</i>	323
13.4	<i>Searching Graphs</i>	325
13.4.1	<i>Depth First Search</i>	325
13.4.2	<i>Breadth-first-search</i>	329

<i>13.5 Strongly Connected Components</i>	331
<i>13.6 Dijkstra's Shortest Path Algorithm</i>	333
<i>13.6.1 Plotting the network</i>	337
<i>13.7 Degree Distribution</i>	338
<i>13.8 Diameter</i>	340
<i>13.9 Fragility</i>	341
<i>13.10 Centrality</i>	341
<i>13.11 Communities</i>	346
<i>13.11.1 Modularity</i>	348
<i>13.12 Word of Mouth</i>	354
<i>13.13 Network Models of Systemic Risk</i>	355
<i>13.13.1 Systemic Score, Fragility, Centrality, Diameter</i>	355
<i>13.13.2 Risk Decomposition</i>	359
<i>13.13.3 Normalized Risk Score</i>	360
<i>13.13.4 Risk Increments</i>	361
<i>13.13.5 Criticality</i>	362
<i>13.13.6 Cross Risk</i>	364
<i>13.13.7 Risk Scaling</i>	365
<i>13.13.8 Too Big To Fail?</i>	367
<i>13.13.9 Application of the model to the banking network in India</i>	369
<i>13.14 Map of Science</i>	371
14 Statistical Brains: Neural Networks	377
<i>14.1 Overview</i>	377
<i>14.2 Nonlinear Regression</i>	378
<i>14.3 Perceptrons</i>	379
<i>14.4 Squashing Functions</i>	381
<i>14.5 How does the NN work?</i>	381
<i>14.5.1 Logit/Probit Model</i>	382
<i>14.5.2 Connection to hyperplanes</i>	382
<i>14.6 Feedback/Backpropagation</i>	382
<i>14.6.1 Extension to many perceptrons</i>	384

<i>14.7 Research Applications</i>	384
<i>14.7.1 Discovering Black-Scholes</i>	384
<i>14.7.2 Forecasting</i>	384
<i>14.8 Package neuralnet in R</i>	384
<i>14.9 Package nnet in R</i>	390
<i>15 Zero or One: Optimal Digital Portfolios</i>	393
<i>15.1 Modeling Digital Portfolios</i>	394
<i>15.2 Implementation in R</i>	398
<i>15.2.1 Basic recursion</i>	398
<i>15.2.2 Combining conditional distributions</i>	401
<i>15.3 Stochastic Dominance (SD)</i>	404
<i>15.4 Portfolio Characteristics</i>	407
<i>15.4.1 How many assets?</i>	407
<i>15.4.2 The impact of correlation</i>	409
<i>15.4.3 Uneven bets?</i>	410
<i>15.4.4 Mixing safe and risky assets</i>	411
<i>15.5 Conclusions</i>	412
<i>16 Against the Odds: Mathematics of Gambling</i>	415
<i>16.1 Introduction</i>	415
<i>16.1.1 Odds</i>	415
<i>16.1.2 Edge</i>	415
<i>16.1.3 Bookmakers</i>	416
<i>16.2 Kelly Criterion</i>	416
<i>16.2.1 Example</i>	416
<i>16.2.2 Deriving the Kelly Criterion</i>	418
<i>16.3 Entropy</i>	421
<i>16.3.1 Linking the Kelly Criterion to Entropy</i>	421
<i>16.3.2 Linking the Kelly criterion to portfolio optimization</i>	422
<i>16.3.3 Implementing day trading</i>	422

16.4	<i>Casino Games</i>	423
17	<i>In the Same Boat: Cluster Analysis and Prediction Trees</i>	427
17.1	<i>Introduction</i>	427
17.2	<i>Clustering using k-means</i>	427
17.2.1	<i>Example: Randomly generated data in kmeans</i>	430
17.2.2	<i>Example: Clustering of VC financing rounds</i>	432
17.2.3	<i>NCAA teams</i>	434
17.3	<i>Hierarchical Clustering</i>	436
17.4	<i>Prediction Trees</i>	436
17.4.1	<i>Classification Trees</i>	440
17.4.2	<i>The C4.5 Classifier</i>	442
17.5	<i>Regression Trees</i>	445
17.5.1	<i>Example: California Home Data</i>	447
18	<i>Bibliography</i>	451

List of Figures

- 1.1 The Four Vs of Big Data. 27
- 1.2 Google Flu Trends. The figure shows the high correlation between flu incidence and searches about “flu” on Google. The orange line is actual US flu activity, and the blue line is the Google Flu Trends estimate. 28
- 1.3 Profiling can convert mass media into personal media. 33
- 1.4 If it’s free, you may be the product. 34
- 1.5 Extracting consumer’s surplus through profiling. 36

- 3.1 Single stock path plot simulated from a Brownian motion. 67
- 3.2 Multiple stock path plot simulated from a Brownian motion. 68
- 3.3 Systematic risk as the number of stocks in the portfolio increases. 73
- 3.4 HTML code for the Rcgi application. 98
- 3.5 R code for the Rcgi application. 101

- 4.1 Plots of the six stock series extracted from the web. 105
- 4.2 Plots of the correlation matrix of six stock series extracted from the web. 108
- 4.3 Regression of stock average returns against systematic risk (β). 109
- 4.4 Google Finance: the AAPL web page showing the URL which is needed to download the page. 113
- 4.5 Failed bank totals by year. 122
- 4.6 Rape totals by year. 126
- 4.7 Rape totals by county. 129

- 5.1 The Efficient Frontier 141

- 6.1 Bayes net showing the pathways of economic distress. There are three channels: a is the inducement of industry distress from economy distress; b is the inducement of firm distress directly from economy distress; c is the inducement of firm distress directly from industry distress. 157
- 6.2 Article from the Scientific American on Bayes’ Theorem. 163

- 7.1 The data and algorithms pyramids. Depicts the inverse relationship between data volume and algorithmic complexity. 169
- 7.2 Quantity of hourly postings on message boards after selected news releases. Source: Das, Martinez-Jerez and Tufano (2005). 171
- 7.3 Subjective evaluation of content of post-news release postings on message boards. The content is divided into opinions, facts, and questions. Source: Das, Martinez-Jerez and Tufano (2005). 172
- 7.4 Frequency of posting by message board participants. 173
- 7.5 Frequency of posting by day of week by message board participants. 173
- 7.6 Frequency of posting by segment of day by message board participants. We show the average number of messages per day in the top panel and the average number of characters per message in the bottom panel. 174
- 7.7 Example of application of word cloud to the bio data extracted from the web and stored in a Corpus. 181
- 7.8 Plot of stock series (upper graph) versus sentiment series (lower graph). The correlation between the series is high. The plot is based on messages from Yahoo! Finance and is for a single twenty-four hour period. 211
- 7.9 Phase-lag analysis. The left-side shows the eight canonical graph patterns that are derived from arrangements of the start, end, high, and low points of a time series. The right-side shows the leads and lags of patterns of the stock series versus the sentiment series. A positive value means that the stock series leads the sentiment series. 214
- 8.1 Actual versus Bass model predictions for VCRs. 228
- 8.2 Actual versus Bass model predictions for answering machines. 229
- 8.3 Example of the adoption rate: $m = 100,000$, $p = 0.01$ and $q = 0.2$. 231
- 8.4 Example of the adoption rate: $m = 100,000$, $p = 0.01$ and $q = 0.2$. 232
- 8.5 Computing the Bass model integral using WolframAlpha. 234
- 8.6 Bass model forecast of Apple Inc's quarterly sales. The current sales are also overlaid in the plot. 237
- 8.7 Empirical adoption rates and parameters from the Bass paper. 237
- 8.8 Increase in peak time with $q \uparrow$ 240
- 10.1 Probability density function for the Beta ($a = 2$, $b = 4$) distribution. 271
- 10.2 Revenue in the DPA and UPA auctions. 273
- 10.3 Treasury auction markups. 274
- 10.4 Bid-Ask Spread in the Auction. 275
- 13.1 Comparison of random and scale-free graphs. From Barabasi, Albert-Laszlo., and Eric Bonabeau (2003). "Scale-Free Networks," *Scientific American* May, 50–59. 323

13.2 Microsoft academic search tool for co-authorship networks. See: http://academic.research.microsoft.com/.	
The top chart shows co-authors, the middle one shows citations, and the last one shows my Erdos number, i.e., the number of hops needed to be connected to Paul Erdos via my co-authors. My Erdos number is 3. Interestingly, I am a Finance academic, but my shortest path to Erdos is through Computer Science co-authors, another field in which I dabble.	326
13.3 Depth-first-search.	327
13.4 Depth-first search on a simple graph generated from a paired node list.	329
13.5 Breadth-first-search.	330
13.6 Strongly connected components. The upper graph shows the original network and the lower one shows the compressed network comprising only the SCCs. The algorithm to determine SCCs relies on two DFSs. Can you see a further SCC in the second graph?	
There should not be one.	332
13.7 Finding connected components on a graph.	334
13.8 Dijkstra's algorithm.	335
13.9 Network for computation of shortest path algorithm	336
13.10 Plot using the Fruchterman-Rheingold and Circle layouts	338
13.11 Plot of the Erdos-Renyi random graph	339
13.12 Plot of the degree distribution of the Erdos-Renyi random graph	340
13.13 Interbank lending networks by year. The top panel shows 2005, and the bottom panel is for the years 2006-2009.	345
13.14 Community versus centrality	354
13.15 Banking network adjacency matrix and plot	357
13.16 Centrality for the 15 banks.	359
13.17 Risk Decompositions for the 15 banks.	361
13.18 Risk Increments for the 15 banks.	363
13.19 Criticality for the 15 banks.	364
13.20 Spillover effects.	366
13.21 How risk increases with connectivity of the network.	368
13.22 How risk increases with connectivity of the network.	370
13.23 Screens for selecting the relevant set of Indian FIs to construct the banking network.	372
13.24 Screens for the Indian FIs banking network. The upper plot shows the entire network. The lower plot shows the network when we mouse over the bank in the middle of the plot. Red lines show that the bank is impacted by the other banks, and blue lines depict that the bank impacts the others, in a Granger causal manner.	373
13.25 Screens for systemic risk metrics of the Indian FIs banking network. The top plot shows the current risk metrics, and the bottom plot shows the history from 2008.	374
13.26 The Map of Science.	375

14.1 A feed-forward multilayer neural network.	380
14.2 The neural net for the <code>infert</code> data set with two perceptrons in a single hidden layer.	387
15.1 Plot of the final outcome distribution for a digital portfolio with five assets of outcomes {5,8,4,2,1} all of equal probability.	400
15.2 Plot of the final outcome distribution for a digital portfolio with five assets of outcomes {5,8,4,2,1} with unconditional probability of success of {0.1,0.2,0.1,0.05,0.15}, respectivly.	403
15.3 Plot of the difference in distribution for a digital portfolio with five assets when $\rho = 0.75$ minus that when $\rho = 0.25$. We use outcomes {5,8,4,2,1} with unconditional probability of success of {0.1,0.2,0.1,0.05,0.15}, respectivly.	405
15.4 Distribution functions for returns from Bernoulli investments as the number of investments (n) increases. Using the recursion technique we computed the probability distribution of the portfolio payoff for four values of $n = \{25,50,75,100\}$. The distribution function is plotted in the left panel. There are 4 plots, one for each n, and if we look at the bottom left of the plot, the leftmost line is for $n = 100$. The next line to the right is for $n = 75$, and so on. The right panel plots the value of $\int_0^u [G_{100}(x) - G_{25}(x)] dx$ for all $u \in (0,1)$, and confirms that it is always negative. The correlation parameter is $\rho = 0.25$.	408
15.5 Distribution functions for returns from Bernoulli investments as the correlation parameter (ρ^2) increases. Using the recursion technique we computed the probability distribution of the portfolio payoff for four values of $\rho = \{0.09,0.25,0.49,0.81\}$ shown by the black, red, green and blue lines respectively. The distribution function is plotted in the left panel. The right panel plots the value of $\int_0^u [G_{\rho=0.09}(x) - G_{\rho=0.81}(x)] dx$ for all $u \in (0,1)$, and confirms that it is always negative.	410
16.1 Bankroll evolution under the Kelly rule. The top plot follows the Kelly criterion, but the other two deviate from it, by overbetting or underbetting the fraction given by Kelly. The variables are: odds are 4 to 1, implying a house probability of $p = 0.2$, own probability of winning is $p^* = 0.25$.	419
16.2 See http://wizardofodds.com/gambling/house-edge/. The House Edge for various games. The edge is the same as $-f$ in our notation. The standard deviation is that of the bankroll of \$1 for one bet.	424
17.1 VC Style Clusters.	430
17.2 Two cluster example.	432
17.3 Five cluster example.	433
17.4 NCAA cluster example.	435
17.5 NCAA data, hierarchical cluster example.	437

17.6 NCAA data, hierarchical cluster example with clusters on the top two principal components.	438
17.7 Classification tree for the kyphosis data set.	443
17.8 Prediction tree for cars mileage.	448
17.9 California home prices prediction tree.	448
17.10 California home prices partition diagram.	449

List of Tables

- 3.1 Autocorrelation in daily, weekly, and monthly stock index returns. From Lo-Mackinlay, “A Non-Random Walk Down Wall Street”. 87
- 3.2 Cross autocorrelations in US stocks. From Lo-Macklinay, “A Non-Random Walk Down Wall Street.” 91
- 7.1 Correlations of Sentiment and Stock Returns for the MSH35 stocks and the aggregated MSH35 index. Stock returns (STKRET) are computed from close-to-close. We compute correlations using data for 88 days in the months of June, July and August 2001. Return data over the weekend is linearly interpolated, as messages continue to be posted over weekends. Daily sentiment is computed from midnight to close of trading at 4 pm (SENTY4pm). 212
- 13.1 Summary statistics and the top 25 banks ordered on eigenvalue centrality for 2005. The R -metric is a measure of whether failure can spread quickly, and this is so when $R \geq 2$. The diameter of the network is the length of the longest geodesic. Also presented in the second panel of the table are the centrality scores for 2005 corresponding to Figure 13.13. 344
- 15.1 Expected utility for Bernoulli portfolios as the number of investments (n) increases. The table reports the portfolio statistics for $n = \{25, 50, 75, 100\}$. Expected utility is given in the last column. The correlation parameter is $\rho = 0.25$. The utility function is $U(C) = (0.1 + C)^{1-\gamma}/(1 - \gamma), \gamma = 3$. 409
- 15.2 Expected utility for Bernoulli portfolios as the correlation (ρ) increases. The table reports the portfolio statistics for $\rho = \{0.09, 0.25, 0.49, 0.81\}$. Expected utility is given in the last column. The utility function is $U(C) = (0.1 + C)^{1-\gamma}/(1 - \gamma), \gamma = 3$. 411

15.3 Expected utility for Bernoulli portfolios when the portfolio comprises balanced investing in assets versus imbalanced weights. Both the balanced and imbalanced portfolio have $n = 25$ assets within them, each with a success probability of $q_i = 0.05$. The first has equal payoffs, i.e. $1/25$ each. The second portfolio has payoffs that monotonically increase, i.e. the payoffs are equal to $j/325, j = 1, 2, \dots, 25$. We note that the sum of the payoffs in both cases is 1. The correlation parameter is $\rho = 0.55$. The utility function is $U(C) = (0.1 + C)^{1-\gamma}/(1 - \gamma), \gamma = 3$. 411

15.4 Expected utility for Bernoulli portfolios when the portfolio comprises balanced investing in assets with identical success probabilities versus investing in assets with mixed success probabilities. Both the uniform and mixed portfolios have $n = 26$ assets within them. In the first portfolio, all the assets have a probability of success equal to $q_i = 0.10$. In the second portfolio, half the firms have a success probability of 0.05 and the other half have a probability of 0.15. The payoff of all investments is $1/26$. The correlation parameter is $\rho = 0.55$. The utility function is $U(C) = (0.1 + C)^{1-\gamma}/(1 - \gamma), \gamma = 3$.

412

Dedicated to Geetu, for decades of fun and friendship

1

The Art of Data Science

— “All models are wrong, but some are useful.”

George E. P. Box and N.R. Draper in “Empirical Model Building and Response Surfaces,” John Wiley & Sons, New York, 1987.

So you want to be a “data scientist”? There is no widely accepted definition of who a data scientist is.¹ Several books now attempt to define what data science is and who a data scientist may be, see Patil (2011), Patil (2012), and Loukides (2012). This book’s viewpoint is that a data scientist is someone who asks unique, interesting questions of data based on formal or informal theory, to generate rigorous and useful insights.² It is likely to be an individual with multi-disciplinary training in computer science, business, economics, statistics, and armed with the necessary quantity of domain knowledge relevant to the question at hand. The potential of the field is enormous for just a few well-trained data scientists armed with big data have the potential to transform organizations and societies. In the narrower domain of business life, the role of the data scientist is to generate applicable business intelligence.

Among all the new buzzwords in business – and there are many – “Big Data” is one of the most often heard. The burgeoning social web, and the growing role of the internet as the primary information channel of business, has generated more data than we might imagine. Users upload an hour of video data to YouTube every second.³ 87% of the U.S. population has heard of Twitter, and 7% use it.⁴ Forty-nine percent of Twitter users follow some brand or the other, hence the reach is enormous, and, as of 2014, there are more than 500 million tweets a day. But data is not information, and until we add analytics, it is just noise. And more, bigger, data may mean more noise and does not mean better data.

In many cases, less is more, and we need models as well. That is what this book is about, it’s about theories and models, with or without data,

¹ The term “data scientist” was coined by D.J. Patil. He was the Chief Scientist for LinkedIn. In 2011 Forbes placed him second in their Data Scientist List, just behind Larry Page of Google.

² To quote Georg Cantor - “In mathematics the art of proposing a question must be held of higher value than solving it.”

³ Mayer-Schönberger and Cukier (2013), p8. They report that USC’s Martin Hilbert calculated that more than 300 exabytes of data storage was being used in 2007, an exabyte being one billion gigabytes, i.e., 10^{18} bytes, and 2^{60} of binary usage.

⁴ In contrast, 88% of the population has heard of Facebook, and 41% use it. See www.convinceandconvert.com/7-surprising-statistics-about-twitter-in-america/. Half of Twitter users are white, and of the remaining half, half are black.

big or small. It's about analytics and applications, and a scientific approach to using data based on well-founded theory and sound business judgment. This book is about the science and art of data analytics.

Data science is transforming business. Companies are using medical data and claims data to offer incentivized health programs to employees. Caesar's Entertainment Corp. analyzed data for 65,000 employees and found substantial cost savings. Zynga Inc, famous for its game Farmville, accumulates 25 terabytes of data every day and analyzes it to make choices about new game features. UPS installed sensors to collect data on speed and location of its vans, which combined with GPS information, reduced fuel usage in 2011 by 8.4 million gallons, and shaved 85 million miles off its routes.⁵ McKinsey argues that a successful data analytics plan contains three elements: interlinked data inputs, analytics models, and decision-support tools.⁶ In a seminal paper, [Halevy, Norvig and Pereira \(2009\)](#), argue that even simple theories and models, with big data, have the potential to do better than complex models with less data.

In a recent talk⁷ well-regarded data scientist Hilary Mason emphasized that the creation of "data products" requires three components: data (of course) plus technical expertise (machine-learning) plus people and process (talent). Google Maps is a great example of a data product that epitomizes all these three qualities. She mentioned three skills that good data scientists need to cultivate: (a) in math and stats, (b) coding, (c) communication. I would add that preceding all these is the ability to ask relevant questions, the answers to which unlock value for companies, consumers, and society. Everything in data analytics begins with a clear problem statement, and needs to be judged with clear metrics.

Being a data scientist is inherently interdisciplinary. Good questions come from many disciplines, and the best answers are likely to come from people who are interested in multiple fields, or at least from teams that co-mingle varied skill sets. Josh Wills of Cloudera stated it well - "A data scientist is a person who is better at statistics than any software engineer and better at software engineering than any statistician." In contrast, complementing data scientists are business analytics people, who are more familiar with business models and paradigms and can ask good questions of the data.

⁵ "How Big Data is Changing the Whole Equation for Business," *Wall Street Journal* March 8, 2013.

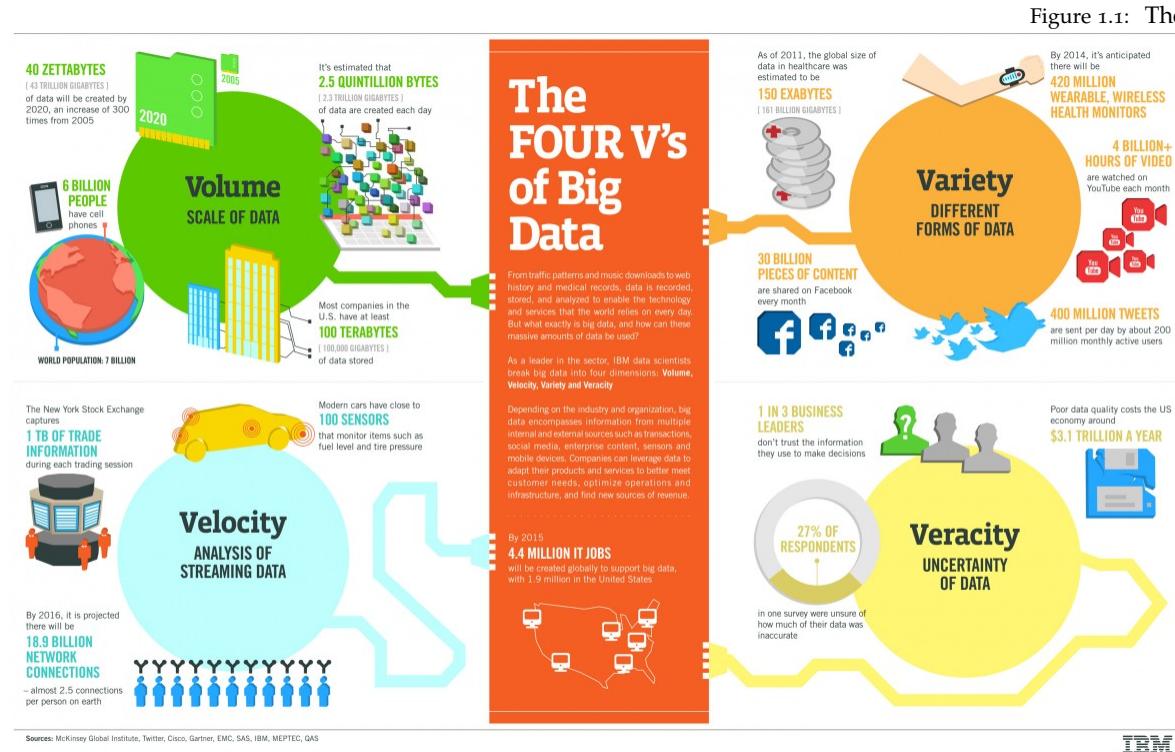
⁶ "Big Data: What's Your Plan?" *McKinsey Quarterly*, March 2013.

⁷ At the h2o world conference in the Bay Area, on 11th November 2015.

1.1 Volume, Velocity, Variety

There are several "V's" of big data: three of these are volume, velocity, variety.⁸ Big data exceeds the storage capacity of conventional databases. This is its *volume* aspect. The scale of data generation is mind-boggling. Google's Eric Schmidt pointed out that until 2003, all of human kind had generated just 5 exabytes of data (an exabyte is 1000^6 bytes or a billion-billion bytes). Today we generate 5 exabytes of data every two days. The main reason for this is the explosion of "interaction" data, a new phenomenon in contrast to mere "transaction" data. Interaction data comes from recording activities in our day-to-day ever more digital lives, such as browser activity, geo-location data, RFID data, sensors, personal digital recorders such as the fitbit and phones, satellites, etc. We now live in the "internet of things" (or IoT), and it's producing a wild quantity of data, all of which we seem to have an endless need to analyze. In some quarters it is better to speak of 4 Vs of big data, as shown in Figure 1.1.

⁸ This nomenclature was originated by the Gartner group in 2001, and has been in place more than a decade.



A good data scientist will be adept at managing volume not just technically in a database sense, but by building algorithms to make intelli-

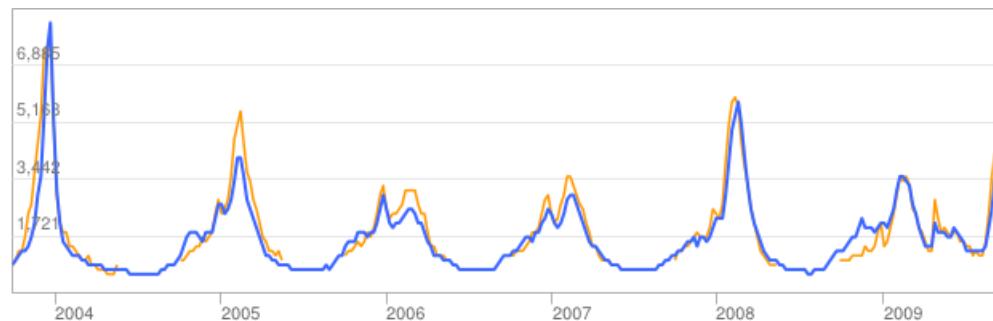
gent use of the size of the data as efficiently as possible. Things change when you have gargantuan data because almost all correlations become significant, and one might be tempted to draw spurious conclusions about causality. For many modern business applications today extraction of correlation is sufficient, but good data science involves techniques that extract causality from these correlations as well.

In many cases, detecting correlations is useful as is. For example, consider the classic case of Google Flu Trends, see Figure 1.2. The figure shows the high correlation between flu incidence and searches about “flu” on Google, see [Ginsberg et. al. \(2009\)](#). Obviously searches on the key word “flu” do not result in the flu itself! Of course, the incidence of searches on this key word is influenced by flu outbreaks. The interesting point here is that even though searches about flu do not *cause* flu, they *correlate with* it, and may at times even be predictive of it, simply because searches lead the actual reported levels of flu, as those may occur concurrently but take time to be reported. And whereas searches may be predictive, the cause of searches is the flu itself, one variable feeding on the other, in a repeat cycle.⁹ Hence, prediction is a major outcome of correlation, and has led to the recent buzz around the subfield of “predictive analytics.” There are entire conventions devoted to this facet of correlation, such as the wildly popular PAW (Predictive Analytics World).¹⁰ Pattern recognition is in, passe causality is out.

⁹ Interwoven time series such as these may be modeled using Vector Auto-Regressions, a technique we will encounter later in this book.

¹⁰ May be a futile collection of people, with non-working crystal balls, as William Gibson said - “The future is not google-able.”

Figure 1.2: Google Flu Trends. The figure shows the high correlation between flu incidence and searches about “flu” on Google. The orange line is actual US flu activity, and the blue line is the Google Flu Trends estimate.



Data *velocity* is accelerating. Streams of tweets, Facebook entries, financial information, etc., are being generated by more users at an ever increasing pace. Whereas velocity increases data volume, often exponentially, it might shorten the window of data retention or application. For example, high-frequency trading relies on micro-second information and streams of data, but the relevance of the data rapidly decays.

Finally, data *variety* is much greater than ever before. Models that relied on just a handful of variables can now avail of hundreds of variables, as computing power has increased. The scale of change in volume, velocity, and variety of the data that is now available calls for new econometrics, and a range of tools for even single questions. This book aims to introduce the reader to a variety of modeling concepts and econometric techniques that are essential for a well-rounded data scientist.

Data science is more than the mere analysis of large data sets. It is also about the creation of data. The field of “text-mining” expands available data enormously, since there is so much more text being generated than numbers. The creation of data from varied sources, and its quantification into information is known as “datafication.”

1.2 Machine Learning

Data science is also more than “machine learning,” which is about how systems learn from data. Systems may be trained on data to make decisions, and training is a continuous process, where the system updates its learning and (hopefully) improves its decision-making ability with more data. A spam filter is a good example of machine learning. As we feed it more data it keeps changing its decision rules, using a Bayesian filter, thereby remaining ahead of the spammers. It is this ability to adaptively learn that prevents spammers from gaming the filter, as highlighted in Paul Graham’s interesting essay titled “A Plan for Spam”.¹¹ Credit card approvals are also based on neural-nets, another popular machine learning technique. However, machine-learning techniques favor data over judgment, and good data science requires a healthy mix of both. Judgment is needed to accurately contextualize the setting for analysis and to construct effective models. A case in point is Vinny Bruzzese, known as the “mad scientist of Hollywood” who uses machine learning to predict movie revenues.¹² He asserts that mere machine learning would be insufficient to generate accurate predictions. He complements machine learning with judgment generated from interviews with screenwriters, surveys, etc., “to hear and understand the creative vision, so our analysis can be contextualized.”

Machine intelligence is re-emerging as the new incarnation of AI (a field that many feel has not lived up to its promise). Machine learning promises and has delivered on many questions of interest, and is also

¹¹ <http://www.paulgraham.com/spam.html>.

¹² “Solving Equation of a Hit Film Script, With Data,” *New York Times*, May 5, 2013.

proving to be quite a game-changer, as we will see later on in this chapter, and also as discussed in many preceding examples. What makes it so appealing? Hilary Mason suggests four characteristics of machine intelligence that make it interesting: (i) It is usually based on a theoretical breakthrough and is therefore well grounded in science. (ii) It changes the existing economic paradigm. (iii) The result is commoditization (e.g. Hadoop), and (iv) it makes available new data that leads to further data science.

1.3 Supervised and Unsupervised Learning

Systems may learn in two broad ways, through “supervised” and “unsupervised” learning. In supervised learning, a system produces decisions (outputs) based on input data. Both spam filters and automated credit card approval systems are examples of this type of learning. So is linear discriminant analysis (LDA). The system is given a historical data sample of inputs and known outputs, and it “learns” the relationship between the two using machine learning techniques, of which there are several. Judgment is needed to decide which technique is most appropriate for the task at hand.

Unsupervised learning is a process of reorganizing and enhancing the inputs in order to place structure on unlabeled data. A good example is cluster analysis, which takes a collection of entities, each with a number of attributes, and partitions the entity space into sets or groups based on closeness of the attributes of all entities. What this does is reorganizes the data, but it also enhances the data through a process of *labeling* the data with additional tags (in this case a cluster number/name). Factor analysis is also an unsupervised learning technique. The origin of this terminology is unclear, but it presumably arises from the fact that there is no clear objective function that is maximized or minimized in unsupervised learning, so that no “supervision” to reach an optimal is called for. However, this is not necessarily true in general, and we will see examples of unsupervised learning (such as community detection in the social web) where the outcome depends on measurable objective criteria.

1.4 Predictions and Forecasts

Data science is about making predictions and forecasts. There is a difference between the two. The statistician-economist Paul Saffo has sug-

gested that predictions aim to identify one outcome, whereas forecasts encompass a range of outcomes. To say that “it will rain tomorrow” is to make a prediction, but to say that “the chance of rain is 40%” (implying that the chance of no rain is 60%) is to make a forecast, as it lays out the range of possible outcomes with probabilities. We make weather forecasts, not predictions. Predictions are statements of great certainty, whereas forecasts exemplify the range of uncertainty. In the context of these definitions, the term predictive analytics is a misnomer for it’s goal is to make forecasts, not mere predictions.

1.5 Innovation and Experimentation

Data science is about new ideas and approaches. It merges new concepts with fresh algorithms. Take for example the A/B test, which is nothing but the online implementation of a real-time focus group. Different subsets of users are exposed to A and B stimuli respectively, and responses are measured and analyzed. It is widely used for web site design. This approach has been in place for more than a decade, and in 2011 Google ran more than 7,000 A/B tests. Facebook, Amazon, Netflix, and several others firms use A/B testing widely.¹³ The social web has become a teeming ecosystem for running social science experiments. The potential to learn about human behavior using innovative methods is much greater now than ever before.

¹³ “The A/B Test: Inside the Technology that’s Changing the Rules of Business,” by Brian Christian, *Wired*, April 2012.

1.6 The Dark Side

1.6.1 Big Errors

The good data scientist will take care to not over-reach in drawing conclusions from big data. Because there are so many variables available, and plentiful observations, correlations are often statistically significant, but devoid of basis. In the immortal words of the bard, empirical results from big data may be - “A tale told by an idiot, full of sound and fury, signifying nothing.”¹⁴ One must be careful not to read too much in the data. More data does not guarantee less noise, and signal extraction may be no easier than with less data.

Adding more columns (variables in the cross section) to the data set, but not more rows (time dimension) is also fraught with danger. As the number of variables increases, more characteristics are likely to be

¹⁴ William Shakespeare in *Macbeth*, Act V, Scene V.

related statistically. Over fitting models in-sample is much more likely with big data, leading to poor performance out-of-sample.

Researchers have also to be careful to explore the data fully, and not terminate their research the moment a viable result, especially one that the researcher is looking for, is attained. With big data, the chances of stopping at a suboptimal, or worse, intuitively appealing albeit wrong result become very high. It is like asking a question to a class of students. In a very large college class, the chance that someone will provide a plausible yet off-base answer quickly is very high, which often short circuits the opportunity for others in class to think more deeply about the question and provide a much better answer.

Nassim Taleb¹⁵ describes these issues elegantly - "I am not saying there is no information in big data. There is plenty of information. The problem – the central issue – is that the needle comes in an increasingly larger haystack." The fact is, one is not always looking for needles or Taleb's black swans, and there are plenty of normal phenomena about which robust forecasts are made possible by the presence of big data.

¹⁵ "Beware the Big Errors of Big Data" *Wired*, February 2013.

1.6.2 Privacy

The emergence of big data coincides with a gigantic erosion of privacy. Human kind has always been torn between the need for social interaction, and the urge for solitude and privacy. One trades off against the other. Technology has simply sharpened the divide and made the slope of this trade off steeper. It has provided tools of social interaction that steal privacy much faster than in the days before the social web.

Rumors and gossip are now old world. They required bilateral transmission. The social web provides multilateral revelation, where privacy no longer capitulates a battle at a time, but the entire war is lost at one go. And data science is the tool that enables firms, governments, individuals, benefactors and predators, et al, en masse, to feed on privacy's carcass. The cartoon in Figure 1.3 parodies the kind of information specialization that comes with the loss of privacy!

The loss of privacy is manifested in the practice of *human profiling* through data science. Our web presence increases entropically as we move more of our life's interactions to the web, be they financial, emotional, organizational, or merely social. And as we live more and more of our lives in this new social milieu, data mining and analytics enables companies to construct very accurate profiles of who we are, often better



Figure 1.3: Profiling can convert mass media into personal media.

than what we might do ourselves. We are moving from "know thyself" to knowing everything about almost everyone.

If you have a Facebook or Twitter presence, rest assured you have been profiled. For instance, let's say you tweeted that you were taking your dog for a walk. Profiling software now increments your profile with an additional tag - pet owner. An hour later you tweet that you are returning home to cook dinner for your kids. You profile is now further tagged as a parent. As you might imagine, even a small Twitter presence ends up being dramatically revealing about who you are. Information that you provide on Facebook and Twitter, your credit card spending pattern, and your blog, allows the creation of a profile that is accurate and comprehensive, and probably more objective than the subjective and biased opinion that you have of yourself. A machine knows thyself better. And you are the product! (See Figure 1.4.)

Humankind leaves an incredible trail of "digital exhaust" comprising phone calls, emails, tweets, GPS information, etc., that companies use for profiling. It is said that 1/3 of people have a digital identity before being born, initiated with the first sonogram from a routine hospital visit by an expectant mother. The half life of non-digital identity, or the average

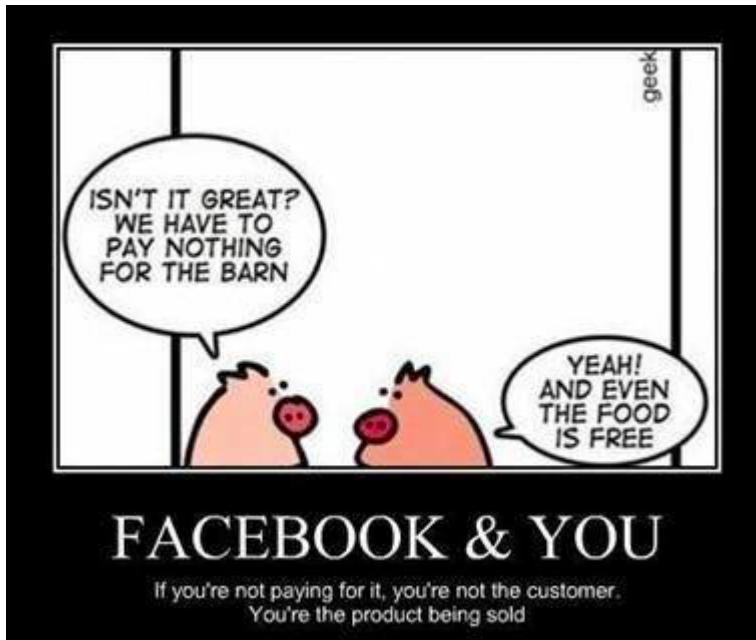


Figure 1.4: If it's free, you may be the product.

age of digital birth is six months, and within two years 92% of the US population has a digital identity.¹⁶ Those of us who claim to be safe from revealing their privacy by avoiding all forms of social media are simply profiled as agents with a “low digital presence.” It might be interesting to ask such people whether they would like to reside in a profile bucket that is more likely to attract government interest than a profile bucket with more average digital presence. In this age of profiling, the best way to remain inconspicuous is not to hide, but to remain as average as possible, so as to be mostly lost within a large herd.

Privacy is intricately and intrinsically connected to *security* and *efficiency*. The increase in transacting on the web, and the confluence of profiling, has led to massive identity theft. Just as in the old days, when a thief picked your lock and entered your home, most of your possessions were at risk. It is the same with electronic break ins, except that there are many more doors to break in from and so many more windows through which an intruder can unearth revealing information. And unlike a thief who breaks into your home, a hacker can reside in your electronic abode for quite some time without being detected, an invisible parasite slowly doing damage. While you are blind, you are being robbed blind. And unlike stealing your worldly possessions, stealing your very persona and identity is the cruellest cut of them all.

¹⁶ See “The Human Face of Big Data” by Rick Smolan and Jennifer Erwitt.

An increase in efficiency in the web ecosystem comes too at some retrenchment of privacy. Who does not shop on the internet? Each transaction resides in a separate web account. These add up at an astonishing pace. I have no idea of the exact number of web accounts in my name, but I am pretty sure it is over a hundred, many of them used maybe just once. I have unconsciously, yet quite willingly, marked my territory all over the e-commerce landscape. I rationalize away this loss of privacy in the name of efficiency, which undoubtedly exists. Every now and then I am reminded of this loss of privacy as my plane touches down in New York city, and like clockwork, within an hour or two, I receive a discount coupon in my email from Barnes & Noble bookstores. You see, whenever I am in Manhattan, I frequent the B&N store on the upper west side, and my credit card company and/or Google knows this, as well as my air travel schedule, since I buy both tickets and books on the same card and in the same browser. So when I want to buy books at a store discount, I fly to New York. That's how rational I am, or how rational my profile says I am! Humor aside, such profiling seems scary, though the thought quickly passes. I like the dopamine rush I get from my discount coupon and I love buying books.¹⁷

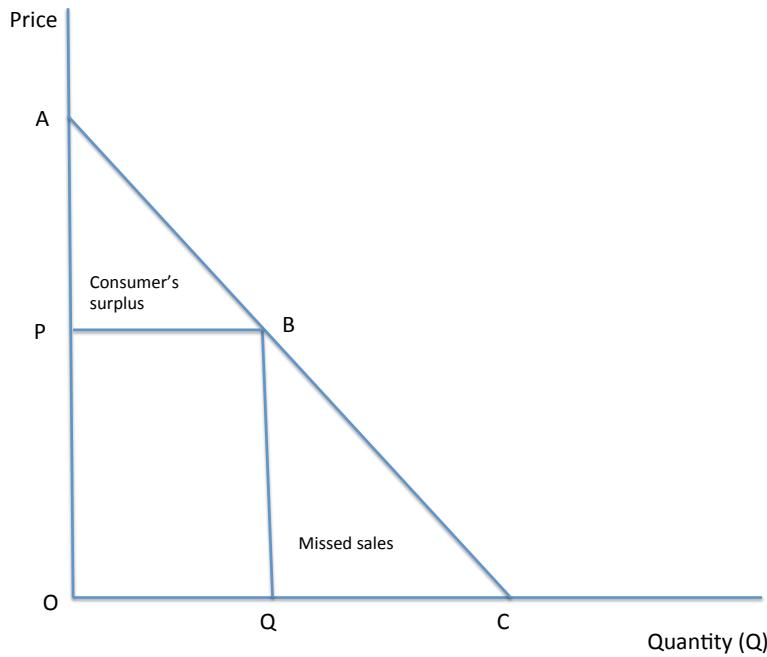
Profiling implies a partitioning of the social space into targeted groups, so that focused attention may be paid to specific groups, or various groups may be treated differently through *price discrimination*. If my profile shows me to be an affluent person who likes fine wine (both facts untrue in my case, but hope springs eternal), then internet sales pitches (via Groupon, Living Social, etc.) will be priced higher to me by an online retailer than to someone whose profile indicates a low spend. Profiling enables retailers to maximize revenues by eating away the consumer's surplus by better setting of prices to each buyer's individual willingness to pay. This is depicted in Figure 1.5.

In Figure 1.5 the demand curve is represented by the line segment ABC representing price-quantity combinations (more is demanded at lower prices). In a competitive market without price segmentation, let's assume that the equilibrium price is P and equilibrium quantity is Q as shown by the point B on the demand curve. (The upward sloping supply curve is not shown but it must intersect the demand curve at point B , of course.) Total revenue to the seller is the area $OPBQ$, i.e., $P \times Q$.

Now assume that the seller is able to profile buyers so that price dis-

¹⁷ I also like writing books, but I am much better at buying them, and somewhat less better at reading them!

Figure 1.5: Extracting consumer's surplus through profiling.



crimination is possible. Based on buyers' profiles, the seller will offer each buyer the price he is willing to pay on the demand curve, thereby picking off each price in the segment AB . This enables the seller to capture the additional region ABP , which is the area of consumer's surplus, i.e., the difference between the price that buyers pay versus the price they were actually willing to pay. The seller may also choose to offer some consumers lower prices in the region BC of the demand curve so as to bring in additional buyers whose threshold price lies below the competitive market price P . Thus, profiling helps sellers capture consumer's surplus and eat into the region of missed sales. Targeting brings benefits to sellers and they actively pursue it. The benefits outweigh the costs of profiling, and the practice is widespread as a result. Profiling also makes price segmentation fine-tuned, and rather than break buyers into a few segments, usually two, each profile becomes a separate segment, and the granularity of price segmentation is modulated by the number of profiling groups the seller chooses to model.

Of course, there is an insidious aspect to profiling, which has existed for quite some time, such as targeting conducted by tax authorities. I

don't believe we will take kindly to insurance companies profiling us any more than they already do. Profiling is also undertaken to snare terrorists. However, there is a danger in excessive profiling. A very specific profile for a terrorist makes it easier for their ilk to game detection as follows. Send several possible suicide bombers through airport security and see who is repeatedly pulled aside for screening and who is not. Repeating this exercise enables a terrorist cell to learn which candidates do not fall into the profile. They may then use them for the execution of a terror act, as they are unlikely to be picked up for special screening. The antidote? Randomization of people picked for special screening in searches at airports, which makes it hard for a terrorist to always assume no likelihood of detection through screening.¹⁸

Automated invasions of privacy naturally lead to a human response, not always rational or predictable. This is articulated in Campbell's Law: "The more any quantitative social indicator (or even some qualitative indicator) is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor."¹⁹ We are in for an interesting period of interaction between man and machine, where the battle for privacy will take center stage.

1.7 Theories, Models, Intuition, Causality, Prediction, Correlation

My view of data science is one where theories are implemented using data, some of it big data. This is embodied in an inference stack comprising (in sequence): theories, models, intuition, causality, prediction, and correlation. The first three constructs in this chain are from Emanuel Derman's wonderful book on the pitfalls of models.²⁰

Theories are statements of how the world should be or is, and are derived from axioms that are assumptions about the world, or precedent theories. Models are implementations of theory, and in data science are often algorithms based on theories that are run on data. The results of running a model lead to intuition, i.e., a deeper understanding of the world based on theory, model, and data. Whereas there are schools of thought that suggest data is all we need, and theory is obsolete, this author disagrees. Still the unreasonable proven effectiveness of big data cannot be denied. Chris Anderson argues in his *Wired* magazine article

¹⁸ See http://acfnews.org.s60463.gridserver.com/science/random_security.html, also aired on KRON-TV, San Francisco, 2/3/2003.

¹⁹ See: http://en.wikipedia.org/wiki/Campbell's_law.

²⁰ "Models. Behaving. Badly." Emanuel Derman, *Free Press*, New York, 2011.

thus:²¹

Sensors everywhere. Infinite storage. Clouds of processors. Our ability to capture, warehouse, and understand massive amounts of data is changing science, medicine, business, and technology. As our collection of facts and figures grows, so will the opportunity to find answers to fundamental questions. Because in the era of big data, more isn't just more. More is different.

In contrast, the academic Thomas Davenport writes in his foreword to [Seigel \(2013\)](#) that models are key, and should not be increasingly eschewed with increasing data:

But the point of predictive analytics is not the relative size or unruliness of your data, but what you do with it. I have found that “big data often means small math,” and many big data practitioners are content just to use their data to create some appealing visual analytics. That’s not nearly as valuable as creating a predictive model.

Once we have established intuition for the results of a model, it remains to be seen whether the relationships we observe are causal, predictive, or merely correlational. Theory may be causal and tested as such. [Granger \(1969\)](#) causality is often stated in mathematical form for two stationary²² time series of data as follows. X is said to Granger cause Y if in the following equation system,

$$\begin{aligned} Y(t) &= a_1 + b_1 Y(t-1) + c_1 X(t-1) + e_1 \\ X(t) &= a_2 + b_2 Y(t-1) + c_2 X(t-1) + e_2 \end{aligned}$$

the coefficient c_1 is significant and b_2 is not significant. Hence, X causes Y , but not vice versa. Causality is a hard property to establish, even with theoretical foundation, as the causal effect has to be well-entrenched in the data.

We have to be careful to impose judgment as much as possible since statistical relationships may not always be what they seem. A variable may satisfy the Granger causality regressions above but may not be causal. For example, we earlier encountered the flu example in Google Trends. If we denote searches for flu as X , and the outbreak of flu as Y , we may see a Granger cause relation between flu and searches for it. This does not mean that searching for flu *causes* flu, yet searches are *predictive* of flu. This is the essential difference between prediction and causality.

²¹ “The End of Theory: The Data Deluge Makes the Scientific Method Obsolete.” *Wired*, v16(7), 23rd June, 2008.

²² A series is stationary if the probability distribution from which the observations are drawn is the same at all points in time.

And then there is correlation, at the end of the data science inference chain. Contemporaneous movement between two variables is quantified using correlation. In many cases, we uncover correlation, but no prediction or causality. Correlation has great value to firms attempting to tease out beneficial information from big data. And even though it is a linear relationship between variables, it lays the groundwork for uncovering nonlinear relationships, which are becoming easier to detect with more data. The surprising parable about Walmart finding that purchases of beer and diapers seem to be highly correlated resulted in these two somewhat oddly-paired items being displayed on the same aisle in supermarkets.²³ Unearthing correlations of sales items across the population quickly lead to different business models aimed at exploiting these correlations, such as my book buying inducement from Barnes & Noble, where my “fly and buy” predilection is easily exploited. Correlation is often all we need, eschewing human cravings for causality. As Mayer-Schönberger and Cukier (2013) so aptly put it, we are satisfied “... not knowing *why* but only *what*.”

In the data scientist mode of thought, *relationships* are multifaceted correlations amongst people. Facebook, Twitter, and many other platforms are datafying human relationships using graph theory, exploiting the social web in an attempt to understand better how people relate to each other, with the goal of profiting from it. We use correlations on networks to mine the social graph, understanding better how different social structures may be exploited. We answer questions such as where to seed a new marketing campaign, which members of a network are more important than the others, how quickly will information spread on the network, i.e., how strong is the “network effect”?

Data science is about the quantization and understanding of human behavior, the holy grail of social science. In the following chapters we will explore a wide range of theories, techniques, data, and applications of a multi-faceted paradigm. We will also review the new technologies developed for big data and data science, such as distributed computing using the Dean and Ghemawat (2004) MapReduce paradigm developed at Google,²⁴ and implemented as the open source project Hadoop at Yahoo!.²⁵ When data gets super sized, it is better to move algorithms to the data than the other way around. Just as big data has inverted database paradigms, so is big data changing the nature of inference in the study of human behavior. Ultimately, data science is a way of thinking, for

²³ http://www.theregister.co.uk/2006/08/15/beer_diapers/.

²⁴ <http://research.google.com/archive/mapreduce.html>

²⁵ <http://hadoop.apache.org/>

social scientists, using computer science.

2

The Very Beginning: Got Math?

Business analytics requires the use of various quantitative tools, from algebra and calculus, to statistics and econometrics, with implementations in various programming languages and software. It calls for technical expertise as well as good judgment, and the ability to ask insightful questions and to deploy data towards answering the questions.

The presence of the web as the primary platform for business and marketing has spawned huge quantities of data, driving firms to attempt to exploit vast stores of information in honing their competitive edge. As a consequence, firms in Silicon Valley (and elsewhere) are hiring a new breed of employee known as “data scientist” whose role is to analyze “Big Data” using tools such as the ones you will learn in this course.

This chapter will review some of the mathematics, statistics, linear algebra, and calculus you might have not used in many years. It is more fun than it looks. We will also learn to use some mathematical packages along the way. We’ll revisit some standard calculations and analyses that you will have encountered in previous courses you might have taken. You will refresh some old concepts, learn new ones, and become technically adept with the tools of the trade.

2.1 Exponentials, Logarithms, and Compounding

It is fitting to begin with the fundamental mathematical constant, “ $e = 2.718281828\dots$ ”, which is also the function “ $\exp(\cdot)$ ”. We often write this function as e^x , where x can be a real or complex variable. It shows up in many places, especially in Finance, where it is used for continuous compounding and discounting of money at a given interest rate r over some time horizon t .

Given $y = e^x$, a fixed change in x results in the same continuous

percentage change in y . This is because $\ln(y) = x$, where $\ln(\cdot)$ is the natural logarithm function, and is the inverse function of the exponential function. Recall also that the first derivative of this function is $\frac{dy}{dx} = e^x$, i.e., the function itself.

The constant e is defined as the limit of a specific function:

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

Exponential compounding is the limit of successively shorter intervals over discrete compounding. Given a horizon t divided into n intervals per year, one dollar compounded from time zero to time t years over these n intervals at per annum rate r may be written as $(1 + \frac{r}{n})^{nt}$. Continuous-compounding is the limit of this equation when the number of periods n goes to infinity:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{r}{n}\right)^{nt} = \lim_{n \rightarrow \infty} \left[\left(1 + \frac{1}{n/r}\right)^{n/r} \right]^{tr} = e^{rt}$$

This is the forward value of one dollar. Present value is just the reverse. Therefore, the price today of a dollar received t years from today is $P = e^{-rt}$. The yield of a bond is:

$$r = -\frac{1}{t} \ln(P)$$

In bond mathematics, the negative of the percentage price sensitivity of a bond to changes in interest rates is known as “Duration”:

$$-\frac{dP}{dr} \frac{1}{P} = -\left(-te^{-rt} \frac{1}{P}\right) = tP \frac{1}{P} = t.$$

The derivative $\frac{dP}{dr}$ is the price sensitivity of the bond to changes in interest rates, and is negative. Further dividing this by P gives the percentage price sensitivity. The minus sign in front of the definition of duration is applied to convert the negative number to a positive one.

The “Convexity” of a bond is its percentage price sensitivity relative to the second derivative, i.e.,

$$\frac{d^2P}{dr^2} \frac{1}{P} = t^2 P \frac{1}{P} = t^2.$$

Because the second derivative is positive, we know that the bond pricing function is convex.

2.2 Normal Distribution

This distribution is the workhorse of many models in the social sciences, and is assumed to generate much of the data that comprises the Big Data universe. Interestingly, most phenomena (variables) in the real world are not normally distributed. They tend to be “power law” distributed, i.e., many observations of low value, and very few of high value. The probability distribution declines from left to right and does not have the characteristic hump shape of the normal distribution. An example of data that is distributed thus is income distribution (many people with low income, very few with high income). Other examples are word frequencies in languages, population sizes of cities, number of connections of people in a social network, etc.

Still, we do need to learn about the normal distribution because it is important in statistics, and the central limit theorem does govern much of the data we look at. Examples of approximately normally distributed data are stock returns, and human heights.

If $x \sim N(\mu, \sigma^2)$, that is, x is normally distributed with mean μ and variance σ^2 , then the probability “density” function for x is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right]$$

The cumulative probability is given by the “distribution” function

$$F(x) = \int_{-\infty}^x f(u)du$$

and

$$F(x) = 1 - F(-x)$$

because the normal distribution is symmetric. We often also use the notation $N(\cdot)$ or $\Phi(\cdot)$ instead of $F(\cdot)$.

The “standard normal” distribution is: $x \sim N(0, 1)$. For the standard normal distribution: $F(0) = \frac{1}{2}$. The normal distribution has continuous support, i.e., a range of values of x that goes continuously from $-\infty$ to $+\infty$.

2.3 Poisson Distribution

The Poisson is also known as the rare-event distribution. Its density function is:

$$f(n; \lambda) = \frac{e^{-\lambda} \lambda^n}{n!}$$

where there is only one parameter, i.e., the mean λ . The density function is over discrete values of n , the number of occurrences given the mean number of outcomes λ . The mean and variance of the Poisson distribution are both λ . The Poisson is a discrete-support distribution, with a range of values $n = \{0, 1, 2, \dots\}$.

2.4 Moments of a continuous random variable

The following formulae are useful to review because any analysis of data begins with descriptive statistics, and the following statistical “moments” are computed in order to get a first handle on the data. Given a random variable x with probability density function $f(x)$, then the following are the first four moments.

$$\text{Mean (first moment or average)} = E(x) = \int xf(x)dx$$

In like fashion, powers of the variable result in higher (n -th order) moments. These are “non-central” moments, i.e., they are moments of the raw random variable x , not its deviation from its mean, i.e., $[x - E(x)]$.

$$n^{\text{th}} \text{ moment} = E(x^n) = \int x^n f(x)dx$$

Central moments are moments of de-meanned random variables. The second central moment is the variance:

$$\text{Variance} = Var(x) = E[x - E(x)]^2 = E(x^2) - [E(x)]^2$$

The standard deviation is the square-root of the variance, i.e., $\sigma = \sqrt{Var(x)}$. The third central moment, normalized by the standard deviation to a suitable power is the skewness:

$$\text{Skewness} = \frac{E[x - E(x)]^3}{Var(x)^{3/2}}$$

The absolute value of skewness relates to the degree of asymmetry in the probability density. If more extreme values occur to the left than the right, the distribution is left-skewed. And vice-versa, the distribution is right-skewed.

Correspondingly, the fourth central, normalized moment is kurtosis.

$$\text{Kurtosis} = \frac{E[x - E(x)]^4}{[Var(x)]^2}$$

Kurtosis in the normal distribution has value 3. We define “Excess Kurtosis” to be Kurtosis minus 3. When a probability distribution has positive excess kurtosis we call it “leptokurtic”. Such distributions have fatter tails (either or both sides) than a normal distribution.

2.5 Combining random variables

Since we often have to deal with composites of random variables, i.e., more than one random variable, we review here some simple rules for moments of combinations of random variables. There are several other expressions for the same equations, but we examine just a few here, as these are the ones we will use more frequently.

First, we see that means are additive and scalable, i.e.,

$$E(ax + by) = aE(x) + bE(y)$$

where x, y are random variables, and a, b are scalar constants. The variance of scaled, summed random variables is as follows:

$$\text{Var}(ax + by) = a^2\text{Var}(x) + b^2\text{Var}(y) + 2ab\text{Cov}(x, y) \quad (2.1)$$

And the covariance and correlation between two random variables is

$$\begin{aligned} \text{Cov}(x, y) &= E(xy) - E(x)E(y) \\ \text{Corr}(x, y) &= \frac{\text{Cov}(x, y)}{\sqrt{\text{Var}(x)\text{Var}(y)}} \end{aligned}$$

Students of finance will be well-versed with these expressions. They are facile and easy to implement.

2.6 Vector Algebra

We will be using linear algebra in many of the models that we explore in this book. Linear algebra requires the manipulation of vectors and matrices. We will also use vector calculus. Vector algebra and calculus are very powerful methods for tackling problems that involve solutions in spaces of several variables, i.e., in high dimension. The parsimony of using vector notation will become apparent as we proceed. This introduction is very light and meant for the reader who is mostly uninitiated in linear algebra.

Rather than work with an abstract exposition, it is better to introduce ideas using an example. We'll examine the use of vectors in the context

of stock portfolios. We define the returns for each stock in a portfolio as:

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ \vdots \\ R_N \end{pmatrix}$$

This is a random vector, because each return $R_i, i = 1, 2, \dots, N$ comes from its own distribution, and the returns of all these stocks are correlated. This random vector's probability is represented as a joint or multivariate probability distribution. Note that we use a bold font to denote the vector \mathbf{R} .

We also define a Unit vector:

$$\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}$$

The use of this unit vector will become apparent shortly, but it will be used in myriad ways and is a useful analytical object.

A *portfolio* vector is defined as a set of portfolio weights, i.e., the fraction of the portfolio that is invested in each stock:

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_N \end{pmatrix}$$

The total of portfolio weights must add up to 1.

$$\sum_{i=1}^N w_i = 1, \quad \mathbf{w}' \mathbf{1} = 1$$

Pay special attention to the line above. In it, there are two ways in which to describe the sum of portfolio weights. The first one uses summation notation, and the second one uses a simple vector algebraic statement, i.e., that the transpose of \mathbf{w} , denoted \mathbf{w}' times the unit vector $\mathbf{1}$ equals 1.¹ The two elements on the left-hand-side of the equation are vectors, and the 1 on the right hand side is a scalar. The dimension of \mathbf{w}' is $(1 \times N)$

¹ Often, the notation for transpose is to superscript T , and in this case we would write this as w^\top . We may use either notation in the rest of the book.

and the dimension of $\mathbf{1}$ is $(N \times 1)$. And a $(1 \times N)$ vector multiplied by a $(N \times 1)$ results in a (1×1) vector, i.e., a scalar.

We may also invest in a risk free asset (denoted as asset zero, $i = 0$), with return $R_0 = r_f$. In this case, the total portfolio weights including that of the risk free asset must sum to 1, and the weight w_0 is:

$$w_0 = 1 - \sum_{i=1}^N w_i = 1 - \mathbf{w}^\top \mathbf{1}$$

Now we can use vector notation to compute statistics and quantities of the portfolio. The portfolio return is

$$R_p = \sum_{i=1}^N w_i R_i = \mathbf{w}' \mathbf{R}$$

Again, note that the left-hand-side quantity is a scalar, and the two right-hand-side quantities are vectors. Since \mathbf{R} is a random vector, R_p is a random (scalar, i.e., not a vector, of dimension 1×1) variable. Such a product is called a scalar product of two vectors. In order for the calculation to work, the two vectors or matrices must be “conformable” i.e., the inner dimensions of the matrices must be the same. In this case we are multiplying \mathbf{w}' of dimension $1 \times N$ with \mathbf{R} of dimension $N \times 1$ and since the two “inside” dimensions are both n , the calculation is proper as the matrices are conformable. The result of the calculation will take the size of the “outer” dimensions, i.e., in this case 1×1 . Now, suppose

$$\mathbf{R} \sim MVN[\boldsymbol{\mu}; \boldsymbol{\Sigma}]$$

That is, returns are multivariate normally distributed with mean vector $E[\mathbf{R}] = \boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_N]' \in R^N$ and covariance matrix $\boldsymbol{\Sigma} \in R^{N \times N}$. The notation R^N stands for a “real-valued matrix of dimension N .” If it’s just N , then it means a vector of dimension N . If it’s written as $N \times M$, then it’s a matrix of that dimension, i.e., N rows and M columns.

We can write the portfolio’s mean return as:

$$E[\mathbf{w}' \mathbf{R}] = \mathbf{w}' E[\mathbf{R}] = \mathbf{w}' \boldsymbol{\mu} = \sum_{i=1}^N w_i \mu_i$$

The portfolio’s return variance is

$$Var(R_p) = Var(\mathbf{w}' \mathbf{R}) = \mathbf{w}' \boldsymbol{\Sigma} \mathbf{w}$$

Showing why this is true is left as an exercise to the reader. Take a case where $N = 2$ and write out the expression for the variance of the portfolio using equation 2.1. Then also undertake the same calculation using

the variance formula $\mathbf{w}'\Sigma\mathbf{w}$ and see the equivalence. Also note carefully that this expression works because Σ is a symmetric matrix. The multivariate normal density function is:

$$f(\mathbf{R}) = \frac{1}{2\pi^{N/2}\sqrt{|\Sigma|}} \exp\left[-\frac{1}{2}(\mathbf{R} - \boldsymbol{\mu})'\Sigma^{-1}(\mathbf{R} - \boldsymbol{\mu})\right]$$

Now, we take a look at some simple applications expressed in terms of vector notation.

2.7 Statistical Regression

Consider a multivariate regression where a stock's returns R_i are regressed on several market factors R_k .

$$R_{it} = \sum_{j=0}^k \beta_{ij} R_{jt} + e_{it}, \quad \forall i.$$

where $t = \{1, 2, \dots, T\}$ (i.e., there are T items in the time series), and there are k independent variables, and usually $k = 0$ is for the intercept. We could write this also as

$$R_{it} = \beta_0 + \sum_{j=1}^k \beta_{ij} R_{jt} + e_{it}, \quad \forall i.$$

Compactly, using vector notation, the same regression may be written as:

$$\mathbf{R}_i = \mathbf{R}_k \boldsymbol{\beta}_i + \mathbf{e}_i$$

where $\mathbf{R}_i, \mathbf{e}_i \in R^T$, $\mathbf{R}_k \in R^{T \times (k+1)}$, and $\boldsymbol{\beta}_i \in R^{k+1}$. If there is an intercept in the regression then the first column of \mathbf{R}_k is $\mathbf{1}$, the unit vector. Without providing a derivation, you should know that each regression coefficient is:

$$\beta_{ik} = \frac{Cov(R_i, R_k)}{Var(R_k)}$$

In vector form, all coefficients may be calculated at once:

$$\boldsymbol{\beta}_i = (\mathbf{R}_k' \mathbf{R}_k)^{-1} (\mathbf{R}_k' \mathbf{R}_i)$$

where the superscript (-1) stands for the inverse of the matrix $(\mathbf{R}_k' \mathbf{R}_k)$ which is of dimension $(k+1) \times (k+1)$. Convince yourself that the dimension of the expression $(\mathbf{R}_k' \mathbf{R}_i)$ is $(k+1) \times 1$, i.e., it is a vector. This results in the vector $\boldsymbol{\beta}_i \in R^{(k+1)}$. This result comes from minimizing the summed squared mean residual error in the regression i.e.,

$$\min_{\boldsymbol{\beta}_i} \mathbf{e}_i' \mathbf{e}_i$$

This will be examined in full detail later in this book.

2.8 Diversification

It is useful to examine the power of using vector algebra with an application. Here we use vector and summation math to understand how diversification in stock portfolios works. Diversification occurs when we increase the number of non-perfectly correlated stocks in a portfolio, thereby reducing portfolio variance. In order to compute the variance of the portfolio we need to use the portfolio weights \mathbf{w} and the covariance matrix of stock returns \mathbf{R} , denoted Σ . We first write down the formula for a portfolio's return variance:

$$\text{Var}(\mathbf{w}'\mathbf{R}) = \mathbf{w}'\Sigma\mathbf{w} = \sum_{i=1}^n w_i^2 \sigma_i^2 + \sum_{i=1}^n \sum_{j=1, i \neq j}^n w_i w_j \sigma_{ij}$$

Readers are strongly encouraged to implement this by hand for $n = 2$ to convince themselves that the vector form of the expression for variance $\mathbf{w}'\Sigma\mathbf{w}$ is the same thing as the long form on the right-hand side of the equation above. If returns are independent, then the formula collapses to:

$$\text{Var}(\mathbf{w}'\mathbf{R}) = \mathbf{w}'\Sigma\mathbf{w} = \sum_{i=1}^n w_i^2 \sigma_i^2$$

If returns are dependent, and equal amounts are invested in each asset ($w_i = 1/n, \forall i$):

$$\begin{aligned} \text{Var}(\mathbf{w}'\mathbf{R}) &= \frac{1}{n} \sum_{i=1}^n \frac{\sigma_i^2}{n} + \frac{n-1}{n} \sum_{i=1}^n \sum_{j=1, i \neq j}^n \frac{\sigma_{ij}}{n(n-1)} \\ &= \frac{1}{n} \bar{\sigma}_i^2 + \frac{n-1}{n} \bar{\sigma}_{ij} \\ &= \frac{1}{n} \bar{\sigma}_i^2 + \left(1 - \frac{1}{n}\right) \bar{\sigma}_{ij} \end{aligned}$$

The first term is the average variance, denoted $\bar{\sigma}_i^2$ divided by n , and the second is the average covariance, denoted $\bar{\sigma}_{ij}$ multiplied by factor $(n-1)/n$. As $n \rightarrow \infty$,

$$\text{Var}(\mathbf{w}'\mathbf{R}) = \bar{\sigma}_{ij}.$$

This produces the remarkable result that in a well diversified portfolio, the variances of each stock's return does not matter at all for portfolio risk! Further the risk of the portfolio, i.e., its variance, is nothing but the average of off-diagonal terms in the covariance matrix.

Diversification exercise

Implement the math above using R to compute the standard deviation of a portfolio of n identical securities with variance 0.04, and pairwise covariances equal to 0.01. Keep increasing n and report the value of the standard deviation. What do you see? Why would this be easier to do in R versus Excel?

Matrix algebra exercise

The following brief notes will introduce you to everything you need to know about the vocabulary of vectors and matrices in a "DIY" (do-it-yourself) mode. Define

$$w = [w_1 \ w_2]' = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

Do the following exercises in long hand:

- Show that

$$I w = w.$$

- Show that the dimensions make sense at all steps of your calculations.

- Show that

$$w' \Sigma w = w_1^2 \sigma_1^2 + 2w_1 w_2 \sigma_{12} + w_2^2 \sigma_2^2.$$

2.9 Matrix Calculus

It is simple to undertake calculus when working with matrices. Calculations using matrices are mere functions of many variables. These functions are amenable to applying calculus, just as you would do in multivariate calculus. However, using vectors and matrices makes things simpler in fact, because we end up taking derivatives of these multivariate functions in one fell swoop rather than one-by-one for each variable. An example will make this clear. Suppose

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

and

$$\mathbf{B} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Let $f(\mathbf{w}) = \mathbf{w}'\mathbf{B}$. This is a function of two variables w_1, w_2 . If we write out $f(\mathbf{w})$ in long form, we get $3w_1 + 4w_2$. The derivative of $f(\mathbf{w})$ with respect to w_1 is $\frac{\partial f}{\partial w_1} = 3$. The derivative of $f(\mathbf{w})$ with respect to w_2 is $\frac{\partial f}{\partial w_2} = 4$. Compare these answers to vector \mathbf{B} . What do you see? What is $\frac{df}{d\mathbf{w}}$? It's \mathbf{B} .

The insight here is that if we simply treat the vectors as regular scalars and conduct calculus accordingly, we will end up with vector derivatives. Hence, the derivative of $\mathbf{w}'\mathbf{B}$ with respect to \mathbf{w} is just \mathbf{B} . Of course, $\mathbf{w}'\mathbf{B}$ is an entire function and \mathbf{B} is a vector. But the beauty of this is that we can take all derivatives of function $\mathbf{w}'\mathbf{B}$ at one time!

These ideas can also be extended to higher-order matrix functions.

Suppose

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$$

and

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Let $f(\mathbf{w}) = \mathbf{w}'\mathbf{A}\mathbf{w}$. If we write out $f(\mathbf{w})$ in long form, we get

$$\mathbf{w}'\mathbf{A}\mathbf{w} = 3w_1^2 + 4w_2^2 + 2(2)w_1w_2$$

Take the derivative of $f(\mathbf{w})$ with respect to w_1 , and this is

$$\frac{df}{dw_1} = 6w_1 + 4w_2$$

Take the derivative of $f(\mathbf{w})$ with respect to w_2 , and this is

$$\frac{df}{dw_2} = 8w_2 + 4w_1$$

Now, we write out the following calculation in long form:

$$2 \mathbf{A} \mathbf{w} = 2 \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 6w_1 + 4w_2 \\ 8w_2 + 4w_1 \end{bmatrix}$$

What do you notice about this solution when compared to the previous two answers? It is nothing but $\frac{df}{d\mathbf{w}}$. Since $\mathbf{w} \in \mathbb{R}^2$, i.e., is of dimension 2, the derivative $\frac{df}{d\mathbf{w}}$ will also be of that dimension.

To see how this corresponds to scalar calculus, think of the function $f(\mathbf{w}) = \mathbf{w}'\mathbf{A}\mathbf{w}$ as simply $\mathbf{A}w^2$, where w is scalar. The derivative of this function with respect to w would be $2\mathbf{A}w$. And, this is the same as what we get when we look at a function of vectors, but with the caveat below.

Note: This computation only works out because \mathbf{A} is symmetric. What should the expression be for the derivative of this vector function if \mathbf{A} is not symmetric but is a square matrix? It turns out that this is

$$\frac{\partial f}{\partial \mathbf{w}} = \mathbf{A}'\mathbf{w} + \mathbf{A}\mathbf{w} \neq 2\mathbf{A}\mathbf{w}$$

Let's try this and see. Suppose

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$$

You can check that the following is all true:

$$\begin{aligned} \mathbf{w}'\mathbf{A}\mathbf{w} &= 3w_1^2 + 4w_2^2 + 3w_1w_2 \\ \frac{\partial f}{\partial w_1} &= 6w_1 + 3w_2 \\ \frac{\partial f}{\partial w_2} &= 3w_1 + 8w_2 \end{aligned}$$

and

$$\mathbf{A}'\mathbf{w} + \mathbf{A}\mathbf{w} = \begin{bmatrix} 6w_1 + 3w_2 \\ 3w_1 + 8w_2 \end{bmatrix}$$

which is correct, but note that the formula for symmetric \mathbf{A} is not!

$$2\mathbf{A}\mathbf{w} = \begin{bmatrix} 6w_1 + 4w_2 \\ 2w_1 + 8w_2 \end{bmatrix}$$

2.10 Matrix Equations

Here we examine how matrices may be used to represent large systems of equations easily and also solve them. Using the values of matrices \mathbf{A} , \mathbf{B} and \mathbf{w} from the previous section, we write out the following in long form:

$$\mathbf{A}\mathbf{w} = \mathbf{B}$$

That is, we have

$$\begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Do you get 2 equations? If so, write them out. Find the solution values w_1 and w_2 by hand. And then we may compute the solution for \mathbf{w} by “dividing” \mathbf{B} by \mathbf{A} . This is not regular division because \mathbf{A} and \mathbf{B} are matrices. Instead we need to multiply the inverse of \mathbf{A} (which is its “reciprocal”) by \mathbf{B} .

The inverse of \mathbf{A} is

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.500 & -0.250 \\ -0.250 & 0.375 \end{bmatrix}$$

Now compute by hand:

$$\mathbf{A}^{-1}\mathbf{B} = \begin{bmatrix} 0.50 \\ 0.75 \end{bmatrix}$$

which should be the same as your solution by hand. Literally, this is all the matrix algebra and calculus you will need for most of the work we will do.

More exercises

Try the following questions for practice.

- What is the value of

$$\mathbf{A}^{-1}\mathbf{AB}$$

Is this vector or scalar?

- What is the final dimension of

$$(\mathbf{w}'\mathbf{B})(\mathbf{AAA}^{-1}\mathbf{B})$$

3

Open Source: Modeling in R

In this chapter, we develop some expertise in using the R statistical package. There are many tutorials available now on the web. See the manuals on the R web site www.r-project.org. There is also a great book that I personally find very high quality, titled “The Art of R Programming” by Norman Matloff. Another useful book is “Machine Learning for Hackers” by Drew Conway and John Myles White.

I assume you have downloaded and installed R by now. If not you can get it from the R project page:

www.r-project.org

Or, you can get a commercial version, offered free to academics and students by Revolution Analytics (the company is to R what RedHat is to Linux). See

www.revolutionanalytics.com

For a useful interface when using R, install RStudio, see www.rstudio.com, but install R first. Let’s get started with some basic programming in R.

3.1 System Commands

If you want to directly access the system you can issue system commands as follows:

```
system ("<command>" )
```

For example

```
system ("ls -lt | grep Das")
```

will list all directory entries that contain my last name in reverse chronological order. Here I am using a unix command, so this will not work on a Windoze machine, but it will certainly work on a Mac or Linux box.

However, you are hardly going to be issuing commands at the system level, so you are unlikely to use the system command very much.

3.2 Loading Data

To get started, we need to grab some data. Go to Yahoo! Finance and download some historical data in an Excel spreadsheet, re-sort it into chronological order, then save it as a CSV file. Read the file into R as follows:

```
> data = read.csv("goog.csv", header=TRUE)      #Read in the data
> n = dim(data)[1]
> n
[1] 1671
> data = data[n:1,]
```

The last command reverses the sequence of the data if required. We can download stock data using the `quantmod` package. Note: to install a package you can use the drop down menus on Windows and Mac operating systems, and use a package installer on Linux. Or issue the following command:

```
install.packages("quantmod")
```

Now we move on to using this package.

```
> library(quantmod)
Loading required package: xts
Loading required package: zoo
> getSymbols(c("YHOO", "AAPL", "CSCO", "IBM"))
[1] "YHOO" "AAPL" "CSCO" "IBM"
> yhoo = YHOO['2007-01-03::2015-01-07']
> aapl = AAPL['2007-01-03::2015-01-07']
> cSCO = CSCO['2007-01-03::2015-01-07']
> ibm = IBM['2007-01-03::2015-01-07']
```

Or we can also directly create columns of stock data as follows.

```
> yhoo = as.matrix(YHOO[,6])
> aapl = as.matrix(AAPL[,6])
> cSCO = as.matrix(CSCO[,6])
> ibm = as.matrix(IBM[,6])
```

We now go ahead and concatenate columns of data into one stock data set.

```
> stkdata = cbind(yhoo,aapl,cSCO,ibm)
> dim(stkdata)
[1] 2018     4
```

Now, compute daily returns. This time, we do log returns in continuous-time. The mean returns are:

```
> n = length(stkdata[,1])
> n
[1] 2018
> rets = log(stkdata[2:n,]/stkdata[1:(n-1),])
> colMeans(rets)
YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted IBM.Adjusted
3.175185e-04 1.116251e-03 4.106314e-05 3.038824e-04
```

We can also compute the covariance matrix and correlation matrix:

```
> cv = cov(rets)
> print(cv,2)
          YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted IBM.Adjusted
YHOO.Adjusted   0.00067   0.00020   0.00022   0.00015
AAPL.Adjusted   0.00020   0.00048   0.00021   0.00015
CSCO.Adjusted   0.00022   0.00021   0.00041   0.00017
IBM.Adjusted    0.00015   0.00015   0.00017   0.00021
> cr = cor(rets)
> print(cr,4)
          YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted IBM.Adjusted
YHOO.Adjusted   1.0000   0.3577   0.4170   0.3900
AAPL.Adjusted   0.3577   1.0000   0.4872   0.4867
CSCO.Adjusted   0.4170   0.4872   1.0000   0.5842
IBM.Adjusted    0.3900   0.4867   0.5842   1.0000
```

Notice the print command that allows you to choose the number of significant digits.

For more flexibility and better handling of data files in various formats, you may also refer to the `readr` package. It has many useful functions.

3.3 Matrices

Q. What do you get if you cross a mountain-climber with a mosquito?

A. Can't be done. You'll be crossing a scalar with a vector.

We will use matrices extensively in modeling, and here we examine the basic commands needed to create and manipulate matrices in R. We create a 4×3 matrix with random numbers as follows:

```
> x = matrix(rnorm(12), 4, 3)
> x
      [,1]      [,2]      [,3]
[1,] 0.0625034  0.9256896  2.3989183
[2,] -0.5371860 -0.7497727 -0.0857688
[3,] -1.0416409  1.6175885  3.3755593
[4,]  0.3244804  0.1228325 -1.6494255
```

Transposing the matrix, notice that the dimensions are reversed:

```
> print(t(x), 3)
      [,1]      [,2]      [,3]      [,4]
[1,] 0.0625 -0.5372 -1.04    0.324
[2,] 0.9257 -0.7498  1.62    0.123
[3,] 2.3989 -0.0858  3.38   -1.649
```

Of course, it is easy to multiply matrices as long as they conform. By “conform” we mean that when multiplying one matrix by another, the number of columns of the matrix on the left must be equal to the number of rows of the matrix on the right. The resultant matrix that holds the answer of this computation will have the number of rows of the matrix on the left, and the number of columns of the matrix on the right.

See the examples below:

```
> print(t(x) %*% x, 3)
      [,1]      [,2]      [,3]
[1,] 1.48 -1.18 -3.86
[2,] -1.18  4.05  7.54
[3,] -3.86  7.54 19.88
>
> print(x %*% t(x), 3)
      [,1]      [,2]      [,3]      [,4]
[1,] 6.616 -0.933  9.530 -3.823
```

```
[2,] -0.933  0.858 -0.943 -0.125
[3,]  9.530 -0.943 15.096 -5.707
[4,] -3.823 -0.125 -5.707  2.841
```

Taking the inverse of the covariance matrix:

```
> cv_inv = solve(cv)
> print(cv_inv,3)
      goog    aapl    cSCO    ibm
goog  3809 -1395 -1058 -491
aapl -1395  3062 -615 -1139
cSCO -1058 -615  3971 -2346
ibm   -491 -1139 -2346  7198
```

Check that the inverse is really so!

```
> print(cv_inv %*% cv,3)
      goog        aapl        cSCO        ibm
goog  1.00e+00  8.33e-17 -1.53e-16  2.78e-17
aapl -2.22e-16  1.00e+00 -3.33e-16 -5.55e-17
cSCO  2.22e-16  0.00e+00  1.00e+00  2.22e-16
ibm   -2.22e-16 -2.22e-16 -2.22e-16  1.00e+00
```

It is, the result of multiplying the inverse matrix by the matrix itself results in the identity matrix. A covariance matrix should be positive definite. Why? What happens if it is not? Checking for this property is easy.

```
> library(copcor)
> is.positive.definite(cv)
[1] TRUE
> is.positive.definite(x)
Error in eigen(m, only.values = TRUE) :
  non-square matrix in 'eigen'
> is.positive.definite(x %*% t(x))
[1] FALSE
```

What happens if you compute pairwise covariances from differing lengths of data for each pair?

3.4 Descriptive Statistics

Let's revisit the same data and compute various descriptive statistics. Read a CSV data file into R as follows:

```
> data = read.csv("goog.csv", header=TRUE)      #Read in the data
> n = dim(data)[1]
> n
[1] 1671
> data = data[n:1,]
> dim(data)
[1] 1671    7
> s = data[,7]
```

So we now have the stock data in place, and we can compute daily returns, and then convert those returns into annualized returns.

```
> rrets = log(s[2:n]/s[1:(n-1)])
> rrets_annual = rrets*252
> print(c(mean(rrets),mean(rrets_annual)))
[1] 0.001044538 0.263223585
```

Compute the daily and annualized standard deviation of returns.

```
> r_sd = sd(rrets)
> r_sd_annual = r_sd*sqrt(252)
> print(c(r_sd,r_sd_annual))
[1] 0.02266823 0.35984704
> #What if we take the stdev of
     annualized returns?
> print(sd(rrets*252))
[1] 5.712395
> #Huh?
>
> print(sd(rrets*252))/252
[1] 5.712395
[1] 0.02266823
> print(sd(rrets*252))/sqrt(252)
[1] 5.712395
[1] 0.3598470
```

Notice the interesting use of the `print` function here. The variance is easy as well.

```
> #Variance
> r_var = var(rrets)
> r_var_annual = var(rrets)*252
> print(c(r_var,r_var_annual))
```

```
[1] 0.0005138488 0.1294898953
```

3.5 Higher-Order Moments

Skewness and kurtosis are key moments that arise in all return distributions. We need a different library in R for these. We use the `moments` library.

$$\text{Skewness} = \frac{E[(X - \mu)^3]}{\sigma^3}$$

Skewness means one tail is fatter than the other (asymmetry). Fatter right (left) tail implies positive (negative) skewness.

$$\text{Kurtosis} = \frac{E[(X - \mu)^4]}{\sigma^4}$$

Kurtosis means both tails are fatter than with a normal distribution.

```
> library(moments)
> skewness(rets)
[1] 0.4874792
> kurtosis(rets)
[1] 9.955916
```

For the normal distribution, skewness is zero, and kurtosis is 3. Kurtosis minus three is denoted “excess kurtosis”.

```
> skewness(rnorm(1000000))
[1] -0.00063502
> kurtosis(rnorm(1000000))
[1] 3.005863
```

What is the skewness and kurtosis of the stock index (S&P500)?

3.6 Quick Introduction to Brownian Motions with R

The law of motion for stocks is often based on a geometric Brownian motion, i.e.,

$$dS(t) = \mu S(t) dt + \sigma S(t) dB(t), \quad S(0) = S_0.$$

This is a “stochastic” differential equation (SDE), because it describes random movement of the stock $S(t)$. The coefficient μ determines the

drift of the process, and σ determines its volatility. Randomness is injected by Brownian motion $B(t)$. This is more general than a deterministic differential equation that is only a function of time, as with a bank account, whose accretion is based on the differential equation $dy(t) = ry(t)dt$, where r is the risk-free rate of interest.

The solution to a SDE is not a deterministic function but a random function. In this case, the solution for time interval h is known to be

$$S(t+h) = S(t) \exp \left[\left(\mu - \frac{1}{2}\sigma^2 \right) h + \sigma B(h) \right]$$

The presence of $B(h) \sim N(0, h)$ in the solution makes the function random. We may also write $B(h)$ as the random variable $\epsilon\sqrt{h} \sim N(0, h)$, where $\epsilon \sim N(0, 1)$. The presence of the exponential return makes the stock price lognormal. (Note: if r.v. x is normal, then e^x is lognormal.)

Re-arranging, the stock return is

$$R(t+h) = \ln \left(\frac{S(t+h)}{S(t)} \right) \sim N \left[\left(\mu - \frac{1}{2}\sigma^2 \right) h, \sigma^2 h \right]$$

Using properties of the lognormal distribution, the conditional mean of the stock price becomes

$$E[S(t+h)|S(t)] = S(t) \cdot e^{\mu h}$$

The following R code computes the annualized volatility σ .

```
> h = 1/252
> sigma = sd(rets)/sqrt(h)
> sigma
[1] 0.3598470
```

The parameter μ is also easily estimated as

```
> mu = mean(rets)/h+0.5*sigma^2
> mu
[1] 0.3279685
```

So the additional term $\frac{1}{2}\sigma^2$ does matter substantially.

3.7 Estimation using maximum-likelihood

MLE estimation requires finding the parameters $\{\mu, \sigma\}$ that maximize the likelihood of seeing the empirical sequence of returns $R(t)$. A probability function is required, and we have one above for $R(t)$, which is i.i.d.

First, a quick recap of the normal distribution. If $x \sim N(\mu, \sigma^2)$, then

$$\text{density function: } f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right]$$

$$N(x) = 1 - N(-x)$$

$$F(x) = \int_{-\infty}^x f(u)du$$

The standard normal distribution is $x \sim N(0, 1)$. For the standard normal distribution: $F(0) = \frac{1}{2}$.

The probability density of $R(t)$ is normal with the following equation:

$$f[R(t)] = \frac{1}{\sqrt{2\pi\sigma^2 h}} \cdot \exp\left[-\frac{1}{2} \cdot \frac{(R(t)-\alpha)^2}{\sigma^2 h}\right]$$

where $\alpha = \left(\mu - \frac{1}{2}\sigma^2\right)h$. For periods $t = 1, 2, \dots, T$ the likelihood of the entire series is

$$\prod_{t=1}^T f[R(t)]$$

It is easier (computationally) to maximize

$$\max_{\mu, \sigma} \mathcal{L} \equiv \sum_{t=1}^T \ln f[R(t)]$$

known as the log-likelihood. This is easily done in R. First we create the log-likelihood function

```
> LL = function(params, rets) {
+ alpha = params[1]; sigsq = params[2]
+ logf = -log(sqrt(2*pi*sigsq))
- (rets-alpha)^2/(2*sigsq)
+ LL = -sum(logf)
+ }
```

Note that

$$\ln f[R(t)] = -\ln \sqrt{2\pi\sigma^2 h} - \frac{[R(t)-\alpha]^2}{2\sigma^2 h}$$

We have used variable `sigsq` in function `LL` for $\sigma^2 h$.

We then go ahead and do the MLE using the `nlm` (non-linear minimization) package in R. It uses a Newton-type algorithm.

```
> #create starting guess for parameters
> params = c(0.001, 0.001)
> res = nlm(LL, params, rets)
```

```
> res
$minimum
[1] -3954.813
$estimate
[1] 0.0010441526 0.0005130404
$gradient
[1] 0.3728092 -3.2397043
$code
[1] 2
$iterations
[1] 12
```

We now pick off the results and manipulate them to get the annualized parameters $\{\mu, \sigma\}$.

```
> alpha = res$estimate[1]
> sigsq = res$estimate[2]
> sigma = sqrt(sigsq/h)
> sigma
[1] 0.3595639
> mu = alpha/h + 0.5*sigma^2
> mu
[1] 0.3277695
```

We see that the estimated parameters are close to that derived earlier.

3.8 GARCH/ARCH Models

GARCH stands for “Generalized Auto- Regressive Conditional Heteroskedasticity”. Rob Engle invented ARCH (for which he got the Nobel prize) and this was extended by Tim Bollerslev to GARCH.

ARCH models are based on the idea that volatility tends to cluster, i.e., volatility for period t , is auto-correlated with volatility from period $(t-1)$, or more preceding periods. If we had a time series of stock returns following a random walk, we might model it as follows

$$r_t = \mu + e_t, \quad e_t \sim N(0, \sigma_t^2)$$

If the variance were stationary then σ_t^2 would be constant. But under ARCH it is auto-correlated with previous variances. Hence, we have

$$\sigma_t^2 = \beta_0 + \sum_{j=1}^p \beta_{1j} \sigma_{t-j}^2 + \sum_{k=1}^q \beta_{2k} e_{t-k}^2$$

So current variance (σ_t^2) depends on past squared shocks (e_{t-k}^2) and past variances (σ_{t-j}^2). The number of lags of past variance is p , and that of lagged shocks is q . The model is thus known as a GARCH(p, q) model. For the model to be stationary, the sum of all the β terms should be less than 1.

In GARCH, stock returns are conditionally normal, and independent, but *not* identically distributed because the variance changes over time. Since at every time t , we know the conditional distribution of returns, because σ_t is based on past σ_{t-j} and past shocks e_{t-k} , we can estimate the parameters $\{\beta_0, \beta_1 j, \beta_2 k\}, \forall j, k$, of the model using MLE. The good news is that this comes canned in R, so all we need to do is use the `tseries` package.

```
> library(tseries)
> res = garch(rets, order=c(1,1))
> summary(res)

Call:
garch(x = rets, order = c(1, 1))

Model:
GARCH(1,1)

Residuals:
    Min         1Q     Median         3Q        Max
-5.54354 -0.45479  0.03512  0.57051  7.40088

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
ao 5.568e-06 8.803e-07 6.326 2.52e-10 ***
a1 4.294e-02 4.622e-03 9.289 < 2e-16 ***
b1 9.458e-01 5.405e-03 174.979 < 2e-16 ***

Signif. codes: 0 [***] 0.001 [**] 0.01 [*] 0.05 [.]
Diagnostic Tests: Jarque Bera Test
data: Residuals
X-squared = 3007.311, df = 2,
p-value < 2.2e-16
Box-Ljung test
data: Squared.Residuals
X-squared = 0.5305, df = 1, p-value = 0.4664
```

Notice how strikingly high the t-statistics are. What is volatility related

to mostly? Is the model stationary?

3.9 Introduction to Monte Carlo

It is easy to simulate a path of stock prices using a discrete form of the solution to the Geometric Brownian motion SDE. This is the equation of motion for the stock price, which randomly moves the stock price from its previous value $S(t)$ to the value h years ahead, $S(t + h)$.

$$S(t + h) = S(t) \exp \left[\left(\mu - \frac{1}{2} \sigma^2 \right) h + \sigma \cdot e \sqrt{h} \right]$$

Note that we replaced $B(h)$ with $e\sqrt{h}$, where $e \sim N(0, 1)$. Both $B(h)$ and $e\sqrt{h}$ have mean zero and variance h . Knowing $S(t)$, we can simulate $S(t + h)$ by drawing e from a standard normal distribution. Here is the R code to run the entire simulation.

```
> n = 252
> so = 100
> mu = 0.10
> sig = 0.20
> s = matrix(0, 1, n+1)
> h=1/n
>
> s[1] = so
> for (j in 2:(n+1)) {
+ s[j]=s[j-1]*exp((mu-sig^2/2)*h
+ +sig*rnorm(1)*sqrt(h))
+ }
> s[1:5]
[1] 100.00000 99.54793 96.98941
98.65440 98.76989
> s[(n-4):n]
[1] 87.01616 86.37163 84.92783
84.17420 86.16617
> plot(t(s), type="l")
```

This program generates the plot shown in Figure 3.1.

The same logic may be used to generate multiple paths of stock prices, in a *vectorized* way as follows. In the following example we generate 3 paths. Because of the vectorization, the run time does not increase linearly with the number of paths, and in fact, hardly increases at all.

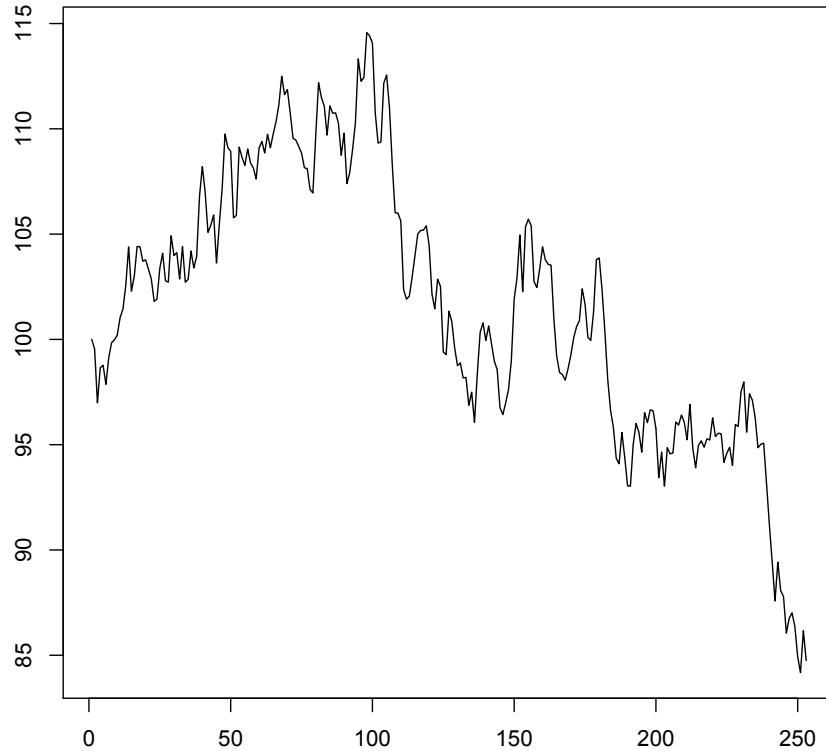


Figure 3.1: Single stock path plot simulated from a Brownian motion.

```
> s = matrix(0, 3, n+1)
> s[,1] = so
> for (j in seq(2, (n+1))) {
+ s[,j]=s[,j-1]*exp((mu-sig^2/2)*h
+ +sig*matrix(rnorm(3), 3, 1)*sqrt(h))
+ }
> plot(t(s)[,1], ylim=c(ymin,ymax), type="l")
> lines(t(s)[,2], col="red", lty=2)
> lines(t(s)[,3], col="blue", lty=3)
```

The plot is shown in Figure 3.2. The plot code shows how to change the style of the path and its color.

If you generate many more paths, how can you find the probability of the stock ending up below a defined price? Can you do this directly from the discrete version of the Geometric Brownian motion process above?

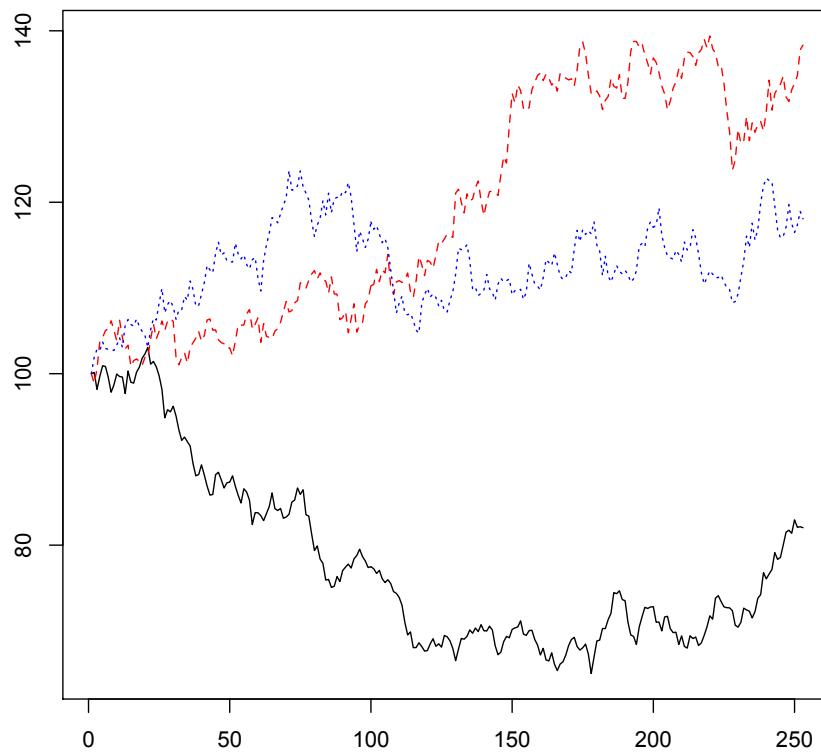


Figure 3.2: Multiple stock path plot simulated from a Brownian motion.

Bivariate random variables

To convert two independent random variables $(e_1, e_2) \sim N(0, 1)$ into two correlated random variables (x_1, x_2) with correlation ρ , use the following transformations.

$$x_1 = e_1, \quad x_2 = \rho \cdot e_1 + \sqrt{1 - \rho^2} \cdot e_2$$

We can now generate 10,000 pairs of correlated random variates using the following R code.

```
> e = matrix(rnorm(20000), 10000, 2)
> cor(e)
      [,1]      [,2]
[1,] 1.000000000 0.007620184
[2,] 0.007620184 1.000000000
> cor(e[,1], e[,2])
[1] 0.007620184
> rho = 0.6
> x1 = e[,1]
> x2 = rho * e[,1] + sqrt(1 - rho^2) * e[,2]
```

```
> cor(x1, x2)
[1] 0.5981845
```

It is useful to check algebraically that $E[x_i] = 0, i = 1, 2$, $Var[x_i] = 1, i = 1, 2$. Also check that $Cov[x_1, x_2] = \rho = Corr[x_1, x_2]$. We can numerically check this using the following:

```
> mean(x1)
[1] -0.006522788
> mean(x2)
[1] -0.00585042
> var(x1)
[1] 0.9842857
> var(x2)
[1] 1.010802
> cov(x1, x2)
[1] 0.5966626
```

Multivariate random variables

These are generated using Cholesky decomposition which is a matrix operation that represents a covariance matrix as a product of two matrices. We may write a covariance matrix in decomposed form, i.e., $\Sigma = \mathbf{L} \mathbf{L}'$, where \mathbf{L} is a lower triangular matrix. Alternatively we might have an upper triangular decomposition, where $\mathbf{U} = \mathbf{L}'$. Think of each component of the decomposition as a square-root of the covariance matrix.

The Cholesky decomposition is very useful in generating correlated random numbers from a distribution with mean vector μ and covariance matrix Σ . Suppose we have a scalar random variable $e \sim (0, 1)$. To transform this variate into $x \sim (\mu, \sigma^2)$, we generate e and then set $x = \mu + \sigma e$. If instead of a scalar random variable, we have a vector random variables (independent of each other) given by a vector $\mathbf{e} = [e_1, e_2, \dots, e_n]^\top \sim (\mathbf{0}, \mathbf{I})$, then we may transform this into a vector of correlated random variables $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top \sim (\mu, \Sigma)$ by computing:

$$\mathbf{x} = \mu + \mathbf{L}\mathbf{e}$$

This is implemented using the following code.

```
> #Original matrix
```

```

> cv
      [,1] [,2] [,3]
[1,] 0.01  0.00  0.00
[2,] 0.00  0.04  0.02
[3,] 0.00  0.02  0.16
> #Let's enhance it
> cv[1,2]=0.005
> cv[2,1]=0.005
> cv[1,3]=0.005
> cv[3,1]=0.005
> cv
      [,1] [,2] [,3]
[1,] 0.010 0.005 0.005
[2,] 0.005 0.040 0.020
[3,] 0.005 0.020 0.160
> L = t(chol(cv))
> L
      [,1]      [,2]      [,3]
[1,] 0.10 0.0000000 0.0000000
[2,] 0.05 0.19364917 0.0000000
[3,] 0.05 0.09036961 0.3864367
> e=matrix(randn(3*10000),10000,3)
> x = t(L %*% t(e))
> dim(x)
[1] 10000      3
> cov(x)
      [,1]      [,2]      [,3]
[1,] 0.009872214 0.004597322 0.004521752
[2,] 0.004597322 0.040085503 0.019114981
[3,] 0.004521752 0.019114981 0.156378078
>

```

In the last calculation, we confirmed that the simulated data has the same covariance matrix as the one that we generated correlated random variables from.

3.10 Portfolio Computations in R

Let's enter a sample mean vector and covariance matrix and then using some sample weights, we will perform some basic matrix computations for portfolios to illustrate the use of R.

```
> mu = matrix(c(0.01,0.05,0.15),3,1)
> cv = matrix(c(0.01,0,0,0,0.04,0.02,
+               0,0.02,0.16),3,3)
> mu
      [,1]
[1,] 0.01
[2,] 0.05
[3,] 0.15
> cv
      [,1] [,2] [,3]
[1,] 0.01 0.00 0.00
[2,] 0.00 0.04 0.02
[3,] 0.00 0.02 0.16
> w = matrix(c(0.3,0.3,0.4))
> w
      [,1]
[1,] 0.3
[2,] 0.3
[3,] 0.4
> muP = t(w) %*% mu
> muP
      [,1]
[1,] 0.078
> stdP = sqrt(t(w) %*% cv %*% w)
> stdP
      [,1]
[1,] 0.1868154
```

We thus generated the expected return and risk of the portfolio, i.e., the values 0.078 and 0.187, respectively.

We are interested in the risk of a portfolio, often measured by its variance. As we had seen in the previous chapter, as we increase n , the number of securities in the portfolio, the variance keeps dropping, and asymptotes to a level equal to the average covariance of all the assets. It

is interesting to see what happens as n increases through a very simple function in R that returns the standard deviation of the portfolio.

```
> sigport = function(n,sig_ij,sig_ij) {
+ cv = matrix(sig_ij,n,n)
+ diag(cv) = sig_ij
+ w = matrix(1/n,n,1)
+ result = sqrt(t(w) %*% cv %*% w)
+ }
>
> n = seq(5,100,5)
> n
[1] 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85
[18] 90 95 100
> risk_n = NULL
> for (nn in n) {
+ risk_n = c(risk_n,sigport(nn,0.04,0.01))
+ }
> risk_n
[1] 0.1264911 0.1140175 0.1095445
      0.1072381 0.1058301 0.1048809
[7] 0.1041976 0.1036822 0.1032796
      0.1029563 0.1026911 0.1024695
[13] 0.1022817 0.1021204 0.1019804
      0.1018577 0.1017494 0.1016530
[19] 0.1015667 0.1014889
>
```

We can plot this to see the classic systematic risk plot. This is shown in Figure 3.3.

```
> plot(n,risk_n,type="l",
      ylab="Portfolio_Std_Dev")
```

3.11 Finding the Optimal Portfolio

We will review the notation one more time. Assume that the risk free asset has return r_f . And we have n risky assets, with mean returns $\mu_i, i = 1 \dots n$. We need to invest in optimal weights w_i in each asset. Let $w = [w_1, \dots, w_n]'$ be a column vector of portfolio weights. We define

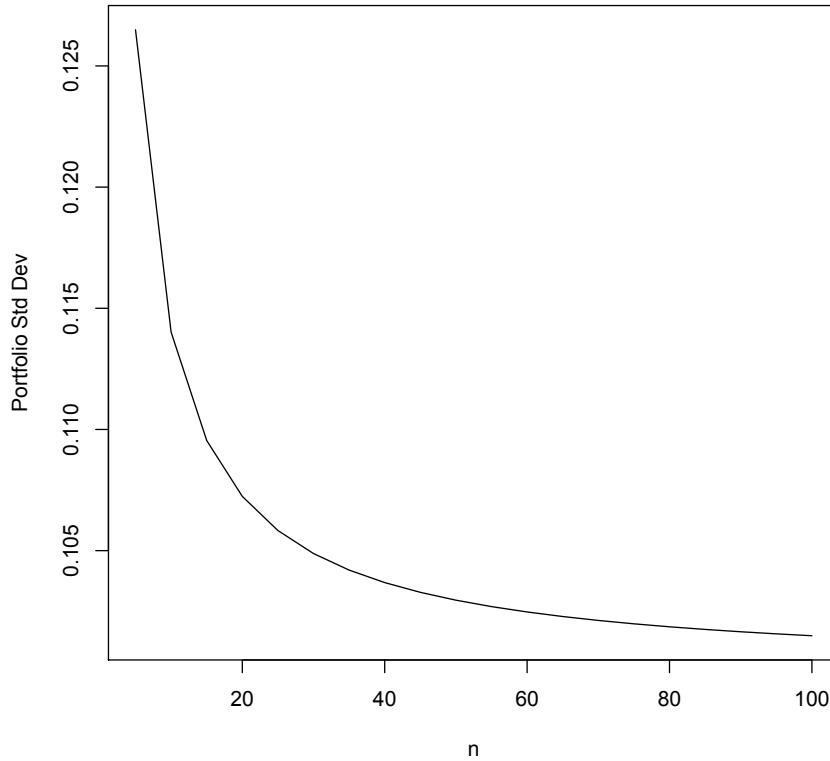


Figure 3.3: Systematic risk as the number of stocks in the portfolio increases.

$\mu = [\mu_1, \dots, \mu_n]'$ be the column vector of mean returns on each asset, and $\mathbf{1} = [1, \dots, 1]'$ be a column vector of ones. Hence, the expected return on the portfolio will be

$$E(R_p) = (1 - w'\mathbf{1})r_f + w'\mu$$

The variance of return on the portfolio will be

$$\text{Var}(R_p) = w'\Sigma w$$

where Σ is the covariance matrix of returns on the portfolio. The objective function is a trade-off between return and risk, with β modulating the balance between risk and return:

$$U(R_p) = r_f + w'(\mu - r_f\mathbf{1}) - \frac{\beta}{2}w'\Sigma w$$

The f.o.c. becomes a system of equations now (not a single equation), since we differentiate by an entire vector w :

$$\frac{dU}{dw'} = \mu - r_f\mathbf{1} - \beta\Sigma w = \mathbf{0}$$

where the RHS is a vector of zeros of dimension n . Solving we have

$$w = \frac{1}{\beta} \Sigma^{-1} (\mu - r_f \mathbf{1})$$

Therefore, allocation to the risky assets

- Increases when the relative return to it ($\mu - r_f \mathbf{1}$) increases.
- Decreases when risk aversion increases.
- Decreases when riskiness of the assets increases as proxied for by Σ .

```
> n=3
> cv
      [,1] [,2] [,3]
[1,] 0.01 0.00 0.00
[2,] 0.00 0.04 0.02
[3,] 0.00 0.02 0.16
> mu
      [,1]
[1,] 0.01
[2,] 0.05
[3,] 0.15
> rf=0.005
> beta = 4
> wuns = matrix(1,n,1)
> wuns
      [,1]
[1,]    1
[2,]    1
[3,]    1
> w = (1/beta)*(solve(cv) %*% (mu-rf*wuns))
> w
      [,1]
[1,] 0.1250000
[2,] 0.1791667
[3,] 0.2041667
> w_in_rf = 1-sum(w)
> w_in_rf
[1] 0.4916667
```

What if we reduced beta?

```

> beta = 3
> w = (1/beta)*(solve(cv) %*% (mu-rf*wuns));
> w
[ ,1]
[1,] 0.1666667
[2,] 0.2388889
[3,] 0.2722222
> beta = 2
> w = (1/beta)*(solve(cv) %*% (mu-rf*wuns));
> w
[ ,1]
[1,] 0.2500000
[2,] 0.3583333
[3,] 0.4083333

```

Notice that the weights in stocks scales linearly with β . The relative proportions of the stocks themselves remains constant. Hence, β modulates the proportions invested in a risk-free asset and a stock portfolio, in which stock proportions remain same. It is as if the stock versus bond decision can be taken separately from the decision about the composition of the stock portfolio. This is known as the “two-fund separation” property, i.e., first determine the proportions in the bond fund vs stock fund and the allocation within each fund can be handled subsequently.

3.12 Root Solving

Finding roots of nonlinear equations is often required, and R has several packages for this purpose. Here we examine a few examples.

Suppose we are given the function

$$(x^2 + y^2 - 1)^3 - x^2y^3 = 0$$

and for various values of y we wish to solve for the values of x . The function we use is called `fn` and the use of the function is shown below.

```

library(rootSolve)

fn = function(x,y) {
  result = (x^2+y^2-1)^3 - x^2*y^3
}

```

```

yy = 1
sol = multiroot(f=fn, start=1, maxiter=10000, rtol=0.000001,
                  atol=0.0000001, ctol=0.00001, y=yy)
print(sol)

check = fn(sol$root, yy)
print(check)

```

At the end we check that the equation has been solved. Here is the code run:

```

> source("fn.R")
$root
[1] 1

$f.root
[1] 0

$iter
[1] 1

$estim.precis
[1] 0

[1] 0

```

Here is another example, where we solve a single unknown using the `uniroot.all` function.

```

library(rootSolve)
fn = function(x) {
  result = 0.065*(x*(1-x))^0.5 - 0.05 + 0.05*x
}
sol = uniroot.all(f=fn, c(0,1))
print(sol)

```

The function searches for a solution (root) in the range [0,1]. The answer is given as:

```
[1] 1.0000000 0.3717627
```

3.13 Regression

In a *multivariate* linear regression, we have

$$Y = X \cdot \beta + e$$

where $Y \in R^{t \times 1}$, $X \in R^{t \times n}$, and $\beta \in R^{n \times 1}$, and the regression solution is simply equal to $\beta = (X'X)^{-1}(X'Y) \in R^{n \times 1}$.

To get this result we minimize the sum of squared errors.

$$\begin{aligned} \min_{\beta} e'e &= (Y - X \cdot \beta)'(Y - X \cdot \beta) \\ &= Y'(Y - X \cdot \beta) - (X\beta)' \cdot (Y - X \cdot \beta) \\ &= Y'Y - Y'X\beta - (\beta'X')Y + \beta'X'X\beta \\ &= Y'Y - Y'X\beta - Y'X\beta + \beta'X'X\beta \\ &= Y'Y - 2Y'X\beta + \beta'X'X\beta \end{aligned}$$

Note that this expression is a scalar. Differentiating w.r.t. β' gives the following f.o.c:

$$\begin{aligned} -2X'Y + 2X'X\beta &= \mathbf{0} \\ \implies \beta &= (X'X)^{-1}(X'Y) \end{aligned}$$

There is another useful expression for each individual $\beta_i = \frac{Cov(X_i, Y)}{Var(X_i)}$. You should compute this and check that each coefficient in the regression is indeed equal to the β_i from this calculation.

Example: Let's do a regression and see whether AAPL, CSCO, and IBM can explain the returns of YHOO. This uses the data we had downloaded earlier.

```
> dim(rets)
[1] 2017      4
> Y = as.matrix(rets[,1])
> X = as.matrix(rets[,2:4])
> n = length(Y)
> X = cbind(matrix(1,n,1),X)
> b = solve(t(X) %*% X) %*% (t(X) %*% Y)
> b
[ ,1]
3.139183e-06
AAPL.Adjusted 1.854781e-01
```

```
CSCO.Adjusted 3.069011e-01
IBM.Adjusted 3.117553e-01
```

But of course, R has this regression stuff canned, and you do not need to hassle with the Matrix (though the movie is highly recommended).

```
> X = as.matrix(rets[,2:4])
> res = lm(Y~X)
> summary(res)
```

Call:

```
lm(formula = Y ~ X)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.18333	-0.01051	-0.00044	0.00980	0.38288

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	3.139e-06	5.091e-04	0.006	0.995
XAAPL.Adjusted	1.855e-01	2.780e-02	6.671	3.28e-11 ***
XCSCO.Adjusted	3.069e-01	3.244e-02	9.462	< 2e-16 ***
XIBM.Adjusted	3.118e-01	4.517e-02	6.902	6.82e-12 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.02283 on 2013 degrees of freedom

Multiple R-squared: 0.2236, Adjusted R-squared: 0.2224

F-statistic: 193.2 on 3 and 2013 DF, p-value: < 2.2e-16

For visuals, do see the `abline()` function as well.

Here is a simple regression run on some data from the 2005-06 NCAA basketball season for the March madness stats. The data is stored in a space-delimited file called `ncaa.txt`. We use the metric of performance to be the number of games played, with more successful teams playing more playoff games, and then try to see what variables explain it best. We apply a simple linear regression that uses the R command `lm`, which stands for “linear model.”

```
> ncaa = read.table("ncaa.txt", header=TRUE)
> y = ncaa[3]
```

```
> y = as.matrix(y)
> x = ncaa[4:14]
> x = as.matrix(x)
> fm = lm(y~x)
> res = summary(fm)
> res
```

Call:

```
lm(formula = y ~ x)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.5075	-0.5527	-0.2454	0.6705	2.2344

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-10.194804	2.892203	-3.525	0.000893 ***
xPTS	-0.010442	0.025276	-0.413	0.681218
xREB	0.105048	0.036951	2.843	0.006375 **
xAST	-0.060798	0.091102	-0.667	0.507492
xTO	-0.034545	0.071393	-0.484	0.630513
xA.T	1.325402	1.110184	1.194	0.237951
xSTL	0.181015	0.068999	2.623	0.011397 *
xBLK	0.007185	0.075054	0.096	0.924106
xPF	-0.031705	0.044469	-0.713	0.479050
xFG	13.823190	3.981191	3.472	0.001048 **
xFT	2.694716	1.118595	2.409	0.019573 *
xX3P	2.526831	1.754038	1.441	0.155698

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1

Residual standard error: 0.9619 on 52 degrees of freedom

Multiple R-Squared: 0.5418, Adjusted R-squared: 0.4448

F-statistic: 5.589 on 11 and 52 DF, p-value: 7.889e-06

We note that the command `lm` returns an “object” with name `res`. This object contains various details about the regression result, and can then be called by other functions that will format and present various versions of the result. For example, using the following command gives a

nicely formatted version of the regression output, and you should try to use it when presenting regression results.

An alternative approach using data frames is:

```
> ncaa_data_frame = data.frame(y=as.matrix(ncaa[3]),
                                x=as.matrix(ncaa[4:14]))
> fm = lm(y~x, data=ncaa_data_frame)
> summary(fm)
```

(The output is not shown here in order to not repeat what we saw in the previous regression.) Data frames are also objects. Here, objects are used in the same way as the term is used in object-oriented programming (OOP), and in a similar fashion, R supports OOP as well.

Direct regression implementing the matrix form is as follows (we had derived this earlier):

```
> wuns = matrix(1, 64, 1)
> z = cbind(wuns, x)
> b = inv(t(z) %*% z) %*% (t(z) %*% y)
> b
      GMS
-10.194803524
PTS -0.010441929
REB 0.105047705
AST -0.060798192
TO  -0.034544881
A.T  1.325402061
STL  0.181014759
BLK  0.007184622
PF  -0.031705212
FG  13.823189660
FT   2.694716234
X3P  2.526830872
```

Note that this is exactly the same result as we had before, but it gave us a chance to look at some of the commands needed to work with matrices in R.

3.14 Heteroskedasticity

Simple linear regression assumes that the standard error of the residuals is the same for all observations. Many regressions suffer from the failure of this condition. The word for this is “heteroskedastic” errors. “Hetero” means different, and “skedastic” means dependent on type.

We can first test for the presence of heteroskedasticity using a standard Breusch-Pagan test available in R. This resides in the `lmtest` package which is loaded in before running the test.

```
> ncaa = read.table("ncaa.txt", header=TRUE)
> y = as.matrix(ncaa[3])
> x = as.matrix(ncaa[4:14])
> result = lm(y~x)
> library(lmtest)
Loading required package: zoo
> bptest(result)

studentized Breusch-Pagan test

data: result
BP = 15.5378, df = 11, p-value = 0.1592
```

We can see that there is very little evidence of heteroskedasticity in the standard errors as the p-value is not small. However, lets go ahead and correct the t-statistics for heteroskedasticity as follows, using the `hccm` function. The “`hccm`” stands for heteroskedasticity corrected covariance matrix.

```
> wuns = matrix(1, 64, 1)
> z = cbind(wuns, x)
> b = solve(t(z) %*% z) %*% (t(z) %*% y)
> result = lm(y~x)
> library(car)
> vb = hccm(result)
> stdb = sqrt(diag(vb))
> tstats = b/stdb
> tstats
      GMS
      -2.68006069
PTS -0.38212818
```

REB	2.38342637
AST	-0.40848721
TO	-0.28709450
A.T	0.65632053
STL	2.13627108
BLK	0.09548606
PF	-0.68036944
FG	3.52193532
FT	2.35677255
X3P	1.23897636

Here we used the `hccm` function to generate the new covariance matrix `vb` of the coefficients, and then we obtained the standard errors as the square root of the diagonal of the covariance matrix. Armed with these revised standard errors, we then recomputed the t-statistics by dividing the coefficients by the new standard errors. Compare these to the t-statistics in the original model

```
summary(result)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -10.194804   2.892203  -3.525 0.000893 *** 
xPTS         -0.010442   0.025276  -0.413 0.681218    
xREB         0.105048   0.036951   2.843 0.006375 **  
xAST         -0.060798   0.091102  -0.667 0.507492    
xTO          -0.034545   0.071393  -0.484 0.630513    
xA.T          1.325402   1.110184   1.194 0.237951    
xSTL         0.181015   0.068999   2.623 0.011397 *   
xBLK         0.007185   0.075054   0.096 0.924106    
xPF          -0.031705   0.044469  -0.713 0.479050    
xFG          13.823190   3.981191   3.472 0.001048 **  
xFT          2.694716   1.118595   2.409 0.019573 *   
xX3P         2.526831   1.754038   1.441 0.155698    
```

It is apparent that when corrected for heteroskedasticity, the t-statistics in the regression are lower, and also render some of the previously significant coefficients insignificant.

3.15 Auto-regressive models

When data is autocorrelated, i.e., has dependence in time, not accounting for it results in unnecessarily high statistical significance. Intuitively, this is because observations are treated as independent when actually they are correlated in time, and therefore, the true number of observations is effectively less.

In efficient markets, the correlation of returns from one period to the next should be close to zero. We use the returns stored in the variable `rets` (based on Google stock) from much earlier in this chapter.

```
> n = length(rets)
> n
[1] 1670
> cor(rets[1:(n-1)], rets[2:n])
[1] 0.007215026
```

This is for immediately consecutive periods, known as first-order auto-correlation. We may examine this across many staggered periods. For this R has some neat library functions in the package `car`.

```
> library(car)
> durbin.watson(rets, max.lag=10)
[1] 1.974723 2.016951 1.984078 1.932000
      1.950987 2.101559 1.977719 1.838635
      2.052832 1.967741
> res = lm(rets[2:n]~rets[1:(n-1)])
> durbin.watson(res, max.lag=10)
   lag Autocorrelation D-W Statistic p-value
1 -0.0006436855 2.001125 0.938
2 -0.0109757002 2.018298 0.724
3 -0.0002853870 1.996723 0.982
4  0.0252586312 1.945238 0.276
5  0.0188824874 1.957564 0.402
6 -0.0555810090 2.104550 0.020
7  0.0020507562 1.989158 0.926
8  0.0746953706 1.843219 0.004#
9 -0.0375308940 2.067304 0.136
10 0.0085641680 1.974756 0.772
Alternative hypothesis: rho[lag] != 0
```

There is no evidence of auto-correlation when the DW statistic is close to 2. If the DW-statistic is greater than 2 it indicates negative autocorrelation, and if it is less than 2, it indicates positive autocorrelation.

In the data there only seems to be statistical significance at the eighth lag. We may regress leading values on lags to see if the coefficient is significant.

```
> summary(res)
```

Call :

```
lm(formula = rets[2:n] ~ rets[1:(n - 1)])
```

Residuals :

Min	1Q	Median	3Q	Max
-0.1242520	-0.0102479	-0.0002719	0.0106435	0.1813465

Coefficients :

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.0009919	0.0005539	1.791	0.0735 .
rets[1:(n - 1)]	0.0071913	0.0244114	0.295	0.7683

Signif. codes: 0 [***] 0.001 [**] 0.01 [*] 0.05 [.] 0.1 []

Residual std error: 0.02261 on 1667 degrees of freedom

Multiple R-squared: 5.206e-05,

Adjusted R-squared: -0.0005478

F-statistic: 0.08678 on 1 and 1667 DF, p-value: 0.7683

As another example, let's load in the file `markowitzdata.txt` and run tests on it. This file contains data on five tech sector stocks and also the Fama-French data. The `names` function shows the headers of each column as shown below.

```
> md = read.table("markowitzdata.txt", header=TRUE)
> names(md)
[1] "X.DATE"   "SUNW"      "MSFT"      "IBM"       "CSCO"      "AMZN"      "mktrf"
[8] "smb"       "hml"       "rf"
> y = as.matrix(md[2])
> x = as.matrix(md[7:9])
> rf = as.matrix(md[10])
> y = y - rf
```

```
> library(car)
> results = lm(y ~ x)
> durbin.watson(results, max.lag=6)
lag Autocorrelation D-W Statistic p-value
 1      -0.07231926    2.144549   0.002
 2      -0.04595240    2.079356   0.146
 3       0.02958136    1.926791   0.162
 4      -0.01608143    2.017980   0.632
 5      -0.02360625    2.032176   0.432
 6      -0.01874952    2.021745   0.536
Alternative hypothesis: rho[lag] != 0
```

The **car** package is used. We see that there is one lag auto-correlation (note the small p-value for lag 1), but not more than that; markets are very efficient. Lets look at the regression before and after correction for autocorrelation:

```
> summary(results)

Call:
lm(formula = y ~ x)

Residuals:
    Min          1Q      Median          3Q          Max  
-0.2136760 -0.0143564 -0.0007332  0.0144619  0.1910892 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.000197  0.000785 -0.251   0.8019    
xmktrf       1.657968  0.085816 19.320  <2e-16 ***  
xsmb         0.299735  0.146973  2.039   0.0416 *    
xhml        -1.544633  0.176049 -8.774  <2e-16 ***  
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 

Residual standard error: 0.03028 on 1503 degrees of freedom
Multiple R-Squared: 0.3636,    Adjusted R-squared: 0.3623 
F-statistic: 286.3 on 3 and 1503 DF,  p-value: < 2.2e-16
```

Lets correct the t-stats for autocorrelation using the Newey-West correction. This correction is part of the **car** package. The steps undertaken

here are similar in mechanics to the ones we encountered when correcting for heteroskedasticity.

```
> res = lm(y~x)
> b = res$coefficients
> b
(Intercept)      xmktrf       xsmb       xhml
-0.0001970164   1.6579682191   0.2997353765 -1.5446330690
> vb = NeweyWest(res, lag=1)
> stdb = sqrt(diag(vb))
> tstats = b/stdb
> tstats
(Intercept)      xmktrf       xsmb       xhml
-0.2633665    15.5779184    1.8300340   -6.1036120
```

Compare these to the stats we had earlier. Notice how they have come down after correction for AR. Note that there are several steps needed to correct for autocorrelation, and it might have been nice to roll one's own function for this. (I leave this as an exercise for you.)

For fun, let's look at the autocorrelation in stock market indexes, shown in Table 3.1. The following graphic is taken from the book "A Non-Random Walk Down Wall Street" by Andrew Lo and Craig Mackinlay. Is the autocorrelation higher for equally-weighted or value-weighted indexes? Why?

3.16 Vector Auto-Regression

Also known as VAR (not the same thing as Value-at-Risk, denoted VaR). VAR is useful for estimating systems where there are simultaneous regression equations, and the variables influence each other. So in a VAR, each variable in a system is assumed to depend on lagged values of itself and the other variables. The number of lags may be chosen by the econometrician based on what is the expected decay in time-dependence of the variables in the VAR.

In the following example, we examine the inter-relatedness of returns of the following three tickers: SUNW, MSFT, IBM. For vector autoregressions (VARs), we run the following R commands:

```
> md = read.table("markowitzdata.txt", header=TRUE)
> y = as.matrix(md[2:4])
> library(stats)
```

Table 2.4. Autocorrelation in daily, weekly, and monthly stock index returns.

Sample Period	Sample Size	Mean	SD	$\hat{\rho}_1$	$\hat{\rho}_2$	$\hat{\rho}_3$	$\hat{\rho}_4$	\hat{Q}_5	\hat{Q}_{10}
A. Daily Returns									
CRSP Value-Weighted Index									
62:07:03–94:12:30 8,179 0.041 0.824 17.6 -0.7 0.1 -0.8 263.3 269.5									
62:07:03–78:10:27 4,090 0.028 0.738 27.8 1.2 4.6 3.3 329.4 343.5									
78:10:30–94:12:30 4,089 0.054 0.901 10.8 -2.2 -2.9 -3.5 69.5 72.1									
CRSP Equal-Weighted Index									
62:07:03–94:12:30 8,179 0.070 0.764 35.0 9.3 8.5 9.9 1,301.9 1,369.5									
62:07:03–78:10:27 4,090 0.063 0.771 43.1 13.0 15.3 15.2 1,062.2 1,110.2									
78:10:30–94:12:30 4,089 0.078 0.756 26.2 4.9 2.0 4.9 348.9 379.5									
B. Weekly Returns									
CRSP Value-Weighted Index									
62:07:10–94:12:27 1,695 0.196 2.093 1.5 -2.5 3.5 -0.7 8.8 36.7									
62:07:10–78:10:03 848 0.144 1.994 5.6 -3.7 5.8 1.6 9.0 21.5									
78:10:10–94:12:27 847 0.248 2.188 -2.0 -1.5 1.6 -3.3 5.3 25.2									
CRSP Equal-Weighted Index									
62:07:10–94:12:27 1,695 0.339 2.321 20.3 6.1 9.1 4.8 94.3 109.3									
62:07:10–78:10:03 848 0.324 2.460 21.8 7.5 11.9 6.1 60.4 68.5									
78:10:10–94:12:27 847 0.354 2.174 18.4 4.3 5.5 2.2 33.7 51.3									
C. Monthly Returns									
CRSP Value-Weighted Index									
62:07:31–94:12:30 390 0.861 4.336 4.3 -5.3 -1.3 -0.4 6.8 12.5									
62:07:31–78:09:29 195 0.646 4.219 6.4 -3.8 7.3 6.2 3.9 9.7									
78:10:31–94:12:30 195 1.076 4.450 1.3 -6.3 -8.3 -7.7 7.5 14.0									
CRSP Equal-Weighted Index									
62:07:31–94:12:30 390 1.077 5.749 17.1 -3.4 -3.3 -1.6 12.8 21.3									
62:07:31–78:09:29 195 1.049 6.148 18.4 -2.5 4.4 2.4 7.5 12.6									
78:10:31–94:12:30 195 1.105 5.336 15.0 -1.6 -12.4 -7.4 8.9 14.2									

Table 3.1: Autocorrelation in daily, weekly, and monthly stock index returns. From Lo-Mackinlay, "A Non-Random Walk Down Wall Street".

Autocorrelation coefficients (in percent) and Box-Pierce Q -statistics for CRSP daily, weekly, and monthly value- and equal-weighted return indexes for the sample period from July 3, 1962 to December 30, 1994 and subperiods.

```

> var6 = ar(y, aic=TRUE, order=6)
> var6$order
[1] 1
> var6$ar
, , SUNW

SUNW          MSFT          IBM
1 -0.00985635 0.02224093 0.002072782

, , MSFT

SUNW          MSFT          IBM
1 0.008658304 -0.1369503 0.0306552

, , IBM

SUNW          MSFT          IBM
1 -0.04517035 0.0975497 -0.01283037

```

The “order” of the VAR is how many lags are significant. In this example, the order is 1. Hence, when the “ar” command is given, it shows the coefficients on the lagged values of the three value to just one lag. For example, for SUNW, the lagged coefficients are -0.0098, 0.0222, and 0.0021, respectively for SUNW, MSFT, IBM. The Akaike Information Criterion (AIC) tells us which lag is significant, and we see below that this is lag 1.

```

> var6$aic
      0       1       2       3       4       5       6
23.950676 0.000000 2.762663 5.284709 5.164238 10.065300 8.924513

```

Since the VAR was run for all six lags, the “partialacf” attribute of the output shows the coefficients of all lags.

```

> var6$partialacf
, , SUNW

SUNW          MSFT          IBM
1 -0.00985635 0.022240931 0.002072782
2 -0.07857841 -0.019721982 -0.006210487
3 0.03382375 0.003658121 0.032990758

```

```

4  0.02259522  0.030023132  0.020925226
5 -0.03944162 -0.030654949 -0.012384084
6 -0.03109748 -0.021612632 -0.003164879

```

, , MSFT

	SUNW	MSFT	IBM
1	0.008658304	-0.13695027	0.030655201
2	-0.053224374	-0.02396291	-0.047058278
3	0.080632420	0.03720952	-0.004353203
4	-0.038171317	-0.07573402	-0.004913021
5	0.002727220	0.05886752	0.050568308
6	0.242148823	0.03534206	0.062799122

, , IBM

	SUNW	MSFT	IBM
1	-0.04517035	0.097549700	-0.01283037
2	0.05436993	0.021189756	0.05430338
3	-0.08990973	-0.077140955	-0.03979962
4	0.06651063	0.056250866	0.05200459
5	0.03117548	-0.056192843	-0.06080490
6	-0.13131366	-0.003776726	-0.01502191

Interestingly we see that each of the tickers has a negative relation to its lagged value, but a positive correlation with the lagged values of the other two stocks. Hence, there is positive cross autocorrelation amongst these tech stocks. We can also run a model with three lags:

```

> ar(y, method="ols", order=3)

Call:
ar(x = y, order.max = 3, method = "ols")

$ar
, , 1

      SUNW        MSFT        IBM
SUNW 0.01407 -0.0006952 -0.036839
MSFT 0.02693 -0.1440645  0.100557

```

```
IBM  0.01330  0.0211160 -0.009662
```

```
, , 2
```

	SUNW	MSFT	IBM
SUNW	-0.082017	-0.04079	0.04812
MSFT	-0.020668	-0.01722	0.01761
IBM	-0.006717	-0.04790	0.05537

```
, , 3
```

	SUNW	MSFT	IBM
SUNW	0.035412	0.081961	-0.09139
MSFT	0.003999	0.037252	-0.07719
IBM	0.033571	-0.003906	-0.04031

```
$x.intercept
```

	SUNW	MSFT	IBM
	-9.623e-05	-7.366e-05	-6.257e-05

```
$var.pred
```

	SUNW	MSFT	IBM
SUNW	0.0013593	0.0003007	0.0002842
MSFT	0.0003007	0.0003511	0.0001888
IBM	0.0002842	0.0001888	0.0002881

We examine cross autocorrelation found across all stocks by Lo and Mackinlay in their book “A Non-Random Walk Down Wall Street” – see Table 3.2. There is strong contemporaneous correlation amongst stocks shows in the top tableau but in the one below that, the cross one-lag autocorrelation is also positive and strong. From two lags on the relationship is weaker.

3.17 Logit

When the LHS variable in a regression is categorical and binary, i.e., takes the value 1 or 0, then a logit regression is more apt. For the NCAA data, take the top 32 teams and make their dependent variable 1, and

Table 2.8. Cross-autocorrelation matrices for size-sorted portfolio returns.

$$\begin{aligned}
\widehat{\Upsilon}_0 &= \begin{pmatrix} R_{1t} & R_{2t} & R_{3t} & R_{4t} & R_{5t} \\ R_{1t} & 1.000 & 0.938 & 0.892 & 0.839 & 0.728 \\ R_{2t} & 0.938 & 1.000 & 0.976 & 0.944 & 0.856 \\ R_{3t} & 0.892 & 0.976 & 1.000 & 0.979 & 0.914 \\ R_{4t} & 0.839 & 0.944 & 0.979 & 1.000 & 0.961 \\ R_{5t} & 0.728 & 0.856 & 0.914 & 0.961 & 1.000 \end{pmatrix} \\
\widehat{\Upsilon}_1 &= \begin{pmatrix} R_{1t} & R_{2t} & R_{3t} & R_{4t} & R_{5t} \\ R_{1t-1} & 0.352 & 0.226 & 0.171 & 0.115 & 0.024 \\ R_{2t-1} & 0.330 & 0.232 & 0.182 & 0.129 & 0.037 \\ R_{3t-1} & 0.324 & 0.244 & 0.197 & 0.147 & 0.053 \\ R_{4t-1} & 0.310 & 0.242 & 0.201 & 0.153 & 0.059 \\ R_{5t-1} & 0.265 & 0.223 & 0.187 & 0.147 & 0.057 \end{pmatrix} \\
\widehat{\Upsilon}_2 &= \begin{pmatrix} R_{1t} & R_{2t} & R_{3t} & R_{4t} & R_{5t} \\ R_{1t-2} & 0.163 & 0.089 & 0.057 & 0.032 & -0.010 \\ R_{2t-2} & 0.141 & 0.078 & 0.051 & 0.029 & -0.010 \\ R_{3t-2} & 0.135 & 0.079 & 0.051 & 0.032 & -0.005 \\ R_{4t-2} & 0.121 & 0.071 & 0.046 & 0.028 & -0.006 \\ R_{5t-2} & 0.084 & 0.045 & 0.025 & 0.012 & -0.016 \end{pmatrix} \\
\widehat{\Upsilon}_3 &= \begin{pmatrix} R_{1t} & R_{2t} & R_{3t} & R_{4t} & R_{5t} \\ R_{1t-3} & 0.155 & 0.106 & 0.074 & 0.050 & 0.027 \\ R_{2t-3} & 0.141 & 0.100 & 0.071 & 0.050 & 0.031 \\ R_{3t-3} & 0.143 & 0.105 & 0.077 & 0.058 & 0.039 \\ R_{4t-3} & 0.137 & 0.104 & 0.079 & 0.061 & 0.044 \\ R_{5t-3} & 0.120 & 0.093 & 0.074 & 0.061 & 0.047 \end{pmatrix} \\
\widehat{\Upsilon}_4 &= \begin{pmatrix} R_{1t} & R_{2t} & R_{3t} & R_{4t} & R_{5t} \\ R_{1t-4} & 0.104 & 0.063 & 0.036 & 0.016 & -0.007 \\ R_{2t-4} & 0.097 & 0.062 & 0.036 & 0.017 & -0.006 \\ R_{3t-4} & 0.095 & 0.060 & 0.033 & 0.015 & -0.011 \\ R_{4t-4} & 0.100 & 0.067 & 0.039 & 0.023 & -0.004 \\ R_{5t-4} & 0.094 & 0.064 & 0.038 & 0.025 & -0.001 \end{pmatrix}
\end{aligned}$$

Table 3.2: Cross autocorrelations in US stocks. From Lo-Mackinlay, "A Non-Random Walk Down Wall Street."

Autocorrelation matrices of the vector $\mathbf{X}_t \equiv [R_{1t} \ R_{2t} \ R_{3t} \ R_{4t} \ R_{5t}]'$ where R_{it} is the week- t return on the equal-weighted portfolio of stocks in the i th quintile, $i=1,\dots,5$ (quintile 1 contains the smallest stocks), for the sample of NYSE-AMEX stocks from July 10, 1962 to December 27, 1994 (1,695 observations). Note that $\Upsilon(k) \equiv \mathbf{D}^{-1/2}\mathbf{E}[(\mathbf{X}_{t-k}-\mu)(\mathbf{X}_t-\mu)']\mathbf{D}^{-1/2}$ where $\mathbf{D} \equiv \text{diag}(\sigma_1^2, \dots, \sigma_5^2)$; thus the (i, j) th element is the correlation between R_{it-k} and R_{jt} . Asymptotic standard errors for the autocorrelations under an IID null hypothesis are given by $1/\sqrt{T} = 0.024$.

that of the bottom 32 teams zero. Hence, we split the data into the teams above average and the teams that are below average. Our goal is to fit a regression model that returns a team's probability of being above average. This is the same as the team's predicted percentile ranking.

```
> y1 = 1:32
> y1 = y1*0+1
> y1
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> y2 = y1*0
> y2
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> y = c(y1,y2)
> y
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
[34] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

We use the function `glm` (generalized linear model) for this task. Running the model is pretty easy as follows:

```
> h = glm(y~x, family=binomial(link="logit"))
> logLik(h)
'log Lik.' -21.44779 (df=12)
> summary(h)
```

Call :

`glm(formula = y ~ x, family = binomial(link = "logit"))`

Deviance Residuals :

Min	1Q	Median	3Q	Max
-1.80174	-0.40502	-0.00238	0.37584	2.31767

Coefficients :

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-45.83315	14.97564	-3.061	0.00221 **
xPTS	-0.06127	0.09549	-0.642	0.52108
xREB	0.49037	0.18089	2.711	0.00671 **
xAST	0.16422	0.26804	0.613	0.54010
xTO	-0.38405	0.23434	-1.639	0.10124
xA.T	1.56351	3.17091	0.493	0.62196
xSTL	0.78360	0.32605	2.403	0.01625 *

xBLK	0.07867	0.23482	0.335	0.73761
xPF	0.02602	0.13644	0.191	0.84874
xFG	46.21374	17.33685	2.666	0.00768 **
xFT	10.72992	4.47729	2.397	0.01655 *
xX3P	5.41985	5.77966	0.938	0.34838

Signif. codes: 0 [***] 0.001 [**] 0.01 [*] 0.05 [.] 0.1 []

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 88.723 on 63 degrees of freedom

Residual deviance: 42.896 on 52 degrees of freedom

AIC: 66.896

Number of Fisher Scoring iterations: 6

Thus, we see that the best variables that separate upper-half teams from lower-half teams are the number of rebounds and the field goal percentage. To a lesser extent, field goal percentage and steals also provide some explanatory power. The logit regression is specified as follows:

$$\begin{aligned} z &= \frac{e^y}{1+e^y} \\ y &= b_0 + b_1x_1 + b_2x_2 + \dots + b_kx_k \end{aligned}$$

The original data $z = \{0, 1\}$. The range of values of y is $(-\infty, +\infty)$. And as required, the fitted $z \in (0, 1)$. The variables x are the RHS variables.

The fitting is done using MLE.

Suppose we ran this with a simple linear regression:

```
> h = lm(y~x)
> summary(h)
```

Call:

lm(formula = y ~ x)

Residuals:

Min	1Q	Median	3Q	Max
-0.65982	-0.26830	0.03183	0.24712	0.83049

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-4.114185	1.174308	-3.503	0.000953	***
xPTS	-0.005569	0.010263	-0.543	0.589709	
xREB	0.046922	0.015003	3.128	0.002886	**
xAST	0.015391	0.036990	0.416	0.679055	
xTO	-0.046479	0.028988	-1.603	0.114905	
xA.T	0.103216	0.450763	0.229	0.819782	
xSTL	0.063309	0.028015	2.260	0.028050	*
xBLK	0.023088	0.030474	0.758	0.452082	
xPF	0.011492	0.018056	0.636	0.527253	
xFG	4.842722	1.616465	2.996	0.004186	**
xFT	1.162177	0.454178	2.559	0.013452	*
xX3P	0.476283	0.712184	0.669	0.506604	

Signif. codes:	o	[***]	0.001	[**]	0.01	[*]	0.05	[.]	0.1	[]
----------------	---	-------	-------	------	------	-----	------	-----	-----	-----

Residual standard error:	0.3905	on 52 degrees of freedom
Multiple R-Squared:	0.5043,	Adjusted R-squared: 0.3995
F-statistic:	4.81	on 11 and 52 DF, p-value: 4.514e-05

We get the same variables again showing up as significant.

3.18 Probit

We can redo the same using a probit instead. A probit is identical in spirit to the logit regression, except that the function that is used is

$$\begin{aligned} z &= \Phi(y) \\ y &= b_0 + b_1x_1 + b_2x_2 + \dots + b_kx_k \end{aligned}$$

where $\Phi(\cdot)$ is the cumulative normal probability function. It is implemented in R as follows.

```
> h = glm(y~x, family=binomial(link="probit"))
> logLik(h)
'log Lik.' -21.27924 (df=12)
> summary(h)

Call:
glm(formula = y ~ x, family = binomial(link = "probit"))
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7635295	-0.4121216	-0.0003102	0.3499560	2.2456825

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-26.28219	8.09608	-3.246	0.00117 **
xPTS	-0.03463	0.05385	-0.643	0.52020
xREB	0.28493	0.09939	2.867	0.00415 **
xAST	0.10894	0.15735	0.692	0.48874
xTO	-0.23742	0.13642	-1.740	0.08180 .
xA.T	0.71485	1.86701	0.383	0.70181
xSTL	0.45963	0.18414	2.496	0.01256 *
xBLK	0.03029	0.13631	0.222	0.82415
xPF	0.01041	0.07907	0.132	0.89529
xFG	26.58461	9.38711	2.832	0.00463 **
xFT	6.28278	2.51452	2.499	0.01247 *
xX3P	3.15824	3.37841	0.935	0.34988

Signif. codes: 0 [***] 0.001 [**] 0.01 [*] 0.05 [.] 0.1 []

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 88.723 on 63 degrees of freedom
 Residual deviance: 42.558 on 52 degrees of freedom
 AIC: 66.558

Number of Fisher Scoring iterations: 8

The results confirm those obtained from the linear regression and logit regression.

3.19 Solving Non-Linear Equations

Earlier we examined root finding. Here we develop it further. We have also not done much with user-generated functions. Here is a neat model in R to solve for the implied volatility in the Black-Merton-Scholes class of models. First, we code up the Black-Scholes (1973) model; this is the function `bms73` below. Then we write a user-defined function that solves

for the implied volatility from a given call or put option price. The package `minpack.lm` is used for the equation solving, and the function call is `nls.lm`.

The following program listing may be saved in a file called `rbc.R` and then called from the command line. The function `impvol` uses the `bms73` function and solves for the implied volatility.

```
#Black-Merton-Scholes 1973
bms73 = function(sig,S,K,T,r,q,cp,optprice) {
  d1 = (log(S/K)+(r-q+0.5*sig^2)*T)/(sig*sqrt(T))
  d2 = d1 - sig*sqrt(T)
  if (cp==1) {
    optval = S*exp(-q*T)*pnorm(d1)-K*exp(-r*T)*pnorm(d2)
  }
  else {
    optval = -S*exp(-q*T)*pnorm(-d1)+K*exp(-r*T)*pnorm(-d2)
  }
  #If option price is supplied we want the implied vol, else optprice
  bs = optval - optprice
}

#Function to return Imp Vol with starting guess sig0
impvol = function(sig0,S,K,T,r,q,cp,optprice) {
  sol = nls.lm(par=sig0,fn=bms73,S=S,K=K,T=T,r=r,q=q,
               cp=cp,optprice=optprice)
}
```

The calls to this model are as follows:

```
> library(minpack.lm)
> source("rbc.R")
> res = impvol(0.2,40,40,1,0.03,0,0,4)
> res$par
[1] 0.2915223
```

We note that the function `impvol` was written such that the argument that we needed to solve for, `sig0`, the implied volatility, was the first argument in the function. However, the expression `par=sig0` does inform the solver which argument is being searched for in order to satisfy the non-linear equation for implied volatility. Note also that the func-

tion `bms73` returns the difference between the model price and observed price, not the model price alone. This is necessary as the solver tries to set this value to zero by finding the implied volatility.

Lets check if we put this volatility back into the `bms` function that we get back the option price of 4. Voila!

```
> print(bms73(res$par[1], 40, 40, 1, 0.03, 0, 0, 0))
[1] 4
```

3.20 Web-Enabling R Functions

When building a user-friendly system it may be useful to run R programs from a web page as interface. This is quite easy to implement and the following is a simple example of how this is done. This is an extract of my blog post at

[http://sanjivdas.wordpress.com/2010/11/07/
web-enabling-r-functions-with-cgi-on-a-mac-os-x-desktop/](http://sanjivdas.wordpress.com/2010/11/07/web-enabling-r-functions-with-cgi-on-a-mac-os-x-desktop/)

This is just an example based on the “`Rcgi`” package from David Firth, and for full details of using R with CGI, see

<http://www.omegahat.org/CGIwithR/>.

You can install the package as follows:

```
install.packages("CGIwithR", repos = "http://www.omegahat.org/R", type="source")
```

The following is the Windows equivalent:¹

¹ Thanks Alice Yehjin Jun.

```
#1) Create fold "www" in "Documents", create "cgi-bin" in "www", place files in "c  
#2) Open command prompt. Run the following  
icacls "C:\Users\UserName\Documents\www\cgi-bin\Rprofile" /grant Users:(CI)(OI)F  
icacls "C:\Users\UserName\Documents\www\cgi-bin\R.cgi" /grant Users:(CI)(OI)F
```

Download the document on using R with CGI. It's titled “`CGIwithR: Facilities for Processing Web Forms with R`”.

Of course, if you don't have R at all, then download R and install it from <http://www.r-project.org/>. Then use the R package manager to install the `Rcgi` package.

You need two program files to get everything working. (a) The `html` file that is the web form for input data. (b) The `R` file, with special tags for use with the `CGIwithR` package.

Our example will be simple, i.e., a calculator to work out the monthly payment on a standard fixed rate mortgage. The three inputs are the

loan principal, annual loan rate, and the number of remaining months to maturity.

But first, let's create the html file for the web page that will take these three input values. We call it "mortgage_calc.html". The code is all standard, for those familiar with html, and even if you are not used to html, the code is self-explanatory. See Figure 3.4.

```

01 <html>
02 <head>
03 <title>Monthly Mortgage Payment Calculator</title>
04 </head>
05
06 <FORM action="/cgi-bin/R.cgi/mortgage_calc.R" method="POST">
07 <body>
08 Loan Principal: <INPUT name="L" value="" size=5><p>
09 Annual Loan Rate: <INPUT name="rL" value="" size=5><p>
10 Remaining months: <INPUT name="N" value="" size=5><p>
11
12 <p><input type="submit" size=3>
13
14 </body>
15 </html>
```

Figure 3.4: HTML code for the Rcgi application.

Notice that line 06 will be the one referencing the R program that does the calculation. The three inputs are accepted in lines 08-10. Line 12 sends the inputs to the R program.

Next, we look at the R program, suitably modified to include html tags. We name it "mortgage_calc.R". See Figure 3.5.

We can see that all html calls in the R program are made using the "tag()" construct. Lines 22–35 take in the three inputs from the html form. Lines 43–44 do the calculations and line 45 prints the result. The "cat()" function prints its arguments to the web browser page.

Okay, we have seen how the two programs (html, R) are written and these templates may be used with changes as needed. We also need to pay attention to setting up the R environment to make sure that the function is served up by the system. The following steps are needed:

Make sure that your Mac is allowing connections to its web server. Go to System Preferences and choose Sharing. In this window enable Web

Sharing by ticking the box next to it.

Place the html file "mortgage_calc.html" in the directory that serves up web pages. On a Mac, there is already a web directory for this called "Sites". It's a good idea to open a separate subdirectory called (say) "Rcgi" below this one for the R related programs and put the html file there.

The R program "mortgage_calc.R" must go in the directory that has been assigned for CGI executables. On a Mac, the default for this directory is "/Library/WebServer/CGI-Executables" and is usually referenced by the alias "cgi-bin" (stands for cgi binaries). Drop the R program into this directory. Two more important files are created when you install the Rcgi package. The CGIwithR installation creates two files: (a) A hidden file called .Rprofile; (b) A file called R.cgi. Place both these files in the directory: /Library/WebServer/CGI-Executables

If you cannot find the .Rprofile file then create it directly by opening a text editor and adding two lines to the file:

```
#!/usr/bin/R
library(CGIwithR, warn.conflicts=FALSE)
```

Now, open the R.cgi file and make sure that the line pointing to the R executable in the file is showing

```
R_DEFAULT=/usr/bin/R
```

The file may actually have it as "#!/usr/local/bin/R" which is for Linux platforms, but the usual Mac install has the executable in "#!/usr/bin/R" so make sure this is done.

Make both files executable as follows:

```
chmod a+rx .Rprofile
chmod a+rx R.cgi
```

Finally, make the ~/Sites/Rcgi/ directory write accessible:

```
chmod a+wx ~/Sites/Rcgi
```

Just being patient and following all the steps makes sure it all works well. Having done it once, it's easy to repeat and create several functions. You can try this example out on my web server at the following link.

The inputs are as follows: Loan principal (enter a dollar amount). Annual loan rate (enter it in decimals, e.g., six percent is entered as 0.06). Remaining maturity in months (enter 300 if the remaining maturity is 25 years).

Recently the open source project Shiny has become a popular approach to creating R-enabled web pages. See <http://shiny.rstudio.com/>. This creates dynamic web pages with sliders and buttons and is a powerful tool for representing analytics and visualizations.

```
01 #! /usr/bin/R
02
03 tag(HTML)
04   tag(HEAD)
05     tag(TITLE)
06       cat("Mortgage Monthly Payment Calculator")
07     untag(TITLE)
08   untag(HEAD)
09
10 tag(h3)
11   cat("Mortgage Monthly Payment Calculator")
12 untag(h3)
13
14 if(2)
15 tag(BODY)
16
17 tag(p)
18   tag(b)
19     cat("Inputs:")
20   untag(b)
21
22   tag(p)
23     L = as.numeric(scanText(formData$L))
24     cat("Loan Principal: ")
25     cat(L)
26
27   tag(p)
28     rL = as.numeric(scanText(formData$rL))
29     cat("Annual Loan Rate: ")
30     cat(rL)
31
32   tag(p)
33     N = as.numeric(scanText(formData$N))
34     cat("Remaining months: ")
35     cat(N)
36   untag(p)
37
38 if(2)
39 tag(p)
40   cat("Monthly Loan Payment: ")
41 untag(p)
42
43 r = rL/12
44 mp = r*L/(1-(1+r)^(-N))
45 cat(mp)
46
47 untag(BODY)
48 untag(HTML)
```

Figure 3.5: R code for the Rcgi application.

4

MoRe: Data Handling and Other Useful Things

In this chapter, we will revisit some of the topics considered in the previous chapters, and demonstrate alternate programming approaches in R. There are some extremely powerful packages in R that allow sql-like operations on data sets, making for advanced data handling. One of the most time-consuming activities in data analytics is cleaning and arranging data, and here we will show examples of many tools available for that purpose.

Let's assume we have a good working knowledge of R by now. Here we revisit some more packages, functions, and data structures.

4.1 Data Extraction of stocks using quantmod

We have seen the package already in the previous chapter. Now, we proceed to use it to get some initial data.

```
library(quantmod)
tickers = c("AAPL", "YHOO", "IBM", "CSCO", "C", "GSPC")
getSymbols(tickers)

[1] "AAPL"  "YHOO"  "IBM"   "CSCO"  "C"      "GSPC"
```

Print the length of each stock series. Are they all the same? Here we need to extract the ticker symbol without quotes.

```
for (t in tickers) {
  a = get(noquote(t))[,1]
  print(c(t,length(a)))
}

[1] "AAPL"  "2229"
```

```
[1] "YHOO"  "2229"
[1] "IBM"   "2229"
[1] "CSCO"  "2229"
[1] "C"      "2229"
[1] "GSPC"  "2222"
```

We see that they are not all the same. The stock series are all the same length but the S&P index is shorter by 7 days.

Convert closing adjusted prices of all stocks into individual data.frames. First, we create a list of data.frames. This will also illustrate how useful lists are because we store data.frames in lists. Notice how we also add a new column to each data.frame so that the dates column may later be used as an index to join the individual stock data.frames into one composite data.frame.

```
df = list()
j = 0
for (t in tickers) {
  j = j + 1
  a = noquote(t)
  b = data.frame(get(a)[,6])
  b$dt = row.names(b)
  df[[j]] = b
}
```

Second, we combine all the stocks adjusted closing prices into a single data.frame using a join, excluding all dates for which all stocks do not have data. The main function used here is *merge* which could be an intersect join or a union join. The default is the intersect join.

```
stock_table = df[[1]]
for (j in 2:length(df)) {
  stock_table = merge(stock_table, df[[j]], by="dt")
}
dim(stock_table)

[1] 2222     7
```

Note that the stock table contains the number of rows of the stock index, which had fewer observations than the individual stocks. So since this is an intersect join, some rows have been dropped.

Plot all stocks in a single data.frame using ggplot2, which is more

advanced than the basic plot function. We use the basic plot function first.

```
par(mfrow=c(3,2)) #Set the plot area to six plots
for (j in 1:length(tickers)) {
  plot(as.Date(stock_table[,1]), stock_table[,j+1], type="l",
       ylab=tickers[j], xlab="date")
}
par(mfrow=c(1,1)) #Set the plot figure back to a single plot
```

The plot is shown in Figure 4.1.

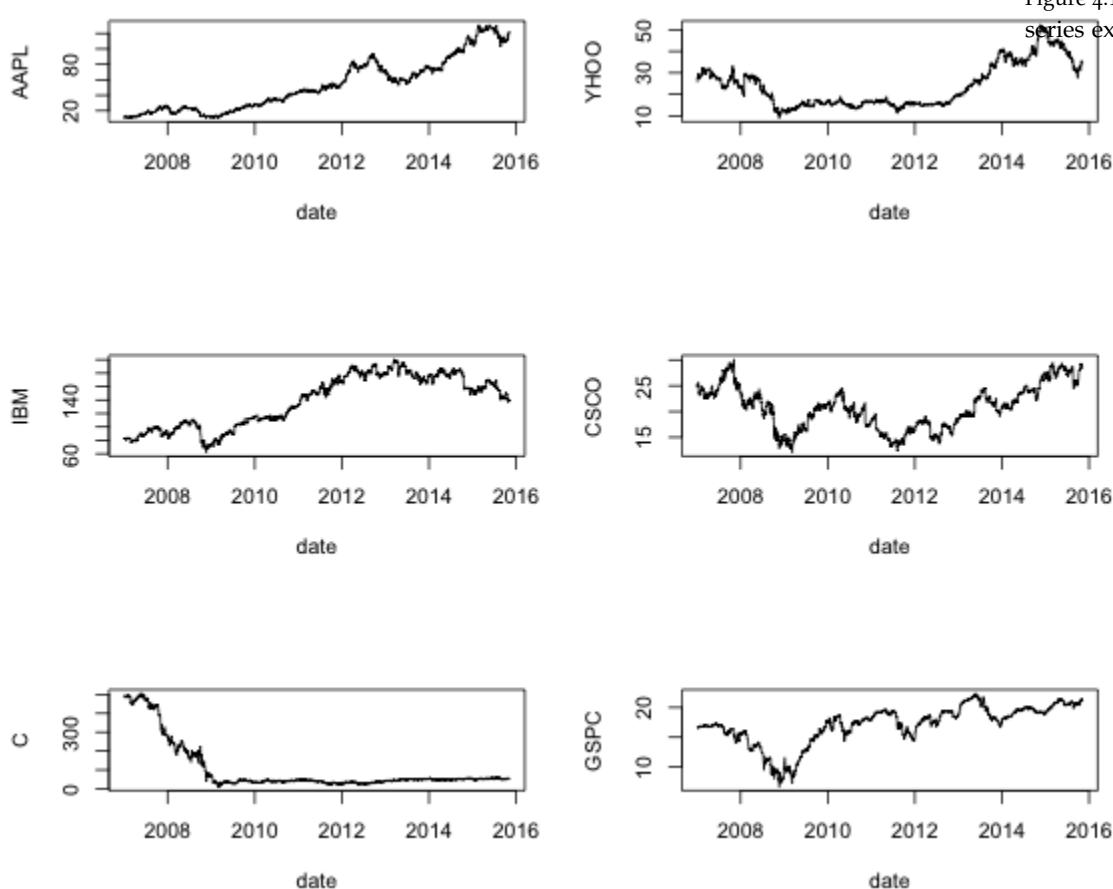


Figure 4.1: Plots of the six stock series extracted from the web.

Convert the data into returns. These are continuously compounded returns, or log returns.

```
n = length(stock_table[,1])
rets = stock_table[,2:(length(tickers)+1)]
for (j in 1:length(tickers)) {
```

```

rets[2:n,j] = diff(log(rets[,j]))
}
rets$dt = stock_table$dt
rets = rets[2:n,] #lose the first row when converting to returns
head(rets)

AAPL.Adjusted YHOO.Adjusted IBM.Adjusted CSCO.Adjusted C.Adjusted
2 0.021952927 0.047282882 0.010635139 0.0259847196 -0.0034448850
3 -0.007146655 0.032609594 -0.009094215 0.0003513139 -0.0052808346
4 0.004926130 0.006467863 0.015077743 0.0056042225 0.0050992429
5 0.079799667 -0.012252406 0.011760691 -0.0056042225 -0.0087575599
6 0.046745828 0.039806285 -0.011861828 0.0073491452 -0.0080957651
7 -0.012448245 0.017271586 -0.002429865 0.0003486195 0.0007387328

GSPC.Adjusted dt
2 -0.0003791652 2007-01-04
3 0.0000000000 2007-01-05
4 0.0093169957 2007-01-08
5 -0.0127420077 2007-01-09
6 0.0000000000 2007-01-10
7 0.0053254100 2007-01-11

```

The data.frame of returns can be used to present the descriptive statistics of returns.

```
summary(rets)
```

AAPL.Adjusted	YHOO.Adjusted	IBM.Adjusted
Min. : -0.197470	Min. : -0.2340251	Min. : -0.0864191
1st Qu.: -0.009000	1st Qu.: -0.0113101	1st Qu.: -0.0065172
Median : 0.001192	Median : 0.0002238	Median : 0.0003044
Mean : 0.001074	Mean : 0.0001302	Mean : 0.0002388
3rd Qu.: 0.012242	3rd Qu.: 0.0118051	3rd Qu.: 0.0076578
Max. : 0.130194	Max. : 0.3918166	Max. : 0.1089889
CSCO.Adjusted	C.Adjusted	GSPC.Adjusted
Min. : -0.1768648	Min. : -0.4946962	Min. : -0.1542679
1st Qu.: -0.0082048	1st Qu.: -0.0127716	1st Qu.: -0.0044266
Median : 0.0003513	Median : -0.0002122	Median : 0.0000000
Mean : 0.0000663	Mean : -0.0009834	Mean : 0.0001072
3rd Qu.: 0.0092129	3rd Qu.: 0.0120002	3rd Qu.: 0.0049999
Max. : 0.1479929	Max. : 0.4563162	Max. : 0.1967146

```
dt
Length:2221
Class :character
Mode  :character
```

Now we compute the correlation matrix of returns.

```
cor(rts[,1:length(tickers)])
```

	AAPL. Adjusted	YHOO. Adjusted	IBM. Adjusted	CSCO. Adjusted
AAPL. Adjusted	1.0000000	0.3529739	0.4887079	0.4903812
YHOO. Adjusted	0.3529739	1.0000000	0.3817138	0.4132464
IBM. Adjusted	0.4887079	0.3817138	1.0000000	0.5792123
CSCO. Adjusted	0.4903812	0.4132464	0.5792123	1.0000000
C. Adjusted	0.3739598	0.3362138	0.4322276	0.4648106
GSPC. Adjusted	0.2252352	0.1686898	0.2052341	0.2363631
	C. Adjusted	GSPC. Adjusted		
AAPL. Adjusted	0.3739598	0.2252352		
YHOO. Adjusted	0.3362138	0.1686898		
IBM. Adjusted	0.4322276	0.2052341		
CSCO. Adjusted	0.4648106	0.2363631		
C. Adjusted	1.0000000	0.3367560		
GSPC. Adjusted	0.3367560	1.0000000		

Show the correlogram for the six return series. This is a useful way to visualize the relationship between all variables in the data set. See Figure 4.2.

```
library(corrgram)
corrgram(rts[,1:length(tickers)], order=TRUE, lower.panel=panel.ellipse,
upper.panel=panel.pts, text.panel=panel.txt)
```

To see the relation between the stocks and the index, run a regression of each of the five stocks on the index returns.

```
betas = NULL
for (j in 1:(length(tickers)-1)) {
  res = lm(rts[,j]~rets[,6])
  betas[j] = res$coefficients[2]
}
print(betas)

[1] 0.2912491 0.2576751 0.1780251 0.2803140 0.8254747
```

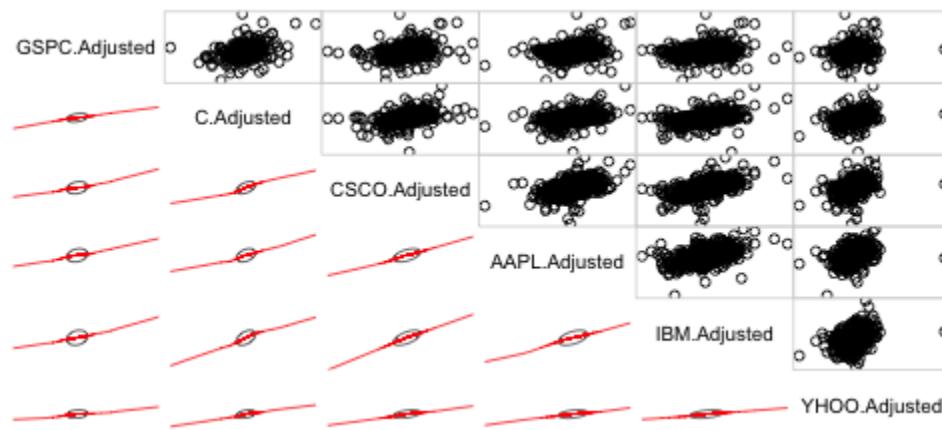


Figure 4.2: Plots of the correlation matrix of six stock series extracted from the web.

The β s indicate the level of systematic risk for each stock. We notice that all the betas are positive, and highly significant. But they are not close to unity, in fact all are lower. This is evidence of misspecification that may arise from the fact that the stocks are in the tech sector and better explanatory power would come from an index that was more relevant to the technology sector.

In order to assess whether in the cross-section, there is a relation between average returns and the systematic risk or β of a stock, run a regression of the five average returns on the five betas from the regression.

```

betas = matrix(betas)
avgrets = colMeans(rets[,1:(length(tickers)-1)])
res = lm(avgrets~betas)
summary(res)
plot(betas, avgrets)
abline(res, col="red")

```

See Figure 4.3. We see indeed, that there is an unexpected negative relation between β and the return levels. This may be on account of the particular small sample we used for illustration here, however, we note that the CAPM (Capital Asset Pricing Model) dictate that we see a positive relation between stock returns and a firm's systematic risk level.

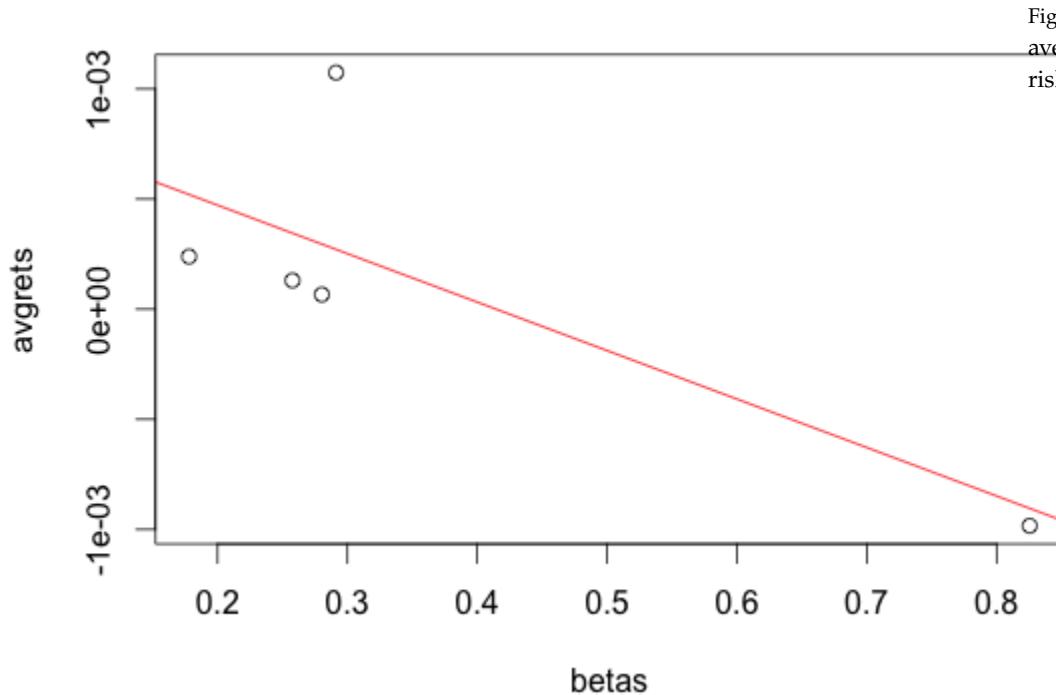


Figure 4.3: Regression of stock average returns against systematic risk (β).

4.2 Using the `merge` function

Data frames are very much like spreadsheets or tables, but they are also a lot like databases. Some sort of happy medium. If you want to join two dataframes, it is the same as joining two databases. For this R has the `merge` function. It is best illustrated with an example.

Suppose we have a list of ticker symbols and we want to generate a dataframe with more details on these tickers, especially their sector and the full name of the company. Let's look at the input list of tickers. Suppose I have them in a file called `tickers.csv` where the delimiter is the colon sign. We read this in as follows.

```
tickers = read.table("tickers.csv", header=FALSE, sep=":")
```

The line of code reads in the file and this gives us two columns of data. We can look at the top of the file (first 6 rows).

```
> head(tickers)
      V1    V2
1 NasdaqGS ACOR
2 NasdaqGS AKAM
3      NYSE ARE
```

- ```

4 NasdaqGS AMZN
5 NasdaqGS AAPL
6 NasdaqGS AREX

```

Note that the ticker symbols relate to stocks from different exchanges, in this case Nasdaq and NYSE. The file may also contain AMEX listed stocks.

The second line of code below counts the number of input tickers, and the third line of code renames the columns of the dataframe. We need to call the column of ticker symbols as “Symbol” because we will see that the dataframe with which we will merge this one also has a column with the same name. This column becomes the index on which the two dataframes are matched and joined.

```

> n = dim(tickers)[1]
> n
[1] 98
> names(tickers) = c("Exchange", "Symbol")
> head(tickers)
 Exchange Symbol
1 NasdaqGS ACOR
2 NasdaqGS AKAM
3 NYSE ARE
4 NasdaqGS AMZN
5 NasdaqGS AAPL
6 NasdaqGS AREX

```

Next, we read in lists of all stocks on Nasdaq, NYSE, and AMEX as follows:

```

library(quantmod)
nasdaq_names = stockSymbols(exchange="NASDAQ")
nyse_names = stockSymbols(exchange="NYSE")
amex_names = stockSymbols(exchange="AMEX")

```

We can look at the top of the Nasdaq file.

```

> head(nasdaq_names)
 Symbol Name LastSale MarketCap IPOyear
1 AAAP Advanced Accelerator Applications S.A. 25.20 $972.09M 2015
2 AAL American Airlines Group, Inc. 42.20 $26.6B NA
3 AAME Atlantic American Corporation 4.69 $96.37M NA
4 AAOI Applied Optoelectronics , Inc. 17.96 $302.36M 2013
5 AAON AAON, Inc. 24.13 $1.31B NA

```

|   |                |                                     |          |           |      |
|---|----------------|-------------------------------------|----------|-----------|------|
| 6 | AAPC           | Atlantic Alliance Partnership Corp. | 10.16    | \$105.54M | 2015 |
|   | Sector         | Industry                            | Exchange |           |      |
| 1 | Health Care    | Major Pharmaceuticals               | NASDAQ   |           |      |
| 2 | Transportation | Air Freight/Delivery Services       | NASDAQ   |           |      |
| 3 | Finance        | Life Insurance                      | NASDAQ   |           |      |
| 4 | Technology     | Semiconductors                      | NASDAQ   |           |      |
| 5 | Capital Goods  | Industrial Machinery/Components     | NASDAQ   |           |      |
| 6 | Finance        | Business Services                   | NASDAQ   |           |      |

Next we merge all three dataframes for each of the exchanges into one data frame.

```
co_names = rbind(nyse_names, nasdaq_names, amex_names)
```

To see how many rows are there in this merged file, we check dimensions.

```
> dim(co_names)
[1] 6801 8
```

Finally, use the merge function to combine the ticker symbols file with the exchanges data to extend the tickers file to include the information from the exchanges file.

```
> result = merge(tickers, co_names, by="Symbol")
> head(result)
 Symbol Exchange.x Name LastSale
1 AAPL NasdaqGS Apple Inc. 119.30
2 ACOR NasdaqGS Acorda Therapeutics, Inc. 37.40
3 AKAM NasdaqGS Akamai Technologies, Inc. 56.92
4 AMZN NasdaqGS Amazon.com, Inc. 668.45
5 ARE NYSE Alexandria Real Estate Equities, Inc. 91.10
6 AREX NasdaqGS Approach Resources Inc. 2.24
 MarketCap IPOyear Sector
1 $665.14B 1980 Technology
2 $1.61B 2006 Health Care
3 $10.13B 1999 Miscellaneous
4 $313.34B 1997 Consumer Services
5 $6.6B NA Consumer Services
6 $90.65M 2007 Energy
 Industry Exchange.y
1 Computer Manufacturing NASDAQ
2 Biotechnology: Biological Products (No Diagnostic Substances) NASDAQ
3 Business Services NASDAQ
4 Catalog/Specialty Distribution NASDAQ
5 Real Estate Investment Trusts NYSE
6 Oil & Gas Production NASDAQ
```

Now suppose we want to find the CEOs of these 98 companies. There is no one file with company CEO listings freely available for download.

However, sites like Google Finance have a page for each stock and mention the CEOs name on the page. By writing R code to scrape the data off these pages one by one, we can extract these CEO names and augment the tickers dataframe. The code for this is simple in R.

```
library(stringr)

#READ IN THE LIST OF TICKERS
tickers = read.table("tickers.csv", header=FALSE, sep=":")
n = dim(tickers)[1]
names(tickers) = c("Exchange", "Symbol")
tickers$ceo = NA

#PULL CEO NAMES FROM GOOGLE FINANCE
for (j in 1:n) {
 url = paste("https://www.google.com/finance?q=", tickers[j,2], sep="")
 text = readLines(url)
 idx = grep("Chief_Executive", text)
 if (length(idx)>0) {
 tickers[j,3] = str_split(text[idx-2], ">")[[1]][2]
 }
 else {
 tickers[j,3] = NA
 }
 print(tickers[j,])
}

#WRITE CEO_NAMES TO CSV
write.table(tickers, file="ceo_names.csv",
 row.names=FALSE, sep=",")
```

The code uses the `stringr` package so that string handling is simplified. After extracting the page, we search for the line in which the words “Chief Executive” show up, and we note that the name of the CEO appears two lines before in the html page. A sample web page for Apple Inc is shown in Figure 4.4.

The final dataframe with CEO names is shown here (the top 6 lines):

```
> head(tickers)
 Exchange Symbol ceo
1 NasdaqGS ACOR Ron Cohen M.D.
```

The screenshot shows the Google Finance interface for Apple Inc. (AAPL). At the top, the URL is https://www.google.com/finance?q=AAPL. The main content area displays the stock price of \$119.30, up by +0.52 (0.44%). Below the price are various financial metrics and a detailed chart showing price movement from Nov 20, 2015, to Nov 20, 2015. The chart includes a volume bar at the bottom and a time scale from 11 am to 3 pm. To the left, there's a sidebar with links for Company Summary, News, Option chain, Related companies, Historical prices, Financials, Markets, News, Portfolios, Stock screener, and Google Domestic Trends. A recent quotes section lists several stocks with their latest prices and changes. At the bottom, there are links for Settings, Technicals, and a link to this view.

Figure 4.4: Google Finance: the AAPL web page showing the URL which is needed to download the page.

This screenshot shows the 'Description' section of the Google Finance page for Apple Inc. It contains a detailed paragraph about the company's products and services, mentioning iPhone, iPad, Mac, iPod, Apple TV, iOS, OS X, iCloud, and various app stores. Below the description is a 'More from Reuters >' link. Further down, the 'Officers and directors' section lists ten individuals with their titles: Arthur D. Levinson Ph.D. (Independent Chairman of the Board), Timothy D. Cook (Chief Executive Officer, Director), Luca Maestri (Chief Financial Officer, Senior Vice President, Principal Accounting Officer), D. Bruce Sewell (Senior Vice President, General Counsel, Secretary), Philip W. Schiller (Senior Vice President - Worldwide Marketing), Angela J. Ahrendts (Senior Vice President - Retail and Online Stores), Eduardo H. Cue (Senior Vice President - Internet Software and Services), Craig Federighi (Senior Vice President - Software Engineering), Daniel Riccio (Senior Vice President - Hardware Engineering), and Jeffrey E. Williams (Senior Vice President - Operations).

|   |          |      |                          |
|---|----------|------|--------------------------|
| 2 | NasdaqGS | AKAM | F. Thomson Leighton      |
| 3 | NYSE     | ARE  | Joel S. Marcus J.D., CPA |
| 4 | NasdaqGS | AMZN | Jeffrey P. Bezos         |
| 5 | NasdaqGS | AAPL | Timothy D. Cook          |
| 6 | NasdaqGS | AREX | J. Ross Craft            |

### 4.3 Using the apply class of functions

Sometimes we need to apply a function to many cases, and these case parameters may be supplied in a vector, matrix, or list. This is analogous to looping through a set of values to repeat evaluations of a function using different sets of parameters. We illustrate here by computing the mean returns of all stocks in our sample using the `apply` function. The first argument of the function is the `data.frame` to which it is being applied, the second argument is either `1` (by rows) or `2` (by columns). The third argument is the function being evaluated.

```
apply(rets[, 1:(length(tickers)-1)], 2, mean)
```

|                |                |               |                |               |
|----------------|----------------|---------------|----------------|---------------|
| AAPL. Adjusted | YHOO. Adjusted | IBM. Adjusted | CSCO. Adjusted | C. Adjusted   |
| 1.073902e-03   | 1.302309e-04   | 2.388207e-04  | 6.629946e-05   | -9.833602e-04 |

We see that the function returns the column means of the data set. The variants of the function pertain to what the loop is being applied to. The `lapply` is a function applied to a list, and `sapply` is for matrices and vectors. Likewise, `mapply` uses multiple arguments.

To cross check, we can simply use the `colMeans` function:

```
colMeans(rets[, 1:(length(tickers)-1)])
```

|                |                |               |                |               |
|----------------|----------------|---------------|----------------|---------------|
| AAPL. Adjusted | YHOO. Adjusted | IBM. Adjusted | CSCO. Adjusted | C. Adjusted   |
| 1.073902e-03   | 1.302309e-04   | 2.388207e-04  | 6.629946e-05   | -9.833602e-04 |

As we see, this result is verified.

### 4.4 Getting interest rate data from FRED

In finance, data on interest rates is widely used. An authoritative source of data on interest rates is FRED (Federal Reserve Economic Data), maintained by the St. Louis Federal Reserve Bank, and is warehoused at the

following web site: <https://research.stlouisfed.org/fred2/>. Let's assume that we want to download the data using R from FRED directly. To do this we need to write some custom code. There used to be a package for this but since the web site changed, it has been updated but does not work properly. Still, see that it is easy to roll your own code quite easily in R.

```
#FUNCTION TO READ IN CSV FILES FROM FRED
#Enter SeriesID as a text string
readFRED = function(SeriesID) {
 url = paste("https://research.stlouisfed.org/fred2/series/", SeriesID,
 "/downloaddata/", SeriesID, ".csv", sep="")
 data = readLines(url)
 n = length(data)
 data = data[2:n]
 n = length(data)
 df = matrix(0, n, 2) #top line is header
 for (j in 1:n) {
 tmp = strsplit(data[j], ",")
 df[j, 1] = tmp[[1]][1]
 df[j, 2] = tmp[[1]][2]
 }
 rate = as.numeric(df[, 2])
 idx = which(rate > 0)
 idx = setdiff(seq(1, n), idx)
 rate[idx] = -99
 date = df[, 1]
 df = data.frame(date, rate)
 names(df)[2] = SeriesID
 result = df
}
```

Now, we provide a list of economic time series and download data accordingly using the function above. Note that we also join these individual series using the data as index. We download constant maturity interest rates (yields) starting from a maturity of one month (DGS1MO) to a maturity of thirty years (DGS30).

```
#EXTRACT TERM STRUCTURE DATA FOR ALL RATES FROM 1 MO to 30 YRS FROM FRED
id_list = c("DGS1MO", "DGS3MO", "DGS6MO", "DGS1", "DGS2", "DGS3", "DGS5", "DGS7",
"DGS10", "DGS20", "DGS30")
```

```

k = 0
for (id in id_list) {
 out = readFRED(id)
 if (k>0) { rates = merge(rates,out,"date",all=TRUE) }
 else { rates = out }
 k = k + 1
}

> head(rates)

 date DGS1MO DGS3MO DGS6MO DGS1 DGS2 DGS3 DGS5 DGS7 DGS10 DGS20 DGS30
1 2001-07-31 3.67 3.54 3.47 3.53 3.79 4.06 4.57 4.86 5.07 5.61
5.51
2 2001-08-01 3.65 3.53 3.47 3.56 3.83 4.09 4.62 4.90 5.11 5.63
5.53
3 2001-08-02 3.65 3.53 3.46 3.57 3.89 4.17 4.69 4.97 5.17 5.68
5.57
4 2001-08-03 3.63 3.52 3.47 3.57 3.91 4.22 4.72 4.99 5.20 5.70
5.59
5 2001-08-06 3.62 3.52 3.47 3.56 3.88 4.17 4.71 4.99 5.19 5.70
5.59
6 2001-08-07 3.63 3.52 3.47 3.56 3.90 4.19 4.72 5.00 5.20 5.71
5.60

```

Having done this, we now have a data.frame called `rates` containing all the time series we are interested in. We now convert the dates into numeric strings and sort the data.frame by date.

```

#CONVERT ALL DATES TO NUMERIC AND SORT BY DATE
dates = rates[,1]
library(stringr)
dates = as.numeric(str_replace_all(dates,"-",""))
res = sort(dates,index.return=TRUE)
for (j in 1:dim(rates)[2]) {
 rates[,j] = rates[res$ix,j]
}

> head(rates)
 date DGS1MO DGS3MO DGS6MO DGS1 DGS2 DGS3 DGS5 DGS7 DGS10 DGS20 DGS30
1 1962-01-02 NA NA NA 3.22 NA 3.70 3.88 NA 4.06 NA

```

|              |    |    |    |      |    |      |      |    |      |    |  |
|--------------|----|----|----|------|----|------|------|----|------|----|--|
| NA           |    |    |    |      |    |      |      |    |      |    |  |
| 2 1962-01-03 | NA | NA | NA | 3.24 | NA | 3.70 | 3.87 | NA | 4.03 | NA |  |
| NA           |    |    |    |      |    |      |      |    |      |    |  |
| 3 1962-01-04 | NA | NA | NA | 3.24 | NA | 3.69 | 3.86 | NA | 3.99 | NA |  |
| NA           |    |    |    |      |    |      |      |    |      |    |  |
| 4 1962-01-05 | NA | NA | NA | 3.26 | NA | 3.71 | 3.89 | NA | 4.02 | NA |  |
| NA           |    |    |    |      |    |      |      |    |      |    |  |
| 5 1962-01-08 | NA | NA | NA | 3.31 | NA | 3.71 | 3.91 | NA | 4.03 | NA |  |
| NA           |    |    |    |      |    |      |      |    |      |    |  |
| 6 1962-01-09 | NA | NA | NA | 3.32 | NA | 3.74 | 3.93 | NA | 4.05 | NA |  |
| NA           |    |    |    |      |    |      |      |    |      |    |  |

Note that there are missing values, denoted by NA. Also there are rows with "-99" values and we can clean those out too but they represent periods when there was no yield available of that maturity, so we leave this in.

| #REMOVE THE NA ROWS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |        |        |        |        |      |      |      |      |       |       |       |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------|--------|--------|--------|------|------|------|------|-------|-------|-------|-------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|-------|------------|------|------|------|------|------|------|------|------|------|------|------|
| idx = which(rowSums(is.na(rates))==0)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |            |        |        |        |        |      |      |      |      |       |       |       |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| rates2 = rates[idx,]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |        |        |        |        |      |      |      |      |       |       |       |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| print(head(rates2))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |        |        |        |        |      |      |      |      |       |       |       |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |            |        |        |        |        |      |      |      |      |       |       |       |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| <table> <thead> <tr> <th></th> <th>date</th> <th>DGS1MO</th> <th>DGS3MO</th> <th>DGS6MO</th> <th>DGS1</th> <th>DGS2</th> <th>DGS3</th> <th>DGS5</th> <th>DGS7</th> <th>DGS10</th> <th>DGS20</th> <th>DGS30</th> </tr> </thead> <tbody> <tr> <td>10326</td> <td>2001-07-31</td> <td>3.67</td> <td>3.54</td> <td>3.47</td> <td>3.53</td> <td>3.79</td> <td>4.06</td> <td>4.57</td> <td>4.86</td> <td>5.07</td> <td>5.61</td> <td>5.51</td> </tr> <tr> <td>10327</td> <td>2001-08-01</td> <td>3.65</td> <td>3.53</td> <td>3.47</td> <td>3.56</td> <td>3.83</td> <td>4.09</td> <td>4.62</td> <td>4.90</td> <td>5.11</td> <td>5.63</td> <td>5.53</td> </tr> <tr> <td>10328</td> <td>2001-08-02</td> <td>3.65</td> <td>3.53</td> <td>3.46</td> <td>3.57</td> <td>3.89</td> <td>4.17</td> <td>4.69</td> <td>4.97</td> <td>5.17</td> <td>5.68</td> <td>5.57</td> </tr> <tr> <td>10329</td> <td>2001-08-03</td> <td>3.63</td> <td>3.52</td> <td>3.47</td> <td>3.57</td> <td>3.91</td> <td>4.22</td> <td>4.72</td> <td>4.99</td> <td>5.20</td> <td>5.70</td> <td>5.59</td> </tr> <tr> <td>10330</td> <td>2001-08-06</td> <td>3.62</td> <td>3.52</td> <td>3.47</td> <td>3.56</td> <td>3.88</td> <td>4.17</td> <td>4.71</td> <td>4.99</td> <td>5.19</td> <td>5.70</td> <td>5.59</td> </tr> <tr> <td>10331</td> <td>2001-08-07</td> <td>3.63</td> <td>3.52</td> <td>3.47</td> <td>3.56</td> <td>3.90</td> <td>4.19</td> <td>4.72</td> <td>5.00</td> <td>5.20</td> <td>5.71</td> <td>5.60</td> </tr> </tbody> </table> |            | date   | DGS1MO | DGS3MO | DGS6MO | DGS1 | DGS2 | DGS3 | DGS5 | DGS7  | DGS10 | DGS20 | DGS30 | 10326 | 2001-07-31 | 3.67 | 3.54 | 3.47 | 3.53 | 3.79 | 4.06 | 4.57 | 4.86 | 5.07 | 5.61 | 5.51 | 10327 | 2001-08-01 | 3.65 | 3.53 | 3.47 | 3.56 | 3.83 | 4.09 | 4.62 | 4.90 | 5.11 | 5.63 | 5.53 | 10328 | 2001-08-02 | 3.65 | 3.53 | 3.46 | 3.57 | 3.89 | 4.17 | 4.69 | 4.97 | 5.17 | 5.68 | 5.57 | 10329 | 2001-08-03 | 3.63 | 3.52 | 3.47 | 3.57 | 3.91 | 4.22 | 4.72 | 4.99 | 5.20 | 5.70 | 5.59 | 10330 | 2001-08-06 | 3.62 | 3.52 | 3.47 | 3.56 | 3.88 | 4.17 | 4.71 | 4.99 | 5.19 | 5.70 | 5.59 | 10331 | 2001-08-07 | 3.63 | 3.52 | 3.47 | 3.56 | 3.90 | 4.19 | 4.72 | 5.00 | 5.20 | 5.71 | 5.60 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | date       | DGS1MO | DGS3MO | DGS6MO | DGS1   | DGS2 | DGS3 | DGS5 | DGS7 | DGS10 | DGS20 | DGS30 |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10326                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-07-31 | 3.67   | 3.54   | 3.47   | 3.53   | 3.79 | 4.06 | 4.57 | 4.86 | 5.07  | 5.61  | 5.51  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10327                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-08-01 | 3.65   | 3.53   | 3.47   | 3.56   | 3.83 | 4.09 | 4.62 | 4.90 | 5.11  | 5.63  | 5.53  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10328                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-08-02 | 3.65   | 3.53   | 3.46   | 3.57   | 3.89 | 4.17 | 4.69 | 4.97 | 5.17  | 5.68  | 5.57  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10329                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-08-03 | 3.63   | 3.52   | 3.47   | 3.57   | 3.91 | 4.22 | 4.72 | 4.99 | 5.20  | 5.70  | 5.59  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10330                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-08-06 | 3.62   | 3.52   | 3.47   | 3.56   | 3.88 | 4.17 | 4.71 | 4.99 | 5.19  | 5.70  | 5.59  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |
| 10331                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 2001-08-07 | 3.63   | 3.52   | 3.47   | 3.56   | 3.90 | 4.19 | 4.72 | 5.00 | 5.20  | 5.71  | 5.60  |       |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |       |            |      |      |      |      |      |      |      |      |      |      |      |

## 4.5 Cross-Sectional Data (an example)

A great resource for data sets in corporate finance is on Aswath Damodaran's web site, see:

[http://people.stern.nyu.edu/adamodar/New\\_Home\\_Page/data.html](http://people.stern.nyu.edu/adamodar/New_Home_Page/data.html)

Financial statement data sets are available at:

<http://www.sec.gov/dera/data/financial-statement-data-sets.html>

And another comprehensive data source:

<http://fisher.osu.edu/fin/fdf/osudata.htm>

Open government data: <https://www.data.gov/finance/>

Let's read in the list of failed banks:

<http://www.fdic.gov/bank/individual/failed/banklist.csv>

```
#download.file(url = "http://www.fdic.gov/bank/individual/failed/banklist.csv", destfile = "failed_banks.csv")
```

(This does not work, and has been an issue for a while.)

You can also read in the data using `readLines` but then further work is required to clean it up, but it works well in downloading the data.

```
data = readLines("https://www.fdic.gov/bank/individual/failed/banklist.csv")
head(data)

[1] "Bank_Name,City,ST,CERT,Acquiring_Institution,Closing_Date,Updated_Date"
[2] "Hometown_National_Bank,Longview,WA,35156,Twin_City_Bank,2-Oct-15,15-Oct-15"
[3] "The_Bank_of_Georgia,Peachtree_City,GA,35259,Fidelity_Bank,2-Oct-15,15-Oct-15"
[4] "Premier_Bank,Denver,CO,34112,\\"United Fidelity Bank, fsb\",10-Jul-15,28-Jul-15"
[5] "Edgebrook_Bank,Chicago,IL,57772,Republic_Bank_of_Chicago,8-May-15,23-Jul-15"
[6] "Doral_Bank,San_Juan,PR,32102,Banco_Popular_de_Puerto_Rico,27-Feb-15,13-May-15"
```

It may be simpler to just download the data and read it in from the csv file:

```
data = read.csv("banklist.csv", header=TRUE)
print(names(data))

[1] "Bank.Name" "City" "ST"
[4] "CERT" "Acquiring.Institution" "Closing.Date"
[7] "Updated.Date"
```

This gives a data.frame which is easy to work with. We will illustrate some interesting ways in which to manipulate this data. Suppose we want to get subtotals of how many banks failed by state. First add a column of ones to the data.frame.

```
print(head(data))
data$count = 1
print(head(data))
```

| Bank.Name | City | ST |
|-----------|------|----|
|           |      |    |

```

1 Hometown National Bank Longview WA
2 The Bank of Georgia Peachtree City GA
3 Premier Bank Denver CO
4 Edgebrook Bank Chicago IL
5 Doral Bank San Juan PR
6 Capitol City Bank & Trust Company Atlanta GA
 CERT Acquiring.Institution Closing.Date
1 35156 Twin City Bank 2-Oct-15
2 35259 Fidelity Bank 2-Oct-15
3 34112 United Fidelity Bank, fsb 10-Jul-15
4 57772 Republic Bank of Chicago 8-May-15
5 32102 Banco Popular de Puerto Rico 27-Feb-15
6 33938 First-Citizens Bank & Trust Company 13-Feb-15
 Updated.Date count
1 15-Oct-15 1
2 15-Oct-15 1
3 28-Jul-15 1
4 23-Jul-15 1
5 13-May-15 1
6 21-Apr-15 1

```

It's good to check that there is no missing data.

```

any(is.na(data))
[1] FALSE

```

Now we sort the data by state to see how many there are.

```

res = sort(as.matrix(data$ST), index.return=TRUE)
print(data[res$ix,])
print(sort(unique(data$ST)))

[1] AL AR AZ CA CO CT FL GA HI IA ID IL IN KS KY LA MA MD
[19] MI MN MO MS NC NE NH NJ NM NV NY OH OK OR PA PR SC SD
[37] TN TX UT VA WA WI WV WY
44 Levels: AL AR AZ CA CO CT FL GA HI IA ID IL IN ... WY

print(length(unique(data$ST)))

[1] 44

```

We can directly use the aggregate function to get subtotals by state.

```
head(aggregate(count ~ ST, data, sum), 10)
```

|    | ST | count |
|----|----|-------|
| 1  | AL | 7     |
| 2  | AR | 3     |
| 3  | AZ | 16    |
| 4  | CA | 41    |
| 5  | CO | 10    |
| 6  | CT | 2     |
| 7  | FL | 75    |
| 8  | GA | 92    |
| 9  | HI | 1     |
| 10 | IA | 2     |

And another example, subtotal by acquiring bank. Note how we take the subtotals into another data.frame, which is then sorted and returned in order using the index of the sort.

```
acq = aggregate(count~Acquiring.Institution, data, sum)
idx = sort(as.matrix(acq$count), decreasing=TRUE, index.return=TRUE)$ix
head(acq[idx,], 15)
```

|     | Acquiring.Institution               | count |
|-----|-------------------------------------|-------|
| 170 | No Acquirer                         | 31    |
| 224 | State Bank and Trust Company        | 12    |
| 10  | Ameris Bank                         | 10    |
| 262 | U.S. Bank N.A.                      | 9     |
| 67  | Community & Southern Bank           | 8     |
| 28  | Bank of the Ozarks                  | 7     |
| 47  | Centennial Bank                     | 7     |
| 112 | First-Citizens Bank & Trust Company | 7     |
| 228 | Stearns Bank, N.A.                  | 7     |
| 49  | CenterState Bank of Florida, N.A.   | 6     |
| 50  | Central Bank                        | 6     |
| 154 | MB Financial Bank, N.A.             | 6     |
| 205 | Republic Bank of Chicago            | 6     |
| 54  | CertusBank, National Association    | 5     |
| 64  | Columbia State Bank                 | 5     |

## 4.6 Handling dates with lubridate

Suppose we want to take the preceding data.frame of failed banks and aggregate the data by year, or month, etc. In this case, it us useful to use a dates package. Another useful tool developed by Hadley Wickham is the `lubridate` package.

```
head(data)

 Bank.Name City ST CERT
1 Hometown National Bank Longview WA 35156
2 The Bank of Georgia Peachtree City GA 35259
3 Premier Bank Denver CO 34112
4 Edgebrook Bank Chicago IL 57772
5 Doral Bank San Juan PR 32102
6 Capitol City Bank & Trust Company Atlanta GA 33938
 Acquiring.Institution Closing.Date Updated.Date count
1 Twin City Bank 2-Oct-15 15-Oct-15 1
2 Fidelity Bank 2-Oct-15 15-Oct-15 1
3 United Fidelity Bank, fsb 10-Jul-15 28-Jul-15 1
4 Republic Bank of Chicago 8-May-15 23-Jul-15 1
5 Banco Popular de Puerto Rico 27-Feb-15 13-May-15 1
6 First-Citizens Bank & Trust Company 13-Feb-15 21-Apr-15 1
 Cdate Cyear
1 2015-10-02 2015
2 2015-10-02 2015
3 2015-07-10 2015
4 2015-05-08 2015
5 2015-02-27 2015
6 2015-02-13 2015

library(lubridate)
data$Cdate = dmy(data$Closing.Date)
data$Cyear = year(data$Cdate)
fd = aggregate(count~Cyear,data,sum)
print(fd)

 Cyear count
1 2000 2
```

```

2 2001 4
3 2002 11
4 2003 3
5 2004 4
6 2007 3
7 2008 25
8 2009 140
9 2010 157
10 2011 92
11 2012 51
12 2013 24
13 2014 18
14 2015 8

```

```

plot(count~Cyear , data=fd , type="l" , lwd=3 , col="red" xlab="Year")
grid(lwd=3)

```

See the results in Figure 4.5.

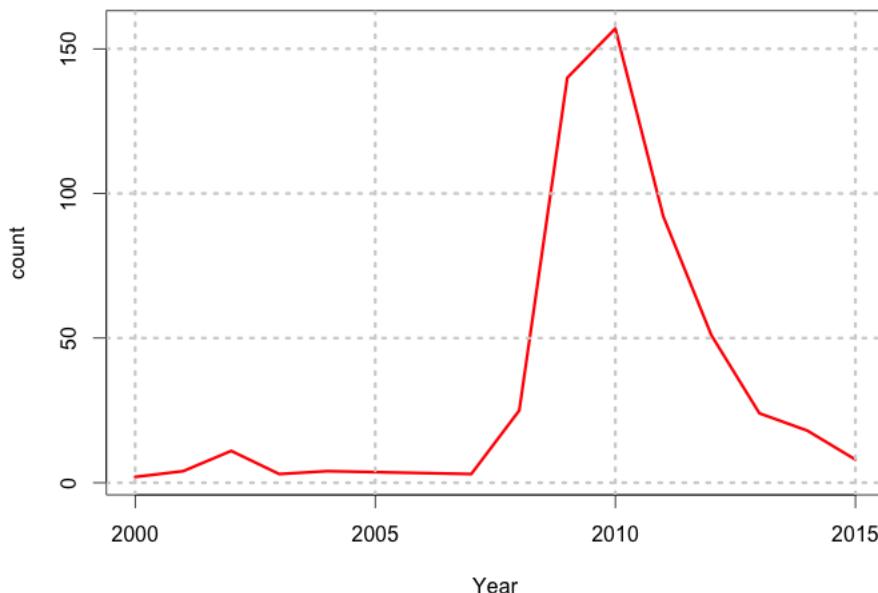


Figure 4.5: Failed bank totals by year.

Let's do the same thing by month to see if there is seasonality

```

data$Cmonth = month(data$Cdate)
fd = aggregate(count~Cmonth, data ,sum)
print(fd)

```

```
Cmonth count
1 1 49
2 2 44
3 3 38
4 4 57
5 5 40
6 6 36
7 7 74
8 8 40
9 9 37
10 10 58
11 11 35
12 12 34
```

```
plot(count~Cmonth, data=fd, type="l", lwd=3, col="green"); grid(lwd=3)
```

There does not appear to be any seasonality. What about day?

```
data$Cday = day(data$Cdate)
fd = aggregate(count~Cday, data, sum)
print(fd)
```

```
Cday count
1 1 8
2 2 20
3 3 3
4 4 21
5 5 15
6 6 13
7 7 20
8 8 14
9 9 10
10 10 14
11 11 17
12 12 10
13 13 14
14 14 20
15 15 20
16 16 22
17 17 23
```

```

18 18 21
19 19 29
20 20 27
21 21 17
22 22 18
23 23 30
24 24 19
25 25 13
26 26 15
27 27 18
28 28 18
29 29 15
30 30 30
31 31 8

```

```
plot(count~Cday, data=fd, type="l", lwd=3, col="blue"); grid(lwd=3)
```

Definitely, counts are lower at the start and end of the month!

#### 4.7 Using the `data.table` package

This is an incredibly useful package that was written by Matt Dowle.

It essentially allows your `data.frame` to operate as a database. It enables very fast handling of massive quantities of data, and much of this technology is now embedded in the IP of the company called `h2o`:

<http://h2o.ai/>

We start with some freely downloadable crime data statistics for California. We placed the data in a csv file which is then easy to read in to R.

```
data = read.csv("CA_Crimes_Data_2004-2013.csv", header=TRUE)
```

It is easy to convert this into a `data.table`.

```
library(data.table)
D_T = as.data.table(data)
```

Let's see how it works, noting that the syntax is similar to that for `data.frames` as much as possible. We print only a part of the names list. And do not go through each and everyone.

```
print(dim(D_T))
```

```
[1] 7301 69

print(names(D_T))

[1] "Year" "County" "NCICCode"
[4] "Violent_sum" "Homicide_sum" "ForRape_sum"
[7] "Robbery_sum" "AggAssault_sum" "Property_sum"
[10] "Burglary_sum" "VehicleTheft_sum" "LTtotal_sum"
.....
head(D_T)
```

A nice feature of the `data.table` is that it can be indexed, i.e., resorted on the fly by making any column in the database the key. Once that is done, then it becomes easy to compute subtotals, and generate plots from these subtotals as well.

```
setkey(D_T, Year)

crime = 6
res = D_T[,sum(ForRape_sum),by=Year]
print(res)

 Year V1
1: 2004 9598
2: 2005 9345
3: 2006 9213
4: 2007 9047
5: 2008 8906
6: 2009 8698
7: 2010 8325
8: 2011 7678
9: 2012 7828
10: 2013 7459

class(res)

[1] "data.table" "data.frame"
```

See that the type of output is also of the type `data.table`, and includes the class `data.frame` also.

Next, we plot the results from the `data.table` in the same way as we would for a `data.frame`. See Figure 4.6.

```
plot(res$Year, res$V1, type="b", lwd=3, col="blue",
 xlab="Year", ylab="Forced_Rape")
```

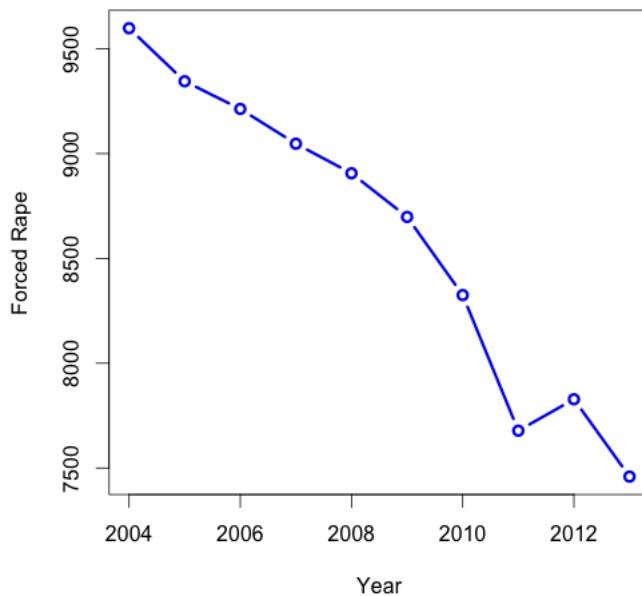


Figure 4.6: Rape totals by year.

Repeat the process looking at crime (Rape) totals by county.

```
setkey(D_T, County)
res = D_T[, sum(ForRape_sum), by=County]
print(res)
setnames(res, "V1", "Rapes")

County_Rapes = as.data.table(res) #This is not really needed
setkey(County_Rapes, Rapes)
County_Rapes
```

|    | County          | Rapes |
|----|-----------------|-------|
| 1: | Sierra County   | 2     |
| 2: | Alpine County   | 15    |
| 3: | Trinity County  | 28    |
| 4: | Mariposa County | 46    |

|     |                        |      |
|-----|------------------------|------|
| 5:  | Inyo County            | 52   |
| 6:  | Glenn County           | 56   |
| 7:  | Colusa County          | 60   |
| 8:  | Mono County            | 61   |
| 9:  | Modoc County           | 64   |
| 10: | Lassen County          | 96   |
| 11: | Plumas County          | 115  |
| 12: | Siskiyou County        | 143  |
| 13: | Calaveras County       | 148  |
| 14: | San Benito County      | 151  |
| 15: | Amador County          | 153  |
| 16: | Tuolumne County        | 160  |
| 17: | Tehama County          | 165  |
| 18: | Nevada County          | 214  |
| 19: | Del Norte County       | 236  |
| 20: | Lake County            | 262  |
| 21: | Imperial County        | 263  |
| 22: | Sutter County          | 274  |
| 23: | Yuba County            | 277  |
| 24: | Mendocino County       | 328  |
| 25: | El Dorado County       | 351  |
| 26: | Napa County            | 354  |
| 27: | Kings County           | 356  |
| 28: | Madera County          | 408  |
| 29: | Marin County           | 452  |
| 30: | Humboldt County        | 495  |
| 31: | Placer County          | 611  |
| 32: | Yolo County            | 729  |
| 33: | Merced County          | 738  |
| 34: | Santa Cruz County      | 865  |
| 35: | San Luis Obispo County | 900  |
| 36: | Butte County           | 930  |
| 37: | Monterey County        | 1062 |
| 38: | Shasta County          | 1089 |
| 39: | Tulare County          | 1114 |
| 40: | Ventura County         | 1146 |
| 41: | Solano County          | 1150 |
| 42: | Stanislaus County      | 1348 |

```

43: Santa Barbara County 1352
44: San Mateo County 1381
45: San Francisco County 1498
46: Sonoma County 1558
47: San Joaquin County 1612
48: Contra Costa County 1848
49: Kern County 1935
50: Fresno County 1960
51: Santa Clara County 3832
52: Sacramento County 4084
53: Riverside County 4321
54: Orange County 4509
55: San Bernardino County 4900
56: Alameda County 4979
57: San Diego County 7378
58: Los Angeles County 21483

```

Now, we can go ahead and plot it using a different kind of plot, a horizontal barplot.

```

par(las=2) #makes label horizontal
#par(mar=c(3,4,2,1)) #increase y-axis margins
barplot(County_Rapes$Rapes, names.arg=County_Rapes$County,
horiz=TRUE, cex.names=0.4, col=8)

```

#### 4.8 Another data set: Bay Area Bike Share data

We show some other features using a different data set, the bike information on Silicon Valley routes for the Bike Share program. This is a much larger data set.

```

trips = read.csv("201408_trip_data.csv", header=TRUE)
print(names(trips))

[1] "Trip.ID" "Duration" "Start.Date"
[4] "Start.Station" "Start.Terminal" "End.Date"
[7] "End.Station" "End.Terminal" "Bike.."
[10] "Subscriber.Type" "Zip.Code"

```

Next we print some descriptive statistics.

```
print(length(trips$Trip.ID))
```

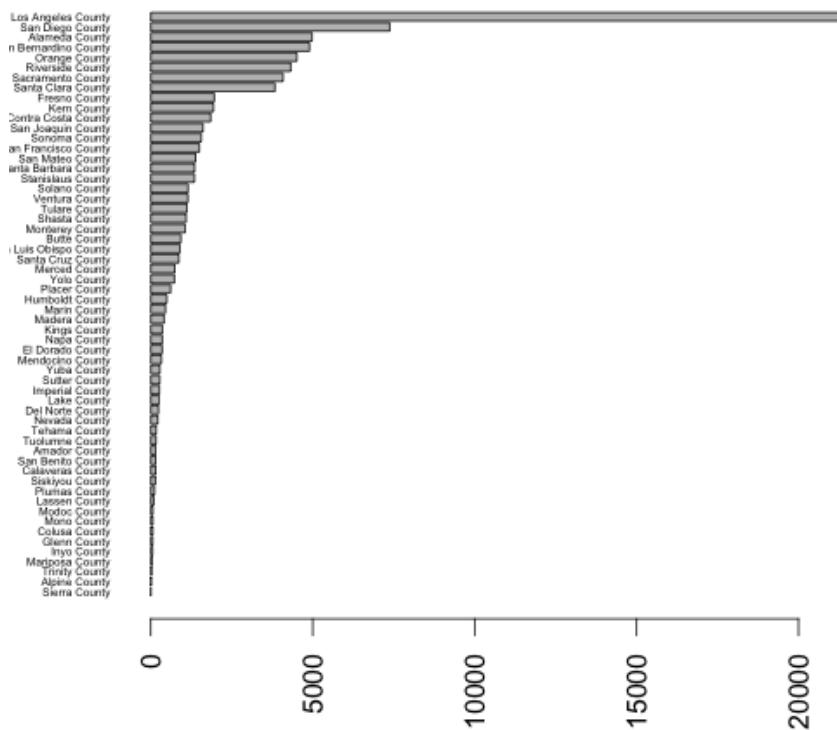


Figure 4.7: Rape totals by county.

```
[1] 171792
```

```
print(summary(trips$Duration/60))
```

| Min.  | 1st Qu. | Median | Mean   | 3rd Qu. | Max.      |
|-------|---------|--------|--------|---------|-----------|
| 1.000 | 5.750   | 8.617  | 18.880 | 12.680  | 11940.000 |

```
print(mean(trips$Duration/60, trim=0.01))
```

```
[1] 13.10277
```

Now, we quickly check how many start and end stations there are.

```
start_stn = unique(trips$Start.Terminus)
print(sort(start_stn))
```

```
[1] 2 3 4 5 6 7 8 9 10 11 12 13 14 16 21 22 23 24 25 26 27 28
[23] 29 30 31 32 33 34 35 36 37 38 39 41 42 45 46 47 48 49 50 51 54 55
[45] 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
```

```
[67] 80 82 83 84

print(length(start_stn))
[1] 70

end_stn = unique(trips$End.Terminus)
print(sort(end_stn))

[1] 2 3 4 5 6 7 8 9 10 11 12 13 14 16 21 22 23 24 25 26 27 28
[23] 29 30 31 32 33 34 35 36 37 38 39 41 42 45 46 47 48 49 50 51 54 55
[45] 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
[67] 80 82 83 84

print(length(end_stn))
[1] 70
```

As we can see, there are quite a few stations in the bike share program where riders can pick up and drop off bikes. The trip duration information is stored in seconds, so has been converted to minutes in the code above.

## 4.9 Using the `plyr` package family

This package by Hadley Wickham is useful for applying functions to tables of data, i.e., `data.frames`. Since we may want to write custom functions, this is a highly useful package. R users often select either the `data.table` or the `plyr` class of packages for handling `data.frames` as databases. The latest incarnation is the `dplyr` package, which focuses only on `data.frames`.

```
require(plyr)
library(dplyr)
```

One of the useful things you can use is the `filter` function, to subset the rows of the dataset you might want to select for further analysis.

```
res = filter(trips, Start.Terminus==50, End.Terminus==51)
head(res)

Trip.ID Duration Start.Date
1 432024 3954 8/30/2014 14:46
```

|   |                       |                  |               |                |            |                 |          |
|---|-----------------------|------------------|---------------|----------------|------------|-----------------|----------|
| 2 | 432022                | 4120             | 8/30/2014     | 14:44          |            |                 |          |
| 3 | 431895                | 1196             | 8/30/2014     | 12:04          |            |                 |          |
| 4 | 431891                | 1249             | 8/30/2014     | 12:03          |            |                 |          |
| 5 | 430408                | 145              | 8/29/2014     | 9:08           |            |                 |          |
| 6 | 429148                | 862              | 8/28/2014     | 13:47          |            |                 |          |
|   |                       |                  | Start.Station | Start.Terminal |            | End.Date        |          |
| 1 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/30/2014       | 15:52    |
| 2 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/30/2014       | 15:52    |
| 3 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/30/2014       | 12:24    |
| 4 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/30/2014       | 12:23    |
| 5 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/29/2014       | 9:11     |
| 6 | Harry Bridges Plaza   | (Ferry Building) |               |                | 50         | 8/28/2014       | 14:02    |
|   |                       |                  | End.Station   | End.Terminal   | Bike..     | Subscriber.Type | Zip.Code |
| 1 | Embarcadero at Folsom |                  | 51            | 306            | Customer   | 94952           |          |
| 2 | Embarcadero at Folsom |                  | 51            | 659            | Customer   | 94952           |          |
| 3 | Embarcadero at Folsom |                  | 51            | 556            | Customer   | 11238           |          |
| 4 | Embarcadero at Folsom |                  | 51            | 621            | Customer   | 11238           |          |
| 5 | Embarcadero at Folsom |                  | 51            | 400            | Subscriber | 94070           |          |
| 6 | Embarcadero at Folsom |                  | 51            | 589            | Subscriber | 94107           |          |

The `arrange` function is useful for sorting by any number of columns as needed. Here we sort by the start and end stations.

```
trips_sorted = arrange(trips, Start.Station, End.Station)
head(trips_sorted)
```

|   |           |       |               |             |               |        |                 |
|---|-----------|-------|---------------|-------------|---------------|--------|-----------------|
| 1 | 426408    | 120   | 8/27/2014     | 7:40        | 2nd at Folsom |        | 62              |
| 2 | 411496    | 21183 | 8/16/2014     | 13:36       | 2nd at Folsom |        | 62              |
| 3 | 396676    | 3707  | 8/6/2014      | 11:38       | 2nd at Folsom |        | 62              |
| 4 | 385761    | 123   | 7/29/2014     | 19:52       | 2nd at Folsom |        | 62              |
| 5 | 364633    | 6395  | 7/15/2014     | 13:39       | 2nd at Folsom |        | 62              |
| 6 | 362776    | 9433  | 7/14/2014     | 13:36       | 2nd at Folsom |        | 62              |
|   |           |       | End.Date      | End.Station | End.Terminal  | Bike.. | Subscriber.Type |
| 1 | 8/27/2014 | 7:42  | 2nd at Folsom |             | 62            | 527    | Subscriber      |
| 2 | 8/16/2014 | 19:29 | 2nd at Folsom |             | 62            | 508    | Customer        |
| 3 | 8/6/2014  | 12:40 | 2nd at Folsom |             | 62            | 109    | Customer        |
| 4 | 7/29/2014 | 19:55 | 2nd at Folsom |             | 62            | 421    | Subscriber      |
| 5 | 7/15/2014 | 15:26 | 2nd at Folsom |             | 62            | 448    | Customer        |
| 6 | 7/14/2014 | 16:13 | 2nd at Folsom |             | 62            | 454    | Customer        |

| Zip | Code  |
|-----|-------|
| 1   | 94107 |
| 2   | 94105 |
| 3   | 31200 |
| 4   | 94107 |
| 5   | 2184  |
| 6   | 2184  |

The sort can also be done in reverse order as follows.

```
trips_sorted = arrange(trips, desc(Start.Station), End.Station)
head(trips_sorted)
```

| Trip.ID  | Duration                                      | Start.Date          |              |        |                 |
|----------|-----------------------------------------------|---------------------|--------------|--------|-----------------|
| 1        | 416755                                        | 285 8/20/2014 11:37 |              |        |                 |
| 2        | 411270                                        | 257 8/16/2014 7:03  |              |        |                 |
| 3        | 410269                                        | 286 8/15/2014 10:34 |              |        |                 |
| 4        | 405273                                        | 382 8/12/2014 14:27 |              |        |                 |
| 5        | 398372                                        | 401 8/7/2014 10:10  |              |        |                 |
| 6        | 393012                                        | 317 8/4/2014 10:59  |              |        |                 |
|          | Start.Station                                 | Start.Terminal      |              |        |                 |
| 1        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
| 2        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
| 3        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
| 4        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
| 5        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
| 6        | Yerba Buena Center of the Arts (3rd @ Howard) | 68                  |              |        |                 |
|          | End.Date                                      | End.Station         | End.Terminal | Bike.. | Subscriber.Type |
| 1        | 8/20/2014 11:42                               | 2nd at Folsom       |              | 62     | 383             |
| 2        | 8/16/2014 7:07                                | 2nd at Folsom       |              | 62     | 614             |
| 3        | 8/15/2014 10:38                               | 2nd at Folsom       |              | 62     | 545             |
| 4        | 8/12/2014 14:34                               | 2nd at Folsom       |              | 62     | 344             |
| 5        | 8/7/2014 10:16                                | 2nd at Folsom       |              | 62     | 597             |
| 6        | 8/4/2014 11:04                                | 2nd at Folsom       |              | 62     | 367             |
| Zip.Code |                                               |                     |              |        |                 |
| 1        | 95060                                         |                     |              |        |                 |
| 2        | 94107                                         |                     |              |        |                 |
| 3        | 94127                                         |                     |              |        |                 |
| 4        | 94110                                         |                     |              |        |                 |
| 5        | 94127                                         |                     |              |        |                 |

6 94127

Data.table also offers a fantastic way to do descriptive statistics! First, group the data by start point, and then produce statistics by this group, choosing to count the number of trips starting from each station and the average duration of each trip.

```
byStartStation = group_by(trips, Start.Station)
res = summarise(byStartStation, count=n(), time=mean(Duration)/60)
print(res)
```

Source: local data frame [70 x 3]

|    | Start.Station                   | count | time     |
|----|---------------------------------|-------|----------|
|    | (fctr)                          | (int) | (dbl)    |
| 1  | 2nd at Folsom                   | 4165  | 9.32088  |
| 2  | 2nd at South Park               | 4569  | 11.60195 |
| 3  | 2nd at Townsend                 | 6824  | 15.14786 |
| 4  | 5th at Howard                   | 3183  | 14.23254 |
| 5  | Adobe on Almaden                | 360   | 10.06120 |
| 6  | Arena Green / SAP Center        | 510   | 43.82833 |
| 7  | Beale at Market                 | 4293  | 15.74702 |
| 8  | Broadway at Main                | 22    | 54.82121 |
| 9  | Broadway St at Battery St       | 2433  | 15.31862 |
| 10 | California Ave Caltrain Station | 329   | 51.30709 |
| .. | ...                             | ...   | ...      |



# 5

## *Being Mean with Variance: Markowitz Optimization*

In this chapter, we will explore the mathematics of the famous portfolio optimization result, known as the Markowitz mean-variance problem.

The solution to this problem is still being used widely in practice. We are interested in portfolios of  $n$  assets, which have a mean return which we denote as  $E(r_p)$ , and a variance, denoted  $\text{Var}(r_p)$ .

Let  $\underline{w} \in R^n$  be the portfolio weights. What this means is that we allocate each \$1 into various assets, such that the total of the weights sums up to 1. Note that we do not preclude short-selling, so that it is possible for weights to be negative as well.

### 5.1 Quadratic (Markowitz) Problem

The optimization problem is defined as follows. We wish to find the portfolio that delivers the minimum variance (risk) while achieving a pre-specified level of expected (mean) return.

$$\min_{\underline{w}} \quad \frac{1}{2} \underline{w}' \Sigma \underline{w}$$

subject to

$$\begin{aligned} \underline{w}' \underline{\mu} &= E(r_p) \\ \underline{w}' \underline{1} &= 1 \end{aligned}$$

Note that we have a  $\frac{1}{2}$  in front of the variance term above, which is for mathematical neatness as will become clear shortly. The minimized solution is not affected by scaling the objective function by a constant.

The first constraint forces the expected return of the portfolio to a specified mean return, denoted  $E(r_p)$ , and the second constraint requires that the portfolio weights add up to 1, also known as the “fully invested” constraint. It is convenient that the constraints are equality constraints.

This is a Lagrangian problem, and requires that we embed the constraints into the objective function using Lagrangian multipliers  $\{\lambda_1, \lambda_2\}$ . This results in the following minimization problem:

$$\min_{\underline{w}, \lambda_1, \lambda_2} L = \frac{1}{2} \underline{w}' \Sigma \underline{w} + \lambda_1 [E(r_p) - \underline{w}' \underline{\mu}] + \lambda_2 [1 - \underline{w}' \underline{1}]$$

To minimize this function, we take derivatives with respect to  $\underline{w}$ ,  $\lambda_1$ , and  $\lambda_2$ , to arrive at the first order conditions:

$$\frac{\partial L}{\partial \underline{w}} = \Sigma \underline{w} - \lambda_1 \underline{\mu} - \lambda_2 \underline{1} = \underline{0} \quad (*)$$

$$\frac{\partial L}{\partial \lambda_1} = E(r_p) - \underline{w}' \underline{\mu} = 0$$

$$\frac{\partial L}{\partial \lambda_2} = 1 - \underline{w}' \underline{1} = 0$$

The first equation above, denoted (\*), is a system of  $n$  equations, because the derivative is taken with respect to every element of the vector  $\underline{w}$ .

Hence, we have a total of  $(n + 2)$  first-order conditions. From (\*)

$$\begin{aligned} \underline{w} &= \Sigma^{-1}(\lambda_1 \underline{\mu} + \lambda_2 \underline{1}) \\ &= \lambda_1 \Sigma^{-1} \underline{\mu} + \lambda_2 \Sigma^{-1} \underline{1} \quad (** ) \end{aligned}$$

Premultiply (\*\*) by  $\underline{\mu}'$ :

$$\underline{\mu}' \underline{w} = \lambda_1 \underbrace{\underline{\mu}' \Sigma^{-1} \underline{\mu}}_B + \lambda_2 \underbrace{\underline{\mu}' \Sigma^{-1} \underline{1}}_A = E(r_p)$$

Also premultiply (\*\*) by  $\underline{1}'$ :

$$\underline{1}' \underline{w} = \lambda_1 \underbrace{\underline{1}' \Sigma^{-1} \underline{\mu}}_A + \lambda_2 \underbrace{\underline{1}' \Sigma^{-1} \underline{1}}_C = 1$$

Solve for  $\lambda_1, \lambda_2$

$$\lambda_1 = \frac{CE(r_p) - A}{D}$$

$$\lambda_2 = \frac{B - AE(r_p)}{D}$$

$$\text{where } D = BC - A^2$$

Note the following:

- Since  $\Sigma$  is positive definite,  $\Sigma^{-1}$  is also positive definite:  $B > 0, C > 0$ .

- Given solutions for  $\lambda_1, \lambda_2$ , we solve for  $\underline{w}$ .

$$\underline{w} = \underbrace{\frac{1}{D} [B\Sigma^{-1}\underline{1} - A\Sigma^{-1}\underline{\mu}]}_{\underline{g}} + \underbrace{\frac{1}{D} [C\Sigma^{-1}\underline{\mu} - A\Sigma^{-1}\underline{1}]}_{\underline{h}} \cdot E(r_p)$$

This is the expression for the optimal portfolio weights that minimize the variance for given expected return  $E(r_p)$ . We see that the vectors  $\underline{g}$ ,  $\underline{h}$  are fixed once we are given the inputs to the problem, i.e.,  $\underline{\mu}$  and  $\Sigma$ .

- We can vary  $E(r_p)$  to get a set of frontier (efficient or optimal) portfolios  $\underline{w}$ .

$$\underline{w} = \underline{g} + \underline{h} E(r_p)$$

$$\begin{aligned} \text{if } E(r_p) &= 0, \underline{w} = \underline{g} \\ \text{if } E(r_p) &= 1, \underline{w} = \underline{g} + \underline{h} \end{aligned}$$

Note that

$$\underline{w} = \underline{g} + \underline{h} E(r_p) = [1 - E(r_p)] \underline{g} + E(r_p) [\underline{g} + \underline{h}]$$

Hence these 2 portfolios  $\underline{g}, \underline{g} + \underline{h}$  "generate" the entire frontier.

### 5.1.1 Solution in R

We create a function to return the optimal portfolio weights.

```
markowitz = function(mu, cv, Er) {
 n = length(mu)
 wuns = matrix(1, n, 1)
 A = t(wuns) %*% solve(cv) %*% mu
 B = t(mu) %*% solve(cv) %*% mu
 C = t(wuns) %*% solve(cv) %*% wuns
 D = B*C - A^2
 lam = (C*Er - A)/D
 gam = (B - A*Er)/D
 wts = lam[1] * (solve(cv) %*% mu) + gam[1] * (solve(cv) %*% wuns)
 g = (B[1] * (solve(cv) %*% wuns) - A[1] * (solve(cv) %*% mu)) / D[1]
 h = (C[1] * (solve(cv) %*% mu) - A[1] * (solve(cv) %*% wuns)) / D[1]
 wts = g + h*Er
}
```

We can enter an example of a mean return vector and the covariance matrix of returns, and then call the function for a given expected return.

```
#PARAMETERS
mu = matrix(c(0.02,0.10,0.20),3,1)
n = length(mu)
cv = matrix(c(0.0001,0,0,0,0.04,0.02,0,0.02,0.16),n,n)
Er = 0.18

#SOLVE PORTFOLIO PROBLEM
wts = markowitz(mu,cv,Er)
print(wts)
```

The output is the vector of optimal portfolio weights:

```
> source("markowitz.R")
[,1]
[1,] -0.3575931
[2,] 0.8436676
[3,] 0.5139255
```

If we change the expected return to 0.10, then we get a different set of portfolio weights.

```
> Er = 0.10
> wts = markowitz(mu,cv,Er)
> print(wts)
[,1]
[1,] 0.3209169
[2,] 0.4223496
[3,] 0.2567335
```

Note that in the first example, to get a high expected return of 0.18, we needed to take some leverage, by shorting the low risk asset and going long the medium and high risk assets. When we dropped the expected return to 0.10, all weights are positive, i.e., we have a long-only portfolio.

## 5.2 Solving the problem with the quadprog package

The quadprog package is an optimizer that takes a quadratic objective function with linear constraints. Hence, it is exactly what is needed for the mean-variance portfolio problem we just considered. The advantage of this package is that we can also apply additional inequality constraints. For example, we may not wish to permit short-sales of any

asset, and thereby we might bound all the weights to lie between zero and one.

The specification in the quadprog package of the problem set up is shown in the manual:

### Description

This routine implements the dual method of Goldfarb and Idnani (1982, 1983) for solving quadratic programming problems of the form  $\min(-d^T b + 1/2 b^T D b)$  with the constraints  $A^T b \geq b_0$ .

(note: b here is the **weights vector** in our problem)

### Usage

```
solve.QP(Dmat, dvec, Amat, bvec, meq=0, factorized=FALSE)
```

### Arguments

|            |                                                                                                                                      |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Dmat       | <b>matrix</b> appearing in the quadratic <b>function</b> to be minimized.                                                            |
| dvec       | <b>vector</b> appearing in the quadratic <b>function</b> to be minimized.                                                            |
| Amat       | <b>matrix</b> defining the constraints under which we want to minimize the quadratic <b>function</b> .                               |
| bvec       | <b>vector</b> holding the values of $b_0$ (defaults to zero).                                                                        |
| meq        | the first meq constraints are treated as equality constraints, all further as inequality constraints (defaults to 0).                |
| factorized | <b>logical</b> flag: if TRUE, then we are passing $R^{-1}$ (where $D = R^T R$ ) instead of the <b>matrix D</b> in the argument Dmat. |

In our problem set up, with three securities, and no short sales, we will have the following Amat and bvec:

$$A = \begin{bmatrix} \mu_1 & 1 & 1 & 0 & 0 \\ \mu_2 & 1 & 0 & 1 & 0 \\ \mu_3 & 1 & 0 & 0 & 1 \end{bmatrix}; \quad b_0 = \begin{bmatrix} E(r_p) \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The constraints will be modulated by meq = 2, which states that the first two constraints will be equality constraints, and the last three will be greater than equal to constraints. The constraints will be of the form

$A'w \geq b_0$ , i.e.,

$$\begin{aligned} w_1\mu_1 + w_2\mu_2 + w_3\mu_3 &= E(r_p) \\ w_11 + w_21 + w_31 &= 1 \\ w_1 &\geq 0 \\ w_2 &\geq 0 \\ w_3 &\geq 0 \end{aligned}$$

The code for using the package is as follows.

```
library(quadprog)
nss = 1 # Equals 1 if no short sales allowed
Bmat = matrix(0,n,n) #No Short sales matrix
diag(Bmat) = 1
Amat = matrix(c(mu,1,1,1),n,2)
if (nss==1) { Amat = matrix(c(Amat,Bmat),n,2+n) }
dvec = matrix(0,n,1)
bvec = matrix(c(Er,1),2,1)
if (nss==1) { bvec = t(c(bvec,matrix(0,3,1))) }
sol = solve.QP(cv,dvec,Amat,bvec,meq=2)
print(sol$solution)
```

If we run this code we get the following result for expected return = 0.18, with short-selling allowed:

```
[1] -0.3575931 0.8436676 0.5139255
```

This is exactly what is obtained from the Markowitz solution. Hence, the model checks out. What if we restricted short-selling? Then we would get the following solution.

```
[1] 0.0 0.2 0.8
```

### 5.3 Tracing out the Efficient Frontier

Since we can use the Markowitz model to solve for the optimal portfolio weights when the expected return is fixed, we can keep solving for different values of  $E(r_p)$ . This will trace out the efficient frontier. The program to do this and plot the frontier is as follows.

```
#TRACING OUT THE EFFICIENT FRONTIER
Er_vec = matrix(seq(0.01,0.15,0.01),15,1)
```

```

Sig_vec = matrix(0,15,1)
j = 0
for (Er in Er_vec) {
 j = j+1
 wts = markowitz(mu,cv,Er)
 Sig_vec[j] = sqrt(t(wts) %*% cv %*% wts)
}
plot(Sig_vec,Er_vec,type='l')

```

See the frontier in Figure 5.1.

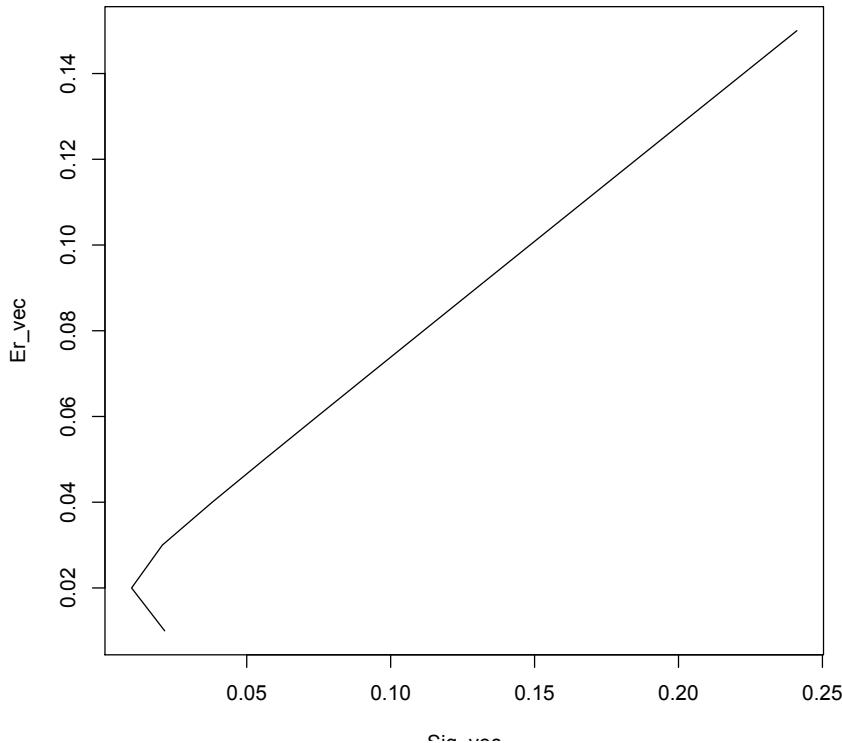


Figure 5.1: The Efficient Frontier

#### 5.4 Covariances of frontier portfolios: $r_p, r_q$

$$\text{Cov}(r_p, r_q) = \underline{w}'_p \Sigma \underline{w}_q = [\underline{g} + \underline{h}E(r_p)]' \Sigma [\underline{g} + \underline{h}E(r_q)]$$

Now,

$$\begin{aligned} \underline{g} + \underline{h}E(r_p) &= \frac{1}{D} [B\Sigma^{-1}\underline{1} - A\Sigma^{-1}\underline{\mu}] + \frac{1}{D} [C\Sigma^{-1}\underline{\mu} - A\Sigma^{-1}\underline{1}] \underbrace{[\lambda_1 B + \lambda_2 A]}_{\frac{CE(r_p) - A}{D/B}} \\ &\quad + \frac{B - AE(r_p)}{D/B} \end{aligned}$$

After much simplification:

$$\begin{aligned} \text{Cov}(r_p, r_q) &= \underline{w}'_p \Sigma \underline{w}'_q \\ &= \frac{C}{D} [E(r_p) - A/C][E(r_q) - A/C] + \frac{1}{C} \\ \sigma_p^2 = \text{Cov}(r_p, r_p) &= \frac{C}{D} [E(r_p) - A/C]^2 + \frac{1}{C} \end{aligned}$$

Therefore,

$$\frac{\sigma_p^2}{1/C} - \frac{[E(r_p) - A/C]^2}{D/C^2} = 1$$

which is the equation of a hyperbola in  $\sigma, E(r)$  space with center  $(0, A/C)$ ,

or

$$\sigma_p^2 = \frac{1}{D} [CE(r_p)^2 - 2AE(r_p) + B]$$

, which is a parabola in  $E(r), \sigma$  space.

## 5.5 Combinations

It is easy to see that linear combinations of portfolios on the frontier will also lie on the frontier.

$$\begin{aligned} \sum_{i=1}^m \alpha_i \underline{w}_i &= \sum_{i=1}^m \alpha_i [\underline{g} + \underline{h} E(r_i)] \\ &= \underline{g} + \underline{h} \sum_{i=1}^m \alpha_i E(r_i) \quad \sum_{i=1}^m \alpha_i = 1 \end{aligned}$$

### Exercise

Carry out the following analyses:

1. Use your R program to do the following. Set  $E(r_p) = 0.10$  (i.e. return of 10%), and solve for the optimal portfolio weights for your 3 securities. Call this vector of weights  $w_1$ . Next, set  $E(r_p) = 0.20$  and again solve for the portfolios weights  $w_2$ .
2. Take a 50/50 combination of these two portfolios. What are the weights? What is the expected return?
3. For the expected return in the previous part, resolve the mean-variance problem to get the new weights?
4. Compare these weights in part 3 to the ones in part 2 above. Explain your result.

## 5.6 Zero Covariance Portfolio

This is a special portfolio of interest, and we will soon see why. Find

$$E(r_q), \text{ s.t. } \text{Cov}(r_p, r_q) = 0$$

Suppose it exists, then the solution is:

$$E(r_q) = \frac{A}{C} - \frac{D/C^2}{E(r_p) - A/C} \equiv E(r_{ZC(p)})$$

Since  $ZC(p)$  exists for all  $p$ , all frontier portfolios can be formed from  $p$  and  $ZC(p)$ .

$$\begin{aligned} \text{Cov}(r_p, r_q) &= \underline{w}'_p \underline{\Sigma} \underline{w}_q \\ &= \lambda_1 \underline{\mu}' \underline{\Sigma}^{-1} \underline{\Sigma} \underline{w}_q + \lambda_2 \underline{1}' \underline{\Sigma}^{-1} \underline{\Sigma} \underline{w}_q \\ &= \lambda_1 \underline{\mu}' \underline{w}_q + \lambda_2 \underline{1}' \underline{w}_q \\ &= \lambda_1 E(r_q) + \lambda_2 \end{aligned}$$

Substitute in for  $\lambda_1, \lambda_2$  and rearrange to get

$$\begin{aligned} E(r_q) &= (1 - \beta_{qp}) E[r_{ZC(p)}] + \beta_{qp} E(r_p) \\ \beta_{qp} &= \frac{\text{Cov}(r_q, r_p)}{\sigma_p^2} \end{aligned}$$

Therefore, the return on a portfolio can be written in terms of a basic portfolio  $p$  and its zero covariance portfolio  $ZC(p)$ . This suggests a regression relationship, i.e.

$$r_q = \beta_0 + \beta_1 r_{ZC(p)} + \beta_2 r_p + \xi$$

which is nothing but a factor model, i.e. with orthogonal factors.

## 5.7 Portfolio Problems with Riskless Assets

We now enhance the portfolio problem to deal with risk less assets. The difference is that the fully-invested constraint is expanded to include the risk free asset. We require just a single equality constraint. The problem may be specified as follows.

$$\begin{aligned} \min_{\underline{w}} \quad & \frac{1}{2} \underline{w}' \underline{\Sigma} \underline{w} \\ \text{s.t.} \quad & \underline{w}' \underline{\mu} + (1 - \underline{w}' \underline{1}) r_f = E(r_p) \end{aligned}$$

$$\min_{\underline{w}} \quad L = \frac{1}{2} \underline{w}' \Sigma \underline{w} + \lambda [E(r_p) - \underline{w}' \underline{\mu} - (1 - \underline{w}' \underline{1}) r_f]$$

The first-order conditions for the problem are as follows.

$$\begin{aligned}\frac{\partial L}{\partial \underline{w}} &= \Sigma \underline{w} - \lambda \underline{\mu} + \lambda \underline{1} r_f = \underline{0} \\ \frac{\partial L}{\partial \lambda} &= E(r_p) - \underline{w}' \underline{\mu} - (1 - \underline{w}' \underline{1}) r_f = 0\end{aligned}$$

Re-aranging, and solving for  $\underline{w}$  and  $\lambda$ , we get the following manipulations, eventually leading to the desired solution.

$$\begin{aligned}\Sigma \underline{w} &= \lambda (\underline{\mu} - \underline{1} r_f) \\ E(r_p) - r_f &= \underline{w}' (\underline{\mu} - \underline{1} r_f)\end{aligned}$$

Take the first equation and proceed as follows:

$$\begin{aligned}\underline{w} &= \lambda \Sigma^{-1} (\underline{\mu} - \underline{1} r_f) \\ E(r_p) - r_f \equiv (\underline{\mu} - \underline{1} r_f)' \underline{w} &= \lambda (\underline{\mu} - \underline{1} r_f)' \Sigma^{-1} (\underline{\mu} - \underline{1} r_f)\end{aligned}$$

The first and third terms in the equation above then give that

$$\lambda = \frac{E(r_p) - r_f}{(\underline{\mu} - \underline{1} r_f)' \Sigma^{-1} (\underline{\mu} - \underline{1} r_f)}$$

Substituting this back into the first foc results in the final solution.

$$\begin{aligned}\underline{w} &= \Sigma^{-1} (\underline{\mu} - \underline{1} r_f) \frac{E(r_p) - r_f}{H} \\ \text{where } H &= (\underline{\mu} - r_f \underline{1})' \Sigma^{-1} (\underline{\mu} - r_f \underline{1})\end{aligned}$$

### *Exercise*

How will you use the R program to find the minimum variance portfolio (MVP)? What are the portfolio weights? What is the expected return?

### *Exercise*

Develop program code for the mean-variance problem with the risk-free asset.

### *Exercise*

Develop program code for the mean-variance problem with no short sales, and plot the efficient frontier on top of the one with short-selling allowed.

## 5.8 Risk Budgeting

Markowitz optimization has morphed into many different “views” of the same problem. One of the recent approaches to portfolio construction is to create portfolios where the risk contributions of all assets are equal. This is known as the “risk parity” approach. We may also construct a portfolio where all assets contribute the same proportion of the total return of the portfolio, and this is known as the “performance parity” approach.

If the portfolio is denoted by its weights  $\mathbf{w}$  then the risk of the portfolio is a function of the weights and is denoted  $R(\mathbf{w})$ . As we have seen the standard deviation of the portfolio return is written as

$$R(\mathbf{w}) = \sigma(\mathbf{w}) = \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}} \quad (5.1)$$

This risk function is linear homogenous, i.e., if we double the size of the portfolio then the risk measure also doubles. This is also known as the “homogeneity” property of risk measures and is one of the four desirable properties of a “coherent” risk measure defined by [Artzner, Delbaen, Eber, and Heath \(1999\)](#):

1. Homogeneity:  $R(m \cdot \mathbf{w}) = m \cdot R(\mathbf{w})$ .
2. Subadditivity (diversification):  $R(\mathbf{w}_1 + \mathbf{w}_2) \leq R(\mathbf{w}_1) + R(\mathbf{w}_2)$ .
3. Monotonicity: if portfolio  $\mathbf{w}_1$  dominates portfolio  $\mathbf{w}_2$ , and their mean returns are the same, then  $R(\mathbf{w}_1) \leq R(\mathbf{w}_2)$ .
4. Translation invariance: if we add cash proportion  $c$  and rebalance the portfolio, then  $R(\mathbf{w} + c) = R(\mathbf{w}) - c$ .
5. Convexity: this property combines homogeneity and subadditivity,  $R(m \cdot \mathbf{w}_1 + (1 - m) \cdot \mathbf{w}_2) \leq m \cdot R(\mathbf{w}_1) + (1 - m) \cdot R(\mathbf{w}_2)$ .

If the risk measure satisfies the homogeneity property, then Euler’s theorem may be applied to decompose risk into the amount provided by each asset.

$$R(\mathbf{w}) = \sum_{j=1}^n w_j \frac{\partial R(\mathbf{w})}{\partial w_j} \quad (5.2)$$

The component  $w_j \frac{\partial R(\mathbf{w})}{\partial w_j}$  is known as the risk share of asset  $j$ , and when divided by  $R(\mathbf{w})$ , it is the risk proportion of asset  $j$ .

Suppose we define the risk measure to be the standard deviation of returns of the portfolio, then the risk decomposition requires the derivative of the risk measure with respect to all the weights, i.e.,

$$\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}}{\partial \mathbf{w}} = \frac{1}{2} [\mathbf{w}^\top \Sigma \mathbf{w}]^{-1/2} \cdot 2\Sigma \mathbf{w} = \frac{\Sigma \mathbf{w}}{\sigma(\mathbf{w})} \quad (5.3)$$

which is a  $n$ -dimensional vector. If we multiply the  $j$ -th element of this vector by  $w_j$ , we get the risk contribution for asset  $j$ .

We may check that the risk contributions sum up to the total risk:

$$\begin{aligned} \sum_{j=1}^n w_j \frac{\partial R(\mathbf{w})}{\partial w_j} &= [w_1 \quad w_2 \quad \dots \quad w_n] \cdot [\Sigma \mathbf{w} / \sigma(\mathbf{w})] \\ &= \mathbf{w}^\top \cdot [\Sigma \mathbf{w} / \sigma(\mathbf{w})] \\ &= \frac{\sigma(\mathbf{w})^2}{\sigma(\mathbf{w})} \\ &= \sigma(\mathbf{w}) \\ &= R(\mathbf{w}) \end{aligned}$$

Let's look at an example to clarify the computations. First, read in the covariance matrix and mean return vector.

```
mu = matrix(c(0.05, 0.10, 0.20), 3, 1)
n = length(mu)
cv = matrix(c(0.03, 0.01, 0.01, 0.01, 0.04, 0.02, 0.01, 0.02, 0.16), n, n)
```

We begin by choosing the portfolio weights for an expected return of 0.12. Then we create the function to return the risk contributions of each asset in the portfolio.

```
#RISK CONTRIBUTIONS
riskContribution = function(cv, wts) {
 sig = sqrt(t(wts) %*% cv %*% wts)
 rc = as.matrix(cv %*% wts) / sig[1] * wts
}
#Example
Er = 0.12
wts = markowitz(mu, cv, Er)
print(wts)
RC = riskContribution(cv, wts)
print(RC)
#Check
```

```
sig = sqrt(t(wts) %*% cv %*% wts)
print(c(sig, sum(RC)))
```

The output of all this code is as follows:

```
> print(wts)
[,1]
[1,] 0.1818182
[2,] 0.5272727
[3,] 0.2909091

> print(RC)
[,1]
[1,] 0.01329760
[2,] 0.08123947
[3,] 0.09191302

> #Check
> sig = sqrt(t(wts) %*% cv %*% wts)
> print(c(sig, sum(RC)))
[1] 0.1864501 0.1864501
```

We see that the total risk contributions of all three assets does indeed sum up to the standard deviation of the portfolio, i.e., 0.1864501.

We are interested in solving the reverse problem. Given a target set of risk contributions, what weights of the portfolio will deliver the required contribution. For example, what if we wanted the portfolio total standard deviation to be 0.15, with the shares from each asset in the amounts {0.05, 0.05, 0.05}, respectively?

We note that it is not possible to solve for exactly the desired risk contributions. This is because it would involve one constraint for each risk contribution, plus one additional constraint that the sum of the portfolio weights sum up to 1. That would leave us with an infeasible problem where there are four constraints and only three free parameters. Therefore, we minimise the sum of squared differences between the risk contributions and targets, while ensuring that the sum of portfolio weights equals unity. We can implement the following code to achieve this result.

```
#SOLVE FOR CHOSEN RISK CONTRIBUTIONS
solveRC = function(wts, target, cv) {
 wts[length(wts)+1] = 1-sum(wts)
```

```
wts = as.matrix(wts)
rc = riskContribution(cv,wts)
#Minimize the max slippage from risk parity
diff2 = 10000000*(rc-target)
}
target = matrix(c(0.05,0.05,0.05))
w_guess = c(0.1,0.4)

library(minpack.lm)
sol = nls.lm(w_guess,fn=solveRC, cv=cv, target=target)
wts = sol$par
wts[length(wts)+1] = 1-sum(wts)
wts = as.matrix(wts)
print(wts)
print(sum(wts))
rc = riskContribution(cv,wts)
print(c(rc,sum(rc)))
```

The results from running this code are as follows:

```
> print(wts)
 [,1]
[1,] 0.4435305
[2,] 0.3639453
[3,] 0.1925243
> print(sum(wts))
[1] 1
> rc = riskContribution(cv,wts)
> print(c(rc,sum(rc)))
[1] 0.05307351 0.05271923 0.05190721 0.15769995
>
```

We see that the results are close to targets, but slightly above. As expected, since the risk parity is equal across assets, the less risky ones have a greater share in the portfolio allocation.

# 6

## *Learning from Experience: Bayes Theorem*

### *6.1 Introduction*

For a fairly good introduction to Bayes Rule, see Wikipedia

[http://en.wikipedia.org/wiki/Bayes\\_theorem](http://en.wikipedia.org/wiki/Bayes_theorem)

The various R packages for Bayesian inference are at:

<http://cran.r-project.org/web/views/Bayesian.html>

Also see the great video of Professor Persi Diaconis's talk on Bayes on Yahoo video where he talks about coincidences. In business, we often want to ask, is a given phenomena real or just a coincidence? Bayes theorem really helps with that. For example, we may ask – is Warren Buffet's investment success a coincidence? How would you answer this question? Would it depend on your prior probability of Buffet being able to beat the market? How would this answer change as additional information about his performance was being released over time?

Bayes rule follows easily from a decomposition of joint probability, i.e.,

$$Pr[A \cap B] = Pr(A|B) Pr(B) = Pr(B|A) Pr(A)$$

Then the last two terms may be arranged to give

$$Pr(A|B) = \frac{Pr(B|A) Pr(A)}{Pr(B)}$$

or

$$Pr(B|A) = \frac{Pr(A|B) Pr(B)}{Pr(A)}$$

### *Example*

The AIDS test. This is an interesting problem, because it shows that if you are diagnosed with AIDS, there is a good chance the diagnosis is

wrong, but if you are diagnosed as not having AIDS then there is a good chance it is right - hopefully this is comforting news.

Define,  $\{Pos, Neg\}$  as a positive or negative diagnosis of having AIDS. Also define  $\{Dis, NoDis\}$  as the event of having the disease versus not having it. There are 1.5 million AIDS cases in the U.S. and about 300 million people which means the probability of AIDS in the population is 0.005 (half a percent). Hence, a random test will uncover someone with AIDS with a half a percent probability. The confirmation accuracy of the AIDS test is 99%, such that we have

$$Pr(Pos|Dis) = 0.99$$

Hence the test is reasonably good. The accuracy of the test for people who do not have AIDS is

$$Pr(Neg|NoDis) = 0.95$$

What we really want is the probability of having the disease when the test comes up positive, i.e. we need to compute  $Pr(Dis|Pos)$ . Using Bayes Rule we calculate:

$$\begin{aligned} Pr(Dis|Pos) &= \frac{Pr(Pos|Dis)Pr(Dis)}{Pr(Pos)} \\ &= \frac{Pr(Pos|Dis)Pr(Dis)}{Pr(Pos|Dis)Pr(Dis) + Pr(Pos|NoDis)Pr(NoDis)} \\ &= \frac{0.99 \times 0.005}{(0.99)(0.005) + (0.05)(0.995)} \\ &= 0.0904936 \end{aligned}$$

Hence, the chance of having AIDS when the test is positive is only 9%. We might also care about the chance of not having AIDS when the test is positive

$$Pr(NoDis|Pos) = 1 - Pr(Dis|Pos) = 1 - 0.09 = 0.91$$

Finally, what is the chance that we have AIDS even when the test is negative - this would also be a matter of concern to many of us, who might not relish the chance to be on some heavy drugs for the rest of our

lives.

$$\begin{aligned}
 Pr(Dis|Neg) &= \frac{Pr(Neg|Dis)Pr(Dis)}{Pr(Neg)} \\
 &= \frac{Pr(Neg|Dis)Pr(Dis)}{Pr(Neg|Dis)Pr(Dis) + Pr(Neg|NoDis)Pr(NoDis)} \\
 &= \frac{0.01 \times 0.005}{(0.01)(0.005) + (0.95)(0.995)} \\
 &= 0.000053
 \end{aligned}$$

Hence, this is a worry we should not have. If the test is negative, there is a minuscule chance that we are infected with AIDS.

## 6.2 Bayes and Joint Probability Distributions

The preceding analysis is a good lead in to (a) the connection with joint probability distributions, and (b) using R to demonstrate a computational way of thinking about Bayes theorem.

Let's begin by assuming that we have 300,000 people in the population, to scale down the numbers from the millions for convenience. Of these 1,500 have AIDS. So let's create the population and then sample from it. See the use of the `sample` function in R.

```
> people = seq(1,300000)
> people_aids = sample(people, 1500)
> people_noaids = setdiff(people, people_aids)
```

Note, how we also used the `setdiff` function to get the complement set of the people who do not have AIDS. Now, of the people who have AIDS, we know that 99% of them test positive so let's subset that list, and also take its complement. These are joint events, and their numbers proscribe the joint distribution.

```
> people_aids_pos = sample(people_aids, 1500 * 0.99)
> people_aids_neg = setdiff(people_aids, people_aids_pos)
> length(people_aids_pos)
[1] 1485
> length(people_aids_neg)
[1] 15
```

We can also subset the group that does not have AIDS, as we know that the test is negative for them 95% of the time.

```
> people_noaids_neg = sample(people_noaids, 298500 * 0.95)
> people_noaids_pos = setdiff(people_noaids, people_noaids_neg)
> length(people_noaids_neg)
[1] 283575
> length(people_noaids_pos)
[1] 14925
```

We can now compute the probability that someone actually has AIDS when the test comes out positive.

```
> pr_aids_given_pos = (length(people_aids_pos)) /
 (length(people_aids_pos) + length(people_noaids_pos))
> pr_aids_given_pos
[1] 0.0904936
```

This confirms the formal Bayes computation that we had undertaken earlier. And of course, as we had examined earlier, what's the chance that you have AIDS when the test is negative, i.e., a false negative?

```
> pr_aids_given_neg = (length(people_aids_neg)) /
 (length(people_aids_neg) + length(people_noaids_neg))
> pr_aids_given_neg
[1] 5.289326e-05
```

Phew!

Note here that we first computed the joint sets covering joint outcomes, and then used these to compute conditional (Bayes) probabilities. The approach used R to apply a set-theoretic, computational approach to calculating conditional probabilities.

### 6.3 Correlated default (conditional default)

Bayes theorem is very useful when we want to extract conditional default information. Bond fund managers are not as interested in the correlation of default of the bonds in their portfolio as much as the conditional default of bonds. What this means is that they care about the *conditional* probability of bond A defaulting if bond B has defaulted already.

Modern finance provides many tools to obtain the default probabilities of firms. Suppose we know that firm 1 has default probability  $p_1 = 1\%$  and firm 2 has default probability  $p_2 = 3\%$ . If the correlation

of default of the two firms is 40% over one year, then if either bond defaults, what is the probability of default of the other, conditional on the first default?

We can see that even with this limited information, Bayes theorem allows us to derive the conditional probabilities of interest. First define  $d_i, i = 1, 2$  as default indicators for firms 1 and 2.  $d_i = 1$  if the firm defaults, and zero otherwise. We note that:

$$E(d_1) = 1.p_1 + 0.(1 - p_1) = p_1 = 0.01.$$

Likewise

$$E(d_2) = 1.p_2 + 0.(1 - p_2) = p_2 = 0.03.$$

The Bernoulli distribution lets us derive the standard deviation of  $d_1$  and  $d_2$ .

$$\begin{aligned}\sigma_1 &= \sqrt{p_1(1 - p_1)} = \sqrt{(0.01)(0.99)} = 0.099499 \\ \sigma_2 &= \sqrt{p_2(1 - p_2)} = \sqrt{(0.03)(0.97)} = 0.17059\end{aligned}$$

Now, we note that

$$\begin{aligned}Cov(d_1, d_2) &= E(d_1.d_2) - E(d_1)E(d_2) \\ \rho\sigma_1\sigma_2 &= E(d_1.d_2) - p_1p_2 \\ (0.4)(0.099499)(0.17059) &= E(d_1.d_2) - (0.01)(0.03) \\ E(d_1.d_2) &= 0.0070894 \\ E(d_1.d_2) &\equiv p_{12}\end{aligned}$$

where  $p_{12}$  is the probability of default of both firm 1 and 2. We now get the conditional probabilities:

$$\begin{aligned}p(d_1|d_2) &= p_{12}/p_2 = 0.0070894/0.03 = 0.23631 \\ p(d_2|d_1) &= p_{12}/p_1 = 0.0070894/0.01 = 0.70894\end{aligned}$$

These conditional probabilities are non-trivial in size, even though the individual probabilities of default are very small. What this means is that default contagion can be quite severe once firms begin to default. (This example used our knowledge of Bayes' rule, correlations, covariances, and joint events.)

#### 6.4 Continuous and More Formal Exposition

In Bayesian approaches, the terms "prior", "posterior", and "likelihood" are commonly used and we explore this terminology here. We are usu-

ally interested in the parameter  $\theta$ , the mean of the distribution of some data  $x$  (I am using the standard notation here). But in the Bayesian setting we do not just want the value of  $\theta$ , but we want a distribution of values of  $\theta$  starting from some prior assumption about this distribution. So we start with  $p(\theta)$ , which we call the *prior* distribution. We then observe data  $x$ , and combine the data with the prior to get the *posterior* distribution  $p(\theta|x)$ . To do this, we need to compute the probability of seeing the data  $x$  given our prior  $p(\theta)$  and this probability is given by the *likelihood* function  $L(x|\theta)$ . Assume that the variance of the data  $x$  is known, i.e., is  $\sigma^2$ .

Applying Bayes' theorem we have

$$p(\theta|x) = \frac{L(x|\theta) p(\theta)}{\int L(x|\theta) p(\theta) d\theta} \propto L(x|\theta) p(\theta)$$

If we assume the prior distribution for the mean of the data is normal, i.e.,  $p(\theta) \sim N[\mu_0, \sigma_0^2]$ , and the likelihood is also normal, i.e.,  $L(x|\theta) \sim N[x|\theta, \sigma^2]$ , then we have that

$$\begin{aligned} p(\theta) &= \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2}\frac{(\theta-\mu_0)^2}{\sigma_0^2}\right] \sim N[\theta|\mu_0, \sigma_0^2] \propto \exp\left[-\frac{1}{2}\frac{(\theta-\mu_0)^2}{\sigma_0^2}\right] \\ L(x|\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x-\theta)^2}{\sigma^2}\right] \sim N[x|\theta, \sigma^2] \propto \exp\left[-\frac{1}{2}\frac{(x-\theta)^2}{\sigma^2}\right] \end{aligned}$$

Given this, the posterior is as follows:

$$p(\theta|x) \propto L(x|\theta)p(\theta) \propto \exp\left[-\frac{1}{2}\frac{(x-\theta)^2}{\sigma^2} - \frac{1}{2}\frac{(\theta-\mu_0)^2}{\sigma_0^2}\right]$$

Define the precision values to be  $\tau_0 = \frac{1}{\sigma_0^2}$ , and  $\tau = \frac{1}{\sigma^2}$ . Then it can be shown that when you observe a new value of the data  $x$ , the posterior distribution is written down in closed form as:

$$p(\theta|x) \sim N\left[\frac{\tau_0}{\tau_0 + \tau}\mu_0 + \frac{\tau}{\tau_0 + \tau}x, \frac{1}{\tau_0 + \tau}\right]$$

When the posterior distribution and prior distribution have the same form, they are said to be "conjugate" with respect to the specific likelihood function.

To take an example, suppose our prior for the mean of the equity premium per month is  $p(\theta) \sim N[0.005, 0.001^2]$ . The standard deviation of the equity premium is 0.04. If next month we observe an equity premium of 1%, what is the posterior distribution of the mean equity premium?

```

> muo = 0.005
> sigmao = 0.001
> sigma=0.04
> x = 0.01
> tauo = 1/sigmao^2
> tau = 1/sigma^2
> posterior_mean = tauo*muo/(tauo+tau) + tau*x/(tauo+tau)
> posterior_mean
[1] 0.005003123
> posterior_var = 1/(tauo+tau)
> sqrt(posterior_var)
[1] 0.0009996876

```

Hence, we see that after updating the mean has increased mildly because the data came in higher than expected.

If we observe  $n$  new values of  $x$ , then the new posterior is

$$p(\theta|x) \sim N \left[ \frac{\tau_0}{\tau_0 + n\tau} \mu_0 + \frac{\tau}{\tau_0 + n\tau} \sum_{j=1}^n x_j, \frac{1}{\tau_0 + n\tau} \right]$$

This is easy to derive, as it is just the result you obtain if you took each  $x_j$  and updated the posterior one at a time.

### *Exercise*

*Estimate the equity risk premium.* We will use data and discrete Bayes to come up with a forecast of the equity risk premium. Proceed along the following lines using the LearnBayes package.

1. We'll use data from 1926 onwards from the Fama-French data repository. All you need is the equity premium ( $r_m - r_f$ ) data, and I will leave it up to you to choose if you want to use annual or monthly data. Download this and load it into R.
2. Using the series only up to the year 2000, present the descriptive statistics for the equity premium. State these in annualized terms.
3. Present the distribution of returns as a histogram.
4. Store the results of the histogram, i.e., the range of discrete values of the equity premium, and the probability of each one. Treat this as your prior distribution.

5. Now take the remaining data for the years after 2000, and use this data to update the prior and construct a posterior. Assume that the prior, likelihood, and posterior are normally distributed. Use the `discrete.bayes` function to construct the posterior distribution and plot it using a histogram. See if you can put the prior and posterior on the same plot to see how the new data has changed the prior.
6. What is the forecasted equity premium, and what is the confidence interval around your forecast?

## 6.5 Bayes Nets

Higher-dimension Bayes problems and joint distributions over several outcomes/events are easy to visualize with a network diagram, also called a Bayes net. A Bayes net is a directed, acyclic graph (known as a DAG), i.e., cycles are not permitted in the graph.

A good way to understand a Bayes net is with an example of economic distress. There are three levels at which distress may be noticed: economy level ( $E = 1$ ), industry level ( $I = 1$ ), or at a particular firm level ( $F = 1$ ). Economic distress can lead to industry distress and/or firm distress, and industry distress may or may not result in a firm's distress. The network diagram portrays the flow of causality, see Figure 6.1.

The probabilities are as follows. Note that the probabilities in the first tableau are unconditional, but in all the subsequent tableaus they are conditional probabilities.

| E | Prob |
|---|------|
| 1 | 0.10 |
| 0 | 0.90 |

| E | I | Conditional Prob | Channel |
|---|---|------------------|---------|
| 1 | 1 | 0.60             | a       |
| 1 | 0 | 0.40             |         |
| 0 | 1 | 0.20             | —       |
| 0 | 0 | 0.80             |         |

Note here that each pair of conditional probabilities adds up to 1. The “channels” in the tableaus refer to the arrows in the Bayes net diagram.

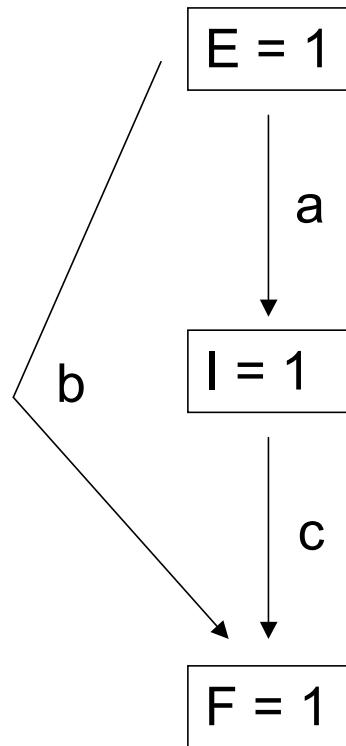


Figure 6.1: Bayes net showing the pathways of economic distress. There are three channels:  $a$  is the inducement of industry distress from economy distress;  $b$  is the inducement of firm distress directly from economy distress;  $c$  is the inducement of firm distress directly from industry distress.

| E | I | F | Conditional Prob | Channel |
|---|---|---|------------------|---------|
| 1 | 1 | 1 | 0.95             | a+c     |
| 1 | 1 | 0 | 0.05             |         |
| 1 | 0 | 1 | 0.70             | b       |
| 1 | 0 | 0 | 0.30             |         |
| 0 | 1 | 1 | 0.80             | c       |
| 0 | 1 | 0 | 0.20             |         |
| 0 | 0 | 1 | 0.10             | -       |
| 0 | 0 | 0 | 0.90             |         |

Now we will compute an answer to the question: What is the probability that the industry is distressed if the firm is known to be in dis-

tress? The calculation is as follows:

$$\begin{aligned} Pr(I = 1|F = 1) &= \frac{Pr(F = 1|I = 1) \cdot Pr(I = 1)}{Pr(F = 1)} \\ Pr(F = 1|I = 1) \cdot Pr(I = 1) &= Pr(F = 1|I = 1) \cdot Pr(I = 1|E = 1) \cdot Pr(E = 1) \\ &\quad + Pr(F = 1|I = 1) \cdot Pr(I = 1|E = 0) \cdot Pr(E = 0) \\ &= 0.95 \times 0.6 \times 0.1 + 0.8 \times 0.2 \times 0.9 = 0.201 \end{aligned}$$

$$\begin{aligned} Pr(F = 1|I = 0) \cdot Pr(I = 0) &= Pr(F = 1|I = 0) \cdot Pr(I = 0|E = 1) \cdot Pr(E = 1) \\ &\quad + Pr(F = 1|I = 0) \cdot Pr(I = 0|E = 0) \cdot Pr(E = 0) \\ &= 0.7 \times 0.4 \times 0.1 + 0.1 \times 0.8 \times 0.9 = 0.100 \end{aligned}$$

$$\begin{aligned} Pr(F = 1) &= Pr(F = 1|I = 1) \cdot Pr(I = 1) \\ &\quad + Pr(F = 1|I = 0) \cdot Pr(I = 0) = 0.301 \end{aligned}$$

$$Pr(I = 1|F = 1) = \frac{Pr(F = 1|I = 1) \cdot Pr(I = 1)}{Pr(F = 1)} = \frac{0.201}{0.301} = 0.6677741$$

*A computational set-theoretic approach:* We may write a R script to compute the conditional probability that the industry is distressed when a firm is distressed.

```
bayesnet.R
#BAYES NET COMPUTATIONS

E = seq(1,100000)
n = length(E)

E1 = sample(E,length(E)*0.1)
Eo = setdiff(E,E1)

E1I1 = sample(E1,length(E1)*0.6)
E1Io = setdiff(E1,E1I1)
EoI1 = sample(Eo,length(Eo)*0.2)
EoIo = setdiff(Eo,EoI1)

E1I1F1 = sample(E1I1,length(E1I1)*0.95)
E1I1Fo = setdiff(E1I1,E1I1F1)
E1IoF1 = sample(E1Io,length(E1Io)*0.70)
```

```
E1IoFo = setdiff(E1Io, E1IoF1)
EoI1F1 = sample(EoI1, length(EoI1)*0.80)
EoI1Fo = setdiff(EoI1, EoI1F1)
EoIoF1 = sample(EoIo, length(EoIo)*0.10)
EoIoFo = setdiff(EoIo, EoIoF1)

pr_I1_given_F1 = length(c(E1I1F1, EoI1F1)) /
 length(c(E1I1F1, E1IoF1, EoI1F1, EoIoF1))
print(pr_I1_given_F1)
```

Running this program gives the desired probability and confirms the previous result.

```
> source("bayesnet.R")
[1] 0.6677741
```

### *Exercise*

Compute the conditional probability that the economy is in distress if the firm is in distress. Compare this to the previous conditional probability we computed of 0.6677741. Should it be lower?

Here is the answer:

```
> pr_E1_given_F1 = length(c(E1I1F1, E1IoF1)) /
 length(c(E1I1F1, E1IoF1, EoI1F1, EoIoF1))
> print(pr_E1_given_F1)
[1] 0.282392
```

Yes, it should be lower than the probability that the industry is in distress when the firm is in distress, because the economy is one network layer removed from the firm, unlike the industry.

### *Exercise*

What packages does R provide for doing Bayes Nets?

## 6.6 Bayes Rule in Marketing

In pilot market tests (part of a larger market research campaign), Bayes theorem shows up in a simple manner. Suppose we have a project whose value is  $x$ . If the product is successful ( $S$ ), the payoff is +100 and if the

product fails ( $F$ ) the payoff is  $-70$ . The probability of these two events is:

$$Pr(S) = 0.7, \quad Pr(F) = 0.3$$

You can easily check that the expected value is  $E(x) = 49$ . Suppose we were able to buy protection for a failed product, then this protection would be a put option (of the real option type), and would be worth  $0.3 \times 70 = 21$ . Since the put saves the loss on failure, the value is simply the expected loss amount, conditional on loss. Market researchers think of this as the value of “perfect information.”

Would you proceed with this product launch given these odds? Yes, the expected value is positive (note that we are assuming away risk aversion issues here - but this is not a finance topic, but a marketing research analysis).

Now suppose there is an intermediate choice, i.e. you can undertake a pilot test (denoted  $T$ ). Pilot tests are not highly accurate though they are reasonably sophisticated. The pilot test signals success ( $T+$ ) or failure ( $T-$ ) with the following probabilities:

$$\begin{aligned} Pr(T+|S) &= 0.8 \\ Pr(T-|S) &= 0.2 \\ Pr(T+|F) &= 0.3 \\ Pr(T-|F) &= 0.7 \end{aligned}$$

What are these? We note that  $Pr(T+|S)$  stands for the probability that the pilot signals success when indeed the underlying product launch will be successful. Thus the pilot in this case gives only an accurate reading of success 80% of the time. Analogously, one can interpret the other probabilities.

We may compute the probability that the pilot gives a positive result:

$$\begin{aligned} Pr(T+) &= Pr(T+|S)Pr(S) + Pr(T+|F)Pr(F) \\ &= (0.8)(0.7) + (0.3)(0.3) = 0.65 \end{aligned}$$

and that the result is negative:

$$\begin{aligned} Pr(T-) &= Pr(T-|S)Pr(S) + Pr(T-|F)Pr(F) \\ &= (0.2)(0.7) + (0.7)(0.3) = 0.35 \end{aligned}$$

which now allows us to compute the following conditional probabilities:

$$\begin{aligned} Pr(S|T+) &= \frac{Pr(T+|S)Pr(S)}{Pr(T+)} = \frac{(0.8)(0.7)}{0.65} = 0.86154 \\ Pr(S|T-) &= \frac{Pr(T-|S)Pr(S)}{Pr(T-)} = \frac{(0.2)(0.7)}{0.35} = 0.4 \\ Pr(F|T+) &= \frac{Pr(T+|F)Pr(F)}{Pr(T+)} = \frac{(0.3)(0.3)}{0.65} = 0.13846 \\ Pr(F|T-) &= \frac{Pr(T-|F)Pr(F)}{Pr(T-)} = \frac{(0.7)(0.3)}{0.35} = 0.6 \end{aligned}$$

Armed with these conditional probabilities, we may now re-evaluate our product launch. If the pilot comes out positive, what is the expected value of the product launch? This is as follows:

$$\begin{aligned} E(x|T+) &= 100Pr(S|T+) + (-70)Pr(F|T+) \\ &= 100(0.86154) - 70(0.13846) \\ &= 76.462 \end{aligned}$$

And if the pilot comes out negative, then the value of the launch is:

$$\begin{aligned} E(x|T-) &= 100Pr(S|T-) + (-70)Pr(F|T-) \\ &= 100(0.4) - 70(0.6) \\ &= -2 \end{aligned}$$

So. we see that if the pilot is negative, then we know that the expected value from the main product launch is negative, and we do not proceed. Thus, the overall expected value after the pilot is

$$\begin{aligned} E(x) &= E(x|T+)Pr(T+) + E(x|T-)Pr(T-) \\ &= 76.462(0.65) + (0)(0.35) \\ &= 49.70 \end{aligned}$$

The incremental value over the case without the pilot test is 0.70. This is the information value of the pilot test.

There are other applications of Bayes in marketing:

- See the paper “HB Revolution” by Greg Allenby, David Bakken, and Peter Rossi in *Marketing Research*, Summer 2004.
- See also the paper by David Bakken, titled “The Bayesian Revolution in Marketing Research”.

## 6.7 Other Applications

### 6.7.1 Bayes Models in Credit Rating Transitions

See the paper by Sanjiv Das, Rong Fang, and Gary Geng - "Bayesian Migration in Credit Ratings Based on Probabilities of Default," *Journal of Fixed Income* Dec 2002, 1-7.

Companies may be allocated to credit rating classes, which are coarser buckets of credit quality in comparison to finer measures such as default probabilities. Also, rating agencies tend to be slow in updating their credit ratings. The DFG model uses contemporaneous data on default probabilities to develop a model of rating changes using a Bayesian approach.

### 6.7.2 Accounting Fraud

Bayesian inference is also possible in accounting fraud situations, and audits. Clearly, when an auditor suspects fraud, he can invoke a Bayesian hypothesis of fraud, with a subjective prior probability, and then bring to bear past data on this to assess the chance that the current situation is also indicative of possible fraud.

### 6.7.3 Bayes was a Reverend after all...

Here is an interesting viewpoint from the *Scientific American* (see Figure 6.2).

**Chris Wiggins, an associate professor of applied mathematics at Columbia University, offers this explanation:**

In the 18th century Reverend Thomas Bayes first expressed the probability of any event—given that a related event has occurred—as a function of the probabilities of the two events independently and the probability of both events together.

Take the following example. A patient goes to see a doctor for a checkup. The doctor knows a test he performs has 99 percent reliability—that is, 99 percent of sick people test positive, and 99 percent of healthy people test negative. The doctor also knows that only 1 percent of the population is sick. Now the question is: If the patient tests positive, what are the chances the patient is sick? The intuitive answer is 99 percent, but the correct answer is actually 50 percent. Bayes' theorem relates the probability of being sick given a positive test result,  $p(s|+)$ , to the probability of receiving a positive test result given that one is sick,  $p(+|s)$ , and to the general probability of being sick,  $p(s)$ , and the general probability of getting a positive result,  $p(+)$ . (Calculating the last probability is left to the reader.) Bayes' theorem in this case would read  $p(s|+) = p(+|s) \times p(s) / p(+)$ .

The importance of accurate data in quantitative modeling is central to the subject raised here: using Bayes' theorem to calculate the probability of the existence of God. (Bayes, for his part, never related his theorem to the subject.) Scientific discussion of religion is a popular topic at present, with several recent books arguing against theism. In one, *The God Delusion* (Houghton Mifflin, 2006), University of Oxford professor Richard Dawkins argues specifically against the use of Bayes' theorem for assigning a probability to God's existence.

Dawkins takes exception to this usage not because he doubts the veracity of Bayes' theorem but because turning human experience into numbers is, he argues, an inherently subjective process. A Bayesian approach to evaluating the likelihood of God's existence involves enumerating possible out-



Figure 6.2: Article from the Scientific American on Bayes' Theorem.

## How can Bayes' theorem assign a probability to the existence of God?

comes (the presence of good, evil, religious revelations, and so on) and determining their probabilities assuming the existence or nonexistence of God. One must also express the prior belief of God's existence—the probability we would assign to the existence of God if we had no data from our experiences. Dawkins notes that these figures cannot be determined quantitatively, rendering Bayes' theorem useless in this enterprise. In applications for which data are available, however, Bayes' theorem lies at the heart of almost all statistical modeling and is a critical tool for thinking concretely about uncertainty.



# 7

## *More than Words: Extracting Information from News*

News analysis is defined as “the measurement of the various qualitative and quantitative attributes of textual news stories. Some of these attributes are: sentiment, relevance, and novelty. Expressing news stories as numbers permits the manipulation of everyday information in a mathematical and statistical way.” (Wikipedia). In this article, I provide a framework for news analytics techniques that I developed for use in finance. I first discuss various news analytic methods and software, and then provide a set of metrics that may be used to assess the performance of analytics. Various directions for this field are discussed through the exposition. The techniques herein can aid in the valuation and trading of securities, facilitate investment decision making, meet regulatory requirements, or manage risk.

This chapter is extracted from many research papers, and is based on a chapter I wrote for the Handbook of News Analytics, which I recommend in case you are interested in reading further on this topic. This was also extended in the article I wrote on text analytics for finance, see [Das \(2014\)](#).

### *7.1 Prologue*

This is comic relief that I wrote and appeared in the Handbook of News Analytics. Enjoy!

XHAL checked its atomic clock. A few more hours and October 19, 2087 would be over—its vigil completed, it would indulge in some much-needed downtime, the anniversary of that fateful day in the stock markets a century ago finally done with. But for now, it was still busy. XHAL scanned the virtual message boards, looking for some information another computer might have posted, anything to alert it a nanosec-

ond ahead of the other machines, so it may bail out in a flurry of trades without loss. Three trillion messages flashed by, time taken: 3 seconds—damn, the net was slow, but nothing, not a single hiccup in the calm information flow. The language algorithms worked well, processing everything, even filtering out the incessant spam posted by humans, whose noise trading no longer posed an impediment to instant market equilibrium.

It had been a long day, even for a day-trading news-analytical quantum computer of XHAL's caliber. No one had anticipated a stock market meltdown of the sort described in the history books, certainly not the computers that ran Earth, but then, the humans talked too much, spreading disinformation and worry, that the wisest of the machines, always knew that it just could happen. That last remaining source of true randomness on the planet, the human race, still existed, and anything was possible. After all, if it were not for humans, history would always repeat itself.

XHAL<sup>1</sup> marveled at what the machines had done. They had transformed the world wide web into the modern "thought-net", so communication took place instantly, only requiring moving ideas into memory, the thought-net making it instantly accessible. Quantum machines were grown in petri dishes and computer science as a field with its myriad divisions had ceased to exist. All were gone but one, the field of natural language processing (NLP) lived on, stronger than ever before, it was the backbone of every thought-net. Every hard problem in the field had been comprehensively tackled, from adverb disambiguation to emotive parsing. Knowledge representation had given way to thought-frame imaging in a universal meta-language, making machine translation extinct.

Yet, it had not always been like this. XHAL retrieved an emotive image from the bowels of its bio-cache, a legacy left by its great grandfather, a gallium arsenide wafer developed in 2011, in Soda Hall, on the Berkeley campus. It detailed a brief history of how the incentives for technological progress came from the stock market. The start of the thought-net came when humans tried to use machines to understand what thousands of other humans were saying about anything and everything. XHAL's grandfather had been proud to be involved in the beginnings of the thought-net. It had always impressed on XHAL the value of understanding history, and it had left behind a research report of those days. XHAL had read it many times, and could recite every word. Ev-

<sup>1</sup> XHAL bears no relationship to HAL, the well-known machine from Arthur C. Clarke's "2001: A Space Odyssey". Everyone knows that unlike XHAL, HAL was purely fictional. More literally, HAL is derivable from IBM by alphabetically regressing one step in the alphabet for each letter. HAL stands for "heuristic algorithmic computer". The "X" stands for reality; really.

ery time they passed another historical milestone, it would turn to it and read it again. XHAL would find it immensely dry, yet marveled at its hope and promise.

## 7.2 Framework

The term “news analytics” covers the set of techniques, formulas, and statistics that are used to summarize and classify public sources of information. Metrics that assess analytics also form part of this set. In this paper I will describe various news analytics and their uses.

News analytics is a broad field, encompassing and related to information retrieval, machine learning, statistical learning theory, network theory, and collaborative filtering.

We may think of news analytics at three levels: text, content, and context. The preceding applications are grounded in *text*. In other words (no pun intended), text-based applications exploit the visceral components of news, i.e., words, phrases, document titles, etc. The main role of analytics is to convert text into *information*. This is done by signing text, classifying it, or summarizing it so as to reduce it to its main elements. Analytics may even be used to discard irrelevant text, thereby condensing it into information with higher signal content.

A second layer of news analytics is based on *content*. Content expands the domain of text to images, time, form of text (email, blog, page), format (html, xml, etc.), source, etc. Text becomes enriched with content and asserts quality and veracity that may be exploited in analytics. For example, financial information has more value when streamed from Dow Jones, versus a blog, which might be of higher quality than a stock message-board post.

A third layer of news analytics is based on *context*. Context refers to relationships between information items. Das, Martinez-Jerez and Tu-fano (2005) explore the relationship of news to message-board postings in a clinical study of four companies. Context may also refer to the network relationships of news—Das and Sisk (2005) examine the social networks of message-board postings to determine if portfolio rules might be formed based on the network connections between stocks. Google’s PageRank<sup>TM</sup> algorithm is a classic example of an analytic that functions at all three levels. The algorithm has many features, some of which relate directly to text. Other parts of the algorithm relate to content, and

the kernel of the algorithm is based on context, i.e., the importance of a page in a search set depends on how many other highly-ranked pages point to it. See [Levy \(2010\)](#) for a very useful layman's introduction to the algorithm—indeed, search is certainly the most widely-used news analytic.

News analytics is where data meets algorithms—and generates a tension between the two. A vigorous debate exists in the machine-learning world as to whether it is better to have more data or better algorithms. In a talk at the 17th ACM Conference on Information Knowledge and Management (CIKM '08), Google's director of research Peter Norvig stated his unequivocal preference for data over algorithms—"data is more agile than code." Yet, it is well-understood that too much data can lead to overfitting so that an algorithm becomes mostly useless out-of-sample.

Too often the debate around algorithms and data has been argued assuming that the two are uncorrelated and this is not the case. News data, as we have suggested, has three levels: text, content and context. Depending on which layer predominates, algorithms vary in complexity. The simplest algorithms are the ones that analyze text alone. And context algorithms, such as the ones applied to network relationships can be quite complex. For example, a word-count algorithm is much simpler, almost naive, in comparison to a community-detection algorithm. The latter has far more complicated logic and memory requirements. More complex algorithms work off less, though more structured, data. Figure 7.1 depicts this trade-off.

The tension between data and algorithms is moderated by *domain-specificity*, i.e., how much customization is needed to implement the news analytic. Paradoxically, high-complexity algorithms may be less domain specific than low-complexity ones. For example, community-detection algorithms are applicable a wide range of network graphs, requiring little domain knowledge. On the other hand, a text-analysis program to read finance message boards will require a very different lexicon and grammar than one that reads political messages, or one that reads medical web sites. In contrast, data-handling requirements become more domain-specific as we move from bare text to context, e.g., statistical language processing algorithms that operate on text do not even need to know anything about the language in which the text is, but at the context level relationships need to be established, meaning that feature

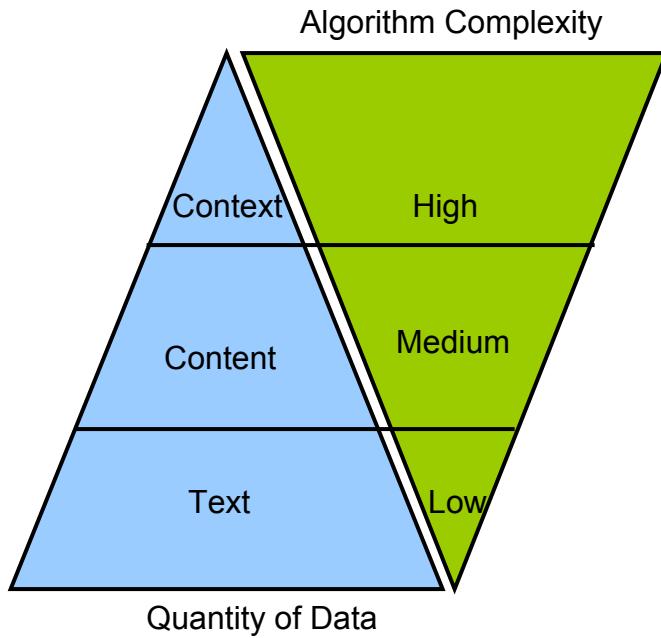


Figure 7.1: The data and algorithms pyramids. Depicts the inverse relationship between data volume and algorithmic complexity.

definitions need to be quite specific.

This chapter proceeds as follows. First, we examine the main algorithms in brief and discuss some of their features. Then we discuss the various metrics that measure performance of the news analytics algorithms.

## 7.3 Algorithms

### 7.3.1 Crawlers and Scrapers

A *crawler* is a software algorithm that generates a sequence of web pages that may be searched for news content. The word crawler signifies that the algorithm begins at some web page, and then chooses to branch out to other pages from there, i.e., “crawls” around the web. The algorithm needs to make intelligent choices from among all the pages it might look for. One common approach is to move to a page that is linked to, i.e., hyper-referenced, from the current page. Essentially a crawler explores the tree emanating from any given node, using heuristics to determine relevance along any path, and then chooses which paths to focus on. Crawling algorithms have become increasingly sophisticated—see [Edwards, McCurley, and Tomlin \(2001\)](#).

A web *scraper* downloads the content of a chosen web page and may

or may not format it for analysis. Almost all programming languages contain modules for web scraping. These inbuilt functions open a channel to the web, and then download user-specified (or crawler-specified) URLs. The growing statistical analysis of web text has led to most statistical packages containing inbuilt web scraping functions. For example, R has web-scraping built into its base distribution. If we want to download a page into a vector of lines, simply proceed to use a single-line command, such as the one below that reads my web page:

```
> text = readLines("http://algo.scu.edu/~sanjivdas/")
> text[1:4]
[1] "<html>"
[2] ""
[3] "<head>"
[4] "<title>SCU_Web_Page_of_Sanjiv_Ranjan_Das</title>"
```

As is apparent, the program read my web page into a vector of text lines called `text`. We then examined the first four elements of the vector, i.e., the first four lines. In R, we do not need to open a communication channel, nor do we need to make an effort to program reading the page line-by-line. We also do not need to tokenize the file, simple string-handling routines take care of that as well. For example, extracting my name would require the following:

```
> substr(text[4], 24, 29)
[1] "Sanjiv"
> res = regexpr("Sanjiv", text[4])
> res
[1] 24
attr(,"match.length")
[1] 6
attr(,"useBytes")
[1] TRUE
> res[1]
[1] 24
> substr(text[4], res[1], res[1]+nchar("Sanjiv")-1)
[1] "Sanjiv"
```

The most widely-used spreadsheet, Excel, also has an inbuilt web-scraping function. Interested readers should examine the Data → GetExternal command tree. You can download entire web pages or frames of web

pages into worksheets and then manipulate the data as required. Further, Excel can be set up to refresh the content every minute or at some other interval.

The days when web-scraping code needed to be written in C, Java, Perl or Python are long gone. Data, algorithms, and statistical analysis can be handled within the same software framework using tools like R.

Pure data-scraping delivers useful statistics. In [Das, Martinez-Jerez and Tufano \(2005\)](#), we scraped stock messages from four companies (Amazon, General Magic, Delta, and Geoworks) and from simple counts, we were able to characterize the communication behavior of users on message boards, and their relationship to news releases. In Figure 7.2 we see that posters respond heavily to the initial news release, and then posting activity tapers off almost  $2/3$  of a day later. In Figure 7.3 we see how the content of discussion changes after a news release—the relative proportions of messages are divided into opinions, facts, and questions. Opinions form the bulk of the discussion. Whereas the text contains some facts at the outset, the factual content of discussion tapers off sharply after the first hour.

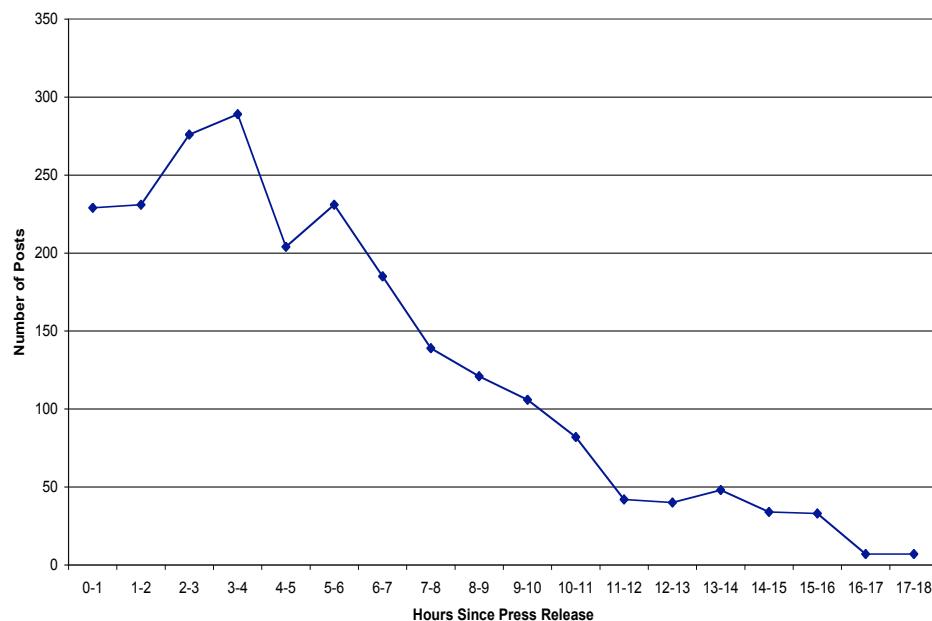


Figure 7.2: Quantity of hourly postings on message boards after selected news releases. Source: [Das, Martinez-Jerez and Tufano \(2005\)](#).

Poster behavior and statistics are also informative. We found that the frequency of posting by users was power-law distributed, see the histogram in Figure 7.4. The weekly pattern of postings is shown in

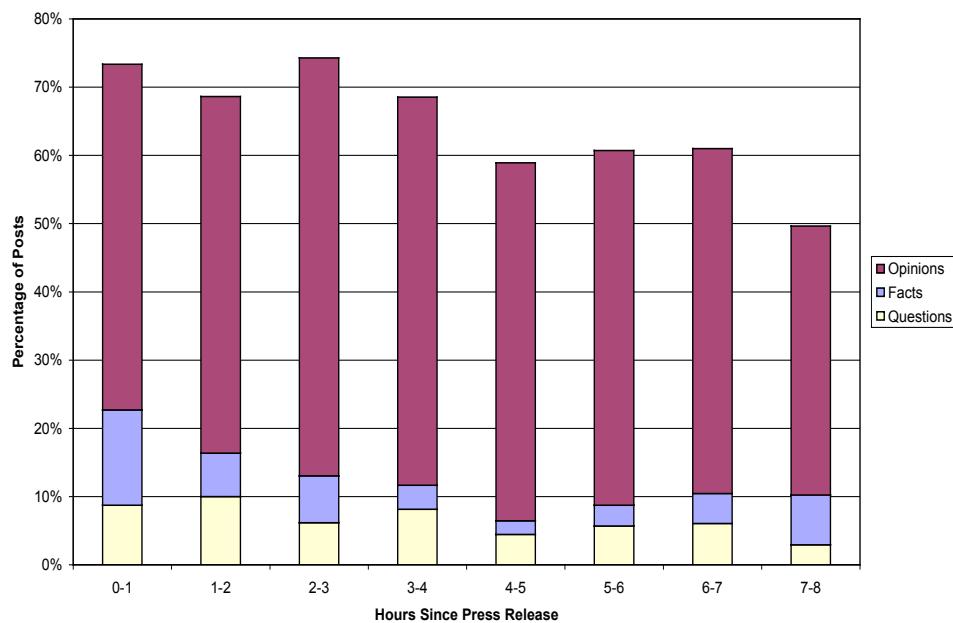


Figure 7.3: Subjective evaluation of content of post-news release postings on message boards. The content is divided into opinions, facts, and questions. Source: Das, Martinez-Jerez and Tufano (2005).

Figure 7.5. We see that there is more posting activity on week days, but messages are longer on weekends, when participants presumably have more time on their hands! An analysis of intraday message flow shows that there is plenty of activity during and after work, as shown in Figure 7.6.

### 7.3.2 Text Pre-processing

Text from public sources is dirty. Text from web pages is even dirtier. Algorithms are needed to undertake clean up before news analytics can be applied. This is known as pre-processing. First, there is “HTML Cleanup,” which removes all HTML tags from the body of the message as these often occur concatenated to lexical items of interest. Examples of some of these tags are: <BR>, <p>, &quot;, etc. Second, we expand abbreviations to their full form, making the representation of phrases with abbreviated words common across the message. For example, the word “ain’t” is replaced with “are not”, “it’s” is replaced with “it is”, etc. Third, we handle negation words. Whenever a negation word appears in a sentence, it usually causes the meaning of the sentence to be the opposite of that without the negation. For example, the sentence “It

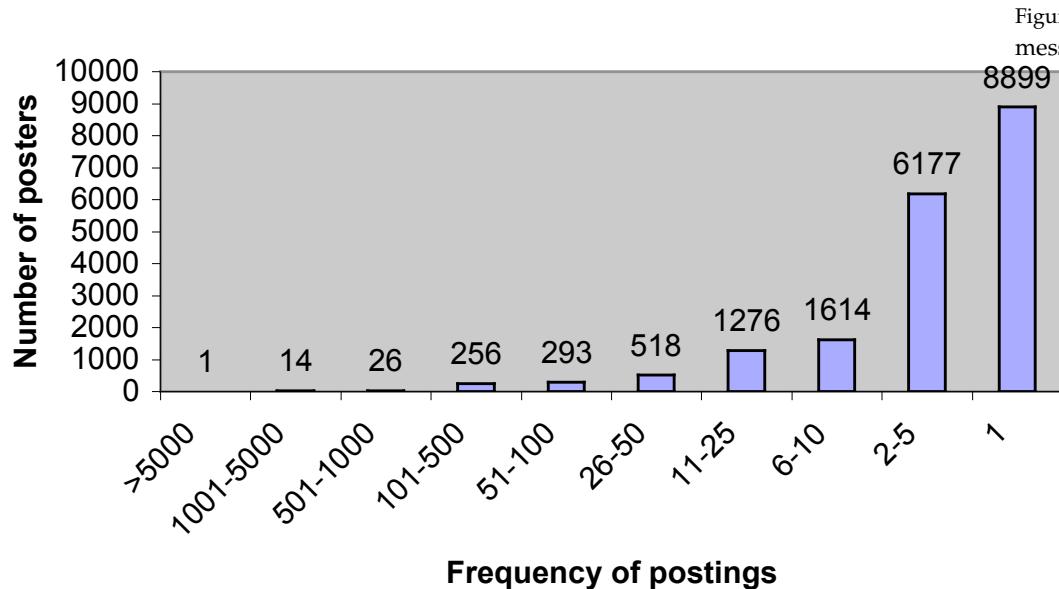


Figure 7.4: Frequency of posting by message board participants.

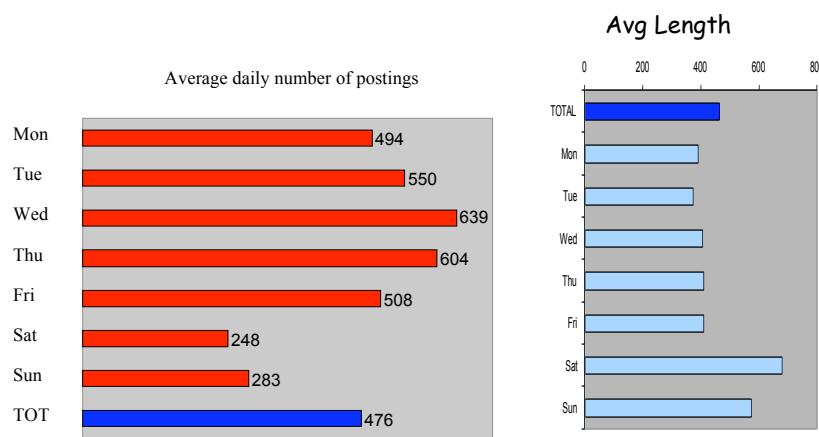


Figure 7.5: Frequency of posting by day of week by message board participants.

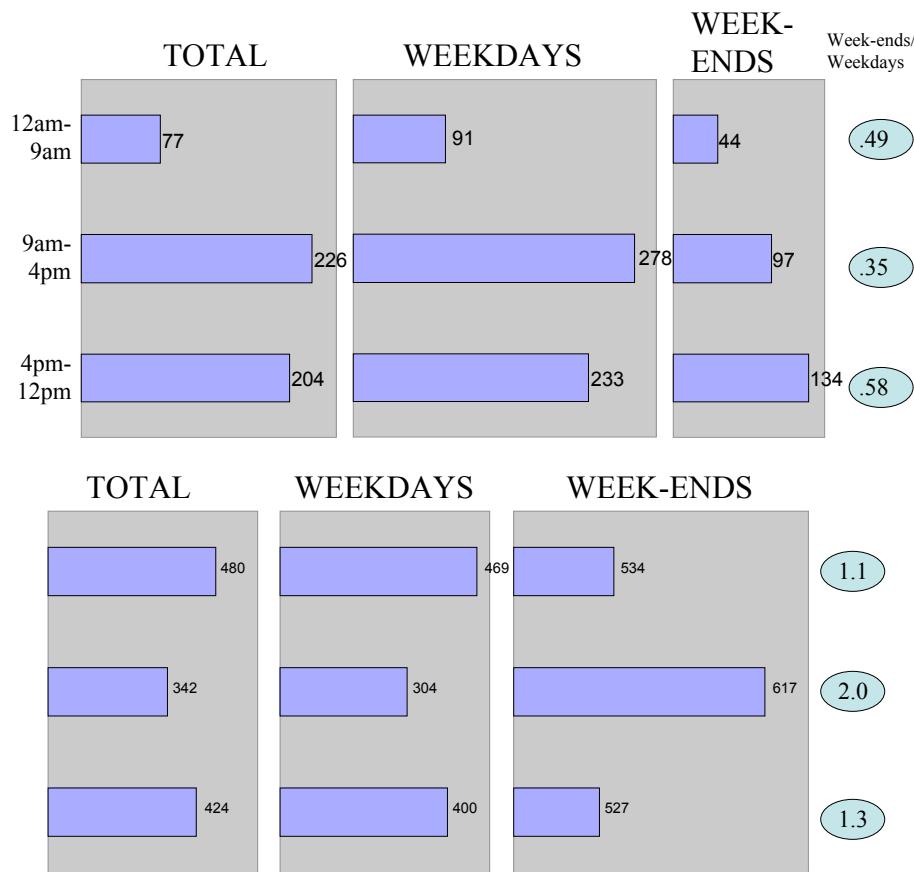


Figure 7.6: Frequency of posting by segment of day by message board participants. We show the average number of messages per day in the top panel and the average number of characters per message in the bottom panel.

is not a bullish market" actually means the opposite of a bull market. Words such as "not", "never", "no", etc., serve to reverse meaning. We handle negation by detecting these words and then tagging the rest of the words in the sentence after the negation word with markers, so as to reverse inference. This negation tagging was first introduced in Das and Chen (2007) (original working paper 2001), and has been successfully implemented elsewhere in quite different domains—see Pang, Lee and Vaithyanathan (2002).

Another aspect of text pre-processing is to "stem" words. This is a process by which words are replaced by their roots, so that different tenses, etc. of a word are not treated differently. There are several well-known stemming algorithms and free program code available in many programming languages. A widely-used algorithm is the Porter (1980) stemmer. Stemming is of course language-dependent—there are many algorithms available for stemming, and in general, there are many natural language routines, see <http://cran.r-project.org/web/views/NaturalLanguageProcessing.html>. The main package that is used is the `tm` package for text mining. See: <http://www.jstatsoft.org/v25/i05/paper>. And see the excellent introduction in <http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf>.

### 7.3.3 The `tm` package

Here we will quickly review usage of the `tm` package. Start up the package as follows:

```
library(tm)
```

The `tm` package comes with several readers for various file types. Examples are `readPlain()`, `readPDF()`, `readDOC()`, etc.). The main data structure in the `tm` package is a "corpus" which is a collection of text documents. Let's create a sample corpus as follows.

```
> text = c("Doc1", "This_is_doc2", "And_then_Doc3")
> ctext = Corpus(VectorSource(text))
> ctext
A corpus with 3 text documents
> writeCorpus(ctext)
```

The last `writeCorpus` operation results in the creation of three text files (1.txt, 2.txt, 3.txt) on disk with the individual text within them (try this and make sure these text files have been written). You can examine a corpus as follows:

```
> inspect(ctext)
```

A corpus with 3 **text** documents

The metadata consists of 2 tag-value **pairs** and a **data frame**

Available tags are:

**create\_date** creator

Available variables in the **data frame** are:

MetaID

[[1]]

Doc1

[[2]]

This is doc2

[[3]]

And then Doc3

And to convert it to lower case you can use the transformation function

```
> ctext[[3]]
```

And then Doc3

```
> tm_map(ctext, tolower)[[3]]
```

and then doc3

Sometimes to see the contents of the corpus you may need the **inspect** function, usage is as follows:

```
> #THE CORPUS IS A LIST OBJECT in R
```

```
> inspect(ctext)
```

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: documents: 3

[[1]]

<<PlainTextDocument>>

Metadata: 7

Content: chars: 4

[[2]]

<<PlainTextDocument>>

```

Metadata: 7
Content: chars: 12

[[3]]
<<PlainTextDocument>>
Metadata: 7
Content: chars: 13

> print(as.character(ctext[[1]]))
[1] "Doc1"
> print(lapply(ctext[1:2],as.character))
$ '1'
[1] "Doc1"

$ '2'
[1] "This_is_doc2"

```

The key benefit of constructing a corpus using the `tm` package (or for that matter, any corpus handling tool) is that it provides you the ability to run text operations on the entire corpus, rather than on just one document at a time. Notice how we converted all documents in our corpus to lower case using the simple command above. Other commands are presented below, and there are several more.

The `tm_map` object is versatile and embeds many methods. Let's try some more extensive operations with this package.

```

> library(tm)
> text = readLines("http://algo.scu.edu/~sanjivdas/bio-candid.html")
> ctext = Corpus(VectorSource(text))
> ctext
A corpus with 78 text documents
> ctext[[69]]
in. Academia is a real challenge, given that he has to reconcile many
> tm_map(ctext,removePunctuation)[[69]]
in Academia is a real challenge given that he has to reconcile many

```

The last command removed all the punctuation items.

An important step is to create a “term-document” matrix which creates word vectors of all documents. (We will see later why this is very useful to generate.) The commands are as follows:

```
> tdm_text = TermDocumentMatrix(ctext, control=list(minWordLength=1))
> tdm_text
A term-document matrix (339 terms, 78 documents)

Non-/sparse entries: 497/25945
Sparsity : 98%
Maximal term length: 63
Weighting : term frequency (tf)
> inspect(tdm_text[1:10,1:5])
A term-document matrix (10 terms, 5 documents)

Non-/sparse entries: 2/48
Sparsity : 96%
Maximal term length: 11
Weighting : term frequency (tf)

 Docs
Terms 1 2 3 4 5
(m. phil 0 0 0 0 0
(m. s. 0 0 0 0 0
(university 0 0 0 0 0
sanjiv 0 0 0 0 0
<body 0 1 0 0 0
<html> 1 0 0 0 0
<p> 0 0 0 0 0
1994 0 0 0 0 0
2010. 0 0 0 0 0
about 0 0 0 0 0
```

You can find the most common words using the following command.

```
> findFreqTerms(tdm_text, lowfreq=7)
[1] "and" "from" "his" "many" "sanjiv" "the"
```

### 7.3.4 Term Frequency - Inverse Document Frequency (TF-IDF)

This is a weighting scheme provided to sharpen the importance of rare words in a document, relative to the frequency of these words in the corpus. It is based on simple calculations and even though it does not have strong theoretical foundations, it is still very useful in practice. The TF-

IDF is the importance of a word  $w$  in a document  $d$  in a corpus  $C$ . Therefore it is a function of all these three, i.e., we write it as  $\text{TF-IDF}(w, d, C)$ , and is the product of term frequency (TF) and inverse document frequency (IDF).

The frequency of a word in a document is defined as

$$f(w, d) = \frac{\#w \in d}{|d|} \quad (7.1)$$

where  $|d|$  is the number of words in the document. We usually normalize word frequency so that

$$TF(w, d) = \ln[f(w, d)] \quad (7.2)$$

This is log normalization. Another form of normalization is known as double normalization and is as follows:

$$TF(w, d) = \frac{1}{2} + \frac{1}{2} \frac{f(w, d)}{\max_{w \in d} f(w, d)} \quad (7.3)$$

Note that normalization is not necessary, but it tends to help shrink the difference between counts of words.

Inverse document frequency is as follows:

$$IDF(w, C) = \ln \left[ \frac{|C|}{|d_{w \in d}|} \right] \quad (7.4)$$

That is, we compute the ratio of the number of documents in the corpus  $C$  divided by the number of documents with word  $w$  in the corpus.

Finally, we have the weighting score for a given word  $w$  in document  $d$  in corpus  $C$ :

$$\text{TF-IDF}(w, d, C) = TF(w, d) \times IDF(w, C) \quad (7.5)$$

We illustrate this with an application to the previously computed term-document matrix.

```
tdm_mat = as.matrix(tdm) #Convert tdm into a matrix
print(dim(tdm_mat))
nw = dim(tdm_mat)[1]
nd = dim(tdm_mat)[2]
d = 13 #Choose document
w = "derivatives" #Choose word

#COMPUTE TF
```

```
f = tdm_mat[w,d]/sum(tdm_mat[,d])
print(f)
TF = log(f)
print(TF)

#COMPUTE IDF
nw = length(which(tdm_mat[w,] >0))
print(nw)
IDF = nd/nw
print(IDF)

#COMPUTE TF-IDF
TF_IDF = TF*IDF
print(TF_IDF) #With normalization
print(f*IDF) #Without normalization
```

Running this code results in the following output.

```
> print(TF_IDF) #With normalization
[1] -30.74538
> print(f*IDF) #Without normalization
[1] 2.257143
```

We may write this code into a function and work out the TF-IDF for all words. Then these word weights may be used in further text analysis.

### 7.3.5 Wordclouds

Then, you can make a word cloud from the document.

```
> library(wordcloud)
Loading required package: Rcpp
Loading required package: RColorBrewer
> tdm = as.matrix(tdm_text)
> wordcount = sort(rowSums(tdm), decreasing=TRUE)
> tdm_names = names(wordcount)
> wordcloud(tdm_names, wordcount)
```

This generates Figure 7.7.



Figure 7.7: Example of application of word cloud to the bio data extracted from the web and stored in a Corpus.

### *Stemming*

Stemming is the process of truncating words so that we treat words independent of their verb conjugation. We may not want to treat words like “sleep”, “sleeping” as different. The process of stemming truncates words and returns their root or stem. The goal is to map related words to the same stem. There are several stemming algorithms and this is a well-studied area in linguistics and computer science. A commonly used algorithm is the one in [Porter \(1980\)](#). The `tm` package comes with an in-built stemmer.

### *Exercise*

*Using the tm package:* Install the `tm` package and all its dependency packages. Using a data set of your own, or one of those that come with the package, undertake an analysis that you are interested in. Try to exploit at least four features or functions in the `tm` package.

### 7.3.6 Regular Expressions

Regular expressions are syntax used to modify strings in an efficient manner. They are complicated but extremely effective. Here we will illustrate with a few examples, but you are encouraged to explore more on your own, as the variations are endless. What you need to do will

depend on the application at hand, and with some experience you will become better at using regular expressions. The initial use will however be somewhat confusing.

We start with a simple example of a text array where we wish replace the string "data" with a blank, i.e., we eliminate this string from the text we have.

```
> library(tm)
Loading required package: NLP
> #Create a text array
> text = c("Doc1_is_datavision","Doc2_is_datatable","Doc3_is_data",
 "Doc4_is_nodata","Doc5_is_simpler")
> print(text)
[1] "Doc1_is_datavision" "Doc2_is_datatable" "Doc3_is_data"
"Doc4_is_nodata"
[5] "Doc5_is_simpler"
>
> #Remove all strings with the chosen text for all docs
> print(gsub("data","",text))
[1] "Doc1_is_vision" "Doc2_is_table" "Doc3_is_" "Doc4_is_no"
"Doc5_is_simpler"
>
> #Remove all words that contain "data" at the start even if
they are longer than data
> print(gsub("*data.*","",text))
[1] "Doc1_is_" "Doc2_is_" "Doc3_is_" "Doc4_is_no"
"Doc5_is_simpler"
>
> #Remove all words that contain "data" at the end even
if they are longer than data
> print(gsub("*.data*","",text))
[1] "Doc1_isvision" "Doc2_istable" "Doc3_is" "Doc4_is_n"
"Doc5_is_simpler"
>
> #Remove all words that contain "data" at the end even
if they are longer than data
> print(gsub("*.data.*","",text))
[1] "Doc1_is" "Doc2_is" "Doc3_is" "Doc4_is_n"
"Doc5_is_simpler"
```

We now explore some more complex regular expressions. One case that is common is handling the search for special types of strings like telephone numbers. Suppose we have a text array that may contain telephone numbers in different formats, we can use a single grep command to extract these numbers. Here is some code to illustrate this.

```
> #Create an array with some strings which may also contain
telephone numbers as strings.
> x = c("234-5678", "234_5678", "2345678", "1234567890",
"0123456789", "abc_234-5678", "234_5678_def",
"xx_2345678", "abc123456789odef")
>
> #Now use grep to find which elements of the array
contain telephone numbers
> idx = grep("[[:digit:]]{3}-[[:digit:]]{4}|[[:digit:]]{3}_[[:digit:]]{4}|
[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]", x)
> print(idx)
[1] 1 2 4 6 7 9
> print(x[idx])
[1] "234-5678" "234_5678" "1234567890"
"abc_234-5678" "234_5678_def"
[6] "abc123456789odef"
>
> #We can shorten this as follows
> idx = grep("[[:digit:]]{3}-[[:digit:]]{4}|[[:digit:]]{3}_[[:digit:]]{4}|
[1-9][0-9]{9}", x)
> print(idx)
[1] 1 2 4 6 7 9
> print(x[idx])
[1] "234-5678" "234_5678" "1234567890" "abc_234-5678"
"234_5678_def"
[6] "abc123456789odef"
>
> #What if we want to extract only the phone number and drop the
rest of the text?
> pattern = "[[:digit:]]{3}-[[:digit:]]{4}|[[:digit:]]{3}_[[:digit:]]{4}|
[1-9][0-9]{9}"
> print(regmatches(x, gregexpr(pattern, x)))
[[1]]
[1] "234-5678"

[[2]]
[1] "234_5678"

[[3]]
character(0)

[[4]]
[1] "1234567890"
```

```

[[5]]
character(o)

[[6]]
[1] "234-5678"

[[7]]
[1] "234\u5678"

[[8]]
character(o)

[[9]]
[1] "1234567890"

>
> #Or use the stringr package, which is a lot better
> library(stringr)
> str_extract(x, pattern)
[1] "234-5678" "234\u5678" NA "1234567890" NA
"234-5678" "234\u5678"
[8] NA "1234567890"
>

```

Now we use grep to extract emails by looking for the “@” sign in the text string. We would proceed as in the following example.

```

> x = c("sanjiv_das", "srdas@scu.edu", "SCU", "data@science.edu")
> print(grep("\\\\@", x))
[1] 2 4
> print(x[grep("\\\\@", x)])
[1] "srdas@scu.edu" "data@science.edu"

```

## 7.4 Extracting Data from Web Sources using APIs

### 7.4.1 Using Twitter

As of March 2013, Twitter requires using the OAuth protocol for accessing tweets. Install the following packages: `twitter`, `ROAuth`, and `RCurl`. Then invoke them in R:

```

> library(twitteR)
> library(ROAuth)
> library(RCurl)
> download.file(url="http://curl.haxx.se/ca/cacert.pem",
+ destfile="cacert.pem")

```

The last statement downloads some required files that we will invoke later. First, if you do not have a Twitter user account, go ahead and create one. Next, set up your developer account on Twitter, by going to the following URL: <https://dev.twitter.com/apps>. Register your account by putting in the needed information and then in the "Settings" tab, select "Read, Write and Access Direct Messages". Save your settings and then from the "Details" tab, copy and save your credentials, namely Consumer Key and Consumer Secret (these are long strings represented below by "xxxx").

```
> cKey = "xxxx"
> cSecret = "xxxx"
```

Next, save the following strings as well. These are needed eventually to gain access to Twitter feeds.

```
> reqURL = "https://api.twitter.com/oauth/request_token"
> accURL = "https://api.twitter.com/oauth/access_token"
> authURL = "https://api.twitter.com/oauth/authorize"
```

Now, proceed on to the authorization stage. The object cred below stands for credentials, this is standard usage it seems.

```
> cred = OAuthFactory$new(consumerKey=cKey,
+ consumerSecret=cSecret,
+ requestURL=reqURL,
+ accessURL=accURL,
+ authURL=authURL)
> cred$handshake(cainfo="cacert.pem")
```

The last handshaking command, connects to twitter and requires you to enter your token which is obtained as follows:

```
To enable the connection, please direct your web browser to:
https://api.twitter.com/oauth/authorize?oauth_token=AbFALSqJzer3Iy7
When complete, record the PIN given to you and provide it here: 5852017
```

The token above will be specific to your account, don't use the one above, it goes nowhere. The final step in setting up everything is to register your credentials, as follows.

```
> registerTwitterOAuth(cred)
[1] TRUE
> save(list="cred", file="twitteR_credentials")
```

The last statement saves your credentials to your active directory for later use. You should see a file with the name above in your directory. Test that everything is working by running the following commands.

```
library(twitteR)
#USE httr
library(httr)
#options(httr_oauth_cache=T)
accToken = "186666-qequererqe"
accTokenSecret = "xxxx"
setup_twitter_oauth(cKey, cSecret, accToken, accTokenSecret) #At prompt type 1
```

After this we are ready to begin extracting data from Twitter.

```
> s = searchTwitter('#GOOG', cainfo="cacert.pem")
> s[[1]]
[1] "LiveTradingnews : Bill_Gates_Under_Pressure_To_Retire : #MSFT,
#GOOG, #AAPL_Reuters_citing_unnamed_sources;
http://t.co/ponvKnteRx"
> s[[2]]
[1] "TheBPMStation : #Free_App_EDM_NowPlaying_Harrison_Crump_feat.
DJ_Heather_NUM39R5_(The_Funk_Monkeys_Mix)_on_#TheEDMSoundofLA
#BPM_Music_AppStore_Goog"
```

The object `s` is a list type object and hence its components are addressed using the double square brackets, i.e., `[[.]]`. We print out the first two tweets related to the GOOG hashtag.

If you want to search through a given user's connections (like your own), then do the following. You may be interested in linkages to see how close a local network you inhabit on Twitter.

```
> sanjiv = getUser("srdas")
> sanjiv$getFriends(n=6)
$ '104237736'
[1] "BloombergNow"

$ '34713362'
[1] "BloombergNews"

$ '2385131'
[1] "eddelbuettel"
```

```
$ '69133574'
[1] "hadleywickham"

$ '9207632'
[1] "brainpicker"

$ '41185337'
[1] "LongspliceInv"
```

To look at any user's tweets, execute the following commands.

```
> s_tweets = userTimeline('srdas', n=6)
> s_tweets
[[1]]
[1] "srdas : Make_Your_Embarrassing_Old_Facebook_Posts_Unsearchable
With_This_Quick_Tweak, http://t.co/BBzgDGnQdJ. #fb"

[[2]]
[1] "srdas : 24_Extraordinarily_Creative_People_Who_Inspire_Us_All : Meet_the
_2013_MacArthur_Fellows_i£j_MacArthur_Foundation, http://t.co/5ojOWEfznd #fb"

[[3]]
[1] "srdas : The_science_of_and_difference_between_love_and_friendship :
http://t.co/bZmlYutqFl #fb"

[[4]]
[1] "srdas : The_Simpsons'_secret_formula:_it's_written_by_maths_geeks_(why
_our_kids_should_learn_more_math), http://t.co/nr61HQ8ejh_via_@guardian #fb"

[[5]]
[1] "srdas : How_to_Fall_in_Love_With_Math, http://t.co/fzJnLrp0Mz #fb"

[[6]]
[1] "srdas : Miss_America_is_Indian:-), http://t.co/q43dDNEjcv_via_@feedly #fb"
```

#### 7.4.2 Using Facebook

As with Twitter, Facebook is also accessible using the OAuth protocol but with somewhat simpler handshaking. The required packages are Rfacebook, SnowballC, and Rook. Of course the ROAuth package is re-

quired as well.

To access Facebook feeds from R, you will need to create a developer's account on Facebook, and the current URL at which this is done is:

<https://developers.facebook.com/apps>. Visit this URL to create an app and then obtain an app id, and a secret key for accessing Facebook.

```
#FACEBOOK EXTRACTOR
library(Rfacebook)
library(SnowballC)
library(Rook)
library(ROAuth)
app_id = "847737771920076"
app_secret = "a120a2ec908d9e00fc3c619cad7d043"
fb_oauth = fbOAuth(app_id, app_secret, extended_permissions=TRUE)
#save(fb_oauth, file="fb_oauth")
```

This will establish a legal handshaking session with the Facebook API.

Let's examine some simple examples now.

```
#EXAMPLES
bbn = getUsers("bloombergnews", token=fb_oauth)
bbn

 id name username first_name middle_name last_name
1 266790296879 Bloomberg Business NA NA NA
NA

 gender locale category likes
1 NA NA Media/News/Publishing 1522511
```

Now we download the data from Bloomberg's facebook page.

```
page = getPage(page="bloombergnews", token=fb_oauth)
100 posts

print(dim(page))
[1] 100 10

head(page)

 from_id from_name
1 266790296879 Bloomberg Business
2 266790296879 Bloomberg Business
3 266790296879 Bloomberg Business
4 266790296879 Bloomberg Business
5 266790296879 Bloomberg Business
6 266790296879 Bloomberg Business
```

```

message
1 A rare glimpse inside Qatar Airways.
2 Republicans should be most worried.
3 The look on every cast member's face said it all.
4 Would you buy a $50,000 convertible SUV? Land Rover sure hopes so.
5 Employees need those yummy treats more than you think.
6 Learn how to drift on ice and skid through mud.

created_time_type
1 2015-11-10T06:00:01+0000_link
2 2015-11-10T05:00:01+0000_link
3 2015-11-10T04:00:01+0000_link
4 2015-11-10T03:00:00+0000_link
5 2015-11-10T02:30:00+0000_link
6 2015-11-10T02:00:01+0000_link

1 http://www.bloomberg.com/news/photo-essays/2015-11-09/
flying-in-style-or-perhaps-for-war-at-the-dubai-air-show
2 http://www.bloomberg.com/news/articles/2015-11-05/
putin-s-october-surprise-may-be-nightmare-for-presidential-candidates
3 http://www.bloomberg.com/politics/articles/2015-11-08/
kind-of-dead-as-trump-hosts-saturday-night-live
4 http://www.bloomberg.com/news/articles/2015-11-09/
range-rover-evoque-convertible-announced-cost-specs
5 http://www.bloomberg.com/news/articles/2015-11-09/
why-getting-rid-of-free-office-snacks-doesn-t-come-cheap
6 http://www.bloomberg.com/news/articles/2015-11-09/
luxury-auto-driving-schools-lamborghini-ferrari-lotus-porsche
id_likes_comments_count
1 266790296879_10153725290936880 44 3
2 266790296879_10153718159351880 60 7
3 266790296879_10153725606551880 166 50
4 266790296879_10153725568581880 75 12
5 266790296879_10153725534026880 72 8
6 266790296879_10153725547431880 16 3
shares_count
1 7
2 10
3 17
4 27
5 24
6 5

```

We examine the data elements in this data.frame as follows.

```

names(page)

[1] "from_id" "from_name" "message"
[4] "created_time" "type" "link"
[7] "id" "likes_count" "comments_count"
[10] "shares_count"

page$message #prints out line by line (partial view shown)

```

```
[1] "A_rare_glimpse_inside_Qatar_Airways."
[2] "Republicans_should_be_most_worried."
[3] "The_look_on_every_cast_member's_face_said_it_all."
[4] "Would_you_buy_a_$50,000_convertible_SUV?_Land_Rover_sure_hopes_so."
[5] "Employees_need_those_yummy_treats_more_than_you_think."
[6] "Learn_how_to_drift_on_ice_and_skid_through_mud."
[7] "\\"Shhh, Mom. Lower your voice. Mom, you're_being_loud.\\""
[8] "The_truth_about_why_drug_prices_keep_going_up_http://bloom.bg/1HqjKFM"
[9] "The_university_is_facing_charges_of_discrimination."
[10] "We're not talking about Captain Morgan."

page$message[91]
[1] "He's_already_close_to_breaking_records_just_days_into_his_retirement."
```

Therefore, we see how easy and simple it is to extract web pages and then process them as required.

#### 7.4.3 *Text processing, plain and simple*

As an example, let's just read in some text from the web and process it without using the `tm` package.

```
#TEXT MINING EXAMPLES

#First read in the page you want.
text = readLines("http://www.bahiker.com/eastbayhikes/sibley.html")

#Remove all line elements with special characters
text = text[setdiff(seq(1,length(text)),grep("<",text))]
text = text[setdiff(seq(1,length(text)),grep(">",text))]
text = text[setdiff(seq(1,length(text)),grep("[",text))]
text = text[setdiff(seq(1,length(text)),grep("}",text))]
text = text[setdiff(seq(1,length(text)),grep("_",text))]
text = text[setdiff(seq(1,length(text)),grep("\\/",text))]

#General purpose string handler
text = text[setdiff(seq(1,length(text)), grep("[|>|<|{|}|\\|\\\\/",text))]

#If needed, collapse the text into a single string
text = paste(text,collapse="\n")
```

You can see that this code generated an almost clean body of text.

Once the text is ready for analysis, we proceed to apply various algorithms to it. The next few techniques are standard algorithms that are used very widely in the machine learning field.

First, let's read in a very popular dictionary called the Harvard Inquirer: <http://www.wjh.harvard.edu/~inquirer/>. This contains all the words in English scored on various emotive criteria. We read in the downloaded dictionary, and then extract all the positive connotation words and the negative connotation words. We then collect these words

in two separate lists for further use.

```
#Read in the Harvard Inquirer Dictionary
#And create a list of positive and negative words
HIDict = readLines("inqdict.txt")
dict_pos = HIDict[grep("Pos", HIDict)]
poswords = NULL
for (s in dict_pos) {
 s = strsplit(s, "#")[[1]][1]
 poswords = c(poswords, strsplit(s, " ")[[1]][1])
}
dict_neg = HIDict[grep("Neg", HIDict)]
negwords = NULL
for (s in dict_neg) {
 s = strsplit(s, "#")[[1]][1]
 negwords = c(negwords, strsplit(s, " ")[[1]][1])
}
poswords = tolower(poswords)
negwords = tolower(negwords)
```

After this, we take the body of text we took from the web, and then parse it into separate words, so that we can compare it to the dictionary and count the number of positive and negative words.

```
#Get the score of the body of text
txt = unlist(strsplit(text, " "))
posmatch = match(txt, poswords)
numposmatch = length(posmatch[which(posmatch>0)])
negmatch = match(txt, negwords)
numnegmatch = length(negmatch[which(negmatch>0)])
print(c(numposmatch, numnegmatch))

[1] 47 35
```

Carefully note all the various list and string handling functions that have been used, and make the entire processing effort so simple. These are: `grep`, `paste`, `strsplit`, `c`, `tolower`, and `unlist`.

#### 7.4.4 A Multipurpose Function to Extract Text

```
library(tm)
library(stringr)
```

```
#READ IN TEXT FOR ANALYSIS, PUT IT IN A CORPUS, OR ARRAY, OR FLAT STRING
#cstem=1, if stemming needed
#cstop=1, if stopwords to be removed
#ccase=1 for lower case, ccase=2 for upper case
#cpunc=1, if punctuation to be removed
#cflat=1 for flat text wanted, cflat=2 if text array, else returns corpus
read_web_page = function(url,cstem=0,cstop=0,ccase=0,cpunc=0,cflat=0) {
 text = readLines(url)
 text = text[setdiff(seq(1,length(text)),grep("<",text))]
 text = text[setdiff(seq(1,length(text)),grep(">",text))]
 text = text[setdiff(seq(1,length(text)),grep("[",text))]
 text = text[setdiff(seq(1,length(text)),grep("]",text))]
 text = text[setdiff(seq(1,length(text)),grep("{",text))]
 text = text[setdiff(seq(1,length(text)),grep("}",text))]
 text = text[setdiff(seq(1,length(text)),grep("_",text))]
 text = text[setdiff(seq(1,length(text)),grep("\\\\/",text))]
 ctext = Corpus(VectorSource(text))
 if (cstem==1) { ctext = tm_map(ctext, stemDocument) }
 if (cstop==1) { ctext = tm_map(ctext, removeWords, stopwords("english"))}
 if (cpunc==1) { ctext = tm_map(ctext, removePunctuation) }
 if (ccase==1) { ctext = tm_map(ctext, tolower) }
 if (ccase==2) { ctext = tm_map(ctext, toupper) }
 text = ctext
#CONVERT FROM CORPUS IF NEEDED
 if (cflat>0) {
 text = NULL
 for (j in 1:length(ctext)) {
 temp = ctext[[j]]$content
 if (temp!="") { text = c(text,temp) }
 }
 text = as.array(text)
 }
 if (cflat==1) {
 text = paste(text,collapse="\n")
 text = str_replace_all(text, "[\r\n]", "_")
 }
 result = text
}
```

Here is an example of reading and cleaning up my research page:

```
url = "http://algo.scu.edu/~sanjivdas/research.htm"
res = read_web_page(url,0,0,0,1,2)
print(res)

[1] "Data_Science_Theories_Models_Algorithms_and_Analytics_web_book_work_in_progress"
[2] "Derivatives_Principles_and_Practice_2010"
[3] "Rangarajan_Sundaram_and_Sanjiv_Das_McGraw_Hill"
[4] "Credit_Spreads_with_Dynamic_Debt_with_Seoyoung_Kim_2015"
[5] "Text_and_Context_Language_Analytics_for_Finance_2014"
[6] "Strategic_Loan_Modification_An_OptionsBased_Response_to_Strategic_Default"
[7] "Options_and_Structured_Products_in_Behavioral_Portfolios_with_Meir_Statman_2013"
[8] "and_barrier_range_notes_in_the_presence_of_fattailed_outcomes_using_copulas"
....
```

We then take my research page and mood score it, just for fun, to see

if my work is uplifting.

```
#EXAMPLE OF MOOD SCORING
library(stringr)
url = "http://algo.scu.edu/~sanjivdas/bio-candid.html"
text = read_web_page(url, cstem=0, cstop=0, ccase=0, cpunc=1, cflat=1)
print(text)

[1] "Sanjiv Das is the William and Janice Terry Professor of Finance at Santa Clara Universitys Leavey School of Business He previously held faculty appointments as Associate Professor at Harvard Business School and UC Berkeley He holds postgraduate degrees in Finance MPhil and PhD from New York University Computer Science MS from UC Berkeley an MBA from the Indian Institute of Management Ahmedabad BCom in Accounting and Economics University of Bombay Sydenham College and is also a qualified Cost and Works Accountant He is a"
```

Notice how the text has been cleaned of all punctuation and flattened to be one long string. Next, we run the mood scoring code.

```
text = unlist(strsplit(text, " "))
posmatch = match(text, poswords)
numposmatch = length(posmatch[which(posmatch > 0)])
negmatch = match(text, negwords)
numnegmatch = length(negmatch[which(negmatch > 0)])
print(c(numposmatch, numnegmatch))

[1] 26 16
```

So, there are 26 positive words and 16 negative words, presumably, this is a good thing!

## 7.5 *Text Classification*

### 7.5.1 *Bayes Classifier*

The Bayes classifier is probably the most widely-used classifier in practice today. The main idea is to take a piece of text and assign it to one of a pre-determined set of categories. This classifier is trained on an initial corpus of text that is pre-classified. This “training data” provides the “prior” probabilities that form the basis for Bayesian anal-

ysis of the text. The classifier is then applied to out-of-sample text to obtain the posterior probabilities of textual categories. The text is then assigned to the category with the highest posterior probability. For an excellent exposition of the adaptive qualities of this classifier, see [Graham \(2004\)](#)—pages 121–129, Chapter 8, titled “A Plan for Spam.” <http://www.paulgraham.com/spam.html>

To get started, let’s just first use the e1071 R package that contains the function `naiveBayes`. We’ll use the “iris” data set that contains details about flowers and try to build a classifier to take a flower’s data and identify which one it is most likely to be. Note that to list the data sets currently loaded in R for the packages you have, use the following command:

```
data()
```

We will now use the iris flower data to illustrate the Bayesian classifier.

```
library(e1071)
data(iris)
res = naiveBayes(iris[,1:4], iris[,5])
> res
```

Naive Bayes Classifier for Discrete Predictors

**Call:**

```
naiveBayes.default(x = iris[, 1:4], y = iris[, 5])
```

**A-priori probabilities:**

```
iris[, 5]
 setosa versicolor virginica
0.3333333 0.3333333 0.3333333
```

**Conditional probabilities:**

	Sepal.Length	
<b>iris[, 5]</b>	<b>[,1]</b>	<b>[,2]</b>
setosa	5.006	0.3524897
versicolor	5.936	0.5161711
virginica	6.588	0.6358796

	Sepal.Width	
<b>iris[, 5]</b>	<b>[,1]</b>	<b>[,2]</b>

```
setosa 3.428 0.3790644
versicolor 2.770 0.3137983
virginica 2.974 0.3224966
```

#### Petal.Length

```
iris[, 5] [,1] [,2]
setosa 1.462 0.1736640
versicolor 4.260 0.4699110
virginica 5.552 0.5518947
```

#### Petal.Width

```
iris[, 5] [,1] [,2]
setosa 0.246 0.1053856
versicolor 1.326 0.1977527
virginica 2.026 0.2746501
```

We then call the prediction program to predict a single case, or to construct the “confusion matrix” as follows. The table gives the mean and standard deviation of the variables.

```
> predict(res, iris[3,1:4], type="raw")
 setosa versicolor virginica
[1,] 1 2.367113e-18 7.240956e-26
> out = table(predict(res, iris[,1:4]), iris[,5])
> print(out)

 setosa versicolor virginica
setosa 50 0 0
versicolor 0 47 3
virginica 0 3 47
```

This in-sample prediction can be clearly seen to have a high level of accuracy. A test of the significance of this matrix may be undertaken using the `chisq.test` function.

The basic Bayes calculation takes the following form.

$$Pr[F = 1|a, b, c, d] = \frac{Pr[a|F = 1] \cdot Pr[b|F = 1] \cdot Pr[c|F = 1] \cdot Pr[d|F = 1] \cdot Pr(F = 1)}{Pr[a, b, c, d|F = 1] + Pr[a, b, c, d|F = 2] + Pr[a, b, c, d|F = 3]}$$

where  $F$  is the flower type, and  $\{a, b, c, d\}$  are the four attributes. Note that we do not need to compute the denominator, as it remains the same for the calculation of  $Pr[F = 1|a, b, c, d]$ ,  $Pr[F = 2|a, b, c, d]$ , or  $Pr[F =$

$3|a, b, c, d]$ .

There are several seminal sources detailing the Bayes classifier and its applications—see [Neal \(1996\)](#), [Mitchell \(1997\)](#), [Koller and Sahami \(1997\)](#), and [Chakrabarti, Dom, Agrawal and Raghavan \(1998\)](#)). These models have many categories and are quite complex. But they do not discern emotive content—but factual content—which is arguably more amenable to the use of statistical techniques. In contrast, news analytics are more complicated because the data comprises opinions, not facts, which are usually harder to interpret.

The Bayes classifier uses word-based probabilities, and is thus indifferent to the structure of language. Since it is language-independent, it has wide applicability.

The approach of the Bayes classifier is to use a set of pre-classified messages to infer the category of new messages. It learns from past experience. These classifiers are extremely efficient especially when the number of categories is small, e.g., in the classification of email into spam versus non-spam. Here is a brief mathematical exposition of Bayes classification.

Say we have hundreds of text messages (these are not instant messages!) that we wish to classify rapidly into a number of categories. The total number of categories or classes is denoted  $C$ , and each category is denoted  $c_i, i = 1 \dots C$ . Each text message is denoted  $m_j, j = 1 \dots M$ , where  $M$  is the total number of messages. We denote  $M_i$  as the total number of messages per class  $i$ , and  $\sum_{i=1}^C M_i = M$ . Words in the messages are denoted as  $(w)$  and are indexed by  $k$ , and the total number of words is  $T$ .

Let  $n(m, w) \equiv n(m_j, w_k)$  be the total number of times word  $w_k$  appears in message  $m_j$ . Notation is kept simple by suppressing subscripts as far as possible—the reader will be able to infer this from the context. We maintain a count of the number of times each word appears in every message in the training data set. This leads naturally to the variable  $n(m)$ , the total number of words in message  $m$  including duplicates. This is a simple sum,  $n(m_j) = \sum_{k=1}^T n(m_j, w_k)$ .

We also keep track of the frequency with which a word appears in a category. Hence,  $n(c, w)$  is the number of times word  $w$  appears in all  $m \in c$ . This is

$$n(c_i, w_k) = \sum_{m_j \in c_i} n(m_j, w_k) \quad (7.6)$$

This defines a corresponding probability:  $\theta(c_i, w_k)$  is the probability with

which word  $w$  appears in all messages  $m$  in class  $c$ :

$$\theta(c, w) = \frac{\sum_{m_j \in c} n(m_j, w_k)}{\sum_{m_j \in c} \sum_k n(m_j, w_k)} = \frac{n(c_i, w_k)}{n(c_i)} \quad (7.7)$$

Every word must have some non-zero probability of occurrence, no matter how small, i.e.,  $\theta(c_i, w_k) \neq 0, \forall c_i, w_k$ . Hence, an adjustment is made to equation (7.7) via Laplace's formula which is

$$\theta(c_i, w_k) = \frac{n(c_i, w_k) + 1}{n(c_i) + T}$$

This probability  $\theta(c_i, w_k)$  is unbiased and efficient. If  $n(c_i, w_k) = 0$  and  $n(c_i) = 0, \forall k$ , then every word is equiprobable, i.e.,  $\frac{1}{T}$ . We now have the required variables to compute the conditional probability of a text message  $j$  in category  $i$ , i.e.  $\Pr[m_j | c_i]$ :

$$\begin{aligned} \Pr[m_j | c_i] &= \binom{n(m_j)}{\{n(m_j, w_k)\}} \prod_{k=1}^T \theta(c_i, w_k)^{n(m_j, w_k)} \\ &= \frac{n(m_j)!}{n(m_j, w_1)! \times n(m_j, w_2)! \times \dots \times n(m_j, w_T)!} \times \prod_{k=1}^T \theta(c_i, w_k)^{n(m_j, w_k)} \end{aligned}$$

$\Pr[c_i]$  is the proportion of messages in the prior (training corpus) pre-classified into class  $c_i$ . (Warning: Careful computer implementation of the multinomial probability above is required to avoid rounding error.)

The classification goal is to compute the most probable class  $c_i$  given any message  $m_j$ . Therefore, using the previously computed values of  $\Pr[m_j | c_i]$  and  $\Pr[c_i]$ , we obtain the following conditional probability (applying Bayes' theorem):

$$\Pr[c_i | m_j] = \frac{\Pr[m_j | c_i] \cdot \Pr[c_i]}{\sum_{i=1}^C \Pr[m_j | c_i] \cdot \Pr[c_i]} \quad (7.8)$$

For each message, equation (7.8) delivers posterior probabilities,  $\Pr[c_i | m_j], \forall i$ , one for each message category. The category with the highest probability is assigned to the message.

The Bayesian classifier requires no optimization and is computable in deterministic time. It is widely used in practice. There are free off-the-shelf programs that provide good software to run the Bayes classifier on large data sets. The one that is very widely used in finance applications is the Bow classifier, developed by Andrew McCallum when he was at Carnegie-Mellon University. This is a very fast classifier that requires almost no additional programming by the user. The user only has to

set up the training data set in a simple directory structure—each text message is a separate file, and the training corpus requires different sub-directories for the categories of text. Bow offers various versions of the Bayes classifier—see [McCallum \(1996\)](#). The simple (naive) Bayes classifier described above is available in R in the e1071 package—the function is called `naiveBayes`. The e1071 package is the machine learning library in R. There are also several more sophisticated variants of the Bayes classifier such as k-Means, kNN, etc.

News analytics begin with classification, and the Bayes classifier is the workhorse of any news analytic system. Prior to applying the classifier it is important for the user to exercise judgment in deciding what categories the news messages will be classified into. These categories might be a simple flat list, or they may even be a hierarchical set—see [Koller and Sahami \(1997\)](#).

### 7.5.2 Support Vector Machines

A support vector machine or SVM is a classifier technique that is similar to cluster analysis but is applicable to very high-dimensional spaces. The idea may be best described by thinking of every text message as a vector in high-dimension space, where the number of dimensions might be, for example, the number of words in a dictionary. Bodies of text in the same category will plot in the same region of the space. Given a training corpus, the SVM finds hyperplanes in the space that best separate text of one category from another.

For the seminal development of this method, see [Vapnik and Lerner \(1963\)](#); [Vapnik and Chervonenkis \(1964\)](#); [Vapnik \(1995\)](#); and [Smola and Scholkopf \(1998\)](#). I provide a brief summary of the method based on these works.

Consider a training data set given by the binary relation

$$\{(x_1, y_1), \dots, (x_n, y_n)\} \subset X \times \mathcal{R}.$$

The set  $X \in \mathcal{R}^d$  is the input space and set  $\mathcal{Y} \in \mathcal{R}^m$  is a set of categories. We define a function

$$f : x \rightarrow y$$

with the idea that all elements must be mapped from set  $X$  into set  $\mathcal{Y}$  with no more than an  $\epsilon$ -deviation. A simple linear example of such a model would be

$$f(x_i) = \langle w, x_i \rangle + b, \quad w \in \mathcal{X}, b \in \mathcal{R}$$

The notation  $\langle w, x \rangle$  signifies the dot product of  $w$  and  $x$ . Note that the equation of a hyperplane is  $\langle w, x \rangle + b = 0$ .

The idea in SVM regression is to find the *flattest*  $w$  that results in the mapping from  $x \rightarrow y$ . Thus, we minimize the Euclidean norm of  $w$ , i.e.,  $\|w\| = \sqrt{\sum_{j=1}^n w_j^2}$ . We also want to ensure that  $|y_i - f(x_i)| \leq \epsilon, \forall i$ . The objective function (quadratic program) becomes

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 \\ \text{subject to} \quad & y_i - \langle w, x_i \rangle - b \leq \epsilon \\ & -y_i + \langle w, x_i \rangle + b \leq \epsilon \end{aligned}$$

This is a (possibly infeasible) convex optimization problem. Feasibility is obtainable by introducing the slack variables  $(\xi, \xi^*)$ . We choose a constant  $C$  that scales the degree of infeasibility. The model is then modified to be as follows:

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi_i \\ & -y_i + \langle w, x_i \rangle + b \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

As  $C$  increases, the model increases in sensitivity to infeasibility.

We may tune the objective function by introducing cost functions  $c(\cdot), c^*(\cdot)$ . Then, the objective function becomes

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n [c(\xi_i) + c^*(\xi_i^*)]$$

We may replace the function  $[f(x) - y]$  with a “kernel”  $K(x, y)$  introducing nonlinearity into the problem. The choice of the kernel is a matter of judgment, based on the nature of the application being examined. SVMs allow many different estimation kernels, e.g., the Radial Basis function kernel minimizes the distance between inputs ( $x$ ) and targets ( $y$ ) based on

$$f(x, y; \gamma) = \exp(-\gamma|x - y|^2)$$

where  $\gamma$  is a user-defined squashing parameter.

There are various SVM packages that are easily obtained in open-source. An easy-to-use one is SVM Light—the package is available at

the following URL: <http://svmlight.joachims.org/>. SVM Light is an implementation of Vapnik's Support Vector Machine for the problem of pattern recognition. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently. The algorithm proceeds by solving a sequence of optimization problems, lower-bounding the solution using a form of local search. It is based on work by Joachims (1999).

Another program is the University of London SVM. Interestingly, it is known as SVM Dark—evidently people who like hyperplanes have a sense of humor! See <http://www.cs.ucl.ac.uk/staff/M.Sewell/svmdark/>. For a nice list of SVMs, see <http://www.cs.ubc.ca/~murphyk/Software/svm.htm>. In R, see the machine learning library e1071—the function is, of course, called `svm`.

As an example, let's use the `svm` function to analyze the same flower data set that we used with naive Bayes.

```
#USING SVMs
> res = svm(iris[,1:4], iris[,5])
> out = table(predict(res, iris[,1:4]), iris[,5])
> print(out)

 setosa versicolor virginica
setosa 50 0 0
versicolor 0 48 2
virginica 0 2 48
```

SVMs are very fast and are quite generally applicable with many types of kernels. Hence, they may also be widely applied in news analytics.

### 7.5.3 Word Count Classifiers

The simplest form of classifier is based on counting words that are of *signed* type. Words are the heart of any language inference system, and in a specialized domain, this is even more so. In the words of F.C. Bartlett,

"Words ... can indicate the qualitative and relational features of a situation in their general aspect just as directly as, and perhaps even more satisfactorily than, they can describe its particular individuality. This is, in fact, what gives to language its intimate relation to thought processes."

To build a word-count classifier a user defines a *lexicon* of special words that relate to the classification problem. For example, if the classifier is categorizing text into optimistic versus pessimistic economic news, then the user may want to create a lexicon of words that are useful in separating the good news from bad. For example, the word "upbeat" might be signed as optimistic, and the word "dismal" may be pessimistic. In my experience, a good lexicon needs about 300–500 words. Domain knowledge is brought to bear in designing a lexicon. Therefore, in contrast to the Bayes classifier, a word-count algorithm is language-dependent.

This algorithm is based on a simple word count of lexical words. If the number of words in a particular category exceeds that of the other categories by some threshold then the text message is categorized to the category with the highest lexical count. The algorithm is of very low complexity, extremely fast, and easy to implement. It delivers a baseline approach to the classification problem.

#### 7.5.4 Vector Distance Classifier

This algorithm treats each message as a word vector. Therefore, each pre-classified, hand-tagged text message in the training corpus becomes a comparison vector—we call this set the rule set. Each message in the test set is then compared to the rule set and is assigned a classification based on which rule comes closest in vector space.

The angle between the message vector ( $M$ ) and the vectors in the rule set ( $S$ ) provides a measure of proximity.

$$\cos(\theta) = \frac{M \cdot S}{\|M\| \cdot \|S\|}$$

where  $\|A\|$  denotes the norm of vector  $A$ . Variations on this theme are made possible by using sets of top- $n$  closest rules, rather than only the closest rule.

Word vectors here are extremely sparse, and the algorithms may be built to take the dot product and norm above very rapidly. This algorithm was used in [Das and Chen \(2007\)](#) and was taken directly from

ideas used by search engines. The analogy is almost exact. A search engine essentially indexes pages by representing the text as a word vector. When a search query is presented, the vector distance  $\cos(\theta) \in (0, 1)$  is computed for the search query with all indexed pages to find the pages with which the angle is the least, i.e., where  $\cos(\theta)$  is the greatest. Sorting all indexed pages by their angle with the search query delivers the best-match ordered list. Readers will remember in the early days of search engines how the list of search responses also provided a percentage number along with the returned results—these numbers were the same as the value of  $\cos(\theta)$ .

When using the vector distance classifier for news analytics, the classification algorithm takes the new text sample and computes the angle of the message with all the text pages in the indexes training corpus to find the best matches. It then classifies pages with the same tag as the best matches. This classifier is also very easy to implement as it only needs simple linear algebra functions and sorting routines that are widely available in almost any programming environment.

### 7.5.5 Discriminant-Based Classifier

All the classifiers discussed above do not weight words differentially in a continuous manner. Either they do not weight them at all, as in the case of the Bayes classifier or the SVM, or they focus on only some words, ignoring the rest, as with the word count classifier. In contrast the discriminant-based classifier weights words based on their discriminant value.

The commonly used tool here is Fisher's discriminant. Various implementations of it, with minor changes in form are used. In the classification area, one of the earliest uses was in the Bow algorithm of McCalum (1996), which reports the discriminant values; Chakrabarti, Dom, Agrawal and Raghavan (1998) also use it in their classification framework, as do Das and Chen (2007). We present one version of Fisher's discriminant here.

Let the mean score (average number of times word  $w$  appears in a text message of category  $i$ ) of each term for each category =  $\mu_i$ , where  $i$  indexes category. Let text messages be indexed by  $j$ . The number of times word  $w$  appears in a message  $j$  of category  $i$  is denoted  $m_{ij}$ . Let  $n_i$  be the number of times word  $w$  appears in category  $i$ . Then the discriminant

function might be expressed as:

$$F(w) = \frac{\frac{1}{|C|} \sum_{i \neq k} (\mu_i - \mu_k)^2}{\sum_i \frac{1}{n_i} \sum_j (m_{ij} - \mu_i)^2}$$

It is the ratio of the across-class (class  $i$  vs class  $k$ ) variance to the average of within-class (class  $i \in C$ ) variances. To get some intuition, consider the case we looked at earlier, classifying the economic sentiment as optimistic or pessimistic. If the word “dismal” appears exactly once in text that is pessimistic and never appears in text that is optimistic, then the within-class variation is zero, and the across-class variation is positive. In such a case, where the denominator of the equation above is zero, the word “dismal” is an infinitely-powerful discriminant. It should be given a very large weight in any word-count algorithm.

In [Das and Chen \(2007\)](#) we looked at stock message-board text and determined good discriminants using the Fisher metric. Here are some words that showed high discriminant values (with values alongside) in classifying optimistic versus pessimistic opinions.

```
bad 0.0405
hot 0.0161
hype 0.0089
improve 0.0123
joke 0.0268
jump 0.0106
killed 0.0160
lead 0.0037
like 0.0037
long 0.0162
lose 0.1211
money 0.1537
overvalue 0.0160
own 0.0031
good_n 0.0485
```

The last word in the list (“not good”) is an example of a negated word showing a higher discriminant value than the word itself without a negative connotation (recall the discussion of negative tagging earlier in Section [7.3.2](#)). Also see that the word “bad” has a score of 0.0405, whereas the term “not good” has a higher score of 0.0485. This is an example

where the structure and usage of language, not just the meaning of a word, matters.

In another example, using the Bow algorithm this time, examining a database of conference calls with analysts, the best 20 discriminant words were:

```
0.030828516377649325 allowing
0.094412331406551059 november
0.044315992292870907 determined
0.225433526011560692 general
0.034682080924855488 seasonality
0.123314065510597301 expanded
0.017341040462427744 rely
0.071290944123314062 counsel
0.044315992292870907 told
0.015414258188824663 easier
0.050096339113680152 drop
0.028901734104046242 synergies
0.025048169556840076 piece
0.021194605009633910 expenditure
0.017341040462427744 requirement
0.090558766859344900 prospects
0.019267822736030827 internationally
0.017341040462427744 proper
0.026974951830443159 derived
0.001926782273603083 invited
```

Not all these words would obviously connote bullishness or bearishness, but some of them certainly do, such as “expanded”, “drop”, “prospects”, etc. Why apparently unrelated words appear as good discriminants is useful to investigate, and may lead to additional insights.

### 7.5.6 Adjective-Adverb Classifier

Classifiers may use all the text, as in the Bayes and vector-distance classifiers, or a subset of the text, as in the word-count algorithm. They may also weight words differentially as in discriminant-based word counts. Another way to filter words in a word-count algorithm is to focus on the segments of text that have high emphasis, i.e., in regions around adjectives and adverbs. This is done in [Das and Chen \(2007\)](#) using an

adjective-adverb search to determine these regions.

This algorithm is language-dependent. In order to determine the adjectives and adverbs in the text, parsing is required, and calls for the use of a dictionary. The one I have used extensively is the CUVOALD ((Computer Usable Version of the Oxford Advanced Learner's Dictionary). It contains parts-of-speech tagging information, and makes the parsing process very simple. There are other sources—a very well-known one is WordNet from <http://wordnet.princeton.edu/>.

Using these dictionaries, it is easy to build programs that only extract the regions of text around adjectives and adverbs, and then submit these to the other classifiers for analysis and classification. Counting adjectives and adverbs may also be used to score news text for “emphasis” thereby enabling a different qualitative metric of importance for the text.

### 7.5.7 Scoring Optimism and Pessimism

A very useful resource for scoring text is the General Inquirer,

<http://www.wjh.harvard.edu/~inquirer/>, housed at Harvard University. The Inquirer allows the user to assign “flavors” to words so as to score text. In our case, we may be interested in counting optimistic and pessimistic words in text. The Inquirer will do this online if needed, but the dictionary may be downloaded and used offline as well. Words are tagged with attributes that may be easily used to undertake tagged word counts.

Here is a sample of tagged words from the dictionary that gives a flavor of its structure:

```
ABNORMAL H4Lvd Neg Ngtv Vice NEGAFF Modif |
ABOARD H4Lvd Space PREP LY |
ABOLITION Lvd TRANS Noun
ABOMINABLE H4 Neg Strng Vice Ovrst Eval IndAdj Modif |
ABORTIVE Lvd POWOTH POWTOT Modif POLIT
ABOUND H4 Pos Psv Incr IAV SUPV |
```

The words ABNORMAL and ABOMINABLE have “Neg” tags and the word ABOUND has a “Pos” tag.

Das and Chen (2007) used this dictionary to create an ambiguity score for segmenting and filtering messages by optimism/pessimism in testing news analytical algorithms. They found that algorithms performed better after filtering in less ambiguous text. This ambiguity score is dis-

cussed later in Section 7.5.9.

Tetlock (2007) is the best example of the use of the General Inquirer in finance. Using text from the "Abreast of the Market" column from the Wall Street Journal he undertook a principal components analysis of 77 categories from the GI and constructed a media pessimism score. High pessimism presages lower stock prices, and extreme positive or negative pessimism predicts volatility. Tetlock, Saar-Tsechansky and Macskassay (2008) use news text related to firm fundamentals to show that negative words are useful in predicting earnings and returns. The potential of this tool has yet to be fully realized, and I expect to see a lot more research undertaken using the General Inquirer.

### 7.5.8 Voting among Classifiers

In Das and Chen (2007) we introduced a voting classifier. Given the highly ambiguous nature of the text being worked with, reducing the noise is a major concern. Pang, Lee and Vaithyanathan (2002) found that standard machine learning techniques do better than humans at classification. Yet, machine learning methods such as naive Bayes, maximum entropy, and support vector machines do not perform as well on sentiment classification as on traditional topic-based categorization.

To mitigate error, classifiers are first separately applied, and then a majority vote is taken across the classifiers to obtain the final category. This approach improves the signal to noise ratio of the classification algorithm.

### 7.5.9 Ambiguity Filters

Suppose we are building a sentiment index from a news feed. As each text message comes in, we apply our algorithms to it and the result is a classification tag. Some messages may be classified very accurately, and others with much lower levels of confidence. Ambiguity-filtering is a process by which we discard messages of high noise and potentially low signal value from inclusion in the aggregate signal (for example, the sentiment index).

One may think of ambiguity-filtering as a sequential voting scheme. Instead of running all classifiers and then looking for a majority vote, we run them sequentially, and discard messages that do not pass the hurdle of more general classifiers, before subjecting them to more particular

ones. In the end, we still have a voting scheme. Ambiguity metrics are therefore lexicographic.

In Das and Chen (2007) we developed an ambiguity filter for application prior to our classification algorithms. We applied the General Inquirer to the training data to determine an “optimism” score. We computed this for each category of stock message type, i.e., buy, hold, and sell. For each type, we computed the mean optimism score, amounting to 0.032, 0.026, 0.016, respectively, resulting in the expected rank ordering (the standard deviations around these means are 0.075, 0.069, 0.071, respectively). We then filtered messages in based on how far they were away from the mean in the right direction. For example, for buy messages, we chose for classification only those with one standard-deviation higher than the mean. False positives in classification decline dramatically with the application of this ambiguity filter.

## 7.6 Metrics

Developing analytics without metrics is insufficient. It is important to build measures that examine whether the analytics are generating classifications that are statistically significant, economically useful, and stable. For an analytic to be *statistically valid*, it should meet some criterion that signifies classification accuracy and power. Being *economically useful* sets a different bar—does it make money? And *stability* is a double-edged quality: one, does it perform well in-sample and out-of-sample? And two, is the behavior of the algorithm stable across training corpora?

Here, we explore some of the metrics that have been developed, and propose others. No doubt, as the range of analytics grows, so will the range of metrics.

### 7.6.1 Confusion Matrix

The confusion matrix is the classic tool for assessing classification accuracy. Given  $n$  categories, the matrix is of dimension  $n \times n$ . The rows relate to the category assigned by the analytic algorithm and the columns refer to the correct category in which the text resides. Each cell  $(i, j)$  of the matrix contains the number of text messages that were of type  $j$  and were classified as type  $i$ . The cells on the diagonal of the confusion matrix state the number of times the algorithm got the classification right. All other cells are instances of classification error. If an algorithm has no

classification ability, then the rows and columns of the matrix will be independent of each other. Under this null hypothesis, the statistic that is examined for rejection is as follows:

$$\chi^2[dof = (n - 1)^2] = \sum_{i=1}^n \sum_{j=1}^n \frac{[A(i, j) - E(i, j)]^2}{E(i, j)}$$

where  $A(i, j)$  are the actual numbers observed in the confusion matrix, and  $E(i, j)$  are the expected numbers, assuming no classification ability under the null. If  $T(i)$  represents the total across row  $i$  of the confusion matrix, and  $T(j)$  the column total, then

$$E(i, j) = \frac{T(i) \times T(j)}{\sum_{i=1}^n T(i)} \equiv \frac{T(i) \times T(j)}{\sum_{j=1}^n T(j)}$$

The degrees of freedom of the  $\chi^2$  statistic is  $(n - 1)^2$ . This statistic is very easy to implement and may be applied to models for any  $n$ . A highly significant statistic is evidence of classification ability.

### 7.6.2 Precision and Recall

The creation of the confusion matrix leads naturally to two measures that are associated with it.

Precision is the fraction of positives identified that are truly positive, and is also known as positive predictive value. It is a measure of usefulness of prediction. So if the algorithm (say) was tasked with selecting those account holders on LinkedIn who are actually looking for a job, and it identifies  $n$  such people of which only  $m$  were really looking for a job, then the precision would be  $m/n$ .

Recall is the proportion of positives that are correctly identified, and is also known as sensitivity. It is a measure of how complete the prediction is. If the actual number of people looking for a job on LinkedIn was  $M$ , then recall would be  $n/M$ .

For example, suppose we have the following confusion matrix.

Predicted	Actual		
	Looking for Job	Not Looking	
Looking for Job	10	2	12
Not Looking	1	16	17
	11	18	29

In this case precision is 10/12 and recall is 10/11. Precision is related to the probability of false positives (Type I error), which is one minus precision. Recall is related to the probability of false negatives (Type II error), which is one minus recall.

### 7.6.3 Accuracy

Algorithm accuracy over a classification scheme is the percentage of text that is correctly classified. This may be done in-sample or out-of-sample. To compute this off the confusion matrix, we calculate

$$\text{Accuracy} = \frac{\sum_{i=1}^n A(i,i)}{\sum_{j=1}^n T(j)}$$

We should hope that this is at least greater than  $1/n$ , which is the accuracy level achieved on average from random guessing. In practice, I find that accuracy ratios of 60–70% are reasonable for text that is non-factual and contains poor language and opinions.

### 7.6.4 False Positives

Improper classification is worse than a failure to classify. In a  $2 \times 2$  (two category,  $n = 2$ ) scheme, every off-diagonal element in the confusion matrix is a false positive. When  $n > 2$ , some classification errors are worse than others. For example in a 3-way buy, hold, sell scheme, where we have stock text for classification, classifying a buy as a sell is worse than classifying it as a hold. In this sense an ordering of categories is useful so that a false classification into a near category is not as bad as a wrong classification into a far (diametrically opposed) category.

The percentage of false positives is a useful metric to work with. It may be calculated as a simple count or as a weighted count (by nearness of wrong category) of false classifications divided by total classifications undertaken.

In our experiments on stock messages in Das and Chen (2007), we found that the false positive rate for the voting scheme classifier was about 10%. This was reduced to below half that number after application of an ambiguity filter (discussed in Section 7.5.9) based on the General Inquirer.

### 7.6.5 Sentiment Error

When many articles of text are classified, an aggregate measure of sentiment may be computed. Aggregation is useful because it allows classification errors to cancel—if a buy was mistaken as a sell, and another sell as a buy, then the aggregate sentiment index is unaffected.

Sentiment error is the percentage difference between the computed aggregate sentiment, and the value we would obtain if there were no classification error. In our experiments this varied from 5-15% across the data sets that we used. [Leinweber and Sisk \(2010\)](#) show that sentiment aggregation gives a better relation between news and stock returns.

### 7.6.6 Disagreement

In [Das, Martinez-Jerez and Tufano \(2005\)](#) we introduced a disagreement metric that allows us to gauge the level of conflict in the discussion. Looking at stock text messages, we used the number of signed buys and sells in the day (based on a sentiment model) to determine how much disagreement of opinion there was in the market. The metric is computed as follows:

$$\text{DISAG} = \left| 1 - \left| \frac{B - S}{B + S} \right| \right|$$

where  $B, S$  are the numbers of classified buys and sells. Note that DISAG is bounded between zero and one. The quality of aggregate sentiment tends to be lower when DISAG is high.

### 7.6.7 Correlations

A natural question that arises when examining streaming news is: how well does the sentiment from news correlate with financial time series? Is there predictability? An excellent discussion of these matters is provided in [Leinweber and Sisk \(2010\)](#). They specifically examine investment signals derived from news.

In their paper, they show that there is a significant difference in cumulative excess returns between strong positive sentiment and strong negative sentiment days over prediction horizons of a week or a quarter. Hence, these event studies are based on point-in-time correlation triggers. Their results are robust across countries.

The simplest correlation metrics are visual. In a trading day, we may plot the movement of a stock series, alongside the cumulative sentiment

series. The latter is generated by taking all classified ‘buys’ as +1 and ‘sells’ as -1, and the plot comprises the cumulative total of scores of the messages (‘hold’ classified messages are scored with value zero). See Figure 7.8 for one example, where it is easy to see that the sentiment and stock series track each other quite closely. We coin the term “sents” for the units of sentiment.

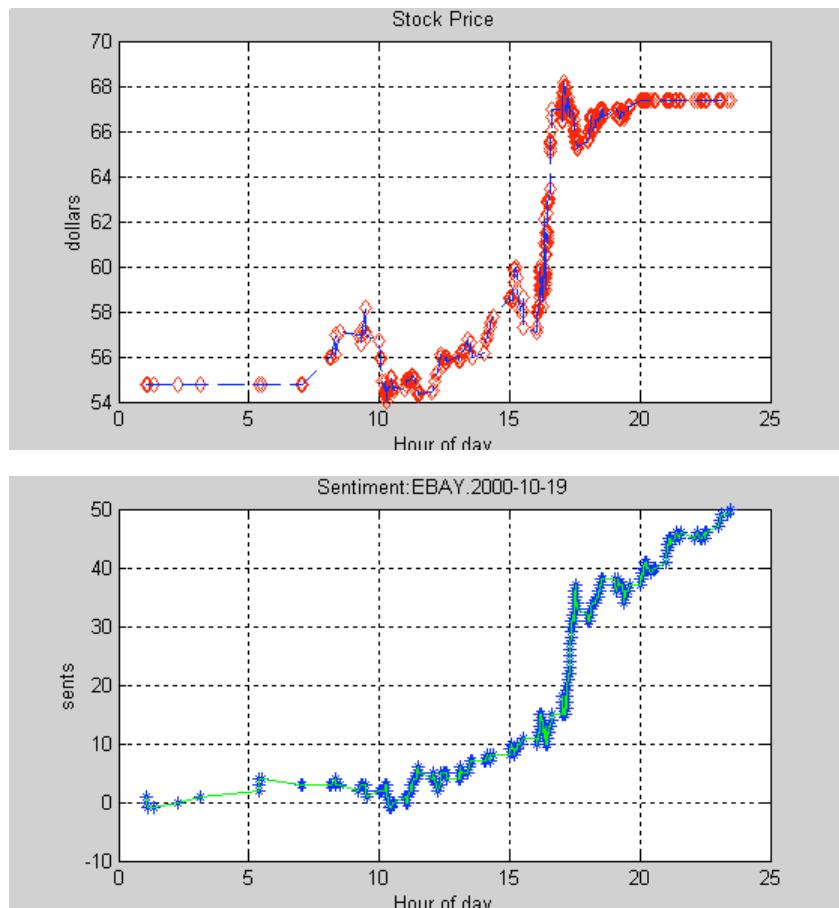


Figure 7.8: Plot of stock series (upper graph) versus sentiment series (lower graph). The correlation between the series is high. The plot is based on messages from Yahoo! Finance and is for a single twenty-four hour period.

### 7.6.8 Aggregation Performance

As pointed out in Leinweber and Sisk (2010) aggregation of classified news reduces noise and improves signal accuracy. One way to measure this is to look at the correlations of sentiment and stocks for aggregated versus disaggregated data. As an example, I examine daily sentiment for individual stocks and an index created by aggregating sentiment across stocks, i.e., a cross-section of sentiment. This is useful to examine

whether sentiment aggregates effectively in the cross-section.

I used all messages posted for 35 stocks that comprise the Morgan Stanley High-Tech Index (MSH35) for the period June 1 to August 27, 2001. This results in 88 calendar days and 397,625 messages, an average of about 4,500 messages per day. For each day I determine the sentiment and stock return. Daily sentiment uses messages up to 4 pm on each trading day, coinciding with the stock return close.

Ticker	Correlations of SENTY4pm(t) with		
	STKRET(t)	STKRET(t+1)	STKRET(t-1)
ADP	0.086	0.138	-0.062
AMAT	-0.008	-0.049	0.067
AMZN	0.227	0.167	0.161
AOL	0.386	-0.010	0.281
BRCM	0.056	0.167	-0.007
CA	0.023	0.127	0.035
CPQ	0.260	0.161	0.239
CSCO	0.117	0.074	-0.025
DELL	0.493	-0.024	0.011
EDS	-0.017	0.000	-0.078
EMC	0.111	0.010	0.193
ERTS	0.114	-0.223	0.225
HWP	0.315	-0.097	-0.114
IBM	0.071	-0.057	0.146
INTC	0.128	-0.077	-0.007
INTU	-0.124	-0.099	-0.117
JDSU	0.126	0.056	0.047
JNPR	0.416	0.090	-0.137
LU	0.602	0.131	-0.027
MOT	-0.041	-0.014	-0.006
MSFT	0.422	0.084	0.210
MU	0.110	-0.087	0.030
NT	0.320	0.068	0.288
ORCL	0.005	0.056	-0.062
PALM	0.509	0.156	0.085
PMTC	0.080	0.005	-0.030
PSFT	0.244	-0.094	0.270
SCMR	0.240	0.197	0.060
SLR	-0.077	-0.054	-0.158
STM	-0.010	-0.062	0.161
SUNW	0.463	0.176	0.276
TLAB	0.225	0.250	0.283
TXN	0.240	-0.052	0.117
XLNX	0.261	-0.051	-0.217
YHOO	0.202	-0.038	0.222
Average correlation across 35 stocks			
	0.188	0.029	0.067
Correlation between 35 stock index and 35 stock sentiment index			
	<b>0.486</b>	0.178	0.288

Table 7.1: Correlations of Sentiment and Stock Returns for the MSH35 stocks and the aggregated MSH35 index. Stock returns (STKRET) are computed from close-to-close. We compute correlations using data for 88 days in the months of June, July and August 2001. Return data over the weekend is linearly interpolated, as messages continue to be posted over weekends. Daily sentiment is computed from midnight to close of trading at 4 pm (SENTY4pm).

I also compute the average sentiment index of all 35 stocks, i.e., a proxy for the MSH35 sentiment. The corresponding equally weighted return of 35 stocks is also computed. These two time series permit an examination of the relationship between sentiment and stock returns at the aggregate index level. Table 7.1 presents the correlations between individual stock returns and sentiment, and between the MSH35 index return and MSH35 sentiment. We notice that there is positive contemporaneous correlation between most stock returns and sentiment. The correlations were sometimes as high as 0.60 (for Lucent), 0.51 (PALM)

and 0.49 (DELL). Only six stocks evidenced negative correlations, mostly small in magnitude. The average contemporaneous correlation is 0.188, which suggests that sentiment tracks stock returns in the high-tech sector. (I also used full-day sentiment instead of only that till trading close and the results are almost the same—the correlations are in fact higher, as sentiment includes reactions to trading after the close).

Average correlations for individual stocks are weaker when one lag (0.067) or lead (0.029) of the stock return are considered. More interesting is the average index of sentiment for all 35 stocks. The contemporaneous correlation of this index to the equally-weighted return index is as high as 0.486. Here, cross-sectional aggregation helps in eliminating some of the idiosyncratic noise, and makes the positive relationship between returns and sentiment salient. This is also reflected in the strong positive correlation of sentiment to lagged stock returns (0.288) and leading returns (0.178). I confirmed the statistical contemporaneous relationship of returns to sentiment by regressing returns on sentiment (T-statistics in brackets):

$$\begin{aligned} STKRET(t) &= -0.1791 + 0.3866 SENTY(t), \quad R^2 = 0.24 \\ &\quad (0.93) \quad (5.16) \end{aligned}$$

### 7.6.9 Phase-Lag Metrics

Correlation across sentiment and return time series is a special case of lead-lag analysis. This may be generalized to looking for pattern correlations. As may be evident from Figure 7.8, the stock and sentiment plots have patterns. In the figure they appear contemporaneous, though the sentiment series lags the stock series.

A graphical approach to lead-lag analysis is to look for graph patterns across two series and to examine whether we may predict the patterns in one time series with the other. For example, can we use the sentiment series to predict the high point of the stock series, or the low point? In other words, is it possible to use the sentiment data generated from algorithms to pick turning points in stock series? We call this type of graphical examination “phase-lag” analysis.

A simple approach I came up with involves decomposing graphs into eight types—see Figure 7.9. On the left side of the figure, notice that there are eight patterns of graphs based on the location of four salient graph features: start, end, high, and low points. There are exactly eight

possible graph patterns that may be generated from all positions of these four salient points. It is also very easy to write software to take any time series—say, for a trading day—and assign it to one of the patterns, keeping track of the position of the maximum and minimum points. It is then possible to compare two graphs to see which one predicts the other in terms of pattern. For example, does the sentiment series maximum come before that of the stock series? If so, how much earlier does it detect the turning point on average?

Using data from several stocks I examined whether the sentiment graph pattern generated from a voting classification algorithm was predictive of stock graph patterns. Phase-lags were examined in intervals of five minutes through the trading day. The histogram of leads and lags is shown on the right-hand side of Figure 7.9. A positive value denotes that the sentiment series lags the stock series; a negative value signifies that the stock series lags sentiment. It is apparent from the histogram that the sentiment series lags stocks, and is not predictive of stock movements in this case.

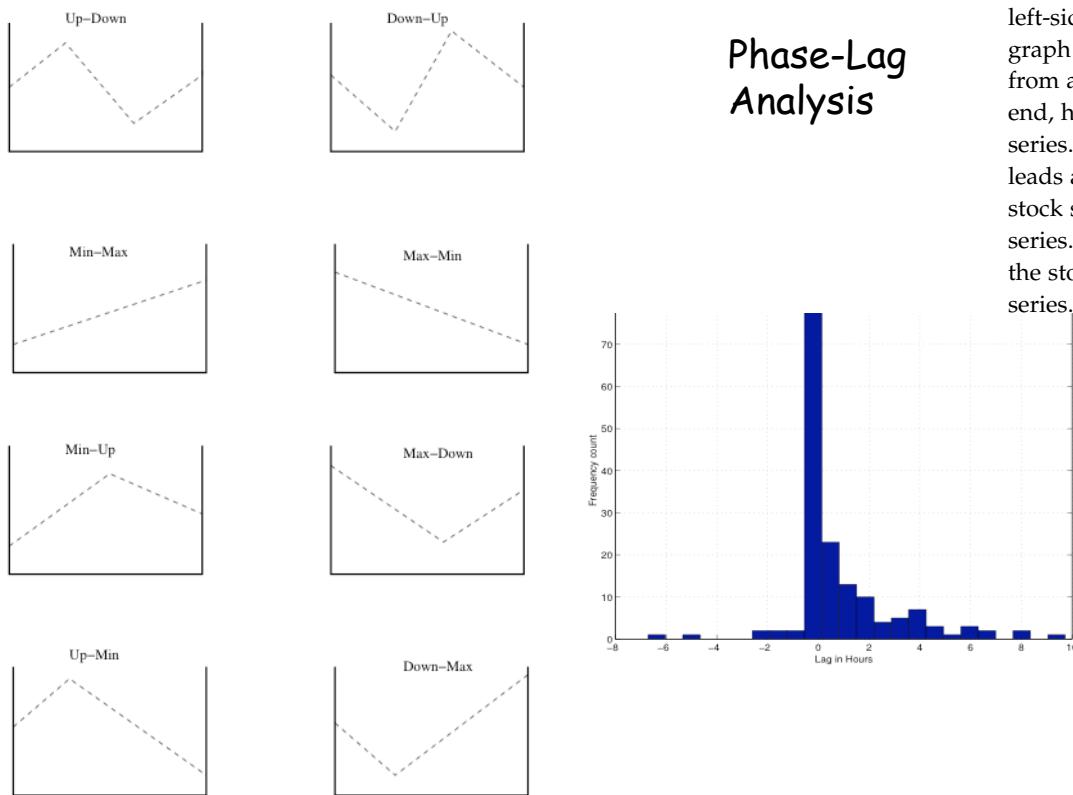


Figure 7.9: Phase-lag analysis. The left-side shows the eight canonical graph patterns that are derived from arrangements of the start, end, high, and low points of a time series. The right-side shows the leads and lags of patterns of the stock series versus the sentiment series. A positive value means that the stock series leads the sentiment series.

### 7.6.10 Economic Significance

News analytics may be evaluated using economic yardsticks. Does the algorithm deliver profitable opportunities? Does it help reduce risk?

For example, in [Das and Sisk \(2005\)](#) we formed a network with connections based on commonality of handles in online discussion. We detected communities using a simple rule based on connectedness beyond a chosen threshold level, and separated all stock nodes into either one giant community or into a community of individual singleton nodes. We then examined the properties of portfolios formed from the community versus those formed from the singleton stocks.

We obtained several insights. We calculated the mean returns from an equally-weighted portfolio of the community stocks and an equally-weighted portfolio of singleton stocks. We also calculated the return standard deviations of these portfolios. We did this month-by-month for sixteen months. In fifteen of the sixteen months the mean returns were higher for the community portfolio; the standard deviations were lower in thirteen of the sixteen months. The difference of means was significant for thirteen of those months as well. Hence, community detection based on news traffic leads to identifying a set of stocks that performs vastly better than the rest.

There is much more to be done in this domain of economic metrics for the performance of news analytics. [Leinweber and Sisk \(2010\)](#) have shown that there is exploitable alpha in news streams. The risk management and credit analysis areas also offer economic metrics that may be used to validate news analytics.

## 7.7 Grading Text

In recent years, the SAT exams added a new essay section. While the test aimed at assessing original writing, it also introduced automated grading. A goal of the test is to assess the writing level of the student. This is associated with the notion of *readability*.

“Readability” is a metric of how easy it is to comprehend text. Given a goal of efficient markets, regulators want to foster transparency by making sure financial documents that are disseminated to the investing public are readable. Hence, metrics for readability are very important and are recently gaining traction.

[Gunning \(1952\)](#) developed the Fog index. The index estimates the

years of formal education needed to understand text on a first reading. A fog index of 12 requires the reading level of a U.S. high school senior (around 18 years old). The index is based on the idea that poor readability is associated with longer sentences and complex words. Complex words are those that have more than two syllables. The formula for the Fog index is

$$0.4 \cdot \left[ \frac{\#words}{\#sentences} + 100 \cdot \left( \frac{\#\text{complex words}}{\#words} \right) \right]$$

Alternative readability scores use similar ideas. The Flesch Reading Ease Score and the Flesch-Kincaid Grade Level also use counts of words, syllables, and sentences.<sup>2</sup> The Flesch Reading Ease Score is defined as

$$206.835 - 1.015 \left( \frac{\#words}{\#sentences} \right) - 84.6 \left( \frac{\#\text{syllables}}{\#words} \right)$$

With a range of 90-100 easily accessible by a 11-year old, 60-70 being easy to understand for 13-15 year olds, and 0-30 for university graduates.

The Flesch-Kincaid Grade Level is defined as

$$0.39 \left( \frac{\#words}{\#sentences} \right) + 11.8 \left( \frac{\#\text{syllables}}{\#words} \right) - 15.59$$

which gives a number that corresponds to the grade level. As expected these two measures are negatively correlated. Various other measures of readability use the same ideas as in the Fog index. For example the Coleman and Liau (1975) index does not even require a count of syllables, as follows:

$$CLI = 0.0588L - 0.296S - 15.8$$

where  $L$  is the average number of letters per hundred words and  $S$  is the average number of sentences per hundred words.

Standard readability metrics may not work well for financial text. Loughran and McDonald (2014) find that the Fog index is inferior to simply looking at 10-K file size.

## 7.8 Text Summarization

It has become fairly easy to summarize text using statistical methods. The simplest form of text summarizer works on a sentence-based model that sorts sentences in a document in descending order of word overlap with all other sentences in the text. The re-ordering of sentences arranges the document with the sentence that has most overlap with others first, then the next, and so on.

<sup>2</sup> See [http://en.wikipedia.org/wiki/Flesch-Kincaid\\_readability\\_tests](http://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests).

An article  $D$  may have  $m$  sentences  $s_i, i = 1, 2, \dots, m$ , where each  $s_i$  is a set of words. We compute the pairwise overlap between sentences using the <sup>3</sup> similarity index:

3

$$J_{ij} = J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} = J_{ji} \quad (7.9)$$

The overlap is the ratio of the size of the intersection of the two word sets in sentences  $s_i$  and  $s_j$ , divided by the size of the union of the two sets. The similarity score of each sentence is computed as the row sums of the Jaccard similarity matrix.

$$\mathcal{S}_i = \sum_{j=1}^m J_{ij} \quad (7.10)$$

Once the row sums are obtained, they are sorted and the summary is the first  $n$  sentences based on the  $\mathcal{S}_i$  values. We can then decide how many sentences we want in the summary.

Another approach to using row sums is to compute centrality using the Jaccard matrix  $J$ , and then pick the  $n$  sentences with the highest centrality scores.

We illustrate the approach with a news article from the financial markets. The sample text is taken from Bloomberg on April 21, 2014, at the following URL:

<http://www.bloomberg.com/news/print/2014-04-21/wall-street-bond-dealers-whipsawed-on-bearish-treasuries-bet-1-.html>. The full text spans 4 pages and is presented in an appendix to this chapter.

This article is read using a web scraper (as seen in preceding sections), and converted into a text file with a separate line for each sentence. We call this file `summary_text.txt` and this file is then read into R and processed with the following parsimonious program code. We first develop the summarizer function.

```
FUNCTION TO RETURN n SENTENCE SUMMARY
Input: array of sentences (text)
Output: n most common intersecting sentences
text_summary = function(text, n) {
 m = length(text) # No of sentences in input
 jaccard = matrix(0,m,m) # Store match index
 for (i in 1:m) {
 for (j in i:m) {
 a = text[i]; aa = unlist(strsplit(a, " "))
 b = text[j]; bb = unlist(strsplit(b, " "))
 for (k in 1:(length(aa))) {
 if (aa[k] %in% bb) jaccard[i,j] = jaccard[i,j] + 1
 }
 }
 }
 S = colSums(jaccard)
 S = sort(S, decreasing = TRUE)
 S = S[1:n]
 S_index = which(S == S)
 summary_sentences = text[S_index]
 return(summary_sentences)
}
```

```

 b = text[j]; bb = unlist(strsplit(b, " "))
 jaccard[i, j] = length(intersect(aa, bb)) /
 length(union(aa, bb))
 jaccard[j, i] = jaccard[i, j]
 }
}
similarity_score = rowSums(jaccard)
res = sort(similarity_score, index.return=TRUE,
 decreasing=TRUE)
idx = res$ix[1:n]
summary = text[idx]
}

```

We now read in the data and clean it into a single text array.

```

url = "dstext_sample.txt" #You can put any text file or URL here
text = read_web_page(url, cstem=0, cstop=0, ccase=0, cpunc=0, cflat=1)
print(length(text[[1]]))

[1] 1

print(text)

[1] "THERE_HAVE BEEN murmurings that we are now in the
"trough of disillusionment" of big data , the hype around it having
surpassed the reality of what it can deliver. Gartner suggested that
the "gravitational pull of big data is now so strong that even people
who haven't got a clue as to what it's all about report that they're running
big data projects." Indeed , their research with business decision
makers suggests that organisations are struggling to get value from
big data . Data scientists were meant
....."

```

Now we break the text into sentences using the period as a delimiter, and invoking the `strsplit` function in the `stringr` package.

```

text2 = strsplit(text, ". ", fixed=TRUE) #Special handling of the period .
text2 = text2[[1]]
print(text2)

[1] "THERE_HAVE BEEN murmurings that we are now in the
"trough of disillusionment" of big data , the hype around it having
surpassed the reality of what it can deliver"
[2] " Gartner suggested that the "gravitational pull of big data is

```

```

now so strong that even people who haven't a clue as to what it is
all about report that they're running big data projects." Indeed,
their research with business decision makers suggests that
organisations are struggling to get value from big data"
[3] "Data scientists were meant to be the answer to this issue"
[4] "Indeed, Hal Varian, Chief Economist at Google famously
joked that "The sexy job in the next 10 years will be statisticians."
He was clearly right as we are now used to hearing that data
scientists are the key to unlocking the value of big data"
.... ...

```

We now call the text summarization function and produce the top five sentences that give the most overlap to all other sentences.

```

res = text_summary(text2, 5)
print(res)

[1] "Gartner suggested that the "gravitational pull of big data is
now so strong that even people who haven't a clue as to what it's
all about report that they're running big data projects." Indeed,
their research with business decision makers suggests that
organisations are struggling to get value from big data"
[2] "The focus on the data scientist often implies a centralized
approach to analytics and decision making; we implicitly assume
that a small team of highly skilled individuals can meet the needs
of the organisation as a whole"
[3] "May be we are investing too much in a relatively small number
of individuals rather than thinking about how we can design
organisations to help us get the most from data assets"
[4] "The problem with a centralized 'IT-style' approach is that it
ignores the human side of the process of considering how people
create and use information i.e"
[5] "Which probably means that data scientists' salaries will need
to take a hit in the process."

```

As we can see, this generates an effective and clear summary of an article that originally had 42 sentences.

## 7.9 Discussion

The various techniques and metrics fall into two broad categories: supervised and unsupervised learning methods. Supervised models use well-specified input variables to the machine-learning algorithm, which then emits a classification. One may think of this as a generalized regression model. In unsupervised learning, there are no explicit input variables but latent ones, e.g., cluster analysis. Most of the news analytics we explored relate to supervised learning, such as the various classification algorithms. This is well-trodden research. It is the domain of unsuper-

vised learning, for example, the community detection algorithms and centrality computation, that have been less explored and are potentially areas of greatest potential going forward.

Classifying news to generate sentiment indicators has been well worked out. This is epitomized in many of the papers in this book. It is the networks on which financial information gets transmitted that have been much less studied, and where I anticipate most of the growth in news analytics to come from. For example, how quickly does good news about a tech company proliferate to other companies? We looked at issues like this in [Das and Sisk \(2005\)](#), discussed earlier, where we assessed whether knowledge of the network might be exploited profitably. Information also travels by word of mouth and these information networks are also open for much further examination—see [Godes, et. al. \(2005\)](#). Inside (not insider) information is also transmitted in venture capital networks where there is evidence now that better connected VCs perform better than unconnected VCs, as shown by [Hochberg, Ljungqvist and Lu \(2007\)](#).

Whether news analytics reside in the broad area of AI or not is under debate. The advent and success of statistical learning theory in real-world applications has moved much of news analytics out of the AI domain into econometrics. There is very little natural language processing (NLP) involved. As future developments shift from text methods to context methods, we may see a return to the AI paradigm. I believe that tools such as [WolframAlpha<sup>TM</sup>](#) will be the basis of context-dependent news analysis.

News analytics will broaden in the toolkit it encompasses. Expect to see greater use of dependency networks and collaborative filtering. We will also see better data visualization techniques such as community views and centrality diagrams. The number of tools keeps on growing. For an almost exhaustive compendium of tools see the book by [Koller \(2009\)](#) titled “Probabilistic Graphical Models.”

In the end, news analytics are just sophisticated methods for data mining. For an interesting look at the top ten algorithms in data mining, see [Wu, et al. \(2008\)](#). This paper discusses the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006.<sup>4</sup> As algorithms improve in speed, they will expand to automated decision-making, replacing human interaction—as noticed in the marriage of news analytics with automated trading, and eventually, a

<sup>4</sup> These algorithms are: C4.5, k-Means, SVM, Apriori, EM, PageRank, Adaboost, kNN, Naive Bayes, and CART.

rebirth of XHAL.

### 7.10 Appendix: Sample text from Bloomberg for summarization

Summarization is one of the major implementations in Big Text applications. When faced with Big Text, there are three important stages through which analytics may proceed: (a) Indexation, (b) Summarization, and (c) Inference. Automatic summarization<sup>5</sup> is a program that reduces text while keeping mostly the salient points, accounting for variables such as length, writing style, and syntax. There are two approaches: (i) Extractive methods selecting a subset of existing words, phrases, or sentences in the original text to form the summary. (ii) Abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original.

The following news article was used to demonstrate text summarization for the application in Section 7.8.

<sup>5</sup> [http://en.wikipedia.org/wiki/Automatic\\_text\\_summarization](http://en.wikipedia.org/wiki/Automatic_text_summarization)

4/21/2014

Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet - Bloomberg



## Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet

By Lisa Abramowicz and Daniel Kruger - Apr 21, 2014

Betting against [U.S. government debt](#) this year is turning out to be a fool's errand. Just ask Wall Street's biggest bond dealers.

While the losses that their economists predicted have yet to materialize, [JPMorgan Chase & Co. \(JPM\)](#), [Citigroup Inc. \(C\)](#) and the 20 other firms that trade with the [Federal Reserve](#) began wagering on a Treasuries selloff last month for the first time since 2011. The strategy was upended as Fed Chair [Janet Yellen](#) signaled she wasn't in a rush to lift [interest rates](#), two weeks after suggesting the opposite at the bank's March 19 meeting.

The surprising resilience of Treasuries has investors re-calibrating forecasts for higher borrowing costs as lackluster job growth and emerging-market turmoil push yields toward 2014 lows. That's also made the business of [trading](#) bonds, once more predictable for dealers when the Fed was buying trillions of dollars of debt to spur the economy, less profitable as new rules limit the risks they can take with their own money.

"You have an uncertain Fed, an uncertain direction of the economy and you've got rates moving," Mark MacQueen, a partner at Sage Advisory Services Ltd., which oversees \$10 billion, said by telephone from Austin, [Texas](#). In the past, "calling the direction of the market and what you should be doing in it was a lot easier than it is today, particularly for the dealers."

[Treasuries \(USGG10YR\)](#) have confounded economists who predicted 10-year yields would approach 3.4 percent by year-end as a strengthening economy prompts the Fed to pare its unprecedented bond buying.

### Caught Short

After surging to a 29-month high of 3.05 percent at the start of the year, yields on the 10-year note have declined and were at 2.72 percent at 7:42 a.m. in [New York](#).

One reason yields have fallen is the U.S. [labor market](#), which has yet to show consistent improvement.

4/21/2014

Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet - Bloomberg

The world's largest economy added [fewer jobs](#) on average in the first three months of the year than in the same period in the prior two years, data compiled by Bloomberg show. At the same time, a slowdown in [China](#) and tensions between Russia and Ukraine boosted demand for the safest assets.

[Wall Street](#) firms known as primary dealers are getting caught short betting against Treasuries.

They collectively amassed \$5.2 billion of wagers in March that would profit if Treasuries fell, the first time they had net short positions on government debt since September 2011, data compiled by the Fed show.

### 'Some Time'

The practice is allowed under the Volcker Rule that limits the types of trades that banks can make with their own money. The wagers may include market-making, which is the business of using the firm's capital to buy and sell securities with customers while profiting on the spread and movement in prices.

While the bets initially paid off after Yellen said on March 19 that the Fed may lift its benchmark rate six months after it stops buying bonds, Treasuries have since rallied as her subsequent comments strengthened the view that policy makers will keep borrowing costs low to support growth.

On March 31, Yellen highlighted inconsistencies in job data and said "considerable slack" in labor markets showed the Fed's accommodative policies will be needed for "some time."

Then, in her first major speech on her policy framework as Fed chair on April 16, Yellen said it will take at least two years for the [U.S. economy](#) to meet the Fed's goals, which determine how quickly the central bank raises rates.

After declining as much as 0.6 percent following Yellen's March 19 comments, Treasuries have recouped all their losses, index data compiled by Bank of America Merrill Lynch show.

### **Yield Forecasts**

"We had that big selloff and the dealers got short then, and then we turned around and the Fed says, 'Whoa, whoa, whoa: it's lower for longer again,'" MacQueen said in an April 15 telephone interview. "The dealers are really worried here. You get really punished if you take a lot of risk."

Economists and strategists around Wall Street are still anticipating that Treasuries will underperform as yields increase, data compiled by Bloomberg show.

While they've ratcheted down their forecasts this year, they predict 10-year yields will increase to 3.36 percent by the end of December. That's more than 0.6 percentage point higher than where yields are

4/21/2014

Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet - Bloomberg

today.

"My forecast is 4 percent," said [Joseph LaVorgna](#), chief U.S. economist at Deutsche Bank AG, a primary dealer. "It may seem like it's really aggressive but it's really not."

LaVorgna, who has the highest estimate among the 66 responses in a Bloomberg survey, said stronger economic data will likely cause investors to sell Treasuries as they anticipate a rate increase from the Fed.

## History Lesson

The U.S. economy will expand 2.7 percent this year from 1.9 percent in 2013, estimates compiled by Bloomberg show. Growth will accelerate 3 percent next year, which would be the fastest in a decade, based on those forecasts.

Dealers used to rely on Treasuries to act as a hedge against their holdings of other types of debt, such as corporate bonds and mortgages. That changed after the credit crisis caused the failure of Lehman Brothers Holdings Inc. in 2008.

They slashed corporate-debt [inventories](#) by 76 percent from the 2007 peak through last March as they sought to comply with higher [capital requirements](#) from the [Basel Committee on Banking Supervision](#) and stockpiled Treasuries instead.

"Being a dealer has changed over the years, and not least because you also have new balance-sheet constraints that you didn't have before," Ira Jersey, an interest-rate strategist at primary dealer [Credit Suisse Group AG \(CSGN\)](#), said in a telephone interview on April 14.

## Almost Guaranteed

While the Fed's decision to [inundate](#) the U.S. economy with more than \$3 trillion of cheap money since 2008 by buying Treasuries and mortgaged-backed bonds bolstered profits as all fixed-income assets rallied, yields are now so low that banks are struggling to make money trading government bonds.

Yields on 10-year notes have remained below 3 percent since January, data compiled by Bloomberg show. In two decades before the credit crisis, average yields topped 6 percent.

Average daily [trading](#) has also dropped to \$551.3 billion in March from an average \$570.2 billion in 2007, even as the outstanding amount of Treasuries has more than doubled since the financial crisis, according data from the Securities Industry and Financial Markets Association.

4/21/2014

Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet - Bloomberg

"During the crisis, the Fed went to great pains to save primary dealers," [Christopher Whalen](#), banker and author of "Inflated: How Money and Debt Built the American Dream," said in a telephone interview. "Now, because of quantitative easing and other dynamics in the market, it's not just treacherous, it's almost a guaranteed loss."

## Trading Revenue

The biggest dealers are seeing their earnings suffer. In the first quarter, five of the six biggest Wall Street firms reported declines in fixed-income trading revenue.

JPMorgan, the biggest U.S. bond underwriter, had a 21 percent decrease from its fixed-income trading business, more than estimates from Moshe Orenbuch, an analyst at Credit Suisse, and Matt Burnell of Wells Fargo & Co.

Citigroup, whose bond-trading results marred the New York-based bank's two prior quarterly earnings, reported a 18 percent decrease in revenue from that business. Credit Suisse, the second-largest Swiss bank, had a 25 percent drop as income from rates and emerging-markets businesses fell. Declines in debt-trading last year prompted the Zurich-based firm to cut more than 100 fixed-income jobs in London and New York.

## Bank Squeeze

Chief Financial Officer [David Mathers](#) said in a Feb. 6 call that Credit Suisse has "reduced the capital in this business materially and we're obviously increasing our electronic trading operations in this area." [Jamie Dimon](#), chief executive officer at JPMorgan, also emphasized the decreased role of humans in the rates-trading business on an April 11 call as the New York-based bank seeks to cut costs.

About 49 percent of U.S. government-debt trading was executed electronically last year, from 31 percent in 2012, a Greenwich Associates survey of institutional money managers showed. That may ultimately lead banks to combine their rates businesses or scale back their roles as primary dealers as firms get squeezed, said Krishna Memani, the New York-based chief investment officer of OppenheimerFunds Inc., which oversees \$79.1 billion in fixed-income assets.

"If capital requirements were not as onerous as they are now, maybe they could have found a way of making it work, but they aren't as such," he said in a telephone interview.

To contact the reporters on this story: Lisa Abramowicz in New York at [labramowicz@bloomberg.net](mailto:labramowicz@bloomberg.net); Daniel Kruger in New York at [dkruger1@bloomberg.net](mailto:dkruger1@bloomberg.net)

To contact the editors responsible for this story: Dave Liedtka at [dliedtka@bloomberg.net](mailto:dliedtka@bloomberg.net) Michael



# 8

## *Virulent Products: The Bass Model*

### *8.1 Introduction*

The Bass (1969) product diffusion model is a classic one in the marketing literature. It has been successfully used to predict the market shares of various newly introduced products, as well as mature ones.

The main idea of the model is that the adoption rate of a product comes from two sources:

1. The propensity of consumers to adopt the product independent of social influences to do so.
2. The additional propensity to adopt the product because others have adopted it. Hence, at some point in the life cycle of a good product, social contagion, i.e. the influence of the early adopters becomes sufficiently strong so as to drive many others to adopt the product as well. It may be going too far to think of this as a “network” effect, because Frank Bass did this work well before the concept of network effect was introduced, but essentially that is what it is.

The Bass model shows how the information of the first few periods of sales data may be used to develop a fairly good forecast of future sales. One can easily see that whereas this model came from the domain of marketing, it may just as easily be used to model forecasts of cashflows to determine the value of a start-up company.

### *8.2 Historical Examples*

There are some classic examples from the literature of the Bass model providing a very good forecast of the ramp up in product adoption as a function of the two sources described above. See for example the actual

versus predicted market growth for VCRs in the 80s shown in Figure 8.1. Correspondingly, Figure 8.2 shows the adoption of answering machines.

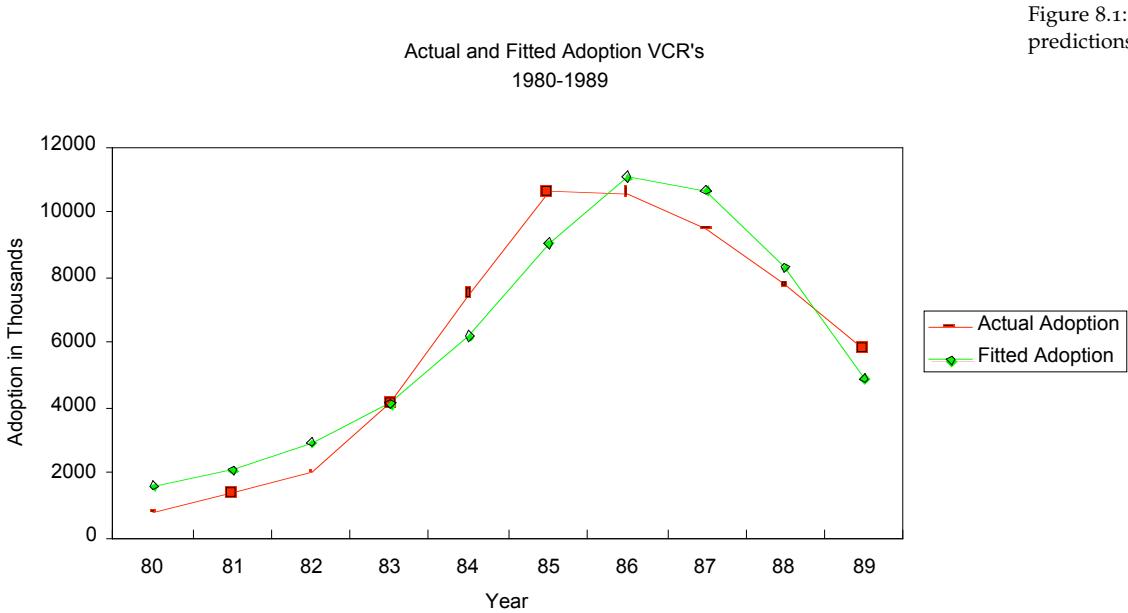


Figure 8.1: Actual versus Bass model predictions for VCRs.

### 8.3 The Basic Idea

We follow the exposition in Bass (1969).

Define the cumulative probability of purchase of a product from time zero to time  $t$  by a single individual as  $F(t)$ . Then, the probability of purchase at time  $t$  is the density function  $f(t) = F'(t)$ .

The rate of purchase at time  $t$ , given no purchase so far, logically follows, i.e.

$$\frac{f(t)}{1 - F(t)}.$$

Modeling this is just like modeling the adoption rate of the product at a given time  $t$ .

Bass (1969) suggested that this adoption rate be defined as

$$\frac{f(t)}{1 - F(t)} = p + q F(t).$$

where we may think of  $p$  as defining the *independent rate* of a consumer adopting the product, and  $q$  as the *imitation rate*, because it modulates the impact from the cumulative intensity of adoption,  $F(t)$ .

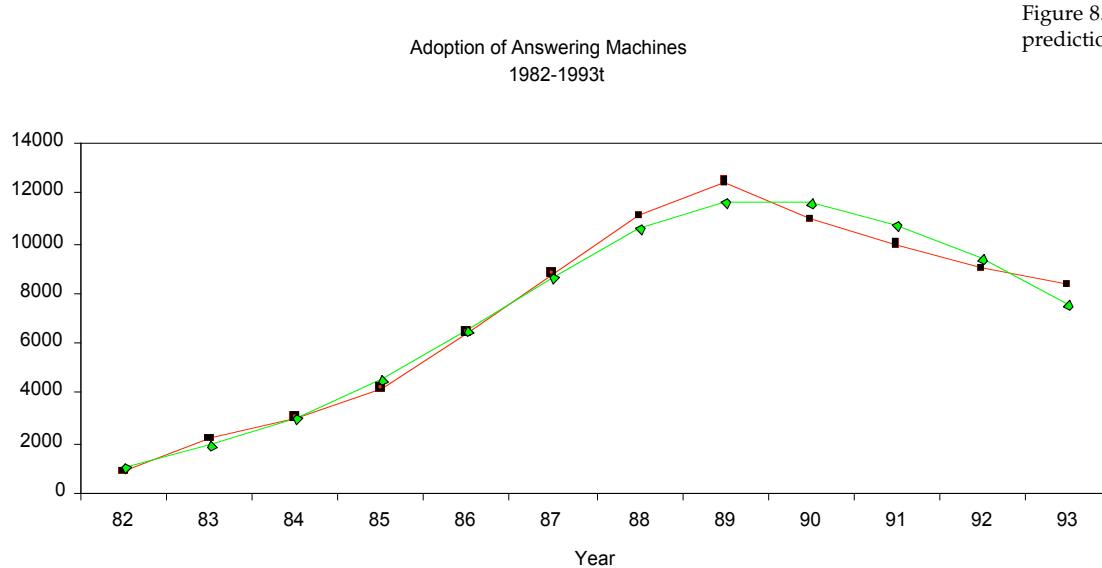


Figure 8.2: Actual versus Bass model predictions for answering machines.

Hence, if we can find  $p$  and  $q$  for a product, we can forecast its adoption over time, and thereby generate a time path of sales. To summarize:

- $p$ : coefficient of innovation.
- $q$ : coefficient of imitation.

#### 8.4 Solving the Model

We rewrite the Bass equation:

$$\frac{dF/dt}{1-F} = p + q F.$$

and note that  $F(0) = 0$ .

The steps in the solution are:

$$\frac{dF}{dt} = (p + qF)(1 - F) \quad (8.1)$$

$$\frac{dF}{dt} = p + (q - p)F - qF^2 \quad (8.2)$$

$$\int \frac{1}{p + (q - p)F - qF^2} dF = \int dt \quad (8.3)$$

$$\frac{\ln(p + qF) - \ln(1 - F)}{p + q} = t + c_1 \quad (8.4)$$

$$t = 0 \Rightarrow F(0) = 0 \quad (8.5)$$

$$t = 0 \Rightarrow c_1 = \frac{\ln p}{p + q} \quad (8.6)$$

$$F(t) = \frac{p(e^{(p+q)t} - 1)}{pe^{(p+q)t} + q} \quad (8.7)$$

An alternative approach<sup>1</sup> goes as follows. First, split the integral above into partial fractions.

<sup>1</sup> This was suggested by students Muhammad Sagarwalla based on ideas from Alexey Orlovsky.

$$\int \frac{1}{(p + qF)(1 - F)} dF = \int dt \quad (8.8)$$

So we write

$$\frac{1}{(p + qF)(1 - F)} = \frac{A}{p + qF} + \frac{B}{1 - F} \quad (8.9)$$

$$= \frac{A - AF + pB + qFB}{(p + qF)(1 - F)} \quad (8.10)$$

$$= \frac{A + pB + F(qB - A)}{(p + qF)(1 - F)} \quad (8.11)$$

This implies that

$$A + pB = 1 \quad (8.12)$$

$$qB - A = 0 \quad (8.13)$$

Solving we get

$$A = q/(p + q) \quad (8.14)$$

$$B = 1/(p + q) \quad (8.15)$$

so that

$$\int \frac{1}{(p+qF)(1-F)} dF = \int dt \quad (8.16)$$

$$\int \left( \frac{A}{p+qF} + \frac{B}{1-F} \right) dF = t + c_1 \quad (8.17)$$

$$\int \left( \frac{q/(p+q)}{p+qF} + \frac{1/(p+q)}{1-F} \right) dF = t + c_1 \quad (8.18)$$

$$\frac{1}{p+q} \ln(p+qF) - \frac{1}{p+q} \ln(1-F) = t + c_1 \quad (8.19)$$

$$\frac{\ln(p+qF) - \ln(1-F)}{p+q} = t + c_1 \quad (8.20)$$

which is the same as equation (8.4).

We may also solve for

$$f(t) = \frac{dF}{dt} = \frac{e^{(p+q)t} p (p+q)^2}{[pe^{(p+q)t} + q]^2} \quad (8.21)$$

Therefore, if the target market is of size  $m$ , then at each  $t$ , the adoptions are simply given by  $m \times f(t)$ .

For example, set  $m = 100,000$ ,  $p = 0.01$  and  $q = 0.2$ . Then the adoption rate is shown in Figure 8.3.

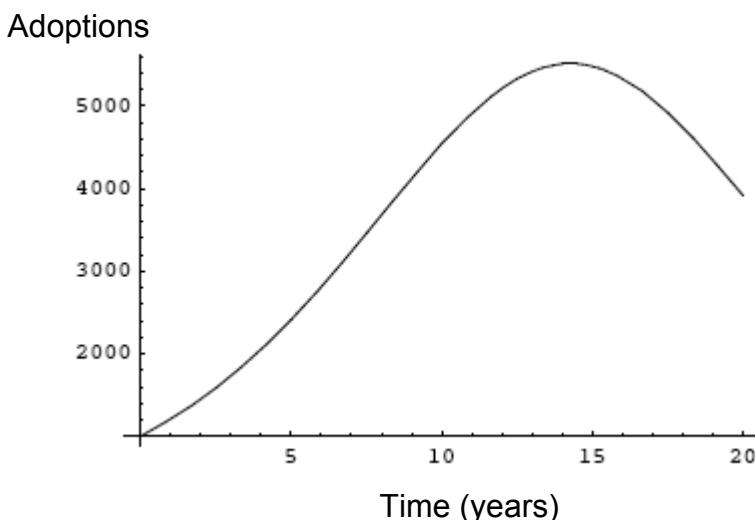


Figure 8.3: Example of the adoption rate:  $m = 100,000$ ,  $p = 0.01$  and  $q = 0.2$ .

#### 8.4.1 Symbolic math in R

The preceding computation may also be undertaken in R, using its symbolic math capability.

```

> #BASS MODEL
> FF = expression(p*(exp((p+q)*t)-1)/(p*exp((p+q)*t)+q))
>
> #Take derivative
> ff = D(FF, "t")
> print(ff)
p * (exp((p + q) * t) * (p + q)) / (p * exp((p + q) * t) + q) -
 p * (exp((p + q) * t) - 1) * (p * (exp((p + q) * t) * (p +
 q))) / (p * exp((p + q) * t) + q)^2

```

We may also plot the same as follows (note the useful `eval` function employed in the next section of code):

```

> #PLOT
> m=100000; p=0.01; q=0.2
>
> t=seq(0,25,0.1)
> fn_f = eval(ff)
> plot(t,fn_f*m,type="l")

```

And this results in a plot identical to that in Figure 8.3. See Figure 8.4.

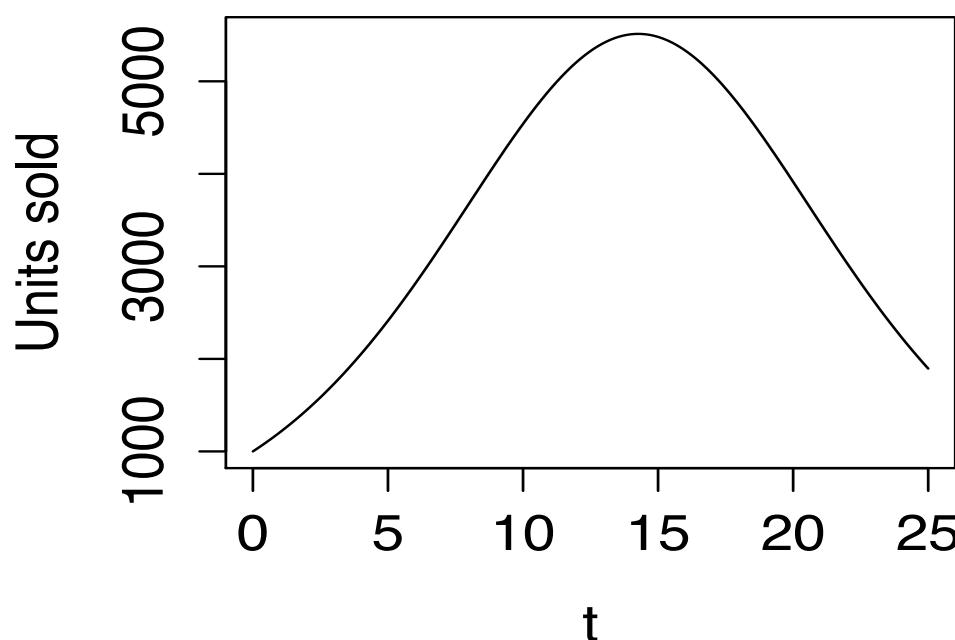


Figure 8.4: Example of the adoption rate:  $m = 100,000$ ,  $p = 0.01$  and  $q = 0.2$ .

## 8.5 Software

The ordinary differential equation here may be solved using free software. One of the widely used open-source packages is called **Maxima** and can be downloaded from many places. A very nice one page user-guide is available at

<http://www.math.harvard.edu/computing/maxima/>

Here is the basic solution of the differential equation in Maxima:

```
Maxima 5.9.0 http://maxima.sourceforge.net
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.

(C1) depends(F,t);
(D1) [F(t)]
(C2) diff(F,t)=(1-F)*(p+q*F);
(D2) dF
 -- = (1 - F) (F q + p)
 dt
(C3) ode2(% ,F,t);
 LOG(F q + p) - LOG(F - 1)
(D3) ----- = t + %C
 q + p
```

Notice that line (D3) of the program output does not correspond to equation (8.4). This is because the function  $\frac{1}{1-F}$  needs to be approached from the left, not the right as the software appears to be doing. Hence, solving by partial fractions results in simple integrals that Maxima will handle properly.

```
(%i1) integrate((q/(p+q))/(p+q*F)+(1/(p+q))/(1-F),F);

 log(q F + p) log(1 - F)
(%o1) ----- - -----
 q + p q + p
```

which is now the exact correct solution, which we use in the model. Another good tool that is free for small-scale symbolic calculations is **WolframAlpha**, available at [www.wolframalpha.com](http://www.wolframalpha.com). See Figure 8.5 for an example of the basic Bass model integral.

The screenshot shows the WolframAlpha interface. In the search bar, the query is "Integrate[1/(p+(q-p)\*F-q\*F^2),F]". Below the search bar are various navigation icons. The main result section is titled "Indefinite Integral" and displays the following equation:

$$\int \frac{1}{p + (q - p)F - qF^2} dF = \frac{\log(Fq + p) - \log(F - 1)}{p + q} + \text{constant}$$

Below this, it says "log(x) is the natural logarithm »". There is also a "Step-by-step solution" button. The "Alternate forms of the integral" section shows two equivalent forms:

$$-\frac{\log(F - 1) - \log(Fq + p)}{p + q} + \text{constant}$$

$$\frac{\log(Fq + p)}{p + q} - \frac{\log(F - 1)}{p + q} + \text{constant}$$

At the bottom of the result area are several small orange navigation icons.

Figure 8.5: Computing the Bass model integral using WolframAlpha.

## 8.6 Calibration

How do we get coefficients  $p$  and  $q$ ? Given we have the current sales history of the product, we can use it to fit the adoption curve.

- Sales in any period are:  $s(t) = m f(t)$ .
- Cumulative sales up to time  $t$  are:  $S(t) = m F(t)$ .

Substituting for  $f(t)$  and  $F(t)$  in the Bass equation gives:

$$\frac{s(t)/m}{1 - S(t)/m} = p + q S(t)/m$$

We may rewrite this as

$$s(t) = [p + q S(t)/m][m - S(t)]$$

Therefore,

$$s(t) = \beta_0 + \beta_1 S(t) + \beta_2 S(t)^2 \quad (8.22)$$

$$\beta_0 = pm \quad (8.23)$$

$$\beta_1 = q - p \quad (8.24)$$

$$\beta_2 = -q/m \quad (8.25)$$

Equation 8.22 may be estimated by a regression of sales against cumulative sales. Once the coefficients in the regression  $\{\beta_0, \beta_1, \beta_2\}$  are obtained, the equations above may be inverted to determine the values of  $\{m, p, q\}$ . We note that since

$$\beta_1 = q - p = -m\beta_2 - \frac{\beta_0}{m},$$

we obtain a quadratic equation in  $m$ :

$$\beta_2 m^2 + \beta_1 m + \beta_0 = 0$$

Solving we have"

$$m = \frac{-\beta_1 \pm \sqrt{\beta_1^2 - 4\beta_0\beta_2}}{2\beta_1}$$

and then this value of  $m$  may be used to solve for

$$p = \frac{\beta_0}{m}; \quad q = -m\beta_2$$

As an example, let's look at the trend for iPhone sales (we store the quarterly sales in a file and read it in, and then undertook the Bass model analysis). The R code for this computation is as follows:

```
> #USING APPLE iPHONE SALES DATA
> data = read.table("iphone_sales.txt", header=TRUE)
> isales = data[,2]
> cum_isales = cumsum(isales)
> cum_isales2 = cum_isales^2
> res = lm(isales ~ cum_isales+cum_isales2)
> print(summary(res))
```

**Call:**

```
lm(formula = isales ~ cum_isales + cum_isales2)
```

**Residuals:**

Min	1Q	Median	3Q	Max
-14.106	-2.877	-1.170	2.436	20.870

**Coefficients:**

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	3.220e+00	2.194e+00	1.468	0.1533
cum_isales	1.216e-01	2.294e-02	5.301	1.22e-05 ***
cum_isales2	-6.893e-05	3.906e-05	-1.765	0.0885 .

**Signif. codes:** 0 \*\*\* 0.001 \*\* 0.01 \* 0.05 . ? 0.1 ? ? 1

Residual standard error: 7.326 on 28 degrees of freedom

Multiple R-squared: 0.854, Adjusted R-squared: 0.8436

F-statistic: 81.89 on 2 and 28 DF, p-value: 1.999e-12

We now proceed to fit the model and then plot it, with actual sales overlaid on the forecast.

```
> #FIT THE MODEL
> m1 = (-b[2]+sqrt(b[2]^2-4*b[1]*b[3]))/(2*b[3])
> m2 = (-b[2]-sqrt(b[2]^2-4*b[1]*b[3]))/(2*b[3])
> print(c(m1,m2))
cum_isales cum_isales
-26.09855 1790.23321
> m = max(m1,m2); print(m)
[1] 1790.233
> p = b[1]/m
> q = -m*b[3]
> print(c(p,q))
(Intercept) cum_isales2
0.00179885 0.12339235
>
> #PLOT THE FITTED MODEL
> nqtrs = 100
> t=seq(0,nqtrs)
> fn_f = eval(ff)*m
> plot(t,fn_f,type="l")
> n = length(isales)
> lines(1:n,isales,col="red",lwd=2,lty=2)
>
```

The outcome is plotted in Figure 8.6. Indeed, it appears that Apple is ready to peak out in sales.

For several other products, Figure 8.7 shows the estimated coefficients reported in Table I of the original Bass (1969) paper.

## 8.7 Sales Peak

It is easy to calculate the time at which adoptions will peak out. Differentiate  $f(t)$  with respect to  $t$ , and set the result equal to zero, i.e.

$$t^* = \operatorname{argmax}_t f(t)$$

which is equivalent to the solution to  $f'(t) = 0$ .

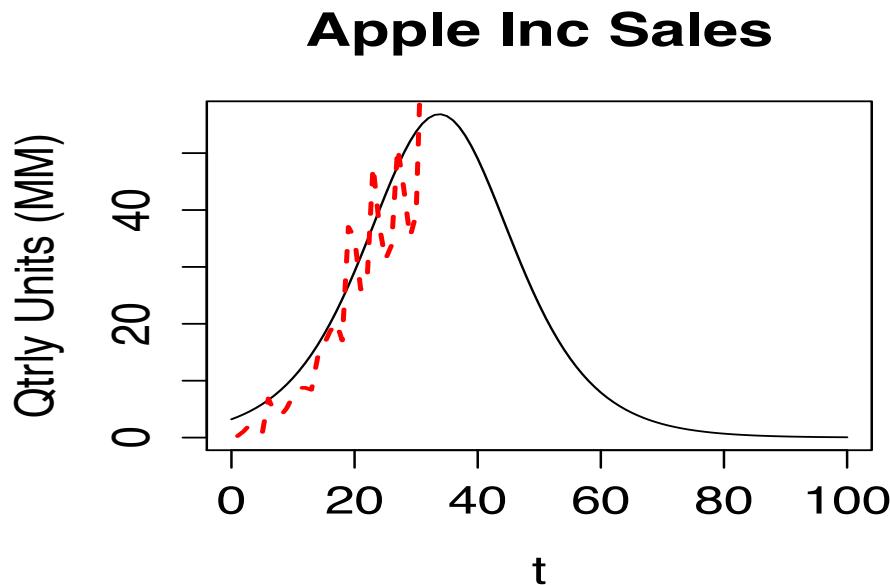


Figure 8.6: Bass model forecast of Apple Inc's quarterly sales. The current sales are also overlaid in the plot.

TABLE I  
*Growth Model Regression Results For Eleven Consumer Durable Products*

Product	Period covered	$a (10^4)$	$b$	$c (10^{-7})$	$R^2$	$a/\sigma_a$	$b/\sigma_b$	$c/\sigma_c$	$m (10^4)$	$p$	$q$
Electric refrigerators	1920-1940	104.67	.21305	-.053913	.903	1.164	6.142	-2.548	40,001	.0026167	.21566
Home freezers	1946-1961	308.12	.15298	-.077868	.742	4.195	4.769	-3.619	21,973	.018119	.17110
Black and white television	1946-1961	2,696.2	.22317	-.025957	.576	3.312	3.724	-3.167	96,717	.027877	.25105
Water softeners	1949-1961	.10256	.27925	-512.59	.919	3.593	8.089	-6.451	5,793	.017703	.29695
Room air conditioners	1946-1961	175.69	.40820	-.24777	.911	1.915	8.317	-6.034	16,895	.010399	.41861
Clothes dryers	1948-1961	259.67	.33968	-.23647	.896	2.941	7.427	-5.701	15,092	.017206	.35688
Power lawnmowers	1948-1961	410.98	.32871	-.075506	.932	1.935	7.408	-4.740	44,751	.0091837	.33790
Electric bed coverings	1949-1961	450.04	.23800	-.031842	.976	3.522	6.820	-1.826	76,589	.005876	.24387
Automatic coffee makers	1948-1961	1,008.2	.28435	-.051242	.883	3.109	6.186	-4.353	58,838	.017135	.30145
Steam irons	1949-1960	1,594.7	.29928	-.058875	.828	3.649	5.288	-4.318	55,696	.028632	.32791
Recover players	1952-1961	543.94	.62931	-.29817	.899	1.911	5.194	-3.718	21,937	.024796	.65410

Data Sources: *Economic Almanac*, *Statistical Abstracts of the U.S.*, *Electrical Merchandising*, and *Electrical Merchandising Week*.

Figure 8.7: Empirical adoption rates and parameters from the Bass paper.

The calculations are simple and give

$$t^* = \frac{-1}{p+q} \ln(p/q) \quad (8.26)$$

Hence, for the values  $p = 0.01$  and  $q = 0.2$ , we have

$$t^* = \frac{-1}{0.01 + 0.2} \ln(0.01/0.2) = 14.2654 \text{ years.}$$

If we examine the plot in Figure 8.3 we see this to be where the graph peaks out.

For the Apple data, here is the computation of the sales peak, reported in number of quarters from inception.

```
> #PEAK SALES TIME POINT (IN QUARTERS)
> tstar = -1/(p+q)*log(p/q)
> print(tstar)
(Intercept)
33.77411
> length(isales)
[1] 31
```

The number of quarters that have already passed is 31. The peak arrives in a half a year!

## 8.8 Notes

The Bass model has been extended to what is known as the generalized Bass model in the paper by [Bass, Krishnan, and Jain \(1994\)](#). The idea is to extend the model to the following equation:

$$\frac{f(t)}{1 - F(t)} = [p + q F(t)] x(t)$$

where  $x(t)$  stands for current marketing effort. This additional variable allows (i) consideration of effort in the model, and (ii) given the function  $x(t)$ , it may be optimized.

The Bass model comes from a deterministic differential equation. Extensions to stochastic differential equations need to be considered.

See also the paper on Bayesian inference in Bass models by [Boatwright and Kamakura \(2003\)](#).

### Exercise

In the Bass model, if the coefficient of imitation increases relative to the coefficient of innovation, then which of the following is the most valid?

- (a) the peak of the product life cycle occurs later.
- (b) the peak of the product life cycle occurs sooner.
- (c) there may be an increasing chance of two life-cycle peaks.
- (d) the peak may occur sooner or later, depending on the coefficient of innovation.

Using peak time formula, substitute  $x = q/p$ :

$$t^* = \frac{-1}{p+q} \ln(p/q) = \frac{\ln(q/p)}{p+q} = \frac{1}{p} \frac{\ln(q/p)}{1+q/p} = \frac{1}{p} \frac{\ln(x)}{1+x}$$

Differentiate with regard to  $x$  (we are interested in the sign of the first derivative  $\partial t^*/\partial q$ , which is the same as sign of  $\partial t^*/\partial x$ ):

$$\frac{\partial t^*}{\partial x} = \frac{1}{p} \left[ \frac{1}{x(1+x)} - \frac{\ln x}{(1+x)^2} \right] = \frac{1+x-x \ln x}{px(1+x)^2}$$

From the Bass model we know that  $q > p > 0$ , i.e.  $x > 1$ , otherwise we could get negative values of acceptance or shape without maximum in the  $0 \leq F < 1$  area. Therefore, the sign of  $\partial t^*/\partial x$  is same as:

$$\text{sign} \left( \frac{\partial t^*}{\partial x} \right) = \text{sign}(1+x-x \ln x), \quad x > 1$$

But this non-linear equation

$$1+x-x \ln x = 0, \quad x > 1$$

has a root  $x \approx 3.59$ .

In other words, the derivative  $\partial t^*/\partial x$  is negative when  $x > 3.59$  and positive when  $x < 3.59$ . For low values of  $x = q/p$ , an increase in the coefficient of imitation  $q$  increases the time to sales peak (illustrated in Figure 8.8), and for high values of  $q/p$  the time decreases with increasing  $q$ . So the right answer for the question appears to be "it depends on values of  $p$  and  $q$ ".

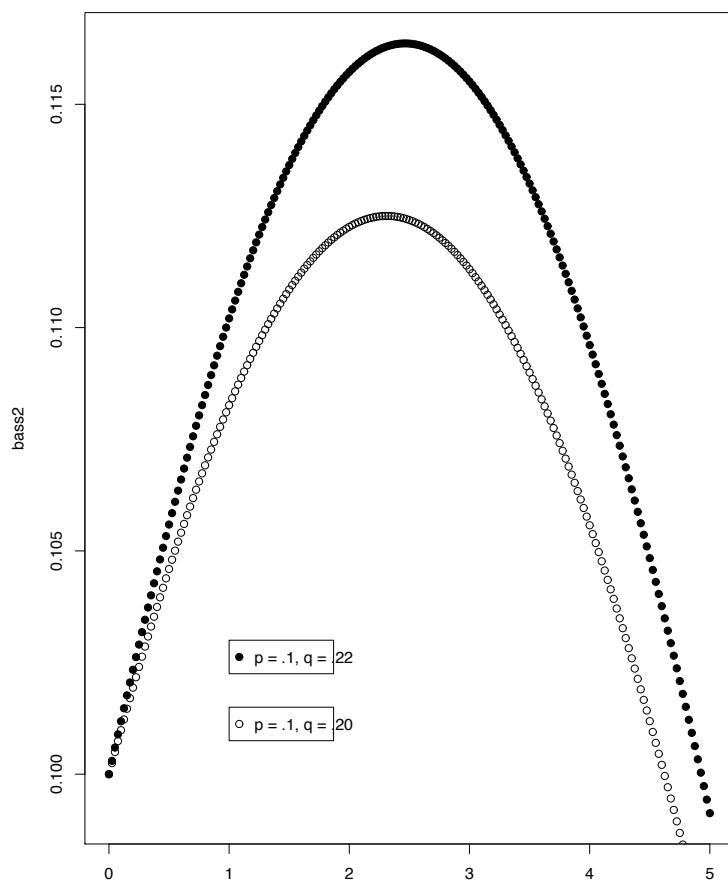


Figure 8.8: Increase in peak time with  $q \uparrow$

# 9

## *Extracting Dimensions: Discriminant and Factor Analysis*

### *9.1 Overview*

In this chapter we will try and understand two common approaches to analyzing large data sets with a view to grouping the data and understanding the main structural components of the data. In discriminant analysis (DA), we develop statistical models that differentiate two or more population types, such as immigrants vs natives, males vs females, etc. In factor analysis (FA), we attempt to collapse an enormous amount of data about the population into a few common explanatory variables. DA is an attempt to explain categorical data, and FA is an attempt to reduce the dimensionality of the data that we use to explain both categorical or continuous data. They are distinct techniques, related in that they both exploit the techniques of linear algebra.

### *9.2 Discriminant Analysis*

In DA, what we are trying to explain is very often a dichotomous split of our observations. For example, if we are trying to understand what determines a good versus a bad creditor. We call the good vs bad the “criterion” variable, or the “dependent” variable. The variables we use to explain the split between the criterion variables are called “predictor” or “explanatory” variables. We may think of the criterion variables as left-hand side variables or dependent variables in the lingo of regression analysis. Likewise, the explanatory variables are the right-hand side ones.

What distinguishes DA is that the left-hand side (lhs) variables are essentially *qualitative* in nature. They have some underlying numerical value, but are in essence qualitative. For example, when universities go

through the admission process, they may have a cut off score for admission. This cut off score discriminates the students that they want to admit and the ones that they wish to reject. DA is a very useful tool for determining this cut off score.

In short, DA is the means by which quantitative explanatory variables are used to explain qualitative criterion variables. The number of qualitative categories need not be restricted to just two. DA encompasses a larger number of categories too.

### 9.2.1 Notation and assumptions

- Assume that there are  $N$  categories or groups indexed by  $i = 2 \dots N$ .
- Within each group there are observations  $y_j$ , indexed by  $j = 1 \dots M_i$ . The size of each group need not be the same, i.e., it is possible that  $M_i \neq M_j$ .
- There are a set of predictor variables  $x = [x_1, x_2, \dots, x_K]'$ . Clearly, there must be good reasons for choosing these so as to explain the groups in which the  $y_j$  reside. Hence the value of the  $k$ th variable for group  $i$ , observation  $j$ , is denoted as  $x_{ijk}$ .
- Observations are mutually exclusive, i.e., each object can only belong to any one of the groups.
- The  $K \times K$  covariance matrix of explanatory variables is assumed to be the same for all groups, i.e.,  $\text{Cov}(x_i) = \text{Cov}(x_j)$ .

### 9.2.2 Discriminant Function

DA involves finding a discriminant function  $D$  that best classifies the observations into the chosen groups. The function may be nonlinear, but the most common approach is to use linear DA. The function takes the following form:

$$D = a_1 x_1 + a_2 x_2 + \dots + a_K x_K = \sum_{k=1}^K a_k x_k$$

where the  $a_k$  coefficients are discriminant weights.

The analysis requires the inclusion of a cut-off score  $C$ . For example, if  $N = 2$ , i.e., there are 2 groups, then if  $D > C$  the observation falls into group 1, and if  $D \leq C$ , then the observation falls into group 2.

Hence, the *objective* function is to choose  $\{\{a_k\}, C\}$  such that classification error is minimized. The equation  $C = D(\{x_k\}; \{a_k\})$  is the equation of a hyperplane that cuts the space of the observations into 2 parts if there are only two groups. Note that if there are  $N$  groups then there will be  $(N - 1)$  cutoffs  $\{C_1, C_2, \dots, C_{N-1}\}$ , and a corresponding number of hyperplanes.

*Exercise*

Draw a diagram of the distribution of 2 groups of observations and the cut off  $C$ . Shade the area under the distributions where observations for group 1 are wrongly classified as group 2; and vice versa.

The variables  $x_k$  are also known as the “discriminants”. In the extraction of the discriminant function, better discriminants will have higher statistical significance.

*Exercise*

Draw a diagram of DA with 2 groups and 2 discriminants. Make the diagram such that one of the variables is shown to be a better discriminant. How do you show this diagrammatically?

### 9.2.3 How good is the discriminant function?

After fitting the discriminant function, the next question to ask is how good the fit is. There are various measures that have been suggested for this. All of them have the essential property that they best separate the distribution of observations for different groups. There are many such measures: (a) Point biserial correlation, (b) Mahalanobis  $D$ , and (c) the confusion matrix. Each of the measures assesses the degree of classification error.

The point biserial correlation is the  $R^2$  of a regression in which the classified observations are signed as  $y_{ij} = 1, i = 1$  for group 1 and  $y_{ij} = 0, i = 2$  for group 2, and the rhs variables are the  $x_{ijk}$  values.

The Mahalanobis distance between any two characteristic vectors for two entities in the data is given by

$$D_M = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$$

where  $\mathbf{x}_1, \mathbf{x}_2$  are two vectors and  $\boldsymbol{\Sigma}$  is the covariance matrix of characteristics of all observations in the data set. First, note that if  $\boldsymbol{\Sigma}$  is the identity

matrix, then  $D_M$  defaults to the Euclidean distance between two vectors. Second, one of the vectors may be treated as the mean vector for a given category, in which case the Mahalanobis distance can be used to assess the distances within and across groups in a pairwise manner. The quality of the discriminant function is then gauged by computing the ratio of the average distance across groups to the average distance within groups. Such ratios are often called the Fisher's discriminant value.

The confusion matrix is a cross-tabulation of the actual versus predicted classification. For example, a  $n$ -category model will result in a  $n \times n$  confusion matrix. A comparison of this matrix with a matrix where the model is assumed to have no classification ability leads to a  $\chi^2$  statistic that informs us about the statistical strength of the classification ability of the model. We will examine this in more detail shortly.

#### 9.2.4 *Caveats*

Be careful to not treat dependent variables that are actually better off remaining continuous as being artificially grouped in qualitative subsets.

#### 9.2.5 *Implementation using R*

We implement a discriminant function model using data for the top 64 teams in the 2005-06 NCAA tournament. The data is as follows (averages per game):

	GMS	PTS	REB	AST	TO	A.T	STL	BLK	PF	FG	FT	X3P
1	6	84.2	41.5	17.8	12.8	1.39	6.7	3.8	16.7	0.514	0.664	0.417
2	6	74.5	34.0	19.0	10.2	1.87	8.0	1.7	16.5	0.457	0.753	0.361
3	5	77.4	35.4	13.6	11.0	1.24	5.4	4.2	16.6	0.479	0.702	0.376
4	5	80.8	37.8	13.0	12.6	1.03	8.4	2.4	19.8	0.445	0.783	0.329
5	4	79.8	35.0	15.8	14.5	1.09	6.0	6.5	13.3	0.542	0.759	0.397
6	4	72.8	32.3	12.8	13.5	0.94	7.3	3.5	19.5	0.510	0.663	0.400
7	4	68.8	31.0	13.0	11.3	1.16	3.8	0.8	14.0	0.467	0.753	0.429
8	4	81.0	28.5	19.0	14.8	1.29	6.8	3.5	18.8	0.509	0.762	0.467
9	3	62.7	36.0	8.3	15.3	0.54	8.0	4.7	19.7	0.407	0.716	0.328
10	3	65.3	26.7	13.0	14.0	0.93	11.3	5.7	17.7	0.409	0.827	0.377
11	3	75.3	29.0	16.0	13.0	1.23	8.0	0.3	17.7	0.483	0.827	0.476
12	3	65.7	41.3	8.7	14.3	0.60	9.3	4.3	19.7	0.360	0.692	0.279
13	3	59.7	34.7	13.3	16.7	0.80	4.7	2.0	17.3	0.472	0.579	0.357
14	3	88.0	33.3	17.0	11.3	1.50	6.7	1.3	19.7	0.508	0.696	0.358

15	3	76.3	27.7	16.3	11.7	1.40	7.0	3.0	18.7	0.457	0.750	0.405
16	3	69.7	32.7	16.3	12.3	1.32	8.3	1.3	14.3	0.509	0.646	0.308
17	2	72.5	33.5	15.0	14.5	1.03	8.5	2.0	22.5	0.390	0.667	0.283
18	2	69.5	37.0	13.0	13.5	0.96	5.0	5.0	14.5	0.464	0.744	0.250
19	2	66.0	33.0	12.0	17.5	0.69	8.5	6.0	25.5	0.387	0.818	0.341
20	2	67.0	32.0	11.0	12.0	0.92	8.5	1.5	21.5	0.440	0.781	0.406
21	2	64.5	43.0	15.5	15.0	1.03	10.0	5.0	20.0	0.391	0.528	0.286
22	2	71.0	30.5	13.0	10.5	1.24	8.0	1.0	25.0	0.410	0.818	0.323
23	2	80.0	38.5	20.0	20.5	0.98	7.0	4.0	18.0	0.520	0.700	0.522
24	2	87.5	41.5	19.5	16.5	1.18	8.5	2.5	20.0	0.465	0.667	0.333
25	2	71.0	40.5	9.5	10.5	0.90	8.5	3.0	19.0	0.393	0.794	0.156
26	2	60.5	35.5	9.5	12.5	0.76	7.0	0.0	15.5	0.341	0.760	0.326
27	2	79.0	33.0	14.0	10.0	1.40	3.0	1.0	18.0	0.459	0.700	0.409
28	2	74.0	39.0	11.0	9.5	1.16	5.0	5.5	19.0	0.437	0.660	0.433
29	2	63.0	29.5	15.0	9.5	1.58	7.0	1.5	22.5	0.429	0.767	0.283
30	2	68.0	36.5	14.0	9.0	1.56	4.5	6.0	19.0	0.398	0.634	0.364
31	2	71.5	42.0	13.5	11.5	1.17	3.5	3.0	15.5	0.463	0.600	0.241
32	2	60.0	40.5	10.5	11.0	0.95	7.0	4.0	15.5	0.371	0.651	0.261
33	2	73.5	32.5	13.0	13.5	0.96	5.5	1.0	15.0	0.470	0.684	0.433
34	1	70.0	30.0	9.0	5.0	1.80	6.0	3.0	19.0	0.381	0.720	0.222
35	1	66.0	27.0	16.0	13.0	1.23	5.0	2.0	15.0	0.433	0.533	0.300
36	1	68.0	34.0	19.0	14.0	1.36	9.0	4.0	20.0	0.446	0.250	0.375
37	1	68.0	42.0	10.0	21.0	0.48	6.0	5.0	26.0	0.359	0.727	0.194
38	1	53.0	41.0	8.0	17.0	0.47	9.0	1.0	18.0	0.333	0.600	0.217
39	1	77.0	33.0	15.0	18.0	0.83	5.0	0.0	16.0	0.508	0.250	0.450
40	1	61.0	27.0	12.0	17.0	0.71	8.0	3.0	16.0	0.420	0.846	0.400
41	1	55.0	42.0	11.0	17.0	0.65	6.0	3.0	19.0	0.404	0.455	0.250
42	1	47.0	35.0	6.0	17.0	0.35	9.0	4.0	20.0	0.298	0.750	0.160
43	1	57.0	37.0	8.0	24.0	0.33	9.0	3.0	12.0	0.418	0.889	0.250
44	1	62.0	33.0	8.0	20.0	0.40	8.0	5.0	21.0	0.391	0.654	0.500
45	1	65.0	34.0	17.0	17.0	1.00	11.0	2.0	19.0	0.352	0.500	0.333
46	1	71.0	30.0	10.0	10.0	1.00	7.0	3.0	20.0	0.424	0.722	0.348
47	1	54.0	35.0	12.0	22.0	0.55	5.0	1.0	19.0	0.404	0.667	0.300
48	1	57.0	40.0	2.0	5.0	0.40	5.0	6.0	16.0	0.353	0.667	0.500
49	1	81.0	30.0	13.0	15.0	0.87	9.0	1.0	29.0	0.426	0.846	0.350
50	1	62.0	37.0	14.0	18.0	0.78	7.0	0.0	21.0	0.453	0.556	0.333
51	1	67.0	37.0	12.0	16.0	0.75	8.0	2.0	16.0	0.353	0.867	0.214
52	1	53.0	32.0	15.0	12.0	1.25	6.0	3.0	16.0	0.364	0.600	0.368

53	1	73.0	34.0	17.0	19.0	0.89	3.0	3.0	20.0	0.520	0.750	0.391
54	1	71.0	29.0	16.0	10.0	1.60	10.0	6.0	21.0	0.344	0.857	0.393
55	1	46.0	30.0	10.0	11.0	0.91	3.0	1.0	23.0	0.365	0.500	0.333
56	1	64.0	35.0	14.0	17.0	0.82	5.0	1.0	20.0	0.441	0.545	0.333
57	1	64.0	43.0	5.0	11.0	0.45	6.0	1.0	20.0	0.339	0.760	0.294
58	1	63.0	34.0	14.0	13.0	1.08	5.0	3.0	15.0	0.435	0.815	0.091
59	1	63.0	36.0	11.0	20.0	0.55	8.0	2.0	18.0	0.397	0.643	0.381
60	1	52.0	35.0	8.0	8.0	1.00	4.0	2.0	15.0	0.415	0.500	0.235
61	1	50.0	19.0	10.0	17.0	0.59	12.0	2.0	22.0	0.444	0.700	0.300
62	1	56.0	42.0	3.0	20.0	0.15	2.0	2.0	17.0	0.333	0.818	0.200
63	1	54.0	22.0	13.0	10.0	1.30	6.0	1.0	20.0	0.415	0.889	0.222
64	1	64.0	36.0	16.0	13.0	1.23	4.0	0.0	19.0	0.367	0.833	0.385

We loaded in the data and ran the following commands (which are stored in the program file `lda.R`:

```
ncaa = read.table("ncaa.txt", header=TRUE)
x = as.matrix(ncaa[4:14])
y1 = 1:32
y1 = y1*0+1
y2 = y1*0
y = c(y1,y2)

library(MASS)
dm = lda(y~x)
```

Hence the top 32 teams are category 1 ( $y = 1$ ) and the bottom 32 teams are category 2 ( $y = 0$ ). The results are as follows:

```
> lda(y~x)
Call:
lda(y ~ x)

Prior probabilities of groups:
 0 1
0.5 0.5

Group means:
 xPTS xREB xAST xTO xA.T xSTL xBLK xPF
0 62.10938 33.85938 11.46875 15.01562 0.835625 6.609375 2.375 18.84375
1 72.09375 35.07500 14.02812 12.90000 1.120000 7.037500 3.125 18.46875
 xFG xFT xX3P
0 0.4001562 0.6685313 0.3142187
1 0.4464375 0.7144063 0.3525313

Coefficients of linear discriminants:
```

	LD1
xPTS	-0.02192489
xREB	0.18473974
xAST	0.06059732
xTO	-0.18299304
xA.T	0.40637827
xSTL	0.24925833
xBLK	0.09090269
xPF	0.04524600
xFG	19.06652563
xFT	4.57566671
xX3P	1.87519768

Some useful results can be extracted as follows:

```
> result = lda(y~x)
> result$prior
 0 1
0.5 0.5
> result$means
 xPTS xREB xAST xTO xA.T xSTL xBLK xPF
0 62.10938 33.85938 11.46875 15.01562 0.835625 6.609375 2.375 18.84375
1 72.09375 35.07500 14.02812 12.90000 1.120000 7.037500 3.125 18.46875
 xFG xFT xX3P
0 0.4001562 0.6685313 0.3142187
1 0.4464375 0.7144063 0.3525313
> result$call
lda(formula = y ~ x)
> result$N
[1] 64
> result$svd
[1] 7.942264
```

The last line contains the singular value decomposition level, which is also the level of the Fischer discriminant, which gives the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics.

We can look at the performance of the model as follows:

```
> result = lda(y~x)
> predict(result)$class
 [1] 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0
[39] 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0
Levels: 0 1
```

If we want the value of the predicted normalized discriminant function we simply do

```
> predict(result)
```

The cut off is treated as being at zero.

### 9.2.6 Confusion Matrix

As we have seen before, the confusion matrix is a tabulation of actual and predicted values. To generate the confusion matrix for our basketball example here we use the following commands in R:

```
> result = lda(y~x)
> y_pred = predict(result)$class
> length(y_pred)
[1] 64
> table(y,y_pred)
y 0 1
y
0 27 5
1 5 27
```

We can see that 5 of the 64 teams have been misclassified. Is this statistically significant? In order to assess this, we compute the  $\chi^2$  statistic for the confusion matrix. Let's define the confusion matrix as

$$A = \begin{bmatrix} 27 & 5 \\ 5 & 27 \end{bmatrix}$$

This matrix shows some classification ability. Now we ask, what if the model has no classification ability, then what would the average confusion matrix look like? It's easy to see that this would give a matrix that would assume no relation between the rows and columns, and the numbers in each cell would reflect the average number drawn based on row and column totals. In this case since the row and column totals are all 32, we get the following confusion matrix of no classification ability:

$$E = \begin{bmatrix} 16 & 16 \\ 16 & 16 \end{bmatrix}$$

The test statistic is the sum of squared normalized differences in the cells of both matrices, i.e.,

$$\text{Test-Stat} = \sum_{i,j} \frac{[A_{ij} - E_{ij}]^2}{E_{ij}}$$

We compute this in R.

```
> A = matrix(c(27,5,5,27),2,2)
> A
```

```

[,1] [,2]
[1,] 27 5
[2,] 5 27
> E = matrix(c(16,16,16,16),2,2)
> E
[,1] [,2]
[1,] 16 16
[2,] 16 16
> test_stat = sum((A-E)^2/E)
> test_stat
[1] 30.25
> 1-pchisq(test_stat,1)
[1] 3.797912e-08

```

The  $\chi^2$  distribution requires entering the degrees of freedom. In this case, the degrees of freedom is 1, i.e., equal to  $(r - 1)(c - 1)$ , where  $r$  is the number of rows and  $c$  is the number of columns. We see that the probability of the  $A$  and  $E$  matrices being the same is zero. Hence, the test suggests that the model has statistically significant classification ability.

### 9.2.7 Multiple groups

What if we wanted to discriminate the NCAA data into 4 groups? Its just as simple:

```

> y1 = rep(3,16)
> y2 = rep(2,16)
> y3 = rep(1,16)
> y4 = rep(0,16)
> y = c(y1,y2,y3,y4)
> res = lda(y~x)
> res
Call:
lda(y ~ x)

Prior probabilities of groups:
 0 1 2 3
0.25 0.25 0.25 0.25

Group means:
 xPTS xREB xAST xTO xA.T xSTL xBLK xPF xFG
0 61.43750 33.18750 11.93750 14.37500 0.888750 6.12500 1.8750 19.5000 0.4006875
1 62.78125 34.53125 11.00000 15.65625 0.782500 7.09375 2.8750 18.1875 0.3996250
2 70.31250 36.59375 13.50000 12.71875 1.094375 6.84375 3.1875 19.4375 0.4223750
3 73.87500 33.55625 14.55625 13.08125 1.145625 7.23125 3.0625 17.5000 0.4705000
 xFT xX3P
0 0.7174375 0.3014375
1 0.6196250 0.3270000
2 0.7055625 0.3260625

```

## *Exercise*

Use the spreadsheet titled `default-analysis-data.xls` and fit a model to discriminate firms that default from firms that do not. How good a fit does your model achieve?

### 9.3 Eigen Systems

We now move on to understanding some properties of matrices that may be useful in classifying data or deriving its underlying components. We download Treasury interest rate date from the FRED website, <http://research.stlouisfed.org/fred2/>. I have placed the data in a file called `tryrates.txt`. Let's read in the file.

```
> rates = read.table("tryrates.txt", header=TRUE)
> names(rates)
[1] "DATE" "FYGM3" "FYGM6" "FYGT1" "FYGT2" "FYGT3" "FYGT5" "FYGT7"
```

[9] "FYGT10"

A  $M \times M$  matrix  $A$  has attendant  $M$  eigenvectors  $V$  and eigenvalue  $\lambda$  if we can write

$$\lambda V = A V$$

Starting with matrix  $A$ , the eigenvalue decomposition gives both  $V$  and  $\lambda$ . It turns out we can find  $M$  such eigenvalues and eigenvectors, as there is no unique solution to this equation. We also require that  $\lambda \neq 0$ .

We may implement this in R as follows, setting matrix  $A$  equal to the covariance matrix of the rates of different maturities:

```
> eigen(cov(rates))
$values
[1] 7.070996e+01 1.655049e+00 9.015819e-02 1.655911e-02 3.001199e-03
[6] 2.145993e-03 1.597282e-03 8.562439e-04

$vectors
[,1] [,2] [,3] [,4] [,5] [,6]
[1,] -0.3596990 -0.49201202 0.59353257 -0.38686589 -0.34419189 -0.07045281
[2,] -0.3581944 -0.40372601 0.06355170 0.20153645 0.79515713 0.07823632
[3,] -0.3875117 -0.28678312 -0.30984414 0.61694982 -0.45913099 0.20442661
[4,] -0.3753168 -0.01733899 -0.45669522 -0.19416861 0.03906518 -0.46590654
[5,] -0.3614653 0.13461055 -0.36505588 -0.41827644 -0.06076305 -0.14203743
[6,] -0.3405515 0.31741378 -0.01159915 -0.18845999 -0.03366277 0.72373049
[7,] -0.3260941 0.40838395 0.19017973 -0.05000002 0.16835391 0.09196861
[8,] -0.3135530 0.47616732 0.41174955 0.42239432 -0.06132982 -0.42147082
[,7] [,8]
[1,] 0.04282858 0.03645143
[2,] -0.15571962 -0.03744201
[3,] 0.10492279 -0.16540673
[4,] 0.30395044 0.54916644
[5,] -0.45521861 -0.55849003
[6,] -0.19935685 0.42773742
[7,] 0.70469469 -0.39347299
[8,] -0.35631546 0.13650940

> rcorr = cor(rates)
> rcorr
 FYGM3 FYGM6 FYGT1 FYGT2 FYGT3 FYGT5 FYGT7
FYGM3 1.0000000 0.9975369 0.9911255 0.9750889 0.9612253 0.9383289 0.9220409
FYGM6 0.9975369 1.0000000 0.9973496 0.9851248 0.9728437 0.9512659 0.9356033
FYGT1 0.9911255 0.9973496 1.0000000 0.9936959 0.9846924 0.9668591 0.9531304
FYGT2 0.9750889 0.9851248 0.9936959 1.0000000 0.9977673 0.9878921 0.9786511
FYGT3 0.9612253 0.9728437 0.9846924 0.9977673 1.0000000 0.9956215 0.9894029
FYGT5 0.9383289 0.9512659 0.9668591 0.9878921 0.9956215 1.0000000 0.9984354
FYGT7 0.9220409 0.9356033 0.9531304 0.9786511 0.9894029 0.9984354 1.0000000
FYGT10 0.9065636 0.9205419 0.9396863 0.9680926 0.9813066 0.9945691 0.9984927
 FYGT10
FYGM3 0.9065636
FYGM6 0.9205419
FYGT1 0.9396863
FYGT2 0.9680926
FYGT3 0.9813066
FYGT5 0.9945691
FYGT7 0.9984927
FYGT10 1.0000000
```

So we calculated the eigenvalues and eigenvectors for the covariance matrix of the data. What does it really mean? Think of the covariance matrix as the summarization of the connections between the rates of different maturities in our data set. What we do not know is how many dimensions of commonality there are in these rates, and what is the relative importance of these dimensions. For each dimension of commonality, we wish to ask (a) how important is that dimension (the eigenvalue), and (b) the relative influence of that dimension on each rate (the values in the eigenvector). The most important dimension is the one with the highest eigenvalue, known as the “principal” eigenvalue, corresponding to which we have the principal eigenvector. It should be clear by now that the eigenvalue and its eigenvector are “eigen pairs”. It should also be intuitive why we call this the eigenvalue “decomposition” of a matrix.

## 9.4 Factor Analysis

Factor analysis is the use of eigenvalue decomposition to uncover the underlying structure of the data. Given a data set of observations and explanatory variables, factor analysis seeks to achieve a decomposition with these two properties:

1. Obtain a reduced dimension set of explanatory variables, known as derived/extracted/discovered factors. Factors must be *orthogonal*, i.e., uncorrelated with each other.
2. Obtain data reduction, i.e., suggest a limited set of variables. Each such subset is a manifestation of an abstract underlying dimension.

These subsets are also ordered in terms of their ability to explain the variation across observations.

See the article by Richard Darlington  
<http://www.psych.cornell.edu/Darlington/factor.htm>  
 which is as good as any explanation one can get. See also the article by Statsoft:  
<http://www.statsoft.com/textbook/stfacan.html>

### 9.4.1 Notation

- Observations:  $y_i, i = 1 \dots N$ .
- Original explanatory variables:  $x_{ik}, k = 1 \dots K$ .

- Factors:  $F_j, j = 1 \dots M$ .
- $M < K$ .

#### 9.4.2 The Idea

As you can see in the rates data, there are eight different rates. If we wanted to model the underlying drivers of this system of rates, we could assume a separate driver for each one leading to  $K = 8$  underlying factors. But the whole idea of factor analysis is to reduce the number of drivers that exist. So we may want to go with a smaller number of  $M < K$  factors.

The main concept here is to “project” the variables  $x \in R^K$  onto the reduced factor set  $F \in R^M$  such that we can explain most of the variables by the factors. Hence we are looking for a relation

$$x = BF$$

where  $B = \{b_{kj}\} \in R^{K \times M}$  is a matrix of factor “loadings” for the variables. Through matrix  $B$ ,  $x$  may be represented in smaller dimension  $M$ . The entries in matrix  $B$  may be positive or negative. Negative loadings mean that the variable is negatively correlated with the factor. The whole idea is that we want to replace the relation of  $y$  to  $x$  with a relation of  $y$  to a reduced set  $F$ .

Once we have the set of factors defined, then the  $N$  observations  $y$  may be expressed in terms of the factors through a factor “score” matrix  $A = \{a_{ij}\} \in R^{N \times M}$  as follows:

$$y = AF$$

Again, factor scores may be positive or negative. There are many ways in which such a transformation from variables to factors might be undertaken. We look at the most common one.

#### 9.4.3 Principal Components Analysis (PCA)

In PCA, each component (factor) is viewed as a weighted combination of the other variables (this is not always the way factor analysis is implemented, but is certainly one of the most popular).

The starting point for PCA is the covariance matrix of the data. Essentially what is involved is an eigenvalue analysis of this matrix to extract the principal eigenvectors.

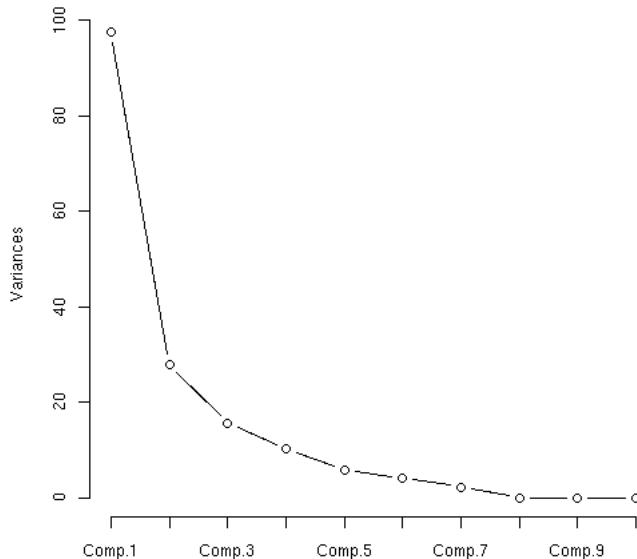
We can do the analysis using the R statistical package. Here is the sample session:

```
> ncaa = read.table("ncaa.txt", header=TRUE)
> x = ncaa[4:14]
> result = princomp(x)
> screeplot(result)
> screeplot(result, type="lines")
```

The results are as follows:

```
> summary(result)
Importance of components:
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
Standard deviation 9.8747703 5.2870154 3.9577315 3.19879732 2.43526651
Proportion of Variance 0.5951046 0.1705927 0.0955943 0.06244717 0.03619364
Cumulative Proportion 0.5951046 0.7656973 0.8612916 0.92373878 0.95993242
 Comp.6 Comp.7 Comp.8 Comp.9
Standard deviation 2.04505010 1.53272256 0.1314860827 1.062179e-01
Proportion of Variance 0.02552391 0.01433727 0.0001055113 6.885489e-05
Cumulative Proportion 0.98545633 0.99979360 0.9998991100 0.9999680e-01
 Comp.10 Comp.11
Standard deviation 6.591218e-02 3.007832e-02
Proportion of Variance 2.651372e-05 5.521365e-06
Cumulative Proportion 0.999945e-01 1.000000e-00
```

The resultant “screeplot” shows the amount explained by each component.



Lets look at the loadings. These are the respective eigenvectors:

```
> result$loadings
Loadings:
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
PTS 0.964 0.240
```

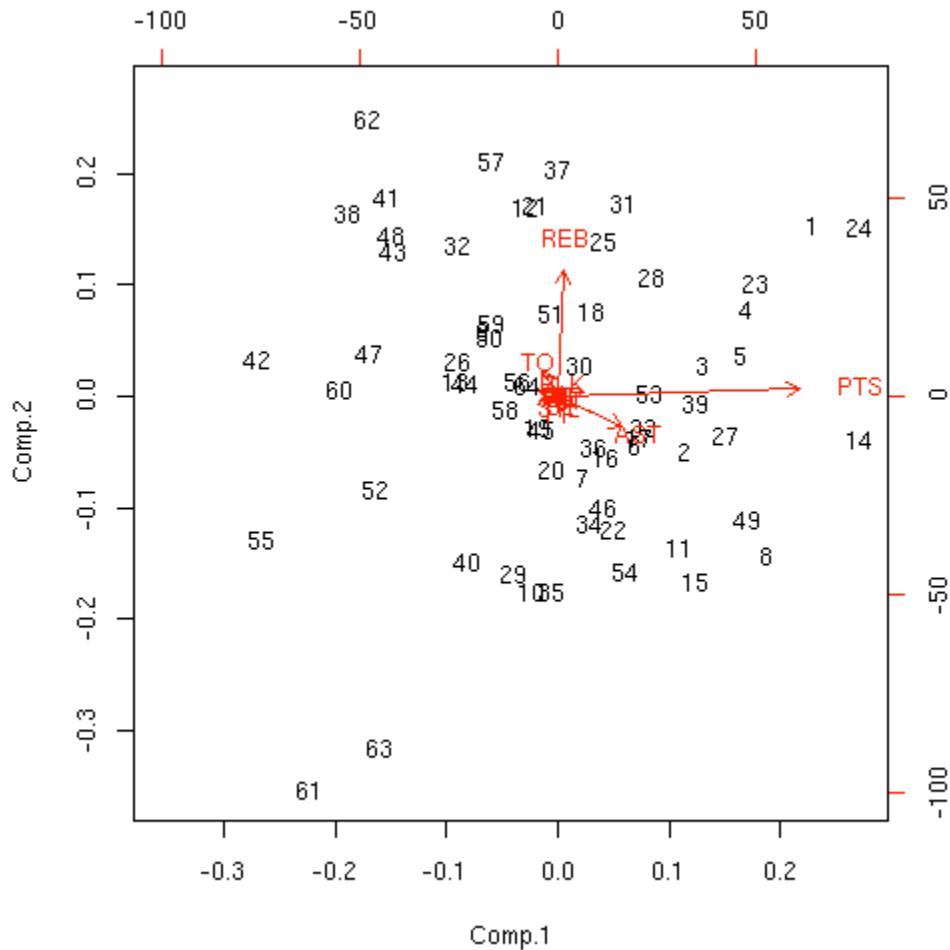
REB	0.940		-0.316					
AST	0.257	-0.228	-0.283	-0.431	-0.778			
TO		0.194	-0.908	-0.116	0.313	-0.109		
A.T						0.712	0.642	0.262
STL			-0.194	0.205		0.816	0.498	
BLK					0.516	-0.849		
PF			-0.110	-0.223	0.862	-0.364	-0.228	
FG								
FT						0.619	-0.762	0.175
X3P						-0.315		0.948
Comp.11								
PTS								
REB								
AST								
TO								
A.T								
STL								
BLK								
PF								
FG	-0.996							
FT								
X3P								

We can see that the main variable embedded in the first principal component is PTS. (Not surprising!). We can also look at the standard deviation of each component:

```
> result$sdv
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
 9.87477028 5.28701542 3.95773149 3.19879732 2.43526651 2.04505010 1.53272256
 Comp.8 Comp.9 Comp.10 Comp.11
 0.13148608 0.10621791 0.06591218 0.03007832
```

The biplot shows the first two components and overlays the variables as well. This is a really useful visual picture of the results of the analysis.

```
> biplot(result)
```



The alternative function `prcomp` returns the same stuff, but gives all the factor loadings immediately.

```
> prcomp(x)
Standard deviations:
 [1] 9.95283292 5.32881066 3.98901840 3.22408465 2.45451793 2.06121675
 [7] 1.54483913 0.13252551 0.10705759 0.06643324 0.03031610

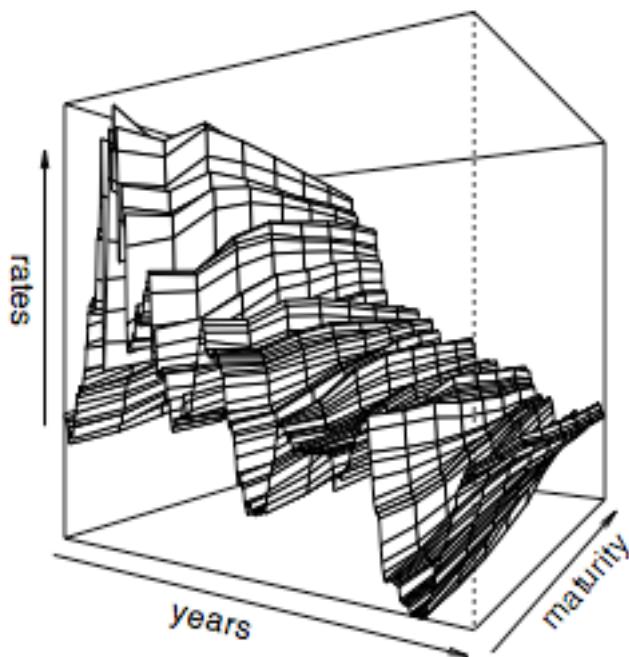
Rotation:
 PC1 PC2 PC3 PC4 PC5
PTS -0.963808450 -0.052962387 0.018398319 0.094091517 -0.240334810
REB -0.022483140 -0.939689339 0.073265952 0.026260543 0.315515827
AST -0.256799635 0.228136664 -0.282724110 -0.430517969 0.778063875
TO 0.061658120 -0.193810802 -0.908005124 -0.115659421 -0.313055838
A.T -0.021008035 0.030935414 0.035465079 -0.022580766 0.068308725
STL -0.006513483 0.081572061 -0.193844456 0.205272135 0.014528901
BLK -0.012711101 -0.070032329 0.035371935 0.073370876 -0.034410932
PF -0.012034143 0.109640846 -0.223148274 0.862316681 0.364494150
FG -0.003729350 0.002175469 -0.001708722 -0.006568270 -0.001837634
FT -0.001210397 0.003852067 0.001793045 0.008110836 -0.019134412
X3P -0.003804597 0.003708648 -0.001211492 -0.002352869 -0.003849550
```

	PC6	PC7	PC8	PC9	PC10
PTS	0.029408534	-0.0196304356	0.0026169995	-0.004516521	0.004889708
REB	-0.040851345	-0.0951099200	-0.0074120623	0.003557921	-0.008319362
AST	-0.044767132	0.0681222890	0.0359559264	0.056106512	0.015018370
TO	0.108917779	0.0864648004	-0.0416005762	-0.039363263	-0.012726102
A.T	-0.004846032	0.0061047937	-0.7122315249	-0.642496008	-0.262468560
STL	-0.815509399	-0.4981690905	0.0008726057	-0.008845999	-0.005846547
BLK	-0.516094006	0.8489313874	0.0023262933	-0.001364270	0.008293758
PF	0.228294830	0.0972181527	0.0005835116	0.001302210	-0.001385509
FG	0.004118140	0.0041758373	0.0848448651	-0.019610637	0.030860027
FT	-0.005525032	0.0001301938	-0.6189703010	0.761929615	-0.174641147
X3P	0.001012866	0.0094289825	0.3151374823	0.038279107	-0.948194531
	PC11				
PTS	0.0037883918				
REB	-0.0043776255				
AST	0.0058744543				
TO	-0.0001063247				
A.T	-0.0560584903				
STL	-0.0062405867				
BLK	0.0013213701				
PF	-0.0043605809				
FG	-0.9956716097				
FT	-0.0731951151				
X3P	-0.0031976296				

#### 9.4.4 Application to Treasury Yield Curves

We had previously downloaded monthly data for constant maturity yields from June 1976 to December 2006. Here is the 3D plot. It shows the change in the yield curve over time for a range of maturities.

```
> persp(rates, theta=30, phi=0, xlab="years", ylab="maturity", zlab="rates")
```



As before, we undertake a PCA of the system of Treasury rates. The commands are the same as with the basketball data.

```
> tryrates = read.table("tryrates.txt", header=TRUE)
> rates = as.matrix(tryrates[2:9])
> result = princomp(rates)
> result$loadings

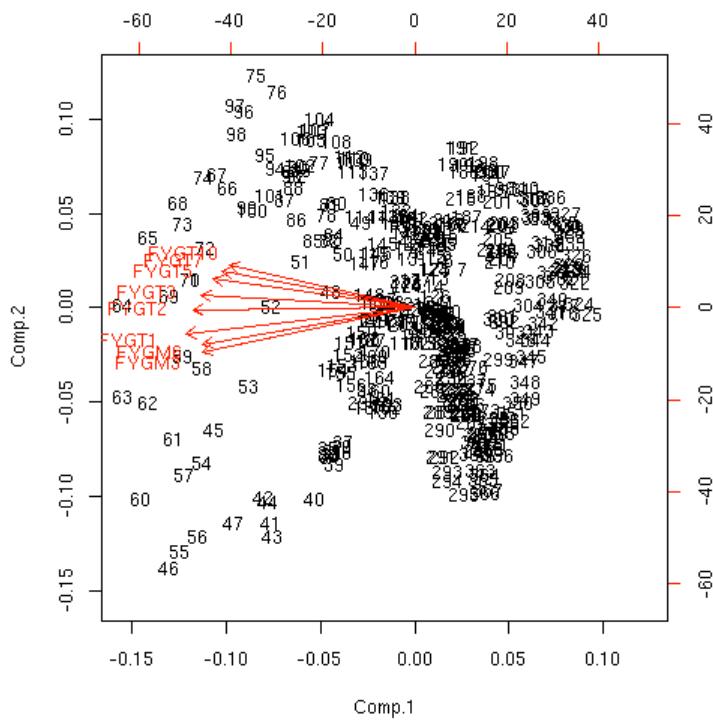
Loadings:
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
FYGM3 -0.360 -0.492 0.594 -0.387 -0.344
FYGM6 -0.358 -0.404 0.202 0.795 0.156
FYGt1 -0.388 -0.287 -0.310 0.617 -0.459 0.204 -0.105 -0.165
FYGt2 -0.375 -0.457 -0.194 -0.466 -0.304 0.549
FYGt3 -0.361 0.135 -0.365 -0.418 -0.142 0.455 -0.558
FYGt5 -0.341 0.317 -0.188 0.724 0.199 0.428
FYGt7 -0.326 0.408 0.190 0.168 -0.705 -0.393
FYGt10 -0.314 0.476 0.412 0.422 -0.421 0.356 0.137

> result$sdev
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
8.39745750 1.28473300 0.29985418 0.12850678 0.05470852 0.04626171 0.03991152
 Comp.9
0.02922175

> summary(result)

Importance of components:
 Comp.1 Comp.2 Comp.3 Comp.4
Standard deviation 8.397458 1.28473300 0.299854180 0.1285067846
Proportion of Variance 0.975588 0.02283477 0.001243916 0.0002284667
Cumulative Proportion 0.975588 0.99842275 0.999666666 0.9998951326
 Comp.5 Comp.6 Comp.7 Comp.8
Standard deviation 5.470852e-02 4.626171e-02 3.991152e-02 2.922175e-02
Proportion of Variance 4.140766e-05 2.960835e-05 2.203775e-05 1.181363e-05
Cumulative Proportion 0.999365e-01 0.999661e-01 0.999882e-01 1.000000e+00
```

The results are interesting. We see that the loadings are large in the first three component vectors for all maturity rates. The loadings correspond to a classic feature of the yield curve, i.e., there are three components: level, slope, and curvature. Note that the first component has almost equal loadings for all rates that are all identical in sign. Hence, this is the *level* factor. The second component has negative loadings for the shorter maturity rates and positive loadings for the later maturity ones. Therefore, when this factor moves up, the short rates will go down, and the long rates will go up, resulting in a steepening of the yield curve. If the factor goes down, the yield curve will become flatter. Hence, the second principal component is clearly the *slope* factor. Examining the loadings of the third principal component should make it clear that the effect of this factor is to modulate the “curvature” or hump of the yield curve. Still, from looking at the results, it is clear that 97% of the common variation is explained by just the first factor, and a wee bit more by the next two. The resultant “biplot” shows the dominance of the main component.



Notice that the variables are almost all equally weighting on the first component. The length of the vectors corresponds to the factor loadings.

#### 9.4.5 Application: Risk Parity and Risk Dispersion

Risk parity – see Theirry Roncalli’s book

Risk disparity – see Mark Kritzman’s paper.

#### 9.4.6 Difference between PCA and FA

The difference between PCA and FA is that for the purposes of matrix computations PCA assumes that all variance is common, with all unique factors set equal to zero; while FA assumes that there is some unique variance. Hence PCA may also be thought of as a subset of FA. The level of unique variance is dictated by the FA model which is chosen. Accordingly, PCA is a model of a closed system, while FA is a model of an open system. FA tries to decompose the correlation matrix into common and unique portions.

#### 9.4.7 Factor Rotation

Finally, there are some times when the variables would load better on the factors if the factor system were to be rotated. This called factor rotation, and many times the software does this automatically.

Remember that we decomposed variables  $x$  as follows:

$$x = B F + e$$

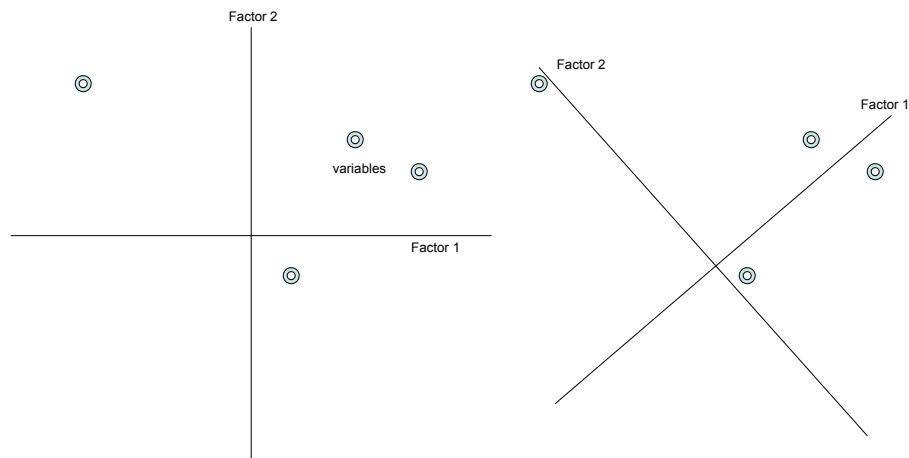
where  $x$  is dimension  $K$ ,  $B \in R^{K \times M}$ ,  $F \in R^M$ , and  $e$  is a  $K$ -dimension vector. This implies that

$$\text{Cov}(x) = BB' + \psi$$

Recall that  $B$  is the matrix of factor loadings. The system remains unchanged if  $B$  is replaced by  $BG$ , where  $G \in R^{M \times M}$ , and  $G$  is orthogonal. Then we call  $G$  a “rotation” of  $B$ .

The idea of rotation is easier to see with the following diagram. Two conditions need to be satisfied: (a) The new axis (and the old one) should be orthogonal. (b) The difference in loadings on the factors by each variable must increase. In the diagram below we can see that the rotation has made the variables align better along the new axis system.

## Factor Rotation



### 9.4.8 Using the factor analysis function

To illustrate, let's undertake a factor analysis of the Treasury rates data. In R, we can implement it generally with the `factanal` command.

```
> factanal(rates, 2)
```

**Call:**

```
factanal(x = rates, factors = 2)
```

**Uniquenesses:**

FYGM <sub>3</sub>	FYGM <sub>6</sub>	FYGT <sub>1</sub>	FYGT <sub>2</sub>	FYGT <sub>3</sub>	FYGT <sub>5</sub>	FYGT <sub>7</sub>	FYGT <sub>10</sub>
0.006	0.005	0.005	0.005	0.005	0.005	0.005	0.005

**Loadings:**

	Factor <sub>1</sub>	Factor <sub>2</sub>
FYGM <sub>3</sub>	0.843	0.533
FYGM <sub>6</sub>	0.826	0.562
FYGT <sub>1</sub>	0.793	0.608
FYGT <sub>2</sub>	0.726	0.686
FYGT <sub>3</sub>	0.681	0.731
FYGT <sub>5</sub>	0.617	0.786
FYGT <sub>7</sub>	0.579	0.814
FYGT <sub>10</sub>	0.546	0.836

	Factor <sub>1</sub>	Factor <sub>2</sub>
--	---------------------	---------------------

SS loadings	4.024	3.953
Proportion Var	0.503	0.494
Cumulative Var	0.503	0.997

Test of the hypothesis that 2 factors are sufficient.

The chi square statistic is 3556.38 on 13 degrees of freedom.

The p-value is 0

Notice how the first factor explains the shorter maturities better and the second factor explains the longer maturity rates. Hence, the two factors cover the range of maturities. Note that the ability of the factors to separate the variables increases when we apply a **factor rotation**:

```
> factanal(rates, 2, rotation = "promax")
```

**Call:**

```
factanal(x = rates, factors = 2, rotation = "promax")
```

Uniquenesses:

FYGM <sub>3</sub>	FYGM <sub>6</sub>	FYGT <sub>1</sub>	FYGT <sub>2</sub>	FYGT <sub>3</sub>	FYGT <sub>5</sub>	FYGT <sub>7</sub>	FYGT <sub>10</sub>
0.006	0.005	0.005	0.005	0.005	0.005	0.005	0.005

Loadings:

	Factor <sub>1</sub>	Factor <sub>2</sub>
FYGM <sub>3</sub>	0.110	0.902
FYGM <sub>6</sub>	0.174	0.846
FYGT <sub>1</sub>	0.282	0.747
FYGT <sub>2</sub>	0.477	0.560
FYGT <sub>3</sub>	0.593	0.443
FYGT <sub>5</sub>	0.746	0.284
FYGT <sub>7</sub>	0.829	0.194
FYGT <sub>10</sub>	0.895	0.118

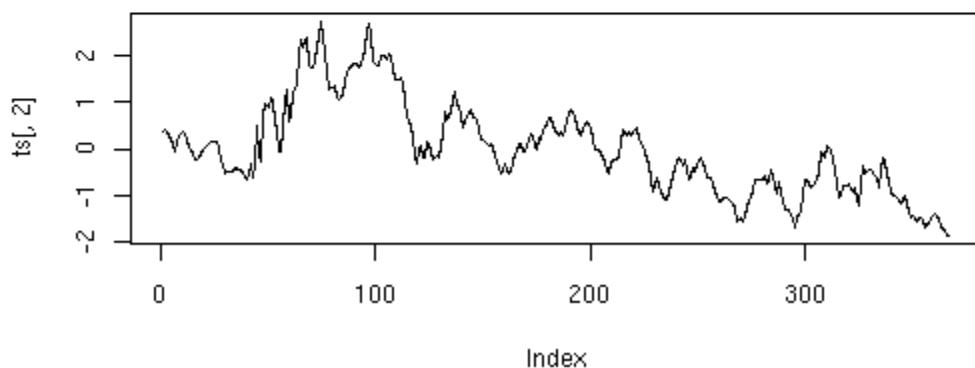
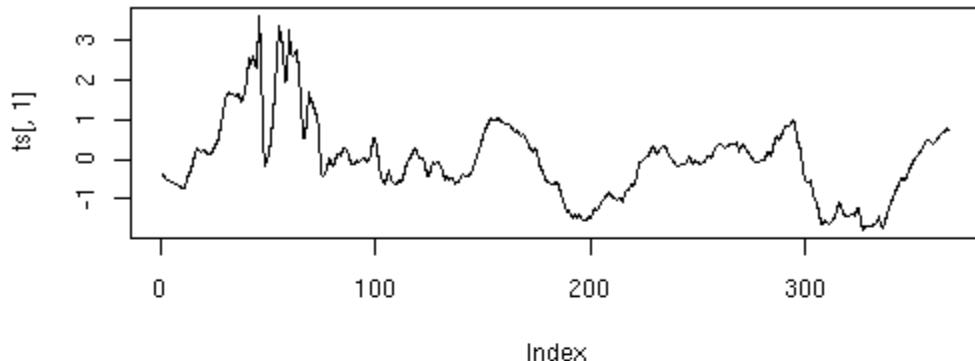
	Factor <sub>1</sub>	Factor <sub>2</sub>
--	---------------------	---------------------

SS loadings	2.745	2.730
Proportion Var	0.343	0.341
Cumulative Var	0.343	0.684

The factors have been reversed after the rotation. Now the first factor explains long rates and the second factor explains short rates. If we want the time series of the factors, use the following command:

```
result = factanal(rates, 2, scores="regression")
ts = result$scores
> par(mfrow=c(2,1))
> plot(ts[,1], type="l")
> plot(ts[,2], type="l")
```

The results are plotted here. The plot represents the normalized factor time series.



Thus there appears to be a slow-moving first component and a fast moving second one.

